Finding Software License Violations Through Binary Code Clone Detection

MSR 2011, Waikiki, Hawaii

Armijn Hemel[1] Karl Trygve Kalleberg[2]
Rob Vermaas[3] **Eelco Dolstra**[3]

[1]The gpl-violations.org Project & Tjaldur Software Governance Solutions

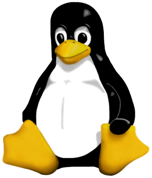
[2]KolibriFX, Norway

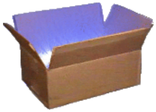
[3]Delft University of Technology, Department of Software Technology, Netherlands

May 21, 2011

Motivation: finding GPL violations

Motivation: finding GPL violations





samba

Motivation: finding GPL violations

GNU General Public License v2

“ You may copy and distribute the Program [...] in object code or executable form [...] provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code [...]; or,

b) Accompany it with a written offer [...] to give any third party [...] a complete machine-readable copy of the corresponding source code [...] ”

samba

The risks of non-compliance

FOR IMMEDIATE RELEASE

DISTRICT COURT OF FRANKFURT ISSUES VERDICT ON GPL VIOLATION OF D-LINK

BERLIN, Germany - September 22, 2006 -- The gpl-violations.org project prevails in court litigation against D-Link Germany GmbH regarding D-Link's alleged inappropriate and copyright infringing use of parts of the Linux Operating System Kernel.

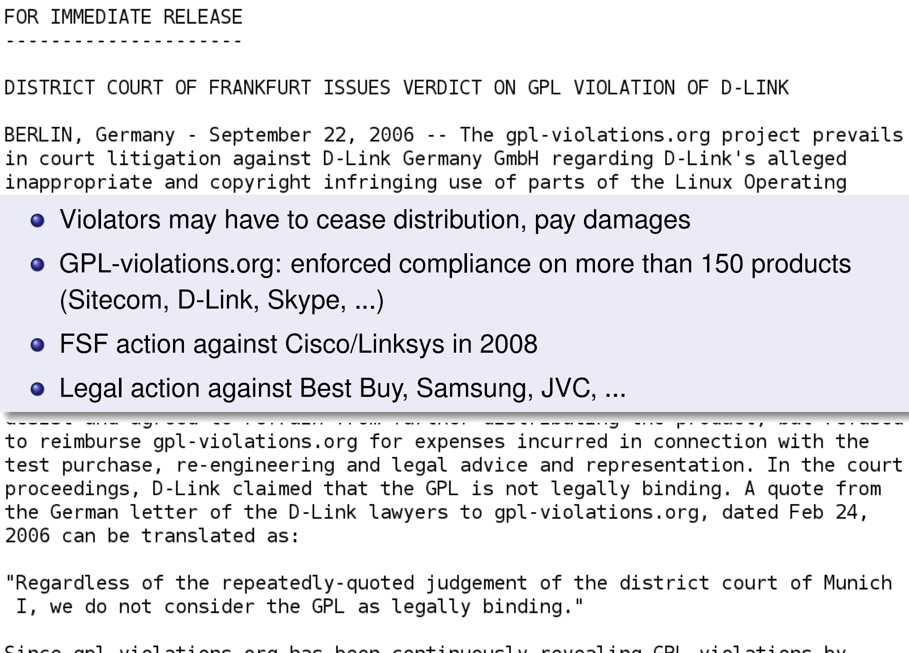
D-Link Germany GmbH, a subsidiary of D-Link Corporation, Taiwan R.O.C., distributed DSM-G600, a network attached storage (NAS) device which uses a Linux-based Operating System. However, this distribution was incompliant with the GNU General Public License (GPL) which covers the Linux Kernel and many other software programs used in the product.

Following-up a legal warning notice, D-Link signed a declaration to cease and desist and agreed to refrain from further distributing the product, but refused to reimburse gpl-violations.org for expenses incurred in connection with the test purchase, re-engineering and legal advice and representation. In the court proceedings, D-Link claimed that the GPL is not legally binding. A quote from the German letter of the D-Link lawyers to gpl-violations.org, dated Feb 24, 2006 can be translated as:

"Regardless of the repeatedly-quoted judgement of the district court of Munich I, we do not consider the GPL as legally binding."

Since gpl-violations.org has been continuously revealing GPL violations by

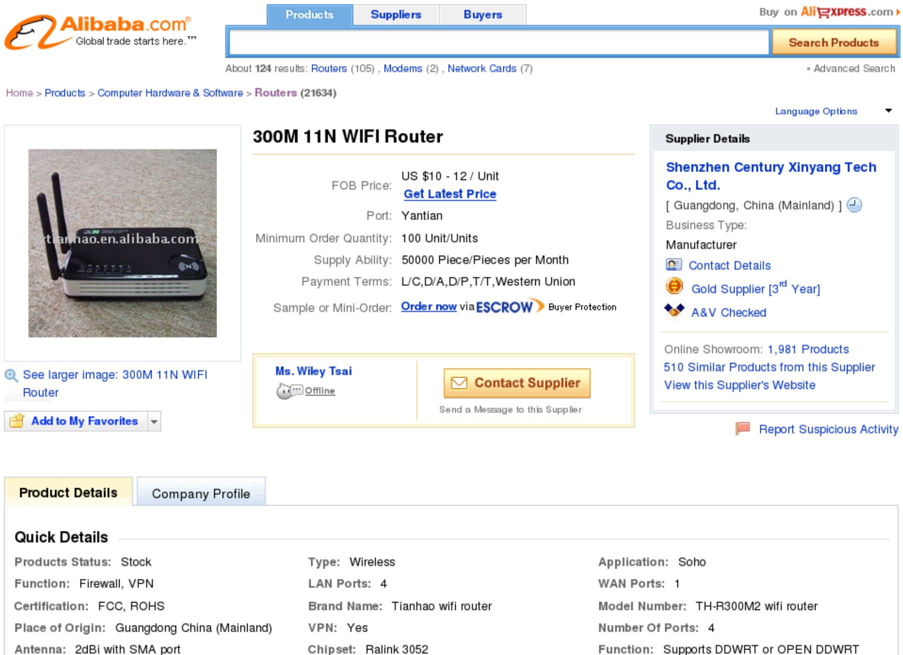
The risks of non-compliance



```
FOR IMMEDIATE RELEASE
---------------------

DISTRICT COURT OF FRANKFURT ISSUES VERDICT ON GPL VIOLATION OF D-LINK

BERLIN, Germany - September 22, 2006 -- The gpl-violations.org project prevails
in court litigation against D-Link Germany GmbH regarding D-Link's alleged
inappropriate and copyright infringing use of parts of the Linux Operating

desist and agreed to refrain from further distributing the product, but refused
to reimburse gpl-violations.org for expenses incurred in connection with the
test purchase, re-engineering and legal advice and representation. In the court
proceedings, D-Link claimed that the GPL is not legally binding. A quote from
the German letter of the D-Link lawyers to gpl-violations.org, dated Feb 24,
2006 can be translated as:

"Regardless of the repeatedly-quoted judgement of the district court of Munich
 I, we do not consider the GPL as legally binding."

Since gpl-violations.org has been continuously revealing GPL violations by
```

- Violators may have to cease distribution, pay damages
- GPL-violations.org: enforced compliance on more than 150 products (Sitecom, D-Link, Skype, ...)
- FSF action against Cisco/Linksys in 2008
- Legal action against Best Buy, Samsung, JVC, ...

