Do Developers Update Their Library Dependencies?

An Empirical Study on the Impact of Security Advisories on Library Migration

Raula Gaikovina Kula[1,2], Daniel M. German[3], Ali Ouni[1,4], Takashi Ishio[1,2], Katsuro Inoue[2]

Nara Institute of Science & Technology[1], Osaka University[2], University of Victoria[3], UAE University[4]

Motivation

Many software projects today advocate the use of third-party libraries because of its many benefits: Our motivation stems from reports of outdated and vulnerable libraries being widespread in the software industry.

- In 2014, Sonatype determined that over 6% of the download requests from the Maven Central repository included known vulnerabilities.

Figure: Heartbleed, Poodle, Shellshock, –all high profile library vulnerabilities were found to have affected a significant portion of the software industry.

The goal of our study is to investigate (1) whether or not dependencies are being updated and (2) the level of developer awareness to dependency migration opportunities such as fixing security vulnerabilities.

Research Questions

- **Tracking Library Migration in Practice**
  - RQ_1: To what extent are developers updating their library dependencies?
- **Developer Responsiveness to Awareness Mechanisms**
  - RQ_2: What is the response to important awareness mechanisms such as a new release announcement and a security advisory on library updates?
  - RQ_3: Why are developers non responsive to a security advisory?

Library Migration & Awareness Mechanisms

- **Tracking System & Libraries:** We define a model of system and library dependency relations. Hence, we formally use the following notations. We define $\mathcal{S}$ for a system, and $\mathcal{L}$ for a library. $\mathcal{L}(\texttt{lib},v)$ denotes version v of a library `lib`, and $\mathcal{S}(\texttt{sys},w)$ for version w of a system `sys`.

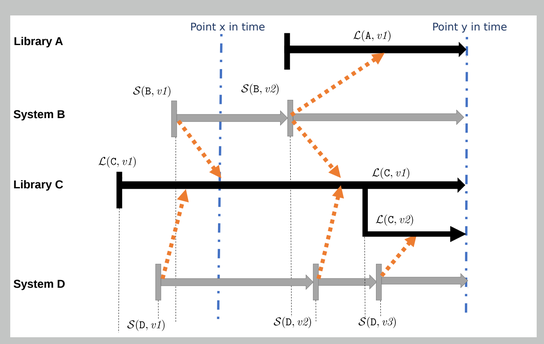


Figure: Library migration between systems and libraries. The orange arrow depicts dependency relations between them.

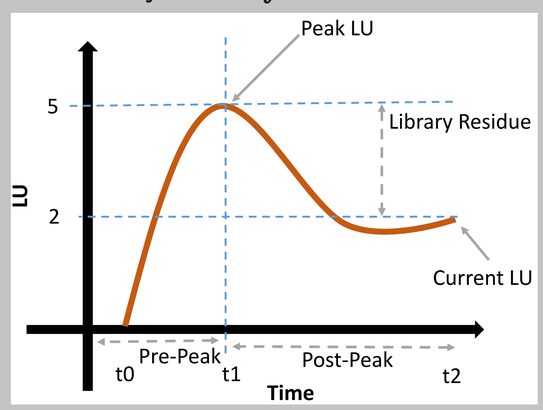


Figure: Simple example of the LU-based metrics. We show the Peak LU at time t1, current LU at time t2 and library residue (Peak LU / Current LU).

- **Effectiveness of Awareness Mechanisms**: We propose a Library Migration Plot (LMP) to visualize impact of awareness mechanisms such as a security advisory.

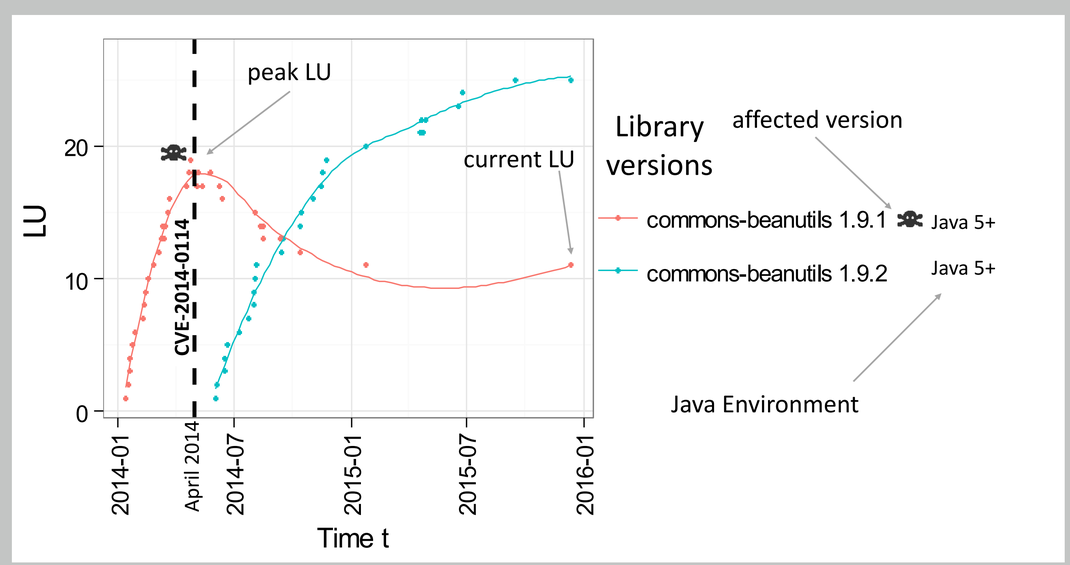
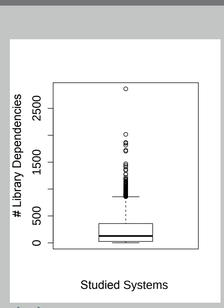


Figure: A Library Migration Plot for libraries $\mathcal{L}(\texttt{beanutils},1.9.1)$ and $\mathcal{L}(\texttt{beanutils},1.9.2)$. In this example, the release of a related security advisory `CVE-2014-0114` (black dashed line) that affects $\mathcal{L}(\texttt{beanutils},1.9.1)$ (marked with crossbones). We also show which JDK (5+) version in which the version supports.

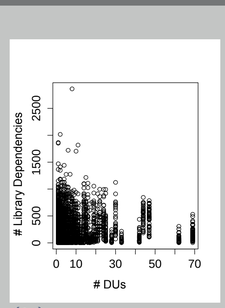
Acknowledgments

- This work is supported by JSPS KANENHI (Grant Numbers JP25220003 and JP26280021) and the "Osaka University Program for Promoting International Joint Research."

RQ_1: Library Migration

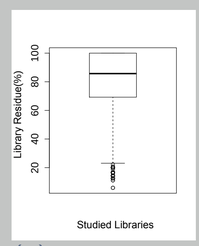


(a) Systems heavily rely on third-party dependencies

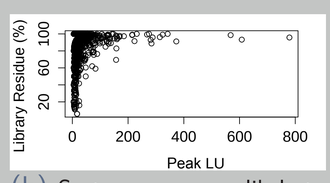


(b) There is little correlation between frequent updates and # dependencies

Figure: System Analysis



(a) Library migration is not common practice for older versions



(b) Systems are more likely to remain with older popular library versions

Figure: Library Analysis

- **Approach:** We analyze 4,659 GitHub Projects and 2,700 library dependencies to understand the extent to which (i) systems use and manage their library dependencies and (ii) library usage trends.
- **Results:** We find that *(i) although system heavily depend on libraries, most systems rarely update their libraries and (ii) systems are less likely migrate their library dependencies, with 81.5% of systems remaining with a popular older version.*

RQ_2: Awareness Mechanisms (Security Advisories)

Table: Eight Case Studies

	Libraries Studied
New Releases	google-guava,junit, log4j
Security Vulnerabilites	commons-beautils, commons-fileupload, commons-httpclient httpcomponents commons-compress

- **Approach:** Using the LMP, we conducted a case study of eight update opportunities (i.e., 3 new releases and 5 security vulnerabilities) to understand developer responsiveness to (i) a new release and (ii) a security advisory disclosure.
- **Results:** *For a new release of a popular library (i) there exist patterns of consistent migration and patterns where an older popular library version is still preferred. For a security advisory disclosure we find cases of developer (ii) non responsiveness to security advisory disclosure, which is sometimes due to an incomplete patch or a latent security advisory.*

RQ_3: Developer Survey

- **Approach:** In a follow-up, we surveyed 16 projects affected by the 5 vulnerabilities in RQ_2, to understand developer awareness and opinions regarding the practice of dependency updates.
- **Results:** *We find that 69% of developers were unaware of their vulnerable dependencies and proceeded to immediately migrate to a safer dependency. Developers evaluate the decision whether or not to update its dependencies based on project specific priorities. Developers cite migration as a practice that requires extra migration effort and added responsibility.*

Conclusions

Although third-party library dependencies is widely practiced, we find that updating is not:

- The study provides motivation for our community develop strategies to improve a developer personal perception of third-party updates.
- Visual aids such as the Library Migration Plots (LMP) could prove useful awareness and motivation for developers quickly update.
- Future work include understanding how developers perceive migration effort and understand responsibilities when using a third-party library dependency.

Replication Dataset

We make available our dataset of 852,322 library dependency migrations at

`https://raux.github.io/Impact-of-Security-Advisories-on-Library-Migrations/`.

Table: Collected dataset

projects creation dates	2004-Oct to 2009-Jan
projects last update	2015-Jan to 2015-Nov
# unique systems (projects)	48,495 (4,659)
# unique library versions	2,736
total size of projects	630 GB
# commits related to `pom.xml`	4,892,770
# library dependency migrations	852,322