Techniques to Detect License Violations of Open-Source Systems

Rainer Koschke

Arbeitsgruppe Softwaretechnik
Fachbereich Mathematik und Informatik
Universität Bremen

BAUHAUS SOFTWARE TECHNOLOGIES

The 9th CREST Open Workshop
Code Provenance and Clone Detection
Tuesday, 23rd November, 2010

The Cisco case

CISCO SYSTEMS ®

Offshore Development Organization → Broadcom → Linksys → Cisco

Pillory `http://gpl-violations.org/`

How short is a copyright violation?

Question

How short may a copy be to be still considered a violation?

How short is a copyright violation?

Question

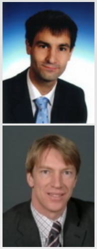
How short may a copy be to be still considered a violation?

Answer

54 LOC (out of 160 KLOC) may be sufficient if they are essential for the operation of a program.

– American Court (Mertzel, 2008)

Questionnaire among more than 100 developers



- copying of open-source software (OSS) by commercial companies is frequent
- developers are lacking understanding of the legal risks
- organizations are lacking policies for OSS use

— Sojer und Henkel (2010)

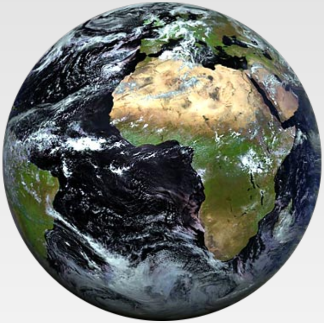
Audits of OSS use

Audits of OSS use

$$323 \cdot 2^{20}[\mathrm{LOC}] \cdot 50[\mathrm{characters}/\mathrm{LOC}] \cdot 0.5[\mathrm{cm}/\mathrm{LOC}]$$
$$= 83,860.256[\mathrm{km}]$$

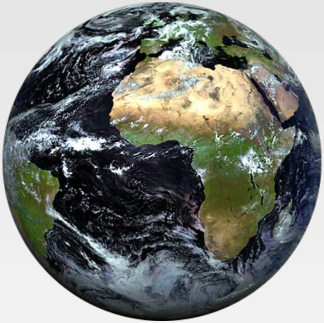
Audits of OSS use

$323 \cdot 2^{20}[\text{LOC}] \cdot 50[\text{characters}/\text{LOC}] \cdot 0.5[\text{cm}/\text{LOC}]$
$= 83,860.256[\text{km}]$

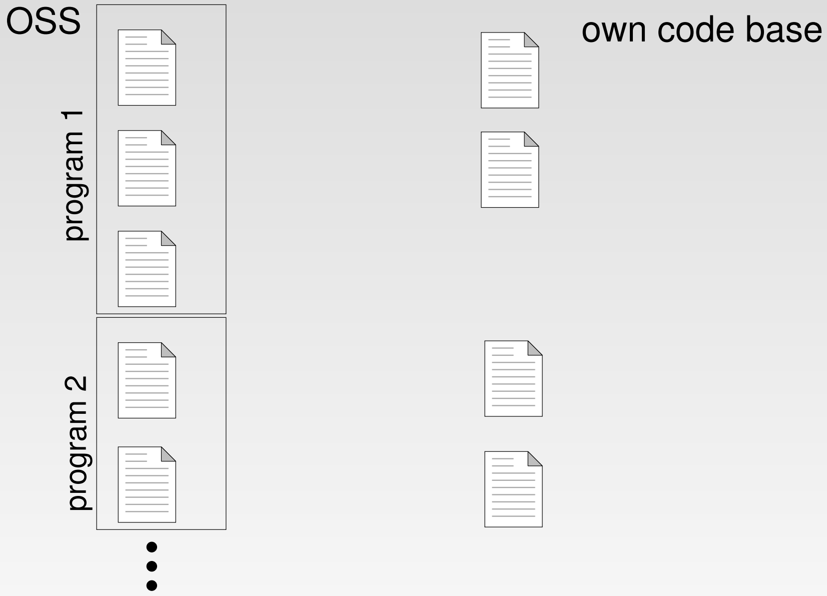


Audits of OSS use

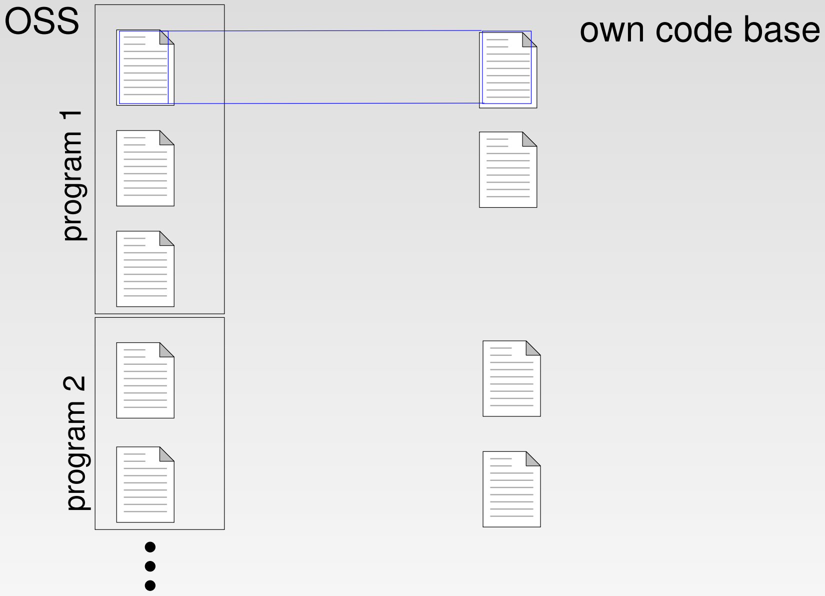
$$323 \cdot 2^{20}[\text{LOC}] \cdot 50[\text{characters/LOC}] \cdot 0.5[\text{cm/LOC}]$$
$$= 83,860.256[\text{km}]$$



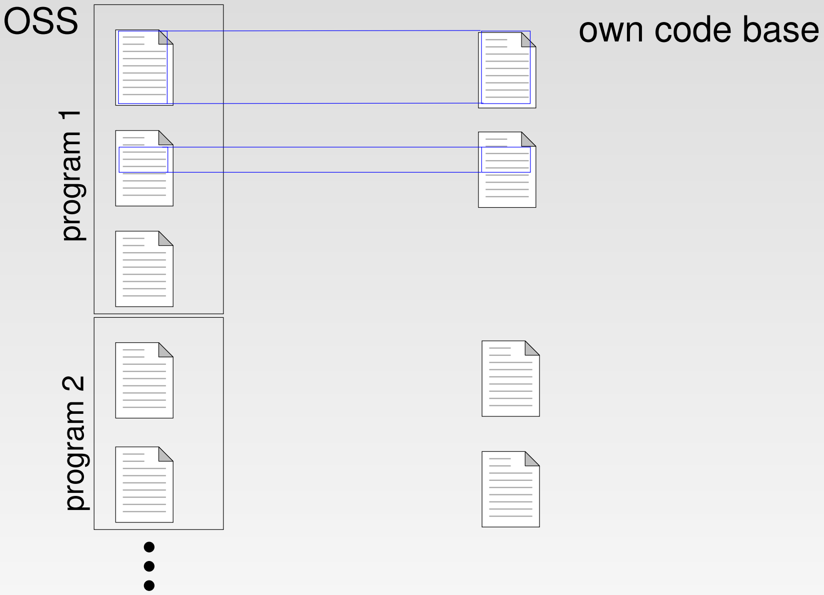
Search for copyright violations



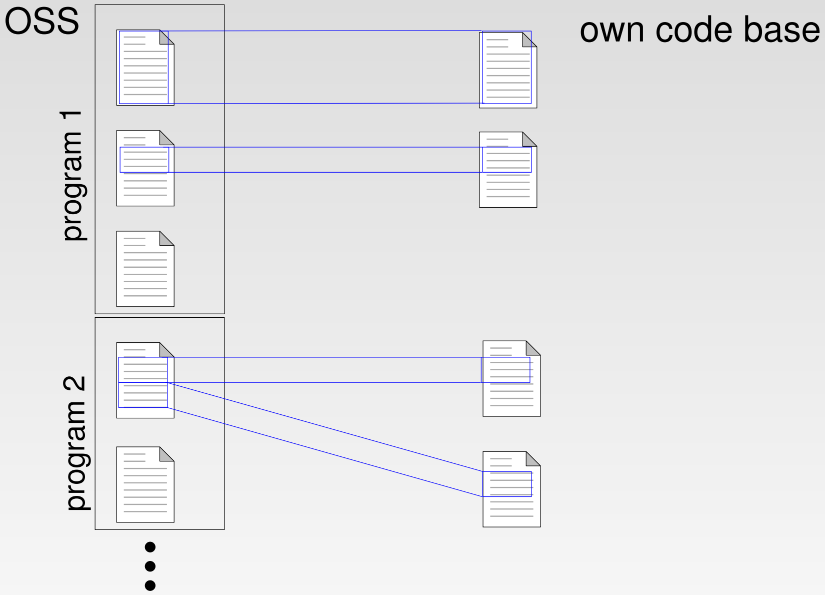
Search for copyright violations



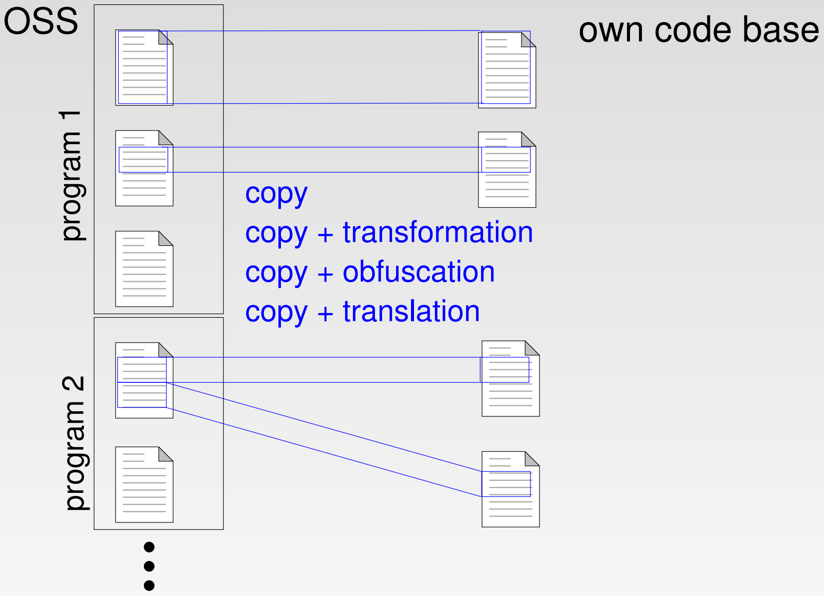
Search for copyright violations



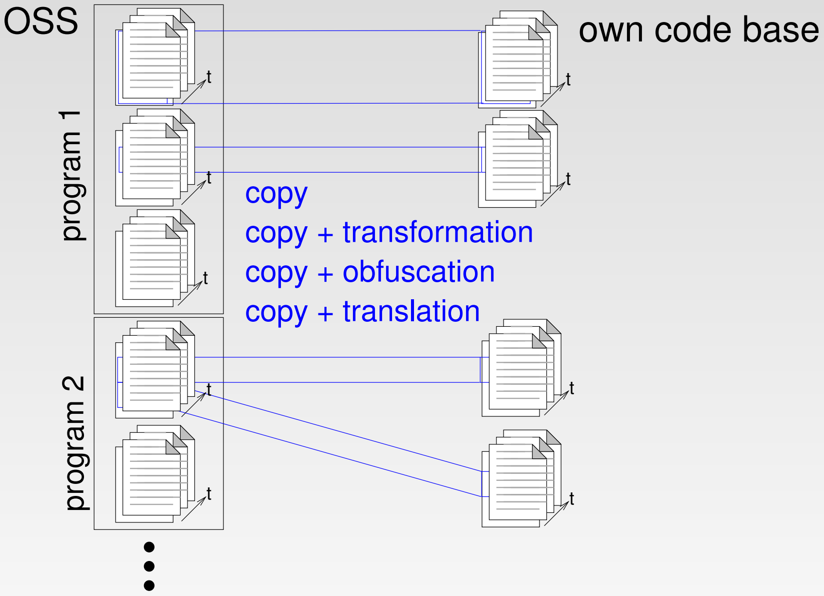
Search for copyright violations



Search for copyright violations



Search for copyright violations



Challenges of copyright violation detection

- Is source code available at all?
- What to compare?
- What granularity (whole files, parts thereof)?
- Degree of tolerance of:
  - transformation
  - obfuscation
  - translation

What to compare?

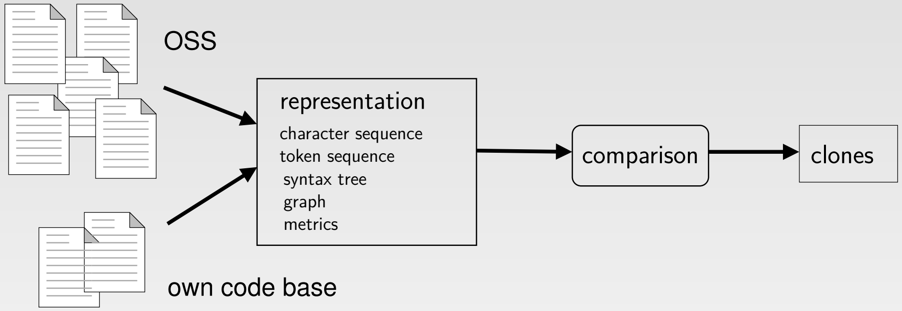
Name	URL
SourceForge	http://sourceforge.net/
freshmeat	http://freshmeat.net/
Debian	http://ftp.debian.org/debian/pool/
Apache Software Foundation	http://apache.org/
CPAN	http://cpan.org/
CRAN	http://cran.r-project.org/
BerliOS Developer	http://developer.berlios.de/
Open Source Scripts	http://opensourcescripts.com/
GNU Savannah	http://savannah.gnu.org/
OpenSymphony	http://www.opensymphony.com/
Koders	http://koders.com/
ObjectWeb	http://objectweb.org/
JBoss	http://jboss.com/
PEAR	http://pear.php.net/
JSAN	http://openjsan.org/
CodePlex	http://www.codeplex.com/
Free Software Directory	http://directory.fsf.org/
Ohloh	http://ohloh.net/
TuxFamily	http://project.tuxfamily.org/
OSOR.eu	http://www.OSOR.eu/
Launchpad	http://www.launchpad.net/
RubyForge	http://rubyforge.org/

— http://en.wikipedia.org/wiki/List_of_free_software_project_directories

Commercial tools

- BitMatch (S.A.F.E.) `http://www.safe-corp.biz/`
  - Comparison of binary files of contained strings
- CodeMatch (S.A.F.E.) `http://www.safe-corp.biz/`
  - Correlation of statements, comments and identifiers in files (Zeidman, 2007, 2008)
- Protex (Black Duck Software), `http://www.blackducksoftware.com/`
  - File comparison via hashing?
- Protecode System 4 (Protecode) `http://www.protecode.com/`
  - File comparison (binary + source) via hashing?
- SIMILE workshop (ESALAB), `http://www.esalab.com/`
  - (Service) Comparison of identifiers, syntax, and semantics (details unknown) in object code

Clone detection



Clone of type 1

Definition

Type 1: Copy without transformation (except reformatting and commenting)

```
int product (int n) {
  int i = 1;
  int result = 1;
  while (i <= n) {
    result = result * i;
    i = i + 1;
  }
  return result;
}
```

```
int product (int n) {
  int i = 1; int result = 1;
  while (i <= n) {
    result=result*i; i=i+1;
  }
  return result;
}
```

Clone of type 2

Definition

Type 2: Copy with parameter substituions (identifiers and literals)

```
int product (int n) {
    int i = 1;
    int result = 1;
    while (i <= n) {
        result = result * i;
        i = i + 1;
    }
    return result;
}
```

```
int prod (int x) {
    int j = 1;
    int r = 1;
    while (j <= x) {
        r = r * j;
        j = j + 1;
    }
    return r;
}
```

product $\rightarrow$ prod, n $\rightarrow$ x, i $\rightarrow$ j, result $\rightarrow$ r

Clone of type 3

Definition

Type 3: Copy with additional/deleted/modified statements/expressions

```
int product (int n) {
  int i = 1;
  int result = 1;
  while (i <= n) {
    result = result * i;
    i = i + 1;
  }
  return result;
}
```

```
int compute (int n, int &sum) {
  int i = 1;
  int result = 1;
  sum = 0;
  while (i <= n) {
    result = result * i;
    sum = sum + i;
    i = i + 1;
  }
  return result;
}
```

Syntactic obfuscation

Definition

Obfuscation: semantic-preserving and syntax-mutating transformations

```
int product (int n) {
  int i = 1;
  int result = 1;
  while (i <= n) {
    result = result * i;
    i = i + 1;
  }
  return result;
}
```

```
int prod (int x) {
  int r = 1;
  for (int j = 0; j < x; j++)
    r = r * (j+1);
  return r;
}
```

Cross-language clones

Definition

Cross-language clones: translation to different programming language

```
int product (int n) {
  int i = 1;
  int result = 1;
  while (i <= n) {
    result = result * i;
    i = i + 1;
  }
  return result;
}
```

```
function product (n : natural)
             return natural
is
  result : natural := 1;
begin
  for i in 1 .. n loop
    result := result * i;
  end loop;
  return result;
end product;
```

Modification tolerance

Comparison of source code with tolerance against transformation/obfuscation/translation

Rep.	Transf./Obfusc.	Translation
Text	—	—
Tokens	token normalization	shared token vocabulary
Syntax	syntactic normalization	unified syntax trees
PDG	reduction to data/control dep.	similar data/control dep.

Modification tolerance

Comparison of source code with tolerance against transformation/obfuscation/translation

Rep.	Transf./Obfusc.	Translation
Text	—	—
Tokens	token normalization	shared token vocabulary
Syntax	syntactic normalization	unified syntax trees
PDG	reduction to data/control dep.	similar data/control dep.

Indirect comparison: comparing the compilation (ByteCode, Microsoft IL, object code)

Summary

- Copyright violation is a relevant problem with bad consequences
- Clone detection techniques may help, they vary in . . .
    - scalability
    - granularity
    - modification tolerance

Summary

- Copyright violation is a relevant problem with bad consequences
- Clone detection techniques may help, they vary in . . .
  - scalability
  - granularity
  - modification tolerance
- Clone detection techniques must be adjusted
- Tools can find only evidence, no proof
- Tools are at least a *best effort*

[Baker 1995] BAKER, Brenda S.: On Finding Duplication and Near-Duplication in Large Software Systems. In: WILLS, L. (Hrsg.) ; NEWCOMB, P. (Hrsg.) ; CHIKOFSKY, E. (Hrsg.): Working Conference on Reverse Engineering. Los Alamitos, California : IEEE Computer Society Press, Juli 1995, S. 86–95

[Baxter u. a. 1998] BAXTER, Ira D. ; YAHIN, Andrew ; MOURA, Leonardo ; SANT'ANNA, Marcelo ; BIER, Lorraine: Clone Detection Using Abstract Syntax Trees. In: KOSHGOFTAAR, T. M. (Hrsg.) ; BENNETT, K. (Hrsg.): International Conference on Software Maintenance, IEEE Computer Society Press, 1998, S. 368–378. – ISBN 0-7803-5255-6, 0-8186-8779-7, 0-8186-8795-9

[Ducasse u. a. 1999] DUCASSE, Stéphane ; RIEGER, Matthias ; DEMEYER, Serge: A Language Independent Approach for Detecting Duplicated Code. In: Proceedings of the International Conference on Software Maintenance (ICSM99), 1999, S. 109–118

[Johnson 1993] JOHNSON, J. H.: Identifying redundancy in source code using fingerprints. In: Conference of the Centre for Advanced Studies on Collaborative research, IBM Press, 1993, S. 171–183

[Johnson 1994] JOHNSON, J. H.: Substring matching for clone detection and change tracking. In: International Conference on Software Maintenance, IEEE Computer Society Press, 1994, S. 120–126

[Komondoor und Horwitz 2001] KOMONDOOR, R. ; HORWITZ, S.: Using slicing to identify duplication in source code. In: Proc. Int. Symposium on Static Analysis, Juli 2001, S. 40–56

[Koschke u. a. 2006] KOSCHKE, Rainer ; FALKE, Raimar ; FRENZEL, Pierre: Clone Detection Using Abstract Syntax Suffix Trees. In: Working Conference on Reverse Engineering, IEEE Computer Society Press, 2006, S. 253–262

[Krinke 2001] KRINKE, Jens: Identifying Similar Code with Program Dependence Graphs. In: Working Conference on Reverse Engineering, 2001, S. 301–309

[Mayrand u. a. 1996] MAYRAND, Jean ; LEBLANC, Claude ; MERLO, Ettore M.: Experiment on the Automatic Detection of Function Clones in a Software System using Metrics. In: Proceedings of the International Conference on Software Maintenance. Washington : IEEE Computer Society Press, November 4–8 1996, S. 244–254. – ISBN 0-8186-7678-7

[Mertzel 2008] MERTZEL, N. J.: Copying 0.03 percent of software code base not "de minimis". In: Journal of Intellectual Property Law & Practice 9 (2008), Nr. 3, S. 547Â–548

[Rieger 2005] RIEGER, Matthias: Effective Clone Detection Without Language Barriers, University of Bern, Switzerland, Dissertation, 2005

[Sojer und Henkel 2010] SOJER, M. ; HENKEL, J.: License Risks from Ad-Hoc Reuse of Code from the Internet: An Empirical Investigation. Preprint. April 2010. – URL http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1594641

[Zeidman 2008] ZEIDMAN, R.: Multidimensional Correlation of Software Source Code. In: Third International Workshop on Systematic Approaches to Digital Forensic Engineering, 2008. SADFE '08, Mai 2008, S. 144 –156

[Zeidman 2007] ZEIDMAN, Robert: Iterative Filtering of Retrieved Information to Increase Relevance. In: Journal of Systemics, Cybernetics and Informatics 5 (2007), Nr. 6, S. 91–96