Inadvertent violations: The supply chain



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

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

The problem: What’s in this binary blob?

```
00000000  50 4b 03 04 14 00 00 00  08 00 29 52 57 3c fa c0  |PK........)RW<..|
00000010  03 a7 26 9e 16 01 f4 ae  19 01 15 00 00 00 76 31  |..&...........v1|
00000020  2e 31 2e 31 2e 31 37 5f  53 4d 43 5f 61 6c 6c 2e  |.1.1.17_SMC_all.|
00000030  65 78 65 ec 3a 6d 78 53  55 9a f7 26 69 9a 42 ca  |exe.:mxSU..&i.B.|
00000040  0d d0 38 65 69 30 60 50  94 96 56 43 91 98 06 03  |..8ei0`P..VC....|
00000050  92 18 9f e1 e3 d6 c8 4d  03 f4 03 69 6b b8 a3 88  |.......M...ik...|
00000060  78 2f 83 da 76 c3 a6 d9  6d 7a 37 0f 38 8b 33 ae  |x/..v...mz7.8.3.|
00000070  33 ce d0 89 ee 8a f8 38  ae 3a 88 1f 30 61 c2 52  |3......8.:..0a.R|
00000080  3a ea 33 ac e3 02 0e 3c  b3 38 ea ee e9 a4 ce d4  |:.3....<.8......|
00000090  85 2d 01 0b 77 df f7 dc  f4 03 1c 67 66 9f fd db  |.-..w......gf...|
000000a0  ab 37 f7 9c f7 bc e7 fd  38 e7 bc 5f a7 ac 5c bb  |.7......8.._..\.|
000000b0  8b d1 33 0c 63 80 57 55  19 e6 00 a3 3d 5e e6 cf  |..3.c.WU....=^..|
000000c0  3f 67 e1 9d 72 fd 5b 53  98 d7 8b de 9f 7d 80 5d  |?g..r.[S.....}.]|
000000d0  f1 fe ec fb 22 9b 1e b5  6f d9 fa f0 03 5b 37 3c  |...."...o....[7<|
000000e0  64 df b8 61 f3 e6 87 25  fb fd 2d f6 ad f2 66 fb  |d..a...%..-...f.|
000000f0  a6 cd f6 e5 ab 83 f6 87  1e 6e 6e 59 50 5c 3c c9  |.........nnYP\<.|
00000100  91 a7 d1 fc c1 99 4b f6  d7 5e dd 3b f2 5e da f5  |......K..^.;.^..|
00000110  f2 de 6a f8 ae 7e e9 cd  bd f3 e0 9b fa c9 3b 7b  |..j..~........;{|
00000120  17 d2 fe 81 bd 9b e0 fb  eb 5d fb f6 56 52 dc d7  |.........]..VR..|
00000130  f6 7e 1f be 37 ee 7a 73  ef 2d f0 fd af 9f be be  |.~..7.zs.-......|
00000140  77 36 7c ef dd b4 31 82  74 46 64 e4 7d 0c b3 82  |w6|...1.tFd.}...|
00000150  35 30 43 1b fd 9e 31 b9  39 76 32 6b 64 98 2a 96  |50C...1.9v2kd.*.|
00000160  61 9a f4 14 76 a1 1b 7e  2c a8 38 ab 69 6f d1 fa  |a...v..~,.8.io..|
00000170  86 fc 9c 91 2f b3 c7 a0  8d c1 a3 a3 bf 96 7c df  |..../.........|.|
00000180  32 0a b7 8c 5b a3 c8 3d  2c b3 07 1b c7 59 e6 85  |2...[..=,....Y..|
```

Binary code clone detection

Goal

- We need a tool that can detect code cloning in binaries
- Detecting a clone doesn’t mean a license violation (which cannot be decided automatically), but it’s a necessary pre-condition

Users

- Copyright holders
- Downstream vendors

Binary clone detection

Previous work

- BAT: a tool for reverse-engineering binaries (used by `gpl-violations.org`)
- BAT did ad-hoc scans for patterns denoting common violations, e.g. the string “`BusyBox v`” indicates the presence of BusyBox
- Sæbjørnsen *et al.*: disassembly-based techniques

Binary clone detection

Previous work

- BAT: a tool for reverse-engineering binaries (used by `gpl-violations.org`)
- BAT did ad-hoc scans for patterns denoting common violations, e.g. the string "`BusyBox v`" indicates the presence of BusyBox
- Sæbjørnsen *et al.*: disassembly-based techniques

This paper

Mine repositories of open source packages to detect cloning of *any* of them in a given binary

Methods:

1. Searching for string literals
2. Compressibility
3. Binary diffs

Method 1: detection using strings

Step 1: extract string literals from lots of open source packages into a *database*

```
printk(KERN_NOTICE "0x%012llx-0x%012llx : \"%s\"\n", (unsigned long long)slave->off
        (unsigned long long)(slave->offset + slave->mtd.size), slave->mtd.name);

/* let's do some sanity checks */
if (slave->offset >= master->size) {
        /* let's register it anyway to preserve ordering */
        slave->offset = 0;
        slave->mtd.size = 0;
        printk(KERN_ERR"mtd: partition \"%s\" is out of reach -- disabled\n",
                part->name);
        goto out_register;
}
if (slave->offset + slave->mtd.size > master->size) {
        slave->mtd.size = master->size - slave->offset;
        printk(KERN_WARNING"mtd: partition \"%s\" extends beyond the end of device
                part->name, master->name, (unsigned long long)slave->mtd.size);
}
if (master->numeraseregions > 1) {
```

