Open-Source Software: Benefits, Security Risks, and Risk Mitigation
TABLE OF CONTENTS

Abstract 3
Introduction 4
Why is OSS so popular? 5
Risks with OSS 7
Security risks 9
Recommendations for remediation 11
How can HCL help? 14
Conclusion 14
References 15
Author Info 16
Abstract

Modern software projects are being increasingly dependent on open-source software. In addition to the frameworks, libraries, and code snippets, we can find the open-source variants of almost all the commercial software available in the market. This paper describes the benefits of using OSS (open source software) components and the security risks associated with OSS components and libraries. Further, it tries to put forward some recommendations to mitigate the security risks introduced because of OSS component usage in enterprise software products.
Introduction

Since the early days of the internet, open-source software has proven to be one of the means providing agility to the software development life cycle (SDLC) process, and delivering the finished software applications to the target customers in the shortest possible time.

Lately, with the advent of so many OSS frameworks, the share of open-source software components as a part of the finished software applications or products has gone up drastically.

According to Synopsys’s Open-Source Security and Risk Analysis (OSSRA) [1] report on the use of open-source software, most of the business verticals used open-source components and libraries as a part of their products. The open-source software usage has become so pervasive that even the connected constrained devices use the open-source component on a big scale. As per the 2020 Synopsys report, the surveying teams conducted an anonymized audit covering 1,250 commercial codebases in 17 industries, including Enterprise Software/Saas; Healthcare, Health Tech, Life Sciences, Financial Services and FinTech, and Internet and Software Infrastructure. The following table projects the usage of OSS in different domain and industries:

<table>
<thead>
<tr>
<th>Industry/Verticals</th>
<th>OSS usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet and Software Infrastructure</td>
<td>83%</td>
</tr>
<tr>
<td>Internet of Things</td>
<td>82%</td>
</tr>
<tr>
<td>EdTech</td>
<td>79%</td>
</tr>
<tr>
<td>Cybersecurity</td>
<td>78%</td>
</tr>
<tr>
<td>Manufacturing, Industrials, Robotics</td>
<td>50%</td>
</tr>
<tr>
<td>Telecommunications and Wireless</td>
<td>46%</td>
</tr>
<tr>
<td>Marketing Tech</td>
<td>78%</td>
</tr>
<tr>
<td>Energy and Clean Tech</td>
<td>72%</td>
</tr>
<tr>
<td>Virtual Reality, Gaming, Entertainment, Media</td>
<td>70%</td>
</tr>
<tr>
<td>Aerospace, Aviation, Automotive, Logistics, Transportation</td>
<td>69%</td>
</tr>
<tr>
<td>Enterprise Software/SaaS</td>
<td>68%</td>
</tr>
<tr>
<td>Computer Hardware and Semiconductors</td>
<td>68%</td>
</tr>
<tr>
<td>Internet and Mobile Apps</td>
<td>68%</td>
</tr>
<tr>
<td>Healthcare, Health-tech, and Life Sciences</td>
<td>65%</td>
</tr>
<tr>
<td>Big Data AI, BI, and ML</td>
<td>64%</td>
</tr>
<tr>
<td>Retail and Ecommerce</td>
<td>63%</td>
</tr>
</tbody>
</table>

Table 1: The Usage of OSS in Different Industries

If one looks at the table above, it can easily be deduced that open-source software has not left any industry untouched.
Why is OSS so popular?

Some software developers and decision-makers connect the use of open-source software with working smart. They go with the idiom that “don’t reinvent the wheel” and relate this sentence with reusability, but they do not pay any attention to the underlining risks associated with the usage of these open-source components. OSS reusability comes with its own security and maintenance risks.

OSS components are like a two-edged sword. The very facts because of which OSS provide benefits, turn out to be a source of risks if not used with due care and due intelligence.

Let us first list the benefits of using open-source software.

**Distributed community support:** All the big enterprise open-source solutions often have a global distributed virtual community developing and supporting the open-source products. They are driven by a shared drive to improve the solution. These communities introduce new innovative design and capabilities faster, and more effectively than the internal team developing proprietary and in-house software.

**The power of the crowd:** Many brains and hands working in tandem can deