修士学位論文

題目

An Empirical Study of Out-dated Third-party Code in Open Source Software

指導教員
井上 克郎 教授

報告者
Pei Xia

平成 25 年 2 月 5 日

大阪大学 大学院情報科学研究科
コンピュータサイエンス専攻 ソフトウェア工学講座

平成 24 年度 修士学位論文

An Empirical Study of Out-dated Third-party Code in Open Source Software

Pei Xia

内容梗概

Using existing source code to build new software systems becomes common. High-quality open source software(OSS) such as zlib, libpng, libcurl etc. are wildly reused as third-party code. However, these existing code are keeping on updating during their life circle. Different versions of third-party code including those with security vulnerabilities are reused by other software and spreading in the OSS all over the world.

This paper presents an empirical study on the code reuse of third-party code. Given a target source code, with the help of code search tool OpenCCFinder, we found and selected a number of open source projects that reused the target source code. Using file clone detection techniques and repository mining techniques we identified the version number of these reused code. Then we analyzed and discovered the defect information of the out-dated third-party code as well as the management information of the open source projects.

The result shows that a large number of open source projects are reusing out-dated third-party code. Moreover, the study observed that a large number of the reused third-party code are not well managed.

主な用語

third-party code reuse
defect detection
open source code search
file clone detection

Contents

1 Introduction

Nowadays, using existing software to build new software systems becomes common. More and more source code from open source software(OSS) can be found on the Internet. Even software in the industry increasingly reuse open source systems due to their reliability and cost benefits.[9]

Integrating third-party code is an important approach in code reuse. A third party code is a reusable software component developed to be either freely distributed or sold by an entity other than the original vendor of the development platform[11]. Many open source projects are considered to be stable and efficient, such as encryption software (OpenSSL), compression software (zlib), databases(MySQL), or graphical tookits(GTK), etc. These code are wildly reused as third-party code by thousands of developers all over the world.

However, while enjoying the benefits, developers also have to concern about the risks brought by reusing third-party code. If the reused code contain critical defects, it will bring damage to the software. For example, In Sept.2010, a Twitter user has demonstrated a cross-site scripting (XSS) vulnerability on the microblogging platform that could allow an attacker to take over users' accounts or spread malware. The third-party javascript code they reused enable a function that could trigger activity such as pop-up box appearing or manipulated with the flaw to redirect a user to a malicious Web Site, which lead to about half a million malicious posts on Twitter.[17]

Third-part code risk management is necessary in software development. Well known open source libraries such as zlib, libpng, libcurl etc. are usually actively maintained libraries. Some of these libraries contain security vulnerabilities in one version that are fixed in later versions. Considering such a scenario that a developer reused a certain version of libpng library and just copy the code to his project. After he implemented the features that he wants, he left the libpng code alone and did not touch it any more. But several days later a security vulnerability announced on libpng official home page and a new version of libpng is released. If he did not notice and follow this update, his software may be affected by the vulnerability. Thus, keeping the code library up-to-date is an important way to avoid some of the risks, while reusing out-dated third-party code would make the software have more chance to be breached by a hacker.

"Out-dated third-party code" in this study represents those code of older versions containing known defects such as software vulnerabilities that should be fixed by upgrading them to a newer version.

As far as we know, currently there is few research focus on the out-dated third-part code reuse and management behavior. Our work is to do an empirical study in this area and collect quantitative data to answer these research questions:

- What is the proportion of out-dated third-party code reused in the open source software?

- What are the potential defects caused by such reuse?
- How do developers manage those out-dated third-party code?

Answering these questions would be helpful in understanding the open source software, evaluating the quality of the softwares who reused third-party code, predicting some of the potential defects in open source software, also would make developers be aware of the importance of third-part code management.

Based on the code clone detection, repository mining techniques and open source code search engines, we proposed a study approach on detecting out-dated third-party code reuse for certain libraries in open source software.

In section 2, we described the detailed study design. Section 3 shows our case studies of 3 wildly reused open source libraries. Section 4 conclude our discussions with some future works. Section 5 shows the related works.

2 Study Approach

In designing the study approach to answer the research questions raised in Section 1, as shown in Figure 1, we firstly selected several open source third-party code from Internet to study with. In the second step, we investigated the potential defects contained in old versions of the subject third-party code. Next, we use OpenCCFinder[22] to find tens of open source projects that have reused the subject third-party code, which we call vendee projects below. *OpenCCFinder* is a tool developed by us in 2011 which we will introduce in subsection 2.3.1. In the fourth step, we get the source files of all version from the repository of subject third-party code, also get the source code of the latest version of vendee projects. Using content hash based file clone detection techniques in Section 2.4.2, we precisely identified the version number of subject third-party code reused in each vendee projects. At last, we manually invastigated the project management information of each vendee projects to get valuable information.

In detail, the study approach can be divided into five steps as follows:

2.1 Choosing Subject Third-party Code

At the beginning, the subject third-party code to study should be decided. A great number of open source projects that reused as third-party libraries can be found on the Internet. For our study, the subject third-party code should be chosen as follows:

- Well-known and widely reused. For example, the zlib library has been reused by famous software such as linux kernel, Mac OS X, XEmacs etc. Reseach results from such kind of third-party code could be more convincing.

- Small-sized. Because in our study we used content hash based file clone detection techniques to identify version number of reused third-party code, we have to download each version of these code to local machine to do the analysis. For the time and space consideration, the size of the subject third-party code should not be too large. Projects with less than 500 files and smaller than 5 megabytes are preferable.

- Actively maintained. Because such projects would have various distributions spreading in the open source software all over the world and reused by different vendee projects. And current situation of third-party code reuse in open source software can be better reflected by these projects rather than stable or old projects.

- Stored and managed by repositories such as git or svn. It is easy for me to get source files of each distribution of the projects and analyze various data of them by using libraries such as EGit or SVNKit.

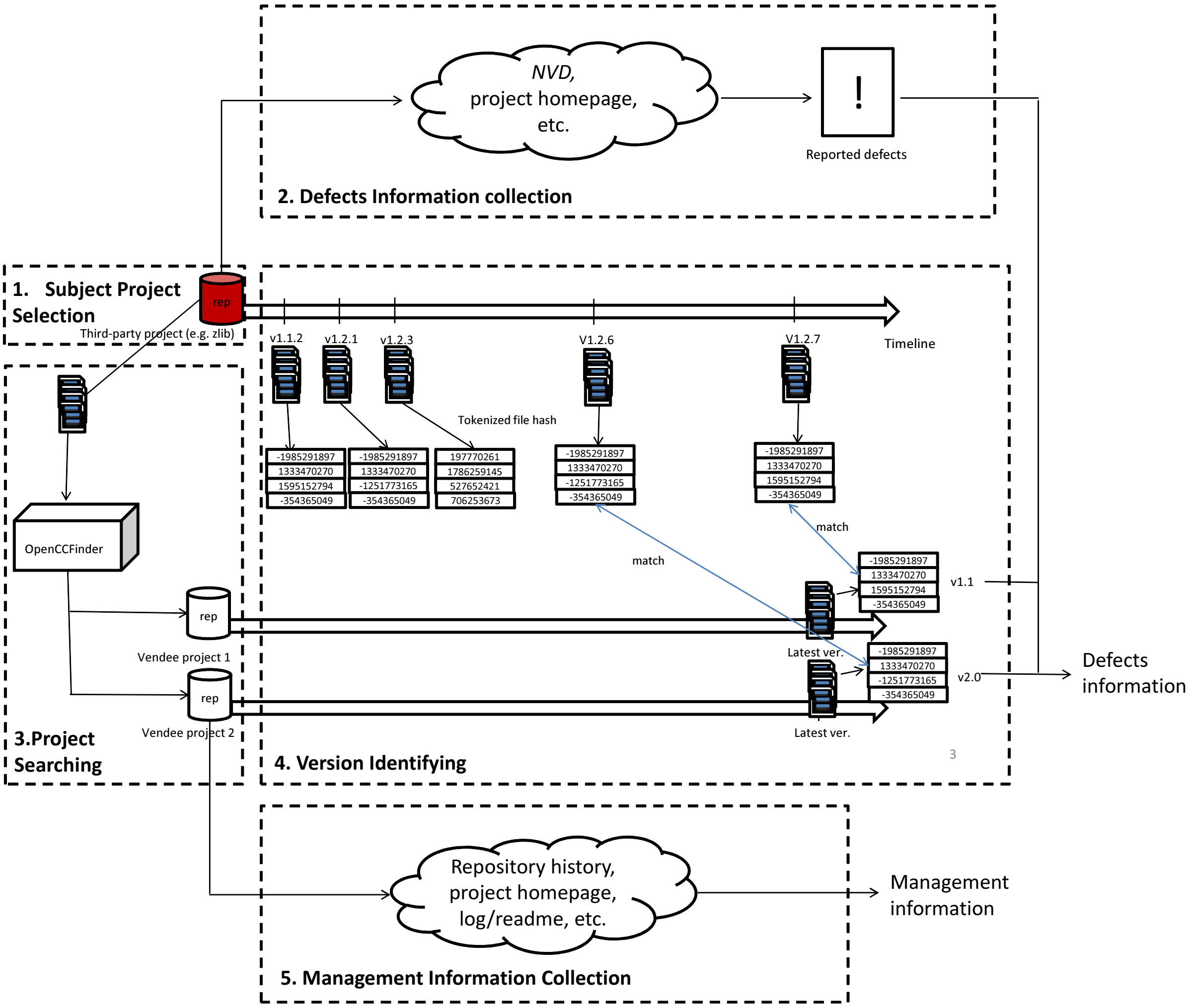


Figure 1: Study Approach Overview

- Reused in the form of source code instead of binary libraries. Because in our study we have to access to the source code for clone detection, if the third-party code is reused in the form of binary libraries, it is difficult for me to study them. Thus, source code in language C is a preferred choice.

Currently we studied three open source libraries, *zlib*, *libpng*, and *libcurl*. The detailed data of them would be represented in Section 4.

2.2 Manual Collection of Defects in Out-dated Third-party Code

In this section, we would like to introduce how we search for the defects information of third-party code of this version. This is to answer the second research question: Are there potential defects caused by reusing out-dated third-party code?

Since we have little background in defect prediction research area, what we have done here is only collecting the existing bug information found by other people. Currently it is done by manually inspection from those three sources: software vulnerability database and project homepage announcement.

2.2.1 Software Vulnerability Database

A software vulnerability database is a platform aimed at collecting, maintaining, and disseminating information about discovered vulnerabilities targeting real computer systems. In this study, we are looking into the $NationalVulnerabilityDatabase$[5] ($NVD$), which is the U.S. government repository of standards based vulnerability management data. It contains famous resources such as $CommonVulnerabilitiesandExposures(CVE)$ and $CERTVulnerabilityNotesDatabase$. In our view, the data searched from NVD are reliable.

Searching with keywords such as project name or filename, NVD returns a list of vulnerabilities information including vulnerability id, summary, published date, CVSS Severity score. Table 1 shows several example results by using the keyword of ”libpng”.

In the summary column, the version name and vulnerability detail are described. As we can see, the vulnerabilities are almost critical bugs which could lead to application crash or execution of arbitrary code. If other projects are reusing old version of libpng library containing such bugs, it would be quite dangerous.

2.2.2 Announcement from Project's Homepage

Another resource of defects information is project homepage announcement. Usually, when some critical defects are found in some versions of an open source project, there would be an announcement on the project's homepage.

For example, on zlib's project homepage,

Table 1: Search results Example of *NVD*

Vulnerability ID	Summary	Published Date	CVSS Severity
CVE-2011-3464	Off-by-one error in the png_formatted_warning function in pngerror.c in libpng 1.5.4 through 1.5.7 might allow remote attackers to cause a denial of service (application crash) and possibly execute arbitrary code via unspecified vectors, which trigger a stack-based buffer overflow.	07/22/2012	7.5 (HIGH)
CVE-2011-3048	The png_set_text_2 function in pngset.c in libpng 1.0.x before 1.0.59, 1.2.x before 1.2.49, 1.4.x before 1.4.11, and 1.5.x before 1.5.10 allows remote attackers to cause a denial of service (crash) or execute arbitrary code via a crafted text chunk in a PNG image file, which triggers a memory allocation failure that is not properly handled, leading to a heap-based buffer overflow.	05/29/2012	6.8 (MEDIUM)
CVE-2011-3045	Integer signedness error in the png_inflate function in pngrutil.c in libpng before 1.4.10beta01, as used in Google Chrome before 17.0.963.83 and other products, allows remote attackers to cause a denial of service (application crash) or possibly execute arbitrary code via a crafted PNG file, a different vulnerability than CVE-2011-3026.	03/22/2012	6.8 (MEDIUM)
CVE-2011-2690	Buffer overflow in libpng 1.0.x before 1.0.55, 1.2.x before 1.2.45, 1.4.x before 1.4.8, and 1.5.x before 1.5.4 makes a function call using a NULL pointer argument instead of an empty-string argument, which allows remote attackers to cause a denial of service (application crash) via a crafted PNG image.	07/17/2011	6.8 (MEDIUM)

Version 1.2.5 fixes bugs in gzseek() and gzeof() that were present in version 1.2.4 (March 2010). All users are encouraged to upgrade immediately.

Version 1.2.3 (July 2005) eliminates potential security vulnerabilities in zlib 1.2.1 and 1.2.2, so all users of those versions should upgrade immediately.

on libpng's homepage,

Vulnerability Warnings: libpng 1.5.4 through 1.5.7 contain a one-byte (stack) buffer-overrun bug in png_formatted_warning(), which could lead to crashes (denial of service) or, conceivably, execution of hostile code. This vulnerability has been assigned ID CVE-2011-3464 and is fixed in version 1.5.8, released 1 February 2012.

Vulnerability Warnings: libpng 1.5.4 (only) introduced a divide-by-zero bug in png_handle_cHRM(), which could lead to crashes (denial of service) in applications that support color correction. This vulnerability has been assigned ID CVE-2011-3328 (CERT VU#477046) and is fixed in version 1.5.5, released 22 September 2011.

In our view, these announcements are quite important. It is reasonable that suggestions from project owners should be taken seriously by whom reusing these code.

2.3 Searching for Projects Reusing Subject Third-party Code

In this section, we will introduce how we search for the vendee projects from open source software in the world.

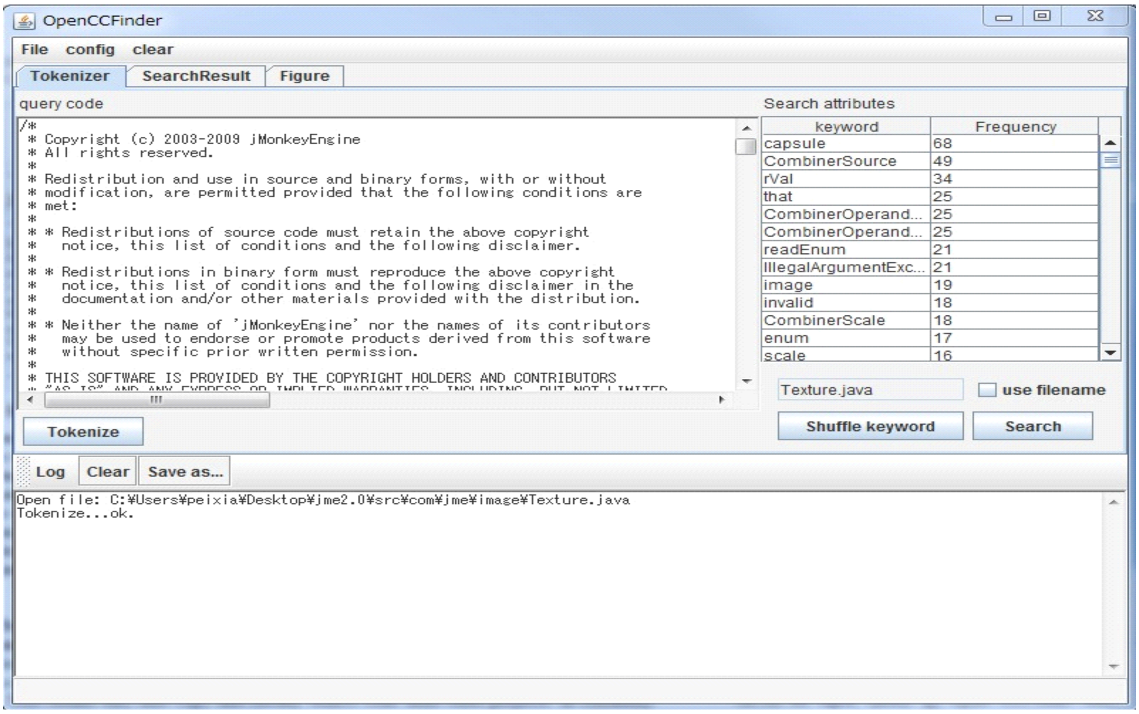
Nowadays, open source software hosting facilities are becoming popular. Millions open source projects are hosted on the Internet. Google Code, GitHub are some of the most popular open source hosting sites. Project hosting on Google Code provides a free collaborative development enviroment for open source projects. According to Google Code Official Blog, it hosts more than 250,000 open source projects. GitHub is a web-based hosting service for software development projects that use the Git revision control system. It hosts millions of projects. There are also similar sites such as Sourceforge, CodePlex, Eclipse Labs, BitBucket, RubyForge, Jave.net, etc. providing open source software hosting service. These sites are playing very important roles in open source communities and covering a large number of open source projects in the world.

From such open source software hosting facilities, we found a list of projects that reused the subject third-party code. In this step, we used a similar code searching system *OpenCCFinder* (Open Code Clone Finder) helping me to do this. *OpenCCFinder* is a system to explore similar code fragments from open source repositories. It developed by me in 2011. This system takes a query code fragment as input, shown in Figure 2, and returns the code fragments containing the code clones with the query, shown in Figure 3. We will introduce this system in detail in following subsections.

2.3.1 The Architecture of OpenCCFinder

Figure 4 shows the architecture of *OpenCCFinder*. It takes an input query Q and returns an output results set R. Input query Q is composed of code fragment q_c and code attribute q_a. q_c may be a complete source file or a part of a source code file, which is in question. q_a is a set of associated information characterizing q_c, such as the file name. q_a is optional and could be added to improve the quality of the output results. Given an input Query Q, *OpenCCFinder* extracts useful information from it and generates queries for external code search engine (e.g. Google code search, SPARS/R etc.), and then analyzes the returned candidate files from external search engines, at last form a final result as output R.

Output result R is composed of results $r1$,$r2$ ... rn. Each result ri is composed of a code file ri_c and its code attribute ri_a. ri_c is a code file which is returned by external code search engines, and ri_a a set of associated information about ri_c, including repository



```
/*
 * Copyright (c) 2003-2009 jMonkeyEngine
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are
 * met:
 *
 * * Redistributions of source code must retain the above copyright
 *   notice, this list of conditions and the following disclaimer.
 *
 * * Redistributions in binary form must reproduce the above copyright
 *   notice, this list of conditions and the following disclaimer in the
 *   documentation and/or other materials provided with the distribution.
 *
 * * Neither the name of 'jMonkeyEngine' nor the names of its contributors
 *   may be used to endorse or promote products derived from this software
 *   without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
```

keyword	Frequency
capsule	68
CombinerSource	49
rVal	34
that	25
CombinerOperand...	25
CombinerOperand...	25
readEnum	21
IllegalArgumentExc...	21
image	19
invalid	18
CombinerScale	18
enum	17
scale	16

```
Open file: C:¥Users¥peixia¥Desktop¥jme2.0¥src¥com¥jme¥image¥Texture.java
Tokenize...ok.
```

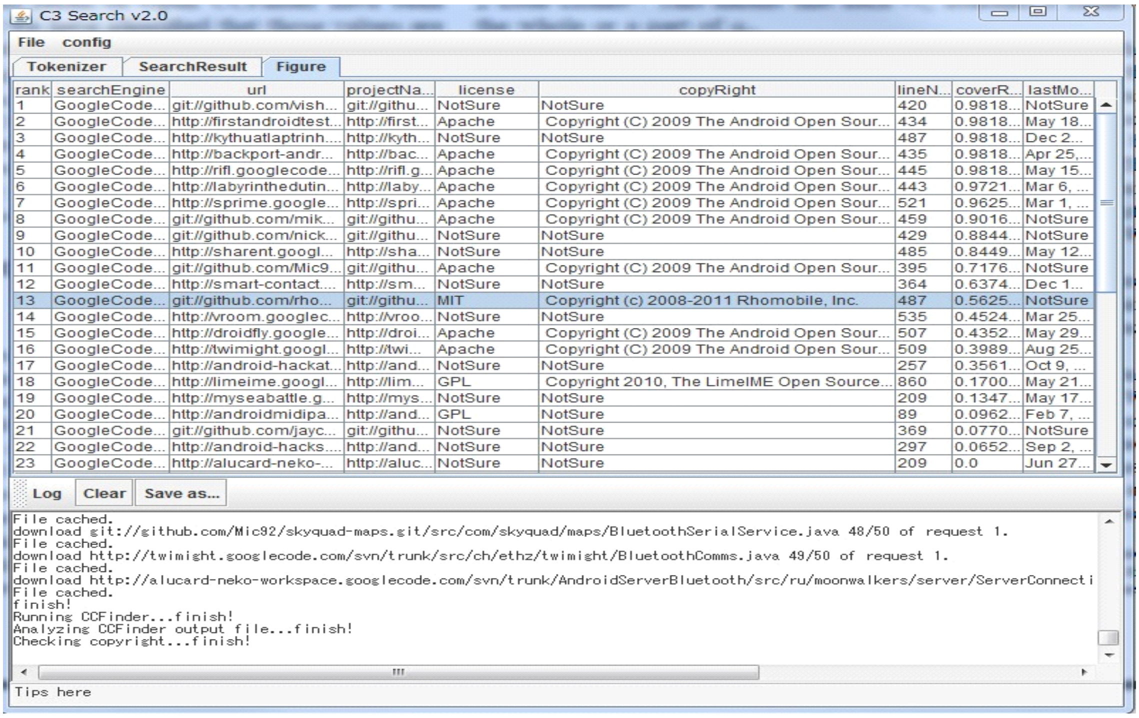
Figure 2: Input of *OpenCCFinder*.

URL, file path, LOC, license, copyright, last modified time, clone cover ratio and clone detail, as shown in Table 3.

Table 2: Subjects Third-party Code Information

Repository url	where the repository of ri can be accessed on the Internet
File Path	the file path of ri in its project
LOC	line of code of ri
License	the software license of the source file
Copyright	the copyright of the software
Last modified time	the latest committed time of ri in its repository
Cover ratio	the code percentage of the queried code q_c that reused by ri

For the external code search engines, we use Google code search[**?**] and search[code][**?**] and code search feature of github[**?**, githubcs]n our tool implementation. The Google code search is a famous code search engine. But currently it only provides code search service of googlecode repositories to the user; The search[code] is a code specific search engine. API documentation, code snippets and open source repositories are indexed and searchable. Currently more than 6 billion code of googlecode. sourceforge, fedora, CodePlex, github, and Bitbucket are collected by search[code]; The code search feature of github can be used to search for source code in github. *OpenCCFinder* merges their results together in order



rank	searchEngine	url	projectNa...	license	copyRight	lineN...	coverR...	lastMo...
1	GoogleCode...	git://github.com/vish...	git://githu...	NotSure	NotSure	420	0.9818...	NotSure
2	GoogleCode...	http://firstandroidtest...	http://first...	Apache	Copyright (C) 2009 The Android Open Sour...	434	0.9818...	May 18...
3	GoogleCode...	http://kythuatlaptrinh....	http://kyth...	NotSure	NotSure	487	0.9818...	Dec 2...
4	GoogleCode...	http://backport-andr...	http://bac...	Apache	Copyright (C) 2009 The Android Open Sour...	435	0.9818...	Apr 25,...
5	GoogleCode...	http://rifl.googlecode...	http://rifl.g...	Apache	Copyright (C) 2009 The Android Open Sour...	445	0.9818...	May 15...
6	GoogleCode...	http://labyrinthedutin...	http://laby...	Apache	Copyright (C) 2009 The Android Open Sour...	443	0.9721...	Mar 6, ...
7	GoogleCode...	http://sprime.google...	http://spri...	Apache	Copyright (C) 2009 The Android Open Sour...	521	0.9625...	Mar 1, ...
8	GoogleCode...	git://github.com/mik...	git://githu...	Apache	Copyright (C) 2009 The Android Open Sour...	459	0.9016...	NotSure
9	GoogleCode...	git://github.com/nick...	git://githu...	NotSure	NotSure	429	0.8844...	NotSure
10	GoogleCode...	http://sharent.googl...	http://sha...	NotSure	NotSure	485	0.8449...	May 12...
11	GoogleCode...	git://github.com/Mic9...	git://githu...	Apache	Copyright (C) 2009 The Android Open Sour...	395	0.7176...	NotSure
12	GoogleCode...	http://smart-contact....	http://sm...	NotSure	NotSure	364	0.6374...	Dec 1...
13	GoogleCode...	git://github.com/rho...	git://githu...	MIT	Copyright (c) 2008-2011 Rhomobile, Inc.	487	0.5625...	NotSure
14	GoogleCode...	http://vroom.googlec...	http://vroo...	NotSure	NotSure	535	0.4524...	Mar 25...
15	GoogleCode...	http://droidfly.google...	http://droi...	Apache	Copyright (C) 2009 The Android Open Sour...	507	0.4352...	May 29...
16	GoogleCode...	http://twimight.googl...	http://twi...	Apache	Copyright (C) 2009 The Android Open Sour...	509	0.3989...	Aug 25...
17	GoogleCode...	http://android-hackat...	http://and...	NotSure	NotSure	257	0.3561...	Oct 9, ...
18	GoogleCode...	http://limeime.googl...	http://lim...	GPL	Copyright 2010, The LimeIME Open Source...	860	0.1700...	May 21...
19	GoogleCode...	http://myseabattle.g...	http://mys...	NotSure	NotSure	209	0.1347...	May 17...
20	GoogleCode...	http://androidmidipa...	http://and...	GPL	NotSure	89	0.0962...	Feb 7, ...
21	GoogleCode...	git://github.com/jayc...	git://githu...	NotSure	NotSure	369	0.0770...	NotSure
22	GoogleCode...	http://android-hacks....	http://and...	NotSure	NotSure	297	0.0652...	Sep 2, ...
23	GoogleCode...	http://alucard-neko-...	http://aluc...	NotSure	NotSure	209	0.0	Jun 27...

```
File cached.
download git://github.com/Mic92/skyquad-maps.git/src/com/skyquad/maps/BluetoothSerialService.java 48/50 of request 1.
File cached.
download http://twimight.googlecode.com/svn/trunk/src/ch/ethz/twimight/BluetoothComms.java 49/50 of request 1.
File cached.
download http://alucard-neko-workspace.googlecode.com/svn/trunk/AndroidServerBluetooth/src/ru/moonwalkers/server/ServerConnecti
File cached.
finish!
Running CCFinder...finish!
Analyzing CCFinder output file...finish!
Checking copyright...finish!
```

Figure 3: Output of *OpenCCFinder*.

to cover more open source repositories.

2.3.2 The Search process of OpenCCFinder

Search process of *OpenCCFinder* can be devided into 6 steps, as shown in Figure 5.

(a) Word Extraction. At the beginning, code fragment q_c in input query Q is tokenized, the words from source code and comments are separated. Camel Case (e.g. helloWorld) or Snake Case (e.g. hello_world) words will not be decomposed into multiple words. User can choose to extract words from source code or from comments, or from both.

(b) Keyword Ranking. Next, the keywords used for query generation are selected from the extracted words. In this step, first *OpenCCFinder* filters out the words that considered being featureless. For example, the reserved words of each source code language, the words in very short length, and the words included in customized filter are filtered out. After the filtering, a simple words importance ranking strategy is applied on the remaining words. Currently there are two strategy implemented in the tool for ranking the words: frequency strategy and random strategy. Frequency

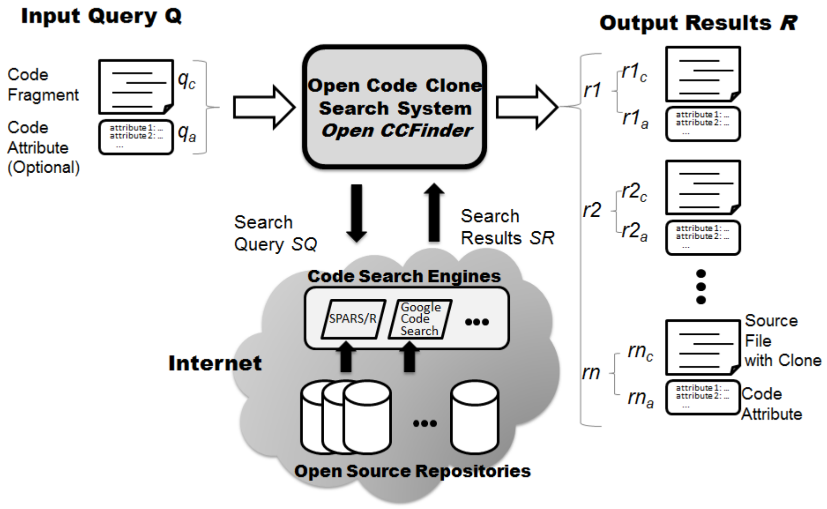


Figure 4: Architecture of *OpenCCFinder*.

strategy is to rank the keywords by the times they appear in the source codes or comment, while random strategy is just to rank the words randomly.

(c) **Searching for Candidates Files.** Using the ranked keywords, a search query SQ for the code search engines is created. As the search engines, here we choose Google Code Search, search[code] and code search feature in github. Each of the search engines accept keywords sequence as their query input, so we use the combination of most frequently used words as SQ. If user wants, the additional input attribute file name also can be given to the search engines.

Then we generate several queries for each search engine to get appropriate candidate files. For each query, the returned results set from search engines should not be very large, for fear of including too many irrelevant results. When the returned results set are too large, we will add one more keyword from the ranked keywords list to the query to narrow the results set. At last we merge the returned results of several queries as the analysis candidate files. The detail process is shown in Algorithm 1 .

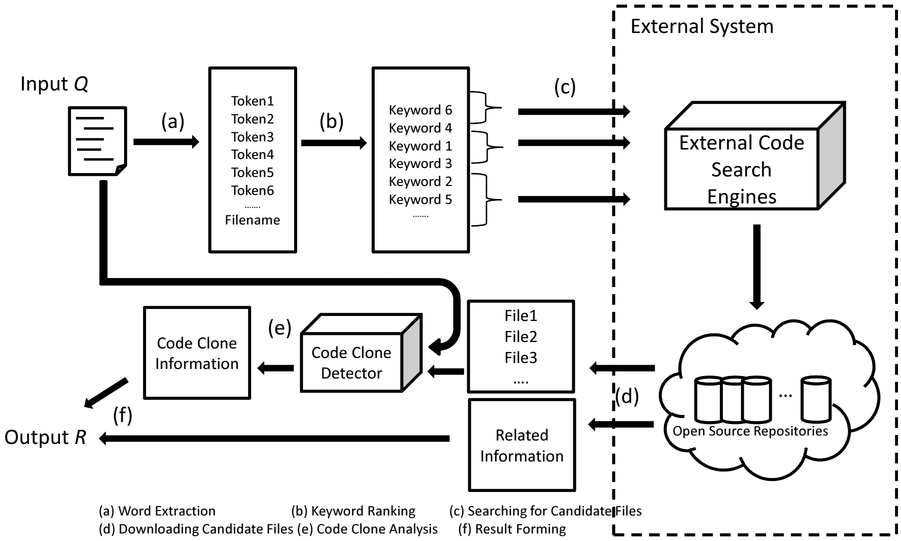


Figure 5: Searching Process of *OpenCCFinder*.

Algorithm 1: The pseudo code of searching for candidate files with ranked keywords

```
Data: Ranked Keyword List
Result: Appropiate Cadidate Files List
begin
    CandidateFiles = $\phi$;
    CurrentKeywords = $\phi$;
    while CadidateFiles is not approriate (Judged by user) do
        PartialCandidateFiles = $\phi$;
        while PartialCandidateFiles is empty or too large size do
            CurrentKeywords = CurrentKeywords $\bigcup$
            keywordsList.nextTopKwyword ;
            PartialCandidateFiles = results searched with CurrentKeywords;
        end
        CurrentKeywords = $\phi$;
        CandidateFiles = CandidateFiles $\bigcup$ PartialCandidateFiles;
    end
    return CandidateFiles
end
```

(d) Downloading Candidate Files. All the candidate files in step (c) are downloaded from Internet. While downloading the file, the tool is also crawling the web to extract useful information for the code attributes such as file path, repository URL, LOC, License, Copyrights, and last modified time if available.

(e) Code Clone Analysis. The code clones between the input query code fragment q_c and each source code ri obtained at Step 4 are computed. We have used a code clone detection tool CCFinder[13], with its parameter setting for the minimum token length 15. 15 tokens is the common configuration for identify code clone in related research area. Then we calculate the cloned code cover ratio of q_c for each Candidates. Cover ratio represents the code percentage of the queried code q_c that reused by ri.

(f) Result Forming. All the candidate files and their code attributes are combined and packed as the output result R of this system, sorted by their cover ratio of q_c.

2.3.3 The Use of *OpenCCFinder* in This Study

In this study, we use *OpenCCFinder* to help me to discover repository url of projects that have reused certain third party code.

For the configuration, the preferred number of vendee projects returned by *OpenCCFinder* should be less than 100. Because there are several manual tasks in the following steps, it would be difficult for me if *OpenCCFinder* return too many results. Our purpose is to estimate the proportion of out-dated third-party code reuse, in our consideration, the scale of experimental samples around 50 could be acceptable.

At first, we selected several source files from zlib, libpng and libcurl. These selected files should be relatively unique, in other words, the identical or similar files of the selected files would better only appears in the same project. For example, in zlib, the files of *gzlib.c*, *zutil.c* is considered unique, while in libcurl, the file of *base*64*.c* is considered not unique.

Secondly, we input the selected files into *OpenCCFinder* and collect the repository url in the results page. *OpenCCFinder* would return a list of similar files of the input files along with the clone cover ratio. We filtered out those results in different file names with a cover ratio smaller than 1 percent, which are probably irrelevant files contain a few common code occasionally. If the results set are too small, we will input next file into OpenCCFinder, and merge the results together.

Thirdly, we filtered out projects that considered to be not appropriate. There are a number of experimental personal projects also hosted on open source project hosting facilities, which are not well managed or abandoned. In our consideration, such projects are not good for our study. So we need to filtered these projects out. The projects remained for our study should be in such conditions:

- The projects should be published formally and managed by a stable organization. They should have their own homepage which contains basic information of projects and mailing list of developers. The project owners should be enterprises or scientific facilities or laboratories, but not students or a single person with unclear background.

- The projects should be actively maintained. At least within the latest 6 months, repository of the projects should have been updated with new source code commits.

Finally, we check out the latest version of these projects and find out the file directories of the return files, which would probably the directories where the third-party code stored. We will explain how we confirm if these projects are reusing the subject third-party code later.

2.4 Identifying Version Number of Reused Third-party Code

In this section, we will introduce how to confirm the third-party code are reused and how to identify the reused version number. And how we can answer the first research question: What is the proportion of out-dated third-party code reused in the open source software?

Through the process described in previous section, we checked out a list of projects from repositories of open source hosting facilities and consider that they are reusing the subject third-party code. The idea of identify version number is comparing the content of each file of the project returned by *OpenCCFinder* with those of third-party libraries. If each file of a certain version (e.g. v1.0) of a third-party code are exactly matched with those in a project returned by OpenCCFinder, it is probably that this project is reusing v1.0 of the third-party code.

File Clone Detection technique can be used to check if two source files are identical.

2.4.1 Introduction of Code Clone

According to Roy&Cordy's Paper[7], A code fragment that has identical or similar code fragment(s) to it in the source code is defined as code clone. Moreover, files of a project are simply copied into another project without any (or just slight) modification are defined as file clones.

A copied fragment can be used with or without minor modifications in a system by the developer. Based on the textual and functional similarity, types of code clones are distinguished as follows:

Type I Identical code fragments except for variations in whitespace (may be also variations in layout) and comments. a copied code fragment is the same as the original. However, there might be some variations in whitespace, comments or layouts. Type I is widely know as Exact clones.

Type II Structurally/syntactically identical fragments except for variations in identifiers, literals, types, layout and comments.

Type III Copied fragments with further modifications. Statements can be changed, added or removed in addition to variations in identifiers, literals, types, layout and comments.

Type IV Two or more code fragments that perform the same computation but implemented through different syntactic variants. They have similar pre and post conditions. Such clones are called semantic clones.

In this study, our purpose is detect file clones between source files from third-party projects and vendee projects collected by OpenCCFinder, and then apply defect prediction to vendee projects. So we only try to find out the type I and type II clones. Although further modifications (type III and type IV) are possible to be done to reused third-party code, we do not identify the version number of reused third-party projects of these code because it is difficult to apply defect prediction by known defects of the third-party to them.

Those are possible and acceptable modifications to reused third-party code in our study:

- Comments are added or removed. It is not strange for developers to add or remove comments when they reuse some source code. Also certain configuration of repositories would lead to automatically adding comments of version and authority information when the file is committed.

- Layouts such as blank lines, line breakers or space characters in source files are changed. Because automatically code formatting tool such as pretty printer are wildly used by developers. The same contents with different formats also leading to different hash values.

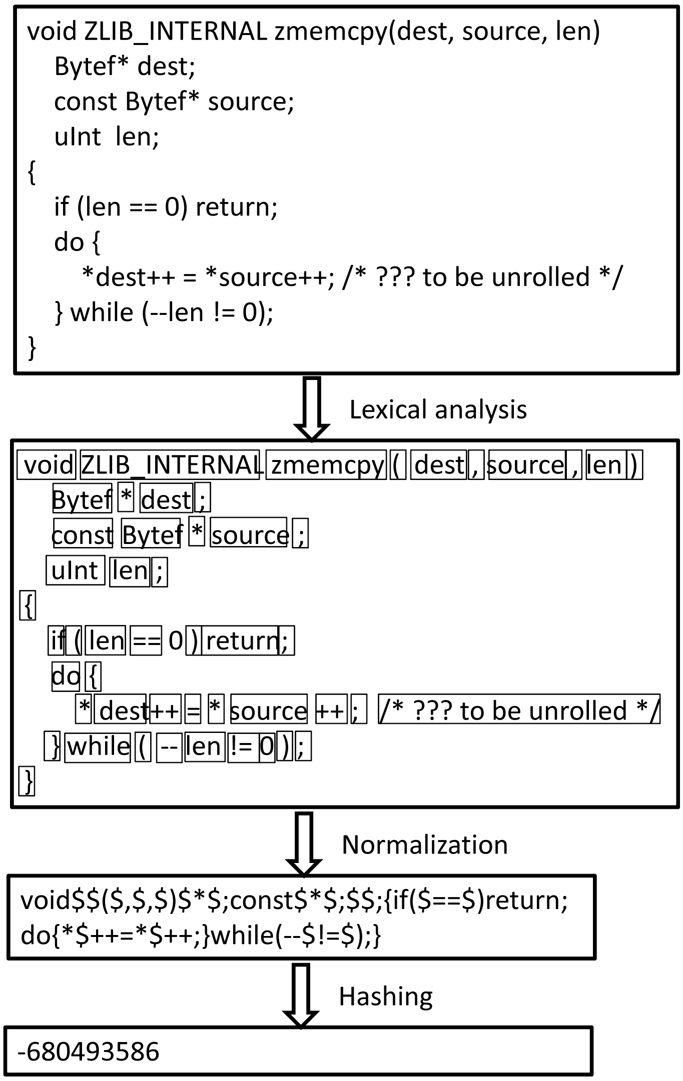
- Rename refactoring is applied. For example, the name of a variable, a structure field, a function or a user-defined type is changed. Rename is the most known and used refactoring which can be applied automatically with many support tools.

These file clones can be detected using tools such as CCFinder. But even with a fast and distributed environment, code clone detection is still a very resource intensive process requiring precise string matching.

Hash based clone detection techniques are considered to be an more efficient way to detect file clones. FCFinder[21] is a tool developed by previous student in our laboratory, but this tool can only detect type I code clones. We improved the existing algorithm of FCFinder and developed a new tool for this study. We will describe the algorithm in detail in next subsection.

2.4.2 Hash Based File Clone Detection Algorithm

The File Clone Detection Algorithm we used in this study is hash based detection. Given two source files, we firstly calculate tokenized hash values for each files, and then compare the two hash values. If the hash value is same, we mark the two files as identical.



```
void ZLIB_INTERNAL zmemcpy(dest, source, len)
    Bytef* dest;
    const Bytef* source;
    uInt  len;
{
    if (len == 0) return;
    do {
        *dest++ = *source++; /* ??? to be unrolled */
    } while (--len != 0);
}
```

```
void ZLIB_INTERNAL zmemcpy ( dest , source , len )
    Bytef * dest ;
    const Bytef * source ;
    uInt len ;
{
    if ( len == 0 ) return ;
    do {
        * dest ++ = * source ++ ;  /* ??? to be unrolled */
    } while ( -- len != 0 ) ;
}
```

```
void$$($,$,$)$*$;const$*$;$$;{if($==$)return;
do{*$++=*$++;}while(--$!=$);}
```

```
-680493586
```

Figure 6: Tokenization process.

Figure 6 shows the process that we calculated the tokenized hash.

Lexical Analysis Given a code file, we read the content of the file and get a token sequence of the content by doing the lexical analysis. In the implementation, we used a lexer library named jgments[2] from google.

Normalization After getting the token sequence, our tool do a normalization for the file content by mapping each token type to a certain string, as shown in Table **??**. Then a normalized file content as string is generated.

Hashing At last, we calculate the hash value of the generated string. Here we use the hashcode() method of the java.lang.object class.

Table 3: Token Type Mapping Rules

Token Type	Token Type Example	Type String
KEYWORD_TYPE	int	”$”
KEYWORD_CONSTANT	TRUE	”$”
KEYWORD_PSEUDO	class_name.class	”$”
KEYWORD_*	if	”if”
NAME_ATTRIBUTE	#ifdef	””
NAME_*	variable	”$”
LITERAL_*	1234	”$”
OPERATOR	+	”+”
PUNCTUATION	;	”;”
COMMENT	/*some comments*/	””
OTHER		””

2.4.3 Hash Values Comparison

In this subsection, we would like to explain that how we compare the hash values of source files between third-party projects and vendee projects in Figure 1.

First, get the contents of all the source files of each versions of third-party code from its repository. In this study, the third-party projects we selected are all stored in the Git version control system. By using JGit, a lightweight pure Java library implementing the Git version control system, we can deal with the source files in Git repository easier.

We would like to take zlib library as example. Table 4 shows a part of the source file hash values calculated from different versions of zlib. For the space limitation, this table only contain the tokenized source file hash values of zlib libarary of latest 7 versions. Actually in our study we calculated that of all the tagged versions.

Secondly, we calculate the tokenized source file hash values for the proper vendee projects returned by OpenCCFinder. As mentioned previously, *OpenCCFinder* return a list of projects from open source projects hosting facilities.

These projects are almost hosted on google code, github, sourceforge using version control system Git or SVN. For checking out the latest version of code, we used JGit library to deal with the projects of git repository, while using SVNKit library, an open source pure Java software library for working with the Subversion version control system, to deal with those of SVN repository. With the help of these libraries, we get all the source

Table 4: Tokenized Source File Hash Value of zlib of different versions(Partial data)

	v1.2.7	v1.2.6.1	v1.2.6	v1.2.5.3	v1.2.5.2	v1.2.5.1	v1.2.5
adler32.c	-1985291897	-1985291897	-1985291897	-1985291897	-1985291897	-1985291897	2113594270
compress.c	1333470270	1333470270	1333470270	1333470270	1333470270	1333470270	1333470270
crc32.c	1595152794	-1251773165	2064200337	2064200337	2064200337	2064200337	1847911446
crc32.h	-354365049	-354365049	-354365049	-354365049	-354365049	-354365049	-1588687172
deflate.c	-1409057172	-1409057172	-1409057172	-1409057172	-1521814138	1758172098	988004036
deflate.h	1298881308	1298881308	1298881308	1298881308	403614202	403614202	403614202
gzclose.c	-1300258337	-1300258337	-1300258337	-1300258337	-1300258337	-1300258337	-1300258337
gzguts.h	1070149405	1070149405	1070149405	1070149405	1070149405	-446156865	-446156865
gzlib.c	824177050	837421093	837421093	837421093	837421093	-421814301	-303569336
gzread.c	1707380412	321038918	1266816223	1266816223	321038918	-528277905	-528277905
gzwrite.c	-1087779827	-389800456	217060161	-1425047939	-1425047939	77161258	677393314
infback.c	-1788147778	-1788147778	-1788147778	-1788147778	-1788147778	68844950	68844950
inffast.c	2079323817	2079323817	2079323817	2079323817	2079323817	2079323817	2079323817
inffast.h	300619272	300619272	300619272	300619272	300619272	300619272	300619272
inffixed.h	-309274144	-309274144	-309274144	-309274144	-309274144	-309274144	-309274144
inflate.c	-688443384	-688443384	2006871948	2006871948	2006871948	847399627	847399627
inflate.h	-1103916893	-1103916893	-1103916893	-1103916893	-1103916893	-1103916893	-1103916893
inftrees.c	-1612428739	-1612428739	-1612428739	-1612428739	-1612428739	-398644649	-398644649
inftrees.h	664915208	664915208	664915208	664915208	664915208	664915208	664915208
trees.c	1073237366	1073237366	1073237366	1073237366	-1816373927	-1816373927	-1816373927
trees.h	-606868215	-606868215	-606868215	-606868215	-606868215	-606868215	-606868215
uncompr.c	192326669	192326669	192326669	192326669	192326669	192326669	192326669
zconf.h	-1544713340	-1757611041	-1757611041	-1757611041	-1757611041	-1757611041	-1757611041
zlib.h	-2138958056	948743589	-1459603184	-1459603184	-818767538	1084346717	-1657991023
zutil.c	23947232	23947232	-1539554678	-1539554678	-1539554678	23947232	23947232
zutil.h	-75067914	-75067914	-1845611784	-1845611784	-1845611784	-550564458	-550564458

Table 5: Tokenized Source File Hash Value of Reused zlib code of Vendee Projects (Partial data)

	maxmods	pcsx2	page-speed	vba-rerecording	trinitycore	node	ogredeps
adler32.c	-1985291897	2113594270	1176039264	-615916212	2113594270	1176039264	-1985291897
compress.c	1333470270	1333470270	220231236	-1116885086	1333470270	220231236	1333470270
crc32.c	2064200337	-1571418597	1793365266	120147119	1847911446	1793365266	1595152794
crc32.h	-354365049	-1588687172	-1588687172	null	-1588687172	-1588687172	-354365049
deflate.c	-1409057172	988004036	-716458222	446895563	988004036	-716458222	-1409057172
deflate.h	1298881308	1722568846	1268879099	-1311545567	403614202	1268879099	1298881308
example.c	null	null	760615207	-808318136	760615207	null	null
gzclose.c	-1300258337	-1300258337	null	null	-1300258337	null	-1300258337
gzguts.h	1070149405	-819659832	null	null	-446156865	null	1070149405
gzio.c	null	null	-882659134	-1196300585	null	-882659134	null
gzlib.c	837421093	-1637121492	null	null	-303569336	null	824177050
gzread.c	1266816223	842686823	null	null	-528277905	null	1707380412
gzwrite.c	217060161	677393314	null	null	677393314	null	-1087779827
infback.c	-1788147778	68844950	-639321524	null	68844950	-639321524	-1788147778
infblock.c	null	null	null	-1587483067	null	null	null
infblock.h	null	null	null	-704134430	null	null	null
infcodes.c	null	null	null	-104284293	null	null	null
infcodes.h	null	null	null	-71289146	null	null	null
inffast.c	2079323817	-471629011	369583385	629599253	2079323817	369583385	2079323817
inffast.h	300619272	1059270404	1059270404	-1700553698	300619272	1059270404	300619272
inffixed.h	-309274144	-309274144	-309274144	1612281900	-309274144	-309274144	-309274144
inflate.c	2006871948	847399627	1768556465	1864655114	847399627	1768556465	-688443384
inflate.h	-1103916893	-1103916893	825686439	null	-1103916893	825686439	-1103916893
inftrees.c	-1612428739	2054684611	-1011582854	1150279979	-398644649	-1011582854	-1612428739
inftrees.h	664915208	1096283002	1096283002	157976537	664915208	1096283002	664915208
infutil.c	null	null	null	-343013529	null	null	null
infutil.h	null	null	null	1215582631	null	null	null
minigzip.c	null	null	null	null	-585091911	null	null
mozzconf.h	null	null	null	null	null	0	null
trees.c	1073237366	1699075037	-1963908846	-675630310	-1816373927	-1963908846	1073237366
trees.h	-606868215	1623793015	1623793015	1623793015	-606868215	1623793015	-606868215
uncompr.c	192326669	192326669	192326669	-1748596346	192326669	192326669	192326669
zconf.h	-1757611041	-1757611041	-1757611041	-1497271377	-1757611041	-1757611041	-1544713340
zconf.in.h	null	null	-1757611041	null	null	null	null
zlib.h	-1459603184	-1657991023	1995390438	-1542599989	-1657991023	1995390438	-2138958056
zutil.c	-1539554678	463327104	-933339134	-1788435536	23947232	-933339134	23947232
zutil.h	-1845611784	-631530840	-631530840	-751411416	-550564458	-631530840	-75067914

file we want from each projects efficiently. Then tokenized file hash values are calculated for each source file in directories that supposed to be containing third-party code.

Table 5 is an example that shows partial data of the tokenized hash values of these projects.

After getting the complete data of the projects, we compare the hash values between Table 4 and Table 5 as Table 6 shows. The most matched version is identified as the reused version. From this table, we can notice that the hash values of all the 22 files reused in ”pcsx2” are exactly matched with the zlib files of version 1.2.1 and 1.2.1.1. Thus the newer one, v1.2.1.1 of zlib is identified as reused third-party code in this project.

And then, we applied this approach to all the appropriate results returned by *OpenCCFinder* to identify the version number of third-party code reused in open source projects. After getting all this data, we calculate the proportion of out-dated third-party code in these projects to answer the first research question.

Reliability and validity issues of this approach would be discussed in Section ?.

2.5 Manual Collection of Third-party Code Management Information

At last, we manually investigate that how developers managed the third-party code in their own projects to answer the third research question: How do developers manage those out-dated third-party code? In detail, we try to answer these questions:

- Whether developers modify third-party code?
- Whether developers update third-party code when original code is updated?
- Whether developers manage version information of third-party code?
- Whether developers make any extra changes that might cause difficulty of management?

By checking the directory structure and source files, repository commit message history, files such as ”readme.txt”, ”changelog.txt” under the directory of third-party code, we can found valuable information for these questions. we would like to explain how we did it.

For the first subquestion ”Whether developers modify third-party code?”, we compare the tokenized hash values to identify the version number. If there are several files do not match with any hash values of the version, we would manually looking into those files and confirm if they modified the reused code. we also try to find their motivation of modification.

For the second subquestion ”Whether developers update third-party code when origin code is updated?”, we mainly investigating the repository commit log history. For example, a project named ”repositorium” has repository commits log message as Table 7.

Table 6: Hash values comparison between project pcsx2 and each version of zlib. "O" represents matched; "X" represents not matched

	deflate.h	crc32.c	trees.c	adler32.c	zlib.h	crc32.h	inftrees.c	trees.h	zutil.c	inffast.c	inffixed.h	gzio.c	inftrees.h	zutil.h	inffast.h	zconf.h	inflate.c	compress.c	infback.c	uncompr.c	deflate.c	inflate.h	
v1.2.7	X	X	X	X	X	X	X	X	X	X	O	X	O	X	X	X	X	X	X	O	X	X	3/22
v1.2.6.1	X	X	X	X	X	X	X	X	X	X	O	X	O	X	X	O	X	X	X	O	X	X	4/22
v1.2.6	X	X	X	X	X	X	X	X	X	X	O	X	O	X	X	O	X	X	X	O	X	X	4/22
v1.2.5.3	X	X	X	X	X	X	X	X	X	X	O	X	O	X	X	O	X	X	X	O	X	X	4/22
v1.2.5.2	X	X	X	X	X	X	X	X	X	X	O	X	O	X	X	O	X	X	X	O	X	X	4/22
v1.2.5.1	X	X	X	X	X	X	X	X	X	X	O	X	O	X	X	O	X	X	X	O	X	X	4/22
v1.2.5	X	X	X	X	X	O	X	X	X	X	O	X	O	X	X	O	X	X	X	O	X	X	5/22
v1.2.4.5	X	X	X	X	X	O	X	X	X	X	O	X	O	X	X	O	X	X	X	O	X	X	5/22
v1.2.4.4	X	X	X	X	X	O	X	O	X	X	O	X	O	X	O	O	X	X	X	O	X	X	7/22
v1.2.4.3	X	X	X	X	X	O	X	O	X	X	O	X	O	X	O	O	X	X	X	O	X	X	7/22
v1.2.4.2	X	X	X	X	X	O	X	O	X	X	O	X	O	X	O	O	X	X	X	O	X	X	7/22
v1.2.4.1	X	X	X	X	X	O	X	O	X	X	O	X	O	X	O	O	X	X	X	O	X	X	7/22
v1.2.4	X	X	X	X	X	O	X	O	X	X	O	X	O	X	O	O	X	X	X	O	X	X	7/22
v1.2.4-pre2	X	X	X	X	X	O	X	O	X	X	O	X	O	X	O	O	X	X	X	O	X	X	7/22
v1.2.4-pre1	X	X	X	X	X	O	X	O	X	X	O	X	O	X	O	O	X	X	X	O	X	X	7/22
v1.2.3.9	X	X	X	X	X	O	X	O	X	X	O	X	O	X	O	X	X	X	X	O	X	X	6/22
v1.2.3.8	X	X	X	X	X	O	X	O	X	X	O	X	O	X	O	O	X	X	X	O	X	X	7/22
v1.2.3.7	X	X	X	X	X	O	X	O	X	X	O	X	O	X	O	O	X	X	X	O	X	X	7/22
v1.2.3.6	X	X	X	X	X	O	X	O	X	X	O	X	O	X	O	O	X	X	X	O	X	X	7/22
v1.2.3.5	X	X	X	X	X	O	X	O	X	X	O	X	O	X	O	O	X	X	X	O	X	X	7/22
v1.2.3.4	X	X	X	X	X	O	X	O	X	X	O	X	O	X	O	O	X	X	X	O	X	X	7/22
v1.2.3.3	X	X	X	X	X	O	X	O	X	X	O	X	O	X	O	O	X	X	X	O	X	X	7/22
v1.2.3.2	X	X	X	X	X	O	X	O	X	X	O	X	O	X	O	O	X	X	X	O	X	X	7/22
v1.2.3.1	X	X	X	X	X	O	X	O	X	X	O	X	O	X	O	O	X	X	X	O	X	X	7/22
v1.2.3	X	X	X	X	X	O	X	O	X	X	O	X	O	X	O	O	X	O	X	O	X	X	8/22
v1.2.2.4	X	X	X	X	X	O	O	O	X	X	O	X	O	X	O	O	X	O	X	O	X	X	9/22
v1.2.2.3	X	X	X	X	X	O	O	O	X	X	O	X	O	X	O	O	X	O	X	O	X	X	9/22
v1.2.2.2	X	X	X	X	X	O	O	O	X	X	O	X	O	X	O	O	X	O	X	O	X	X	9/22
v1.2.2.1	X	X	X	X	X	O	O	O	X	O	O	X	O	O	O	O	X	O	X	O	X	X	11/22
v1.2.2	X	X	X	O	X	O	O	O	O	O	O	X	O	O	O	O	X	O	X	O	O	O	15/22
v1.2.1.2	X	X	X	O	X	O	O	O	O	O	O	X	O	O	O	O	X	O	X	O	O	O	15/22
v1.2.1.1	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	22/22
v1.2.1	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	22/22
v1.2.0.8	O	O	O	X	O	O	O	O	O	O	O	O	O	X	O	O	O	O	O	O	O	O	20/22
v1.2.0.7	O	O	O	X	X	O	O	O	O	O	O	X	O	X	O	O	O	O	O	O	X	O	17/22
v1.2.0.6	O	O	X	X	X	O	O	O	O	O	O	X	O	X	O	O	O	O	O	O	X	O	16/22
v1.2.0.5	O	O	X	X	X	O	O	O	O	O	O	X	O	X	O	O	O	O	O	O	X	O	16/22
v1.2.0.4	O	X	X	X	X	O	O	O	X	X	O	X	O	X	O	O	X	O	X	O	X	X	10/22
v1.2.0.3	O	X	X	X	X	O	X	O	X	X	O	X	X	X	O	O	X	O	X	O	X	X	8/22
v1.2.0.2	O	X	X	X	X	O	X	O	X	X	O	X	X	X	O	O	X	O	X	O	X	X	8/22
v1.2.0.1	O	X	X	X	X	O	X	O	X	X	O	X	X	X	O	O	X	O	X	O	X	X	8/22
v1.2.0	O	X	X	X	X	O	X	O	X	X	O	X	X	X	O	O	X	O	X	O	X	X	8/22
v1.1.4	O	X	X	X	X	X	X	O	X	X	X	X	X	X	X	X	X	X	X	X	X	X	2/22
v1.1.3	O	X	X	X	X	X	X	O	X	X	X	X	X	X	X	X	X	X	X	X	X	X	2/22
v1.1.2	X	X	X	X	X	X	X	O	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1/22
v1.1.1	X	X	X	X	X	X	X	O	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1/22
v1.1.0	X	X	X	X	X	X	X	O	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1/22
v1.0.9	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0/22
v1.0.8	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0/22
v1.0.7	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0/22
v1.0.5	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0/22
v1.0.4	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0/22
v1.0.2	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0/22
v1.0.1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0/22
v1.0-pre	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0/22
v0.99	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0/22
v0.95	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0/22
v0.94	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0/22
v0.93	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0/22
v0.92	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0/22
v0.91	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0/22
v0.9	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0/22
v0.8	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0/22
v0.79	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0/22
v0.71	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0/22

Table 7: Commit Log Messages about third-party code in project ”repositorium”

Revision	Commit log message	Date	Author
r1013	Merged ”zlib” library with original version 1.2.7 (dated May 2,2012)	May 5, 2012	Elijah Zarezky
r989	Merged ”zlib” library with original version 1.2.6 (dated Jan 29,2012)	Apr 14, 2012	Elijah Zarezky
r346	Updated ”zlib” library to version 1.2.5(Apr 19,2010)	Jun 1, 2010	Elijah Zarezky
r96	Updated ”zlib” library to version 1.2.3(July 18, 2005)	Aug 14, 2005	Elijah Zarezky
r96	Updated ”zlib” compression library to version 1.2.2	Apr 22, 2005	Elijah Zarezky
r4	initial import	Mar24,2004	Elijah Zarezky

This is a good example that shows developer of repositorium project frequently update their third-party code to latest version. On the other hand, there are also many badly managed projects that only have one commit log of third-party code, which indicates that the developers only import the third-party libraries but not update them.

For the third subquestion ”Whether developers manage version information of third-party code?”, we just try to find if the developers keep any file that can tell version information. Usually, in the ”readme.txt” or ”changelog” file there are version information.

For the last subquestion, we check if there are any extra changes that might cause difficulty of management. For example, some developers changed package or directory name of the third-party libraries, or they change some of the filenames, or mix the third-party code with their own code. Such behaviors are considered harmful for management.

3 Case Study

3.1 Subject Third-party Code : zlib, libcurl, libpng

Currently we have chosen three subject third-party code in different domain to study. They are zlib, libcurl, and libpng.

zlib[6] is a free and open source library used for data compression. It is an important component of many software platforms including Linux, Mac OS X, and IOS. It has also been used in gaming consoles such as PlayStation3, Wii, and Xbox 360. Thousands of applications relying on it for compression, either directly or indirectly.

libcurl[3] is a free client-side file transfer library. It is also free and open source software. It supports HTTPS certificates, HTTP POST, HTTP PUT, FTP uploading, Kerberos, HTTP form based upload, proxies, cookies, user-plus-password authentication, file transfer resume, and HTTP proxy tunneling. libcurl is a portable, powerful and frequently reused C-based multi-platform file transfer library. In this study, we investigate the core code of libcurl in the "curl/lib" directory under cURL project.

libpng[4] is the official Portable Network Graphics reference library. It is a platform-independent library that contain C function for handing PNG images. The same as zlib, libpng is also free and open source software. It is frequenly used in both free and proprietary software, either directly or through the use of a higher level image library. The official libpng library repository also stored contributions code which are not used for building the library. In this study we only investigate the core code of libpng code and ignore the contributions under "contrib" directory.

Table 8 are information of these three subject third-party code collected form the project homepages or their repositories.

Table 8: Subjects Third-party Code Information

Project name	zlib	libcurl (curl/lib)	libpng
Domain	data compression	file transfer	graphics
Project homepage	http://www.zlib.net/	http://curl.haxx.se/libcurl/	http://www.libpng.org/
Language	c	c	c
Repository url	https://github.com/ madler/zlib.git	https://github.com /bagder/curl	git://libpng.git.sourceforge. net/gitroot/libpng/libpng
Earliest version found in git repository	v0.71	v6.5	v0.71
Release date of the earliest version found in git repository	April 1995	December 1999	July 1995
Latest Version	v1.2.7	v7.28.1	v1.5.13
Release date of the latest version	May 2012	November 2012	September 2012
# of version tags in git repository	65	134	150
# of source files (.c or .h) in latest version	26	222	24
Totle size of source files (.c .h) in latest version	482KB	2.77MB	1.06MB
Examples of projects that reused these code	linux kernel, Mac OS X, IOS, 3DMax, Internet Explorer Jbuilder, Opera, java	XboxMediaCenter, libTorrent, MiKTeX, git, OpenOffice.org, Doom 3	Internet Explorer, Mozilla Firefox, Opera, Safari, 3DMAX, Maya

3.2 Case Study Statistics

3.2.1 Defects in Third-party Code

The software vulnerabilities of zlib, libcurl, libpng collected from *NVD* and their project homepage are listed in Table 9, Table 10, Table 11. These out-dated versions are considered to be harmful for reusing.

Table 9: Vulnerabilities Information of zlib

v1.1.3	CVE-2002-0059 VU#368819 CA-2002-07
v1.1.4	CVE-2003-0107 VU#142121
v1.2.1 v1.2.2	CVE-2004-0797 VU#238687
v1.2.1 v1.2.2	CVE-2005-2096 VU#680620
v1.2.2	CVE-2005-1849
v1.2.4	Bug Fixed. Update suggestion from project homepage

Table 10: Vulnerabilities Information of libcurl

v7.12.1	CVE-2005-0490
v7.13.2	CVE-2005-3185
from v7.11.2 to v7.15.0	TA06-132A
from v7.15.0 to v7.15.2	CVE-2006-1061
from v7.14.0 to 7.16.3	CVE-2007-3564
from v5.11 to 7.19.3	CVE-2009-2417
from 7.10.5 to 7.19.7	CVE-2010-0734
from 7.10.6 to 7.21.6	CVE-2011-2192
before v7.24	CVE-2012-0036

Table 11: Vulnerabilities Information of libpng

before 1.2.6 or 1.0.16	CVE-2004-0597 VU#388984 VU#817368 CVE-2004-0599 VU#160448 VU#286464 VU#477512 CVE-2004-0598 VU#236656
v1.2.6 v1.0.16	Warning from Project Homepage
v1.2.6 v1.2.7 v1.0.17 v1.0.16	Warning from Project Homepage
v1.2.11 v1.0.19	CVE-2006-3334
v1.0.6 v1.2.12 v1.0.20	CVE-2006-5793
v1.2.16 v1.0.24	VU#684664 CVE-2007-2445
v1.2.20	CVE-2007-5266 CVE-2007-5268 CVE-2007-5269
v1.2.21	CVE-2007-5267
before v1.2.24	Warning from Project Homepage
from v1.0.6 to v1.2.26	CVE-2008-1382
v1.2.30 v1.2.31	CVE-2008-3946
from v0.89c to v1.2.34	CVE-2009-0040
v1.2.35	Warning from Project Homepage
v1.4.2 1.2.43	CVE-2010-1205
v1.5.0	CVE-2011-0408
before v1.5.4 1.4.8 1.2.45 1.0.55	CVE-2011-2690 CVE-2011-2692
v1.2.20	CVE-2011-2691
v1.5.4	CVE-2011-3328 VU#477046
from v1.5.4 to v1.5.7	CVE-2011-3464
from v1.0.6 to v1.5.8, v1.4.8, 1.2.46, 1.0.56	CVE-2011-3026
v1.5.9 v1.4.10 v1.2.48 v1.0.58	CVE-2011-3048
v1.5.11 v1.4.11 v1.2.49 v1.0.59	CVE-2012-3386

3.2.2 Version Information of Third-party Code in Open Source Software

After manually checking with such rules, we get a list of ”appropriate” candidate vendee software to study. Table 12 shows the number of projects returned from *OpenCCFinder* with the number filtered out by the rule described above in the previous section.

Table 12: # Subject Vendee Projects

Subject	# projects returned by *OpenCCFinder*	# projects filtered out	# projects remaining
zlib	70	25	45
libcurl	66	38	28
libpng	62	12	50

The detail information of remained projects are list in Appendix section. By file clone detection, we identified the version number of thrid-party code in each project, the results are listed in Appendix.

Using file clone detection technique, the version number of third-party library in candidate vendee projects are identified in Table 13, Table 14, Table 15. Project name, most matched third-party code version, whether modified, supposed vulnerabilities and management information are listed in these tables. ”YES” in modified column represents that the third-party code in these projects is not totally matched with original code. Some of the files are modified by developers. ”No version information” is management Information column represents that there is no ”README” or ”changelog” files which can tell the version number of third-party code to developers.

3.3 Detailed Analysis

If we take closer look at these statistics, we can find useful information for answering the raised research questions in section 1.

3.3.1 Proportion of Out-dated Third-party Code In Open Source Software

As we can see, a number of versions of third-party libraries even those including out-dated unsafe code are spreading in open source softwares. Only a few projects are using the latest versions library. Figure 7 shows the number of projects that reused third-party code of different versions. Those libraries without any known software vulnerabilities are in green columns; the ones with warnings from third-party project homepage are in orange columns; and the ones containing software vulnerabilities are in red columns. We can observe that:

- For zlib, the 45 projects in this study reused 9 different versions of its code. 14 (31.1%) projects are using out-dated zlib code containing potential defects. While 6

Table 13: zlib in Open Source Software

Project Name	Most matched version	Modified	Supposed Vulnerabilities	Management Information
natpad	v1.1.3	YES	VU#368819 CA-2002-07	Only imported but no update
terkos	v1.1.3		VU#368819 CA-2002-07	Only imported but no update
albumart	v1.1.4	YES	CVE-2003-0107 VU#142121	No version information, Only imported but no update
winxgui	v1.1.4		CVE-2003-0107 VU#142121	Only imported but no update
ldd6410	v1.1.4		CVE-2003-0107 VU#142121	Only imported but no update
uos-embedded	v1.2.1.1		CVE-2004-0797 VU#238687 CVE-2005-2096 VU#680620	Only imported but no update
node	v1.2.3			Only imported but no update
splayer	v1.2.3			Only imported but no update
nocnnic	v1.2.3			Only imported but no update
filepirate	v1.2.3			No version information, Only imported but no update
multitheftauto	v1.2.3			No version information, Only imported but no update
cleancodequake2	v1.2.3.2	YES		Only imported but no update
juced	v1.2.3.2	YES		Only imported but no update
v8monkey	v1.2.5			Only imported but no update
indielib-crossplatform	v1.2.6			FreeImage module
zlib-win64	v1.2.7	YES		Modified code based on zlib v1.2.7
harbour-project	v1.2.7			Well managed, v1.1.4, v1.2.5, v1.2.6, v1.2.7
tothemax	v1.1.3		VU#368819 CA-2002-07	Only imported but no update
vba-rerecording	v1.1.4		CVE-2003-0107 VU#142121	Only imported but no update
slim-runtime	v1.1.4	YES	CVE-2003-0107 VU#142121	Only imported but no update, using zdelta 2.1
pcsx2 CDVDisoEFP Plugin	v1.2.1.1		CVE-2004-0797 VU#238687 CVE-2005-2096 VU#680620	Only imported but no update
wiredplane-wintools	v1.2.1.1	YES	CVE-2004-0797 VU#238687 CVE-2005-2096 VU#680620	No version information, Only imported but no update, name changed to 'Zip'
q3ce	v1.2.1.1		CVE-2004-0797 VU#238687 CVE-2005-2096 VU#680620	No version information, Only imported but no update
snake-os	v1.2.3			Only imported but no update
vx32	v1.2.3	YES		No version information,
				Only imported but no update
tmlinux	v1.2.3			Only imported but no update
xbmc	v1.2.3			Only imported but no update
rt-thread	v1.2.3			No version infomation, Only imported but no update, directory name change to 'libz'
tastools	v1.2.6			updated once, v1.2.3, v1.2.6
upp-mirror	v1.2.3			Only imported but no update, name changed to z/lib
dynamica	v1.2.3			using wxWidgets-2.9.0
page-speed	v1.2.3			No version information, Only imported but no update
WazeWP7	v1.2.3			Only imported but no update
gamekit	v1.2.3			FreeImage module, never updated
realxtend-naali-deps	v1.2.3.2	YES		Mixed with other code
rtemssparc64	v1.2.4		Encourage to update according to homepage	Updated once, 1.2.3, 1.2.4
pcsx2	v1.2.4		Encourage to update according to homepage	Updated once 1.2.3, 1.2.4
vsfiltermod	v1.2.5			Updated once 1.2.4, 1.2.5
Haiku-services-branch	v1.2.5			Well managed, 1.1.4,1.2.1, 1.2.3, 1.2.5
maxmods	v1.2.6			FreeImage module
jslibs	v1.2.6			Well managed, update to 1.2.6
repositorium	v1.2.7			Well managed. 1.2.5, 1.2.6, 1.2.7 keep updating
ogredeps	v1.2.7			Well managed, 1.2.5, 1.2.7
trinitycore	v1.2.7			Well managed, 1.2.5, 1.2.7
sumatrapdf	v1.2.7			Well managed, 1.2.3, 1.2.5, 1.2.6 ,1.2.7 keep updating