Method 1: detection using strings

Step 1: extract string literals from lots of open source packages into a *database*

```
printk(KERN_NOTICE "0x%012llx-0x%012llx : \"%s\"\n", (unsigned long long)slave->off
        (unsigned long long)(slave->offset + slave->mtd.size), slave->mtd.name);

/* let's do some sanity checks */
if (slave->offset >= master->size) {
        /* let's register it anyway to preserve ordering */
        slave->offset = 0;
        slave->mtd.size = 0;
        printk(KERN_ERR"mtd: partition \"%s\" is out of reach --
                part->name);
        goto out_register;
}
if (slave->offset + slave->mtd.size > master->size) {
        slave->mtd.size = master->size - slave->offset;
        printk(KERN_WARNING"mtd: partition \"%s\" extends beyond the end of device
                part->name, master->name, (unsigned long long)slave->mtd.size);
}
if (master->numeraseregions > 1) {
```

Method 1: detection using strings

- The corpus: 23,896 packages from Fedora 5, 9, 11 and 14
- 1,728,718 C and C++ source files
- 42,238,120 string literals
- 13 GiB SQLite DB
- Most common string: `"%s"` (3495 packages)
- Most common word: `"version"` (1749 packages)
- Most common sentence: `"Out of memory"` (586 packages)

Method 1: detection using strings

Step 2: extract strings from the binary

```
$ strings /tmp/tmpzevICi/tmplNoDrJ/tmpkghqD0
*,0[
testsetup_long
testsetup
initcall_debug
init=
...
<5>Removing MTD device #%d (%s) with use count %d
dev:    size   erasesize  name
mtd%d: %8.8x %8.8x "%s"
<5>Creating %d MTD partitions on "%s":
memory allocation error while creating partitions for "%s"
<5>Moving partition %d: 0x%08x -> 0x%08x
<5>0x%08x-0x%08x : "%s"
mtd: partition "%s" is out of reach - disabled
mtd: partition "%s" extends beyond the end of device "%s" - size truncated to %#x
mtd: partition "%s" doesn't start on an erase block boundary - force read-only
mtd: partition "%s" doesn't end on an erase block - force read-only
<5>%s partition parsing not available
<5>%d %s partitions found on MTD device %s
```

Method 1: detection using strings

Step 3: match strings against the DB

```
$ strings /tmp/tmpzevICi/tmplNoDrJ/tmpkghqD0
*,0[
testsetup_long
testsetup
initcall_debug
init=
...
<5>Removing MTD device #%d (%s) with use count %d
dev:    size   erasesize  name
mtd%d: %8.8x %8.8x "%s"
<5>Creating %d MTD partitions on "%s":
memory allocation error while creating partitions for "%s"
<5>Moving partition %d: 0x%08x -> 0x%08x
<5>0x%08x-0x%08x : "%s"
mtd: partition "%s" is out of reach - disabled
mtd: partition "%s" extends beyond the end of device "%s" - size truncated to %#x
mtd: partition "%s" doesn't start on an erase block boundary - force read-only
mtd: partition "%s" doesn't end on an erase block - force read-only
<5>%s partition parsing not available
<5>%d %s partitions found on MTD device %s
```

Method 1: detection using strings

Step 3: match strings against the DB

```
$ strings /tmp/tmpzevICi/tmplNoDrJ/tmpkghqD0
*,0[
testsetup_long
testsetup
initcall_debug
init=
...
<5>Removing MTD device #%d (%s) with use count %d
dev:    size   erasesize  name
mtd%d: %8.8x %8.8x "%s"
<5>Creating %d MTD partitions
memory allocation error while
<5>Moving partition %d: 0x%08
<5>0x%08x-0x%08x : "%s"
mtd: partition "%s" is out of reach - disabled
mtd: partition "%s" extends beyond the end of device "%s" - size truncated to %#x
mtd: partition "%s" doesn't start on an erase block boundary - force read-only
mtd: partition "%s" doesn't end on an erase block - force read-only
<5>%s partition parsing not available
<5>%d %s partitions found on MTD device %s
```

Found in
`linux-2.6.15/drivers/mtd/mtdpart.c`!

Method 1: detection using strings

Step 4: compute score for each package, present result

- Strings that occur in multiple packages get a lower score

Method 1: detection using strings

Step 4: compute score for each package, present result

- Strings that occur in multiple packages get a lower score

Score	Package	# Unique	Top strings
21687.30	`linux`	1035	`"%d (%s) %c %d %d %d %lu %lu ..."`
			`"key msqid perms cbytes qnum lspid ..."`
			`"mtd:  partition "%s" extends beyond ..."`
5147.63	`u-boot`	196	`"## Transferring control to NetBSD..."`
			`"image contents (magic number, header..."`
			`"address 'addr' in memory; this includes..."`

...

Method 2: detection using compression

- Basic idea: if the concatenation of two binaries compresses much better than the individual binaries, this is evidence of cloning
- Requires a repository of binary packages; slow and (partially) arch-dependent, but doesn't depend on string literals or source code

Method 2: detection using compression

- Basic idea: if the concatenation of two binaries compresses much better than the individual binaries, this is evidence of cloning
- Requires a repository of binary packages; slow and (partially) arch-dependent, but doesn't depend on string literals or source code

Example: does `svn` contain part/all of `libsqlite3.a`?

$$\begin{aligned} |C(\texttt{svn})| &= 2{,}563{,}804 \\ |C(\texttt{libsqlite3.a})| &= 252{,}872 \\ |C(\texttt{svn libsqlite3.a})| &= 2{,}576{,}616 \end{aligned}$$

So the compression of the concatenation is 240,060 bytes shorter, strong evidence that `svn` contains a clone of `libsqlite3.a`.

$$\mathrm{reuse}_c(x,y) = \frac{|C(x)| + |C(y)| - |C(xy)|}{|C(y)|} \approx .95$$

Method 2: detection using compression

Evaluation: we checked a statically linked `svn` binary from one Linux distribution against a corpus of 134 static libraries from Debian 6.0; cut-off at 0.1.

$\mathrm{reuse}_c(\texttt{svn}, p)$	Package p
0.945	`libsqlite3.a`
0.899	`libexpat.a`
0.868	`libdb.a`
0.842	**libdb_cxx.a**
0.839	`libz.a`
0.823	`libxml2.a`
0.772	`libneon.a`
0.765	`libapr-1.a`
0.694	`libcrypto.a`
0.675	`libssl.a`
0.441	`libpthread.a`

...

Method 3: detection using binary diffs

- Basic idea: if the binary patch from b_1 to b_2 is much shorter than the patch from ε to b_2, then b_1 probably contains a clone of b_2

Method 3: detection using binary diffs

- Basic idea: if the binary patch from b_1 to b_2 is much shorter than the patch from ε to b_2, then b_1 probably contains a clone of b_2

Example: does `svn` contain part/all of `libsqlite3.a`?

$$\begin{aligned} |D(\texttt{svn}, \texttt{libsqlite3.a})| &= 26{,}130 \\ |D(\varepsilon, \texttt{libsqlite3.a})| &= 261{,}138 \end{aligned}$$

Thus `libsqlite3.a` can be cheaply reconstructed from `svn`, strong evidence that `svn` contains a clone of `libsqlite3.a`.

Evaluation

To determine precision and recall, all methods were applied to manually constructed static binaries (rather than third-party firmwares, where the false negatives aren't known).

- String method: recall = 0.83, precision = 0.85.
- Compression method: recall = 0.72, precision = 0.91.
- Diff method: recall = 0.64, precision = 0.89.

String method on some third-party binaries:

Binary	Type	Size (MiB)	*tp*	*fp*	Precision
Vodafone Webby	Firmware	29	42	46	0.48
Asus WL500G	Firmware	2	26	12	0.68
Spotify	Core dump	344	27	61	0.31

Conclusions and future work

Conclusions

- The string method is simple, effective, architecture-independent, easy to interpret
- The compression/diff methods are much slower, architecture-dependent, hard to interpret, but don’t rely on the presence of strings or availability of source code
- The compression method performs better than the diff method

Conclusions and future work

Conclusions

- The string method is simple, effective, architecture-independent, easy to interpret
- The compression/diff methods are much slower, architecture-dependent, hard to interpret, but don't rely on the presence of strings or availability of source code
- The compression method performs better than the diff method

Future work

- Need better way to deal with internal cloning in the source repository
- Evaluate the compression/diff methods on a much larger scale
  - E.g. against all releases/architectures of Debian rather than just 134 static libraries
- Apply this to the Apple App Store / Android Marketplace
  - 350,000 apps in the App Store is bound to give interesting results