Introduction

The use of open-source libraries is a factual reality in the software industry. Their usage has increased exponentially during the last decade as the numbers of available packages. As stated by \cite{Do developers update their library dependencies?}, the Maven Central Repository growth 542.17% from 2010 and 2016. Another popular package repository npm growth from 0 packages at its creation in 2010 to 1,000,000 packages in 2019 (footnote <https://snyk.io/blog/npm-passes-the-1-millionth-package-milestone-what-can-we-learn/> accessed 10/10/2019).

How to leverage the power of open source libraries and still manage the risk of vulnerabilities they carry is a problem that either remain ignored by many software engineers or present considerable challenges to them.

During this paper important studies related to dependencies management will be presented along with the most recent studies about how to identify know vulnerabilities in open source libraries.

Unfortunately, but expected the number of know vulnerabilities also increased proportionaly.

A 2014 study from Sonatype determined that over 6\% of the download requests from the Maven Central Repository were for component versions that included known vulnerabilities. In their review of over 1,500 applications, each of them had an average of 24 severe or critical flaws inherited from their components. \footnote{Report published January 02, 2015 at http://goo.gl/i8J1Zq.}

A white paper produced by Contrast security, stated that over 25\% of all libraries download from Maven Central Repository has vulnerability. Only one vulnerable version of the Java GWT package was downloaded 17,666,703 times \cite{williams2012unfortunate}.

Among the 10 most popular NPM packages, 6 present 1 or more vulnerabilities (<https://snyk.io/blog/npm-passes-the-1-millionth-package-milestone-what-can-we-learn/> accessed 10/10/2019).

The Figure 1 \cite{ On the Impact of Security Vulnerabilities in the npm Package Dependency Network} shows the number of discovered vulnerabilities on the NPM repository. In the same study, they found that out of 610,097 available packages (2017 data) 133,602 packages directly depend on a vulnerable package and 72,470 packages had at least one release that relies on a vulnerable package.

The rest of the document will cover studies on how long vulnerable package remains harmful and how long a fix takes to spread over the dependent applications (section II), what prevents software developers from updating their dependencies (section III) and solutions on how to manage the risk involved on open source libraries adoption.

To analyze for how long a vulnerability is harmful \cite{ On the Impact of Security Vulnerabilities in the npm Package Dependency Network } used a 700 security vulnerabilities report made available by Snyk.io \footnote{Snyk.io} and retrieved the list of its releases from the open source discovery service libraries.io \cite{Nesbitt2017}. Based on the list of release, they identified which ones were affected by the vulnerability.

Figure 2 shows Kaplan-Meier estimator curve \cite{Kaplan1958} for the event “vulnerability is fixed”. The data presented considers the date of the affected release and the date that the fix is available. After 10 months, there is a probability higher than 80% that a high severity vulnerability is still unfixed.

The term know vulnerability refers to a vulnerability to that was disclosure, confirmed by security specialist, had its severity and exploitability scored and has a fix for it. The fact that a fix is available does not finish the harm the vulnerability can cause.