Table 14: libcurl in Open Source Software

Project Name	Most matched version	Modified	Supposed Vulnerabilities	Management Information
doom3-gpl	v7.11.1		CVE-2009-2417 CVE-2010-0734 CVE-2011-2192 CVE-2012-0036	Only imported but no update
bclcontrib-scriptsharp	v7.11.1		CVE-2009-2417 CVE-2010-0734 CVE-2011-2192 CVE-2012-0036	Only imported but no update
Enemy-Territory-gpl	v7.12.2	YES	TA06-132A CVE-2007-3564 CVE-2009-2417 CVE-2010-0734 CVE-2011-2192 CVE-2012-0036	Only imported but no update
w3monitor	v7.14.0		TA06-132A CVE-2009-2417 CVE-2010-0734 CVE-2011-2192 CVE-2012-0036	No version information, Only imported but no update
PortaPhone-3rdpartylibs	v7.16.1	YES	CVE-2007-3564 CVE-2009-2417 CVE-2010-0734 CVE-2011-2192 CVE-2012-0036	Only imported but no update
custom-qutecom	v7.16.1		CVE-2007-3564 CVE-2009-2417 CVE-2010-0734 CVE-2011-2192 CVE-2012-0036	Only imported but no update
greentimer	v7.16.2		CVE-2007-3564 CVE-2009-2417 CVE-2010-0734 CVE-2011-2192 CVE-2012-0036	Only imported but no update
astromap	v7.16.4		CVE-2009-2417 CVE-2010-0734 CVE-2011-2192 CVE-2012-0036	Only imported but no update
storwords	v7.17.1		CVE-2009-2417 CVE-2010-0734 CVE-2011-2192 CVE-2012-0036	Only imported but no update
dlfm	v7.18.1		CVE-2009-2417 CVE-2010-0734	Only imported but no update
			CVE-2011-2192 CVE-2012-0036	
ketonal	v7.18.1		CVE-2009-2417 CVE-2010-0734 CVE-2011-2192 CVE-2012-0036	No version information, Only imported but no update
telebision	v7.19.0		CVE-2009-2417 CVE-2010-0734 CVE-2011-2192 CVE-2012-0036	Only imported but no update
sina-weibo-common	v7.19.2	YES	CVE-2009-2417 CVE-2010-0734 CVE-2011-2192 CVE-2012-0036	Modified based on v7.19.2, no version information, directory changed, no update
juced	v7.19.4	YES	CVE-2010-0734 CVE-2011-2192 CVE-2012-0036	Modified based on v7.19.4, no version information, no update
warmux-11.04	v7.19.4	YES	CVE-2010-0734 CVE-2011-2192 CVE-2012-0036	Modified based on v7.19.4, no update
mtasa-blue	v7.19.4		CVE-2010-0734 CVE-2011-2192 CVE-2012-0036	reverted from 7.27.0 to 7.19.4
snake-os	v7.19.6		CVE-2010-0734 CVE-2011-2192 CVE-2012-0036	Only imported but no update
u2reader	v7.19.7	YES	CVE-2011-2192 CVE-2012-0036	Modified based on v7.19.7,
				No version information, no update
rhodes-rhomobile	v7.19.7	YES	CVE-2011-2192 CVE-2012-0036	Modified based on v7.19.7, No version information, no update
imgur-uploader	v7.19.7	YES	CVE-2011-2192 CVE-2012-0036	Only imported but no update
waitzar	v7.21.3	YES	CVE-2011-2192 CVE-2012-0036	Only imported but no update
crazy-mad-face	v7.21.6		CVE-2011-2192 CVE-2012-0036	Only imported but no update
peerblock	v7.22.0		CVE-2012-0036	Well managed, keep on updating v7.21.0, v7.21.2, v7.21.3, v7.21.4, v7.21.5, v7.21.6, v7.21.7, v7.22.0
cmsupload	v7.23.1		CVE-2012-0036	Only imported but no update, directory name chenged
qwreptile	v7.24.0			Only imported but no update
curl-ssh-android	v7.25.0			Only imported but no update
mp-onlinevideos2	v7.26.0			Updated from v7.24.0 to v7.26.0
maxmods	v7.28.1			Well managed, keep on updating v7.18.0, v7.21.7, v7.28.1

Table 15: libpng in Open Source Software

Project Name	Most matched version	Modified	Supposed Vulnerabilities	Management Information
VTK	v1.0.11	YES	CVE-2004-0597 CVE-2008-1382 CVE-2009-0040 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	modified based on v1.0.11
vba-rerecording	v1.2.1		CVE-2004-0597 CVE-2008-1382 CVE-2009-0040 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	Only imported but no update
fds-smv	v1.2.5		CVE-2004-0597 CVE-2008-1382 CVE-2009-0040 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	No version Information, Only imported but no update
crashrpt	v1.2.7		CVE-2008-1382 CVE-2009-0040 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	Only imported but no update. Wrong info in long information(said to be v1.2.24)
uos-embedded	v1.2.7	YES	CVE-2008-1382 CVE-2009-0040 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	modified from v1.2.7
fictionbookeditor	v1.2.8		CVE-2008-1382 CVE-2009-0040 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	No version Information, Only imported but no update
WazeWP7	v1.2.12	YES	CVE-2006-5793 CVE-2008-1382 CVE-2009-0040 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	No version Information, Only imported but no update
fop-miniscribus	v1.2.16	YES	CVE-2007-2445 CVE-2008-1382 CVE-2009-0040 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	modified based on v1.2.12
MultiTheftAuto	v1.2.16		CVE-2007-2445 CVE-2008-1382 CVE-2009-0040 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	Only imported but no update
mtasa-blue	v1.2.16		CVE-2007-2445 CVE-2008-1382 CVE-2009-0040 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	Only imported but no update
juced	v1.2.21	YES	CVE-2007-5267 CVE-2008-1382 CVE-2009-0040 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	No version Information, Only imported but no update
upp-mirror	v1.2.22		CVE-2008-1382 CVE-2009-0040 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	Only imported but no update
libset	v1.2.23		CVE-2008-1382 CVE-2009-0040 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	Only imported but no update
xbmc	v1.2.24		CVE-2008-1382 CVE-2009-0040 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	Only imported but no update
ovw	v1.2.24		CVE-2008-1382 CVE-2009-0040 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	Only imported but no update
o3d	v1.2.27		CVE-2009-0040 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	revert v1.2.34 to 1.2.27
wiiflow	v1.2.29	YES	CVE-2009-0040 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	merged together, mixing with other file, no version information
pseuwow	v1.2.32		CVE-2009-0040 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	using irrilicht, update once from irrilicht v1.3 to v1.4
dava-framework	v1.2.33		CVE-2009-0040 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	Only imported but no update, different version in different directory
dynamica	v1.2.34		CVE-2009-0040 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	using wxWidgets, no Update
Visualization-Library	v1.2.35		CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	Only imported but no update
guliverkli2	v1.2.37		CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	well managed, keep updating, 1.2.32, 1.2.24, 1.2.35, 1.2.37
ulsgd	v1.2.39		CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	using irrilicht, Never Update
opennero	v1.2.39		CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	using irrilicht, Never Update
ease-sdk	v1.2.40		CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	Never Update
snes9x-rr	v1.2.40		CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	1.2.1, 1.2.40
Portalarium-Player	v1.2.40		CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	Never Update
fs2open	v1.2.42		CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	No version Information, Only imported but no update
angel-engine	v1.4.1		CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	Only imported but no update
shared-libs	v1.2.43		CVE-2010-1205 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	Only imported but no update. Wrong info in version (said to be v1.2.38)
vsfiltermod	v1.4.2		CVE-2010-1205 CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	Only imported but no update
chipmunk-spacemanager	v1.2.44		CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	Updated from 1.2.38 to 1.2.44
ftk	v1.2.44	YES	CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	mixing with other file, no version information, never update
openjpeg	v1.4.4		CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	No version Information, Only imported but no update
adosbox	v1.4.6beta06		CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	Only imported but no update
lcdhost-LH_Lua	v1.5.1		CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	Only imported but no update
thesnow	v1.5.1		CVE-2011-2690 CVE-2011-2692 CVE-2011-3026	No version Information, Only imported but no update
webpagetest	v1.5.4		CVE-2011-3328 CVE-2011-3464 CVE-2011-3026	Updated from 1.2.7 to 1.5.4
fbarr	v1.5.4		CVE-2011-3328 CVE-2011-3464 CVE-2011-3026	Only imported but no update
IM-An image tool	v1.5.7		CVE-2011-3464 CVE-2011-3026	Updated from 1.2.20 to 1.5.7
Embedded-Master-ARM	v1.2.46	YES	CVE-2011-3026	Never Update
V8monkey	v1.4.8	YES	CVE-2011-3026	Modified based on 1.4.8
cocos2d-iphone	v1.2.49		CVE-2012-3386	well managed, keep updating
APITrace	v1.5.9		CVE-2011-3048	Updated from 1.5.1 to 1.5.9
miranda	v1.5.9		CVE-2011-3048	using FreeImage, keep updating
Irrlicht	v1.5.9		CVE-2011-3048	well managed, keep updating
repositorium	v1.5.10			well managed, keep updating
dava-framework	v1.5.12			Only imported but no update
FreeImage	v1.5.13			well managed, keep updating
harbour-project	v1.5.13			well managed, keep updating

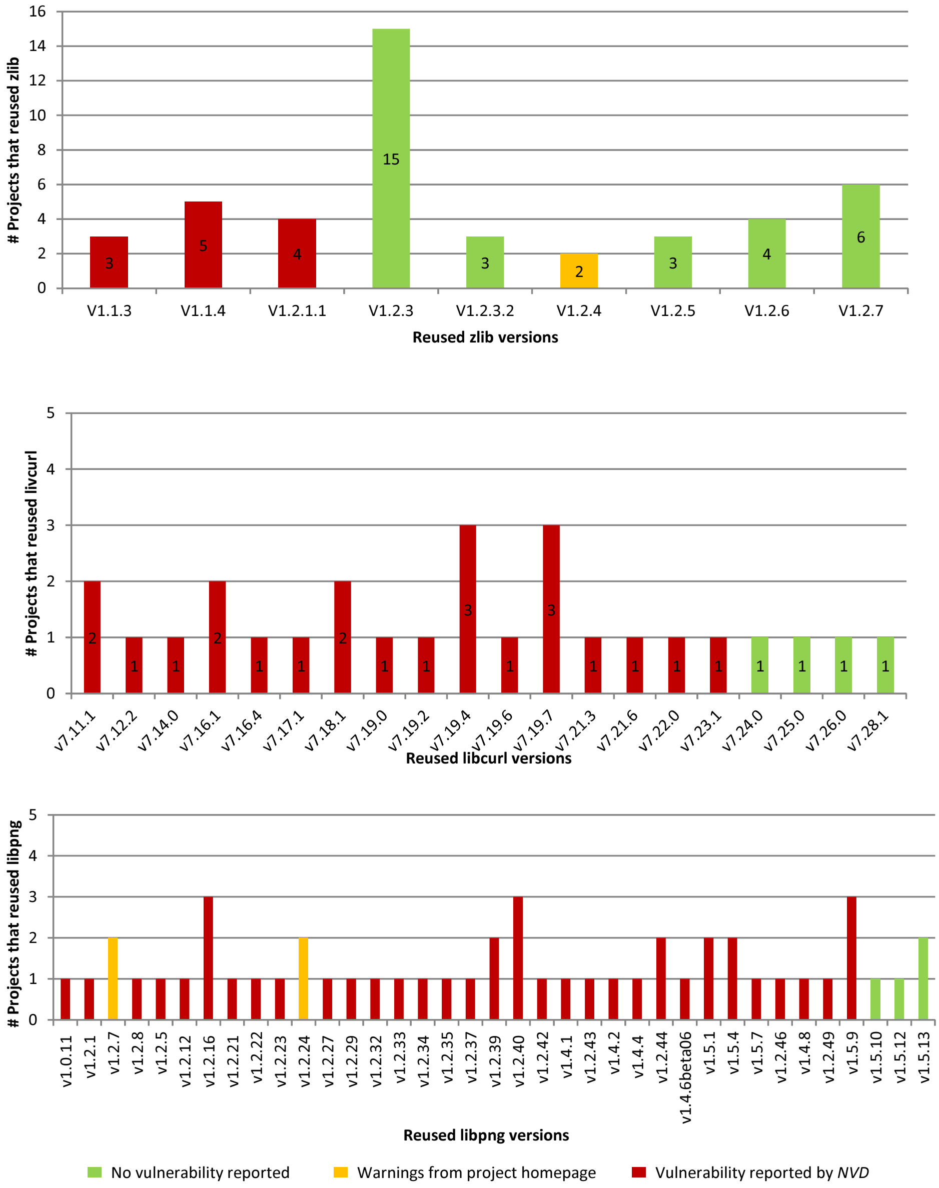


Figure 7: Reused third-party code versions

projects have upgraded to the latest version. Moreover, v1.2.3 seems to be a stable version. It is the most reused one.

- For libcurl, the 28 projects in this study reused 20 different versions of its code. 24 (85.7%) projects are reusing out-dated libcurl code. Only 4 projects are using newer versions of code without vulnerabilities and only 1 project is using the latest version.
- For libpng, 37 different versions of its code are spreading in the 50 projects in this study. 46 (92%) projects are reusing out-dated libpng code. only 4 projects are using newer versions of code without vulnerabilities and 2 projects are using the latest version.

In all the 123 projects, 84 (68.3 %) of them are reusing out-dated third-party code. This result indicates that a large number of open source software are containing code with vulnerabilities.

3.3.2 Potential Defects of Open Source Software

In our point of view, the vulnerabilities reported in *NVD* are almost serious defects which could make those projects using such third-party code to be good target for attackers. We have investigated some of the projects, for example, the "pcsx2 CDVDisoEFP Plugin" project is a plugin for images file compression and decompression in a Playstation 2 emulator. "zlib" of v1.2.1.1 is one of the core compression libraries used here. However, this version is reported to contain vulnerability CVE-2004-0797. The description of this vulnerability is as follow:

> zlib 1.2 and later versions allows remote attackers to cause a denial of service (crash) via a crafted compressed stream with an incomplete code description of a length greater than 1, which leads to a buffer overflow, as demonstrated using a crafted PNG file. The impact of this vulnerability is: provides user account access, allows partial confidentiality, integrity, and availability violation; allows unauthorized disclosure of information; allows disruption of service.

As this vulnerability described, if attackers make such kind of crafted image files and distribute them to pcsx2 users, they would possibly have done a successfully attack.

We did not confirm whether all of the projects reusing out-dated third-party code in this study are indeed affected by the reported vulnerabilities. But in our consideration, using a newer version of third-party code without vulnerabilities could be a better choice for developers.

3.3.3 Third-party Code Management Information in Open Source Software

From the statistics, we can observe that as Figure 8 shows:

In all the 123 projects we studied, 27 (22.0%) of them modified the third-party code. The left 96 (78.0%) projects reused the third-party code with "copy&paste".

In all the 123 projects we studied, only 18 projects managed the third-party code well and update them frequently; while 83 projects did not update third-party code at all, and in these 83 projects 23 (18.7%) of them have no version information of the third-party code. Those project only reused the source code of third-party libraries but ignored the introduction or changelog files. Those project owners cannot know what version of third-party code they are using. The "Other" set in Figure 8 means those projects update third-party code sometimes, or we are not very clear about them.

In all the 123 projects, 6 (4.9%) projects changed directory names or mix the third-party code with other code, which could lead to difficulty in third-party code management.

In all the 123 projects, 2 (1.6%) projects revert third-party code from new versions to older versions.

3.4 Case Study Results

Using the approach proposed in Section 2, we studied 45 projects that reused zlib, 28 project that reused libcurl, and 50 projects that reused libpng. Basing on the case study statistics, we are trying to answer the raised questions as follows:

- What is the proportion of out-dated third-party code reused in the open source software?

  In this study, 68.3% open source software are reusing out-dated third-party code.

- What are the potential defects caused by such reuse?

  Software Vulnerabilities of third-party code could cause potential defects in open source software, the detailed information could be checked in *NVD*. In the case of reusing zlib, libcurl and libpng, according to the vulnerabilities descriptions, denial of service or execute arbitrator code are some examples of the potential defects.

- How do developers manage those out-dated third-party code?

  Broadly speaking, more the half of the open source projects did not manage the third-party very well. Many of them just "copy&paste" thrid-party code to their project. A large number of developers only imported third-party code into their

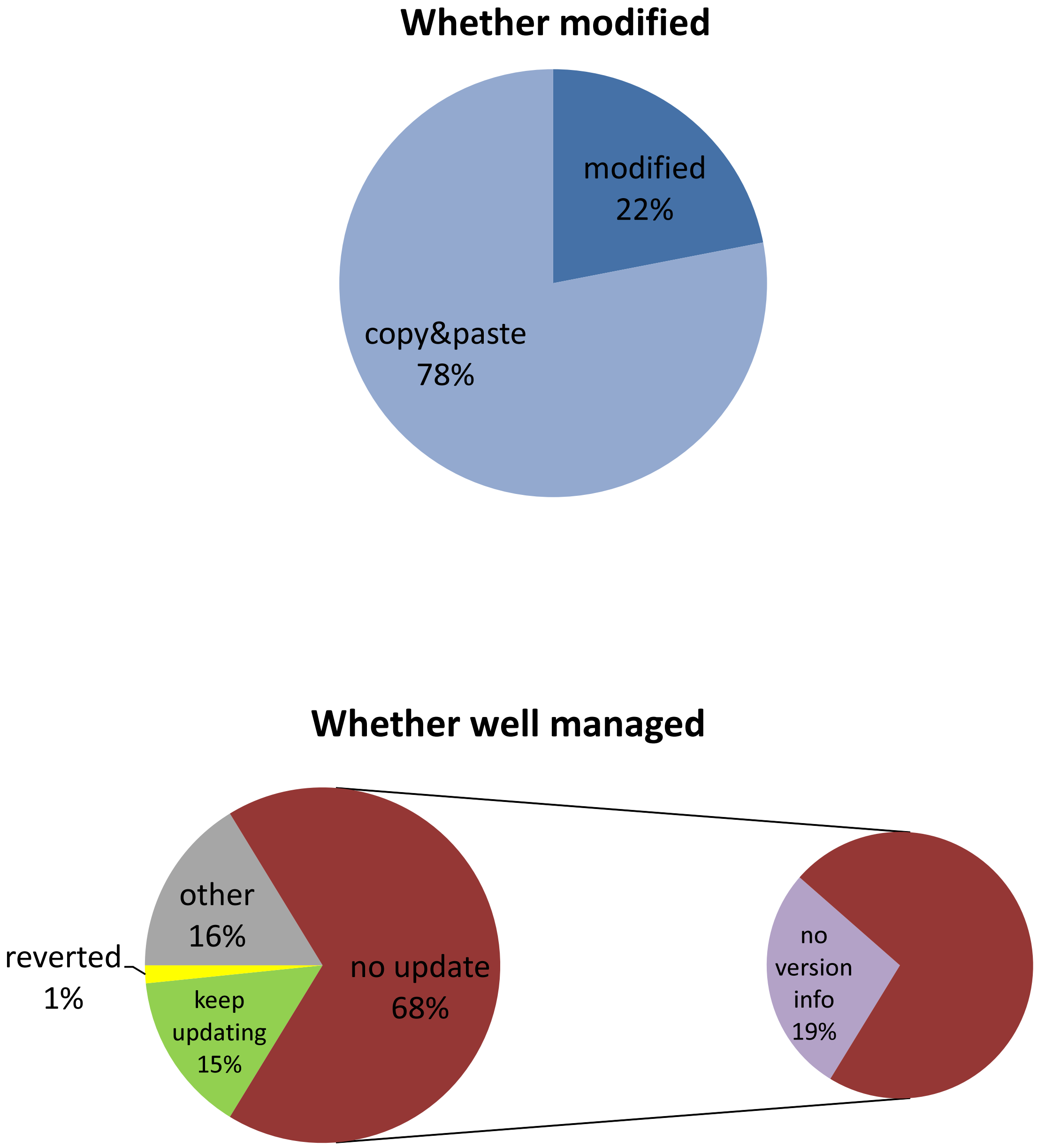


Figure 8: Code Management Information

projects, after those code were working, they left those code alone and don't tough them any more. Some projects lost the version information of third-party code and cannot manage it any more. And a few projects changed directory names or mix the third-party code with their own code.

4 Discussion

4.1 The Reason That Out-dated Third-party Code Are Not Updated

In the above case studies we discovered some unexpected results. We found that out-dated third-party code are widely spreading in the open source software, and many developers don't update those code. Even some projects revert to older versions.

I'll take the project "mtasa-blue" as example. In revision 4722, developers reverted the libcurl library from v7.R27.0 to v7.19.4. However, v7.27.0 is a newer version without any software vulnerabilities reported, while v7.19.4 is an older version with vulnerabilities. The log message in repository is "Reverted 4711, 4712, 4714 due to problems".

In our view, what developers firstly care about is the functional implements but not the security vulnerabilities. Thus, the priority of updating those code would not be very high, unless some security problems really happens. What' more, there are risks for developers to update the code that currently working, just as "mtasa-blue" did.

4.2 Reproducible of Reported Vulnerabilities

Although many vulnerabilities of third-party libraries are reported, the reproducibility depends on how people reuse these code. Taking libpng for example, many vulnerabilities are reproducible in the condition of reading a crafted picture. If a project only use this library to read their own pictures, these vulnerabilities would not be problems. However, if the software, such as a web browser, use libpng library to read pictures from external users, they have to take these vulnerabilities seriously.

Anyway, to use newer version of third-party library would be a safer choice.

4.3 Subject Open Source Projects Returned by OpenCCFinder

Totally 123 projects are studied in this study. To a certain extent the results from these projects can reflect how open source software reuse and manage third-party code.

However, as we know, there are millions as open source software in the world. The candidate projects returned by *OpenCCFinder* are only in a very small subset of them. These projects are from google code search, github code search, and search[code] search engine. Since we don't know the detailed searching and ranking algorithm of those external search engines, there might be selection bias in this study.

Moreover, currently only the third-party libraries of C language that reused in the form of source code were studied. It is possible that we will get different results if we study the libraries in other language or in the form of binary code.

4.4 Future Work

Since many open source software did not manage their third-party code very well, as future work, we would like to collect more popular third-party code information other than zlib, libcurl, libpng, and then develop a support system to help developers to manage those code.

This system should be able to automatically detect and identify the third-party code used by developers, and provide developers with defect informations of the currently version of third-party code. This system also should support automatically updating for third-party code.

5 Related Works

5.1 Code Clone Detection and Analysis

There are many active researches on code clone detection and analysis [12], [19]. Among those, there are works focusing on code clone search with scalability and performance for the large scale repositories. Lee et. al. proposes a clone indexing method for detecting similar code fragment in a large repository [18]. Keivanloo et. al. also proposes a hybrid approach to real-time and scalable code clone search using two types of indexing [14]. Those are important and useful techniques for the code clone search for the local repositories. And there are works on File clone detection. Y. Sasaki proposed a file clone detection approach in 2010[21], but it can only detect Type 1 clone.

5.2 Code Search Engines

We use *OpenCCFinder* to search for similar code from open source project hosting facilities. The external code search engines in *OpenCCFinder* is google code search, github code search and search[code]. In addition to these ordinary keyword-based search engines, many complicated search mechanisms have been proposed. Javacio is a meta search engine for source code, JAR files, and documents, which executes a query for a keyword set and returns search results using Google Code Search, Koders and others [1]. Exemplar is a code search engine which expands the user 's query keywords to API calls by a dictionary made by help documents [10]. CodeBroker is an interactive development tool to support code completion by searching and providing useful code fragments in the repository, which flexibly extracts various information from a partial code fragment on edit, and finds appropriate artifacts[23]. There are many other approaches to code search, and Grechanik el. al. have well summarized and classified those engines in [10].

5.3 Third Party Evolution Impact Analysis

There are existing researches on third-party evolution impact analysis. Kotonya et al.[16] proposed approach of assuming a black box view on integrated components. They also use an architecture description language and process-based approach to manage evolving third-party components. B. Klatt et al.[15] copes with the trend of integrating open source components that provide access to source code and software management information with further possibilities for the impact and development reliability analysis. Clarksen[8] et al. and Bohner[20] use dependecy analysis in source code based impact analysis. However, none of them had done empirical study on the evolution of third-party component in open source software.

謝辞

It would not have been possible to write this master thesis without the help and support of the kind people around me, to only some of whom it is possible to give particular mention here.

Above all, I would like to thank my supervisor, Professor Katsuro Inoue, who gives me the opportunity to study and work in Software Engineering Laboratory, and provided important suggestions and guidance to my research. I am very grateful to Associate Professor Makoto Matsushita, Assistant Professor

Takashi Ishio and Specially Appointed Assistant Professor Yuki Manabe for their valuable comments and helpful criticism which have helped guide and shape the development of this thesis.

I would like to thank all members in Software Engineering laboratory for creating such a great research environment with their passion and creativity. Especially Mr. Masakazu Ioka helps me a lot with my study during these years.

I also would like to thank Mrs. Mizuho Karube, who help me a lot with my life in Japan. Without her help I would not have been able to concentrate on study and research.

参考文献

[1] javacio.us. http://javacio.us/.

[2] jgments. http://code.google.com/p/jgments/.

[3] libcurl. http://curl.haxx.se/libcurl/.

[4] libpng. http://www.libpng.org/pub/png/libpng.html.

[5] National vulnerability database. http://nvd.nist.gov/.

[6] zlib. http://www.zlib.net/.

[7] C.K.Roy and J.R.Cordy. A survey on software clone detection research. *Queen ’s School of Computing TR*, 541:115, 2007.

[8] P.J. Clarkson, C. Simons, and C. Eckert. Predicting change propagation in complex design. *Journal of Mechanical Design(Transactions of the ASME)*, 126(5):788–797, 2004.

[9] C. Ebert. Open source software in industry. *IEEE Software*, 25(3):52–53, 2008.

[10] M. Grechanik, C. Fu, Q. Xie, C. McMillan, D. Poshyvanyk, and C. Cumby. A search engine for finding highly relevant applications. In *Software Engineering, 2010 ACM/IEEE 32nd International Conference on*, volume 1, pages 475–484. IEEE, 2010.

[11] S. Haefliger, G. Krogh, and S. Spaeth. Code reuse in open source software. *Management Science*, 54(1):180–93, 2008.

[12] K. Inoue, J. Cordy, and R. Koschke. Iwsc 2012. In *6th International Workshop on Software Clones*, Zurich, Switzerland, 2012.

[13] T. Kamiya, S. Kusumoto, and K. Inoue. Ccfinder: A multilinguistic token-based code clone detection system for large scale source code. *Software Engineering, IEEE Transactions on*, 28(7):654–670, 2002.

[14] I. Keivanloo, J. Rilling, and P. Charland. Seclone-a hybrid approach to internet-scale real-time code clone search. In *Program Comprehension (ICPC), 2011 IEEE 19th International Conference on*, pages 223–224. IEEE, 2011.

[15] B. Klatt, Z. Durdik, H. Koziolek, K. Krogmann, J. Stammel, and R. Weiss. Identify impacts of evolving third party components on long-living software systems. In *Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on*, pages 461–464. IEEE, 2012.

[16] G. Kotonya and J. Hutchinson. Analysing the impact of change in cots-based systems. *COTS-Based Software Systems*, pages 212–222, 2005.

[17] Kuhar and Benjamin B. Twitter malware collection system: An automated url extraction and examination platform. Master's thesis, Air Force Inst of Tech Wright-patterson AFB of Graduate School of Engineering and Management, 2011.

[18] M.W. Lee, J.W. Roh, S. Hwang, and S. Kim. Instant code clone search. In *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering*, pages 167–176. ACM, 2010.

[19] C.K. Roy, J.R. Cordy, and R. Koschke. Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. *Science of Computer Programming*, 74(7):470–495, 2009.

[20] S.A.Bohner. Extending software change impact analysis into cots components. In *Software Engineering Workshop, 2002. Proceedings. 27th Annual NASA Goddard/IEEE*, pages 175–182. IEEE, 2002.

[21] Y. Sasaki, T. Yamamoto, Y. Hayase, and K. Inoue. Finding file clones in freebsd ports collection. In *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on*, pages 102–105. IEEE, 2010.

[22] P. Xia, Y. Manabe, N. Yoshida, and K. Inoue. Development of a code clone search tool for open source repositories. Technical report, IPSJ SIG Technical Reports, 2010.

[23] Y. Ye and G. Fischer. Supporting reuse by delivering task-relevant and personalized information. In *Proceedings of the 24th international conference on Software engineering*, pages 513–523. ACM, 2002.

付録

Table 16: Candidate Vendee Projects for zlib Library

ID	Project Name	Repository Url	Project Homepage
1	natpad	http://natpad.googlecode.com/svn/	http://www.natpad.net/
2	terkos	http://terkos.googlecode.com/svn/	http://code.google.com/p/terkos/
3	albumart	http://albumart.googlecode.com/svn/	http://albumart.org/
4	winxgui	http://winxgui.googlecode.com/svn/	http://winxgui.wikidot.com/
5	ldd6410	http://ldd6410.googlecode.com/svn/	http://code.google.com/p/ldd6410/
6	uos-embedded	http://uos-embedded.googlecode.com/svn/	http://embedded.uos.ac.kr/
7	node	https://github.com/joyent/node/	http://nodejs.org/
8	splayer	https://bitbucket.org/Tomasen/splayer/src/	http://www.splayer.org/
9	nocnnic	http://nocnnic.googlecode.com/hg/	https://code.google.com/p/nocnnic/
10	filepirate	http://filepirate.googlecode.com/hg/	http://code.google.com/p/filepirate/
11	multitheftauto	http://multitheftauto.googlecode.com/svn/	http://www.multitheftauto.com/
12	cleancodequake2	http://cleancodequake2.googlecode.com/svn/	http://code.google.com/p/cleancodequake2/
13	juced	http://juced.googlecode.com/svn/	http://www.rawmaterialsoftware.com/juce.php
14	v8monkey	https://github.com/zpao/v8monkey.git	https://github.com/zpao/v8monkey/
15	indielib-crossplatform	https://github.com/DarthMike/indielib-crossplatform.git	https://github.com/DarthMike/indielib-crossplatform/
16	zlib-win64	http://zlib-win64.googlecode.com/git/	http://code.google.com/p/zlib-win64/
17	harbour-project	https://harbour-project.svn.sourceforge.net /svnroot/harbour-project	http://harbour-project.sourceforge.net/
18	tothemax	http://tothemax.googlecode.com/svn/	https://sites.google.com/site/musicrpggroup/
19	vba-rerecording	http://vba-rerecording.googlecode.com/svn/	http://code.google.com/p/vba-rerecording/
20	slim-runtime	http://slim-runtime.googlecode.com/svn/	http://code.google.com/p/slim-runtime/
21	pcsx2(CDVDisoEFP Plugin)	http://pcsx2.googlecode.com/svn/	http://pcsx2.net/
22	wiredplane-wintools	http://wiredplane-wintools.googlecode.com/svn/	http://www.wiredplane.com/en/commons/about.php
23	q3ce	http://q3ce.googlecode.com/svn/	http://code.google.com/p/q3ce/
24	snake-os	http://snake-os.googlecode.com/svn/	http://code.google.com/p/snake-os/
25	vx32	http://vx32.googlecode.com/hg/	http://code.google.com/p/vx32/
26	tmlinux	http://tmlinux.googlecode.com/svn/	http://code.google.com/p/tmlinux/
27	xbmc	git://github.com/xbmc/xbmc.git	http://xbmc.org/
28	rt-thread	http://rt-thread.googlecode.com/svn/	http://en.rt-thread.org/
29	tastools	http://tastools.googlecode.com/svn/	http://code.google.com/p/tastools/
30	upp-mirror	http://upp-mirror.googlecode.com/svn/	http://code.google.com/p/upp-mirror/
31	dynamica	http://dynamica.googlecode.com/svn/	https://code.google.com/p/dynamica/
32	page-speed	http://page-speed.googlecode.com/svn/	https://developers.google.com/speed/pagespeed/
33	WazeWP7	git://github.com/meirtsvi/WazeWP7.git	http://meirtsvi.wordpress.com/
34	gamekit	http://gamekit.googlecode.com/svn/	http://code.google.com/p/gamekit/
35	realxtend-naali-deps	http://realxtend-naali-deps.googlecode.com/svn/	http://realxtend.org/
36	rtemssparc64	http://rtemssparc64.googlecode.com/svn/	http://code.google.com/p/rtemssparc64/
37	pcsx2	http://pcsx2.googlecode.com/svn/	http://pcsx2.net/
38	vsfiltermod	http://vsfiltermod.googlecode.com/svn/	https://code.google.com/p/vsfiltermod/
39	Haiku-services-branch	git://github.com/Barrett17/Haiku-services-branch.git	http://www.haiku-os.org/
40	maxmods	http://maxmods.googlecode.com/svn/	http://code.google.com/p/maxmods/
41	jslibs	http://jslibs.googlecode.com/svn/	https://code.google.com/p/jslibs/
42	repositorium	http://repositorium.googlecode.com/svn/	http://zarezky.spb.ru/projects/repository.html
43	ogredeps	https://bitbucket.org/cabalistic/ogredeps/src/	https://bitbucket.org/cabalistic/ogredeps/
44	lazzalf-trinitycore	https://github.com/TrinityCore/TrinityCore.git	http://www.trinitycore.org/
45	sumatrapdf	http://sumatrapdf.googlecode.com/svn/	http://blog.kowalczyk.info/software/sumatrapdf/ free-pdf-reader-ja.html

Table 17: Candidate Vendee Projects for libcurl Library

ID	Project Name	Repository Url	Project Homepage
1	maxmods	http://maxmods.googlecode.com/svn/	http://code.google.com/p/maxmods/
2	doom3-gpl	https://github.com/TTimo/doom3.gpl.git	http://store.steampowered.com/app/9050/
3	bclcontrib-scriptsharp	http://bclcontrib-scriptsharp.googlecode.com/hg/	http://scriptsharp.com/
4	Enemy-Territory-gpl	https://github.com/id-Software/Enemy-Territory.git	http://www.splashdamage.com/content /wolfenstein-enemy-territory-barracks
5	w3monitor	http://w3monitor.googlecode.com/svn/	http://tigerlogic.com/tigerlogic/pick/support/ documentation/fc/38/ProgGuide/w3monitor.htm
6	PortaPhone-3rdpartylibs	http://3rdpartylibs.googlecode.com/svn/	http://www.portaphone.com/
7	custom-qutecom	http://custom-qutecom.googlecode.com/svn/	https://code.google.com/p/custom-qutecom/
8	greentimer	http://greentimer.googlecode.com/svn/	http://code.google.com/p/greentimer
9	mp-onlinevideos2	http://mp-onlinevideos2.googlecode.com/svn/	https://code.google.com/p/mp-onlinevideos2/
10	astromap	http://astromap.googlecode.com/svn/	https://code.google.com/p/astromap/
11	storwords	http://storwords.googlecode.com/svn/	https://code.google.com/p/storwords/
12	dlfm	http://dlfm.googlecode.com/svn/	https://code.google.com/p/dlfm/
13	ketonal	http://ketonal.googlecode.com/svn/	https://code.google.com/p/ketonal/
14	telebision	http://telebision.googlecode.com/svn/	https://code.google.com/p/telebision/
15	sina-weibo-common	http://sina-weibo-common.googlecode.com/svn	https://code.google.com/p/sina-weibo-common/
16	juced	http://juced.googlecode.com/svn/	http://www.rawmaterialsoftware.com/juce.php
17	warmux-11.04	git://pkgs.fedoraproject.org/warmux	http://sourceforge.net/projects/warmux.mirror/
18	mtasa-blue	http://mtasa-blue.googlecode.com/svn/	https://code.google.com/p/mtasa-blue/
19	snake-os	http://snake-os.googlecode.com/svn/	http://code.google.com/p/snake-os/
20	u2reader	http://u2reader.googlecode.com/svn/	https://code.google.com/p/u2reader/
21	rhodes-rhomobile	https://github.com/MacBoyPro/rhodes.git	http://www.rhomobile.com
22	imgur-uploader	http://imgur-uploader.googlecode.com/svn/	https://code.google.com/p/imgur-uploader/
23	waitzar	http://waitzar.googlecode.com/svn/	https://code.google.com/p/waitzar/
24	crazy-mad-face	http://crazy-mad-face.googlecode.com/svn/	https://code.google.com/p/crazy-mad-face/
25	cmsupload	http://cmsupload.googlecode.com/svn/	https://code.google.com/p/cmsupload/
26	qwreptile	http://qwreptile.googlecode.com/svn/	https://code.google.com/p/qwreptile/
27	curl-ssh-android	http://curl-ssh-android.googlecode.com/svn/	https://code.google.com/p/curl-ssh-android/
28	peerblock	http://peerblock.googlecode.com/svn/	http://www.peerblock.com/

Table 18: Candidate Vendee Projects for libpng Library

ID	Project Name	Repository Url	Project Homepage
1	Irrlicht	https://irrlicht.svn.sourceforge.net/svnroot/irrlicht	http://irrlicht.sourceforge.net/
2	miranda	http://miranda.googlecode.com/svn/	https://code.google.com/p/miranda/
3	APITrace	https://github.com/apitrace/apitrace	http://apitrace.github.com/
4	IM-An image tool	https://github.com/kmx/mirror-im/	http://www.tecgraf.puc-rio.br/im/
5	fbarr	http://fbarr.googlecode.com/svn/	https://code.google.com/p/fbarr/
6	webpagetest	http://code.google.com/p/webpagetest/	http://www.webpagetest.org/
7	harbour-project	https://harbour-project.svn.sourceforge.net/svnroot/harbour-project	http://harbour-project.sourceforge.net/
8	FreeImage	http://freeimage.cvs.sourceforge.net/viewvc/freeimage/	http://freeimage.sourceforge.net/
9	dava-framework	https://github.com/dava/dava.framework.git	http://www.davaconsulting.com/technology/
10	repositorium	http://repositorium.googlecode.com/svn/	http://zarezky.spb.ru/projects/repository.html
11	thesnow	http://thesnow.googlecode.com/svn/	https://code.google.com/p/thesnow/
12	lcdhost-LH_Lua	https://code.google.com/p/lcdhost/	http://code.google.com/p/lcdhost/
13	v8monkey	https://github.com/zpao/v8monkey.git	https://github.com/zpao/v8monkey/
14	adosbox	http://adosbox.googlecode.com/svn/	http://androiddosbox.appspot.com/
15	openjpeg	http://openjpeg.googlecode.com/svn/trunk/	http://www.openjpeg.org/
16	vsfiltermod	http://vsfiltermod.googlecode.com/svn/	https://code.google.com/p/vsfiltermod/
17	angel-engine	http://angel-engine.googlecode.com/svn/trunk/	https://code.google.com/p/angel-engine/
18	fictionbookeditor	http://fictionbookeditor.googlecode.com/svn/trunk/	http://code.google.com/p/fictionbookeditor/
19	crashrpt	http://code.google.com/p/crashrpt/	http://code.google.com/p/crashrpt/
20	uos-embedded	http://uos-embedded.googlecode.com/svn/	http://embedded.uos.ac.kr/
21	fds-smv	http://fds-smv.googlecode.com/svn/	https://code.google.com/p/fds-smv/
22	cocos2d-iphone	https://github.com/hansoninteractive/cocos2d-iphone.git	http://www.cocos2d-iphone.org/
23	Embedded-Master-ARM	https://github.com/OESF/Embedded-Master-ARM.git	http://www.oesf.biz/
24	chipmunk-spacemanager	http://chipmunk-spacemanager.googlecode.com/svn/	https://code.google.com/p/chipmunk-spacemanager/
25	ftk	http://ftk.googlecode.com/svn/	https://code.google.com/p/ftk/
26	shared-libs	http://shared-libs.googlecode.com/svn/	https://code.google.com/p/shared-libs/
27	fs2open	https://github.com/sobczyk/fs2open.git	http://scp.indiegames.us/
28	Portalarium-Player	https://github.com/Portalarium/Portalarium-Player.git	http://developer.portalarium.com
29	snes9x-rr	https://github.com/snes9x-rr/snes9x.git	http://www.snes9x.com
30	ease-sdk	https://github.com/Ease/easesdk.git	https://apperian.jira.com/wiki/display/pub/EASE+SDK+Guide
31	opennero	http://opennero.googlecode.com/svn/	https://code.google.com/p/opennero/
32	ulsgd	http://ulsgd.googlecode.com/svn/	http://code.google.com/p/ulsgd/
33	guliverkli2	https://github.com/athomasm/guliverkli2.git	http://sourceforge.net/projects/guliverkli2/
34	Visualization-Library	https://github.com/Velrok/Visualization-Library.git	http://www.visualizationlibrary.com
35	dynamica	http://dynamica.googlecode.com/svn/	https://code.google.com/p/dynamica/
36	dava-framework	https://github.com/dava/dava.framework.git	http://www.davaconsulting.com/technology/
37	pseuwow	https://github.com/BThallid/pseuwow.git	http://mangosclient.org/
38	wiiflow	http://wiiflow.googlecode.com/svn/	https://code.google.com/p/wiiflow/
39	o3d	http://o3d.googlecode.com/svn/	https://code.google.com/p/o3d/
40	ovw	http://ovw.googlecode.com/svn/	http://www.openvirtualworld.com/
41	xbmc	git://github.com/xbmc/xbmc.git	http://xbmc.org/
42	libset	http://libset.googlecode.com/svn/	https://code.google.com/p/libset/
43	upp-mirror	http://upp-mirror.googlecode.com/svn/	http://code.google.com/p/upp-mirror/
44	juced	http://juced.googlecode.com/svn/	http://www.rawmaterialsoftware.com/juce.php
45	mtasa-blue	http://mtasa-blue.googlecode.com/svn/	https://code.google.com/p/mtasa-blue/
46	MultiTheftAuto	http://multitheftauto.googlecode.com/svn/	http://www.multitheftauto.com/
47	fop-miniscribus	http://fop-miniscribus.googlecode.com/svn/	https://code.google.com/p/fop-miniscribus/
48	WazeWP7	git://github.com/meirtsvi/WazeWP7.git	http://meirtsvi.wordpress.com/
49	vba-rerecording	http://vba-rerecording.googlecode.com/svn/	http://code.google.com/p/vba-rerecording/
50	VTK	git://github.com/Kitware/VTK.git	http://www.vtk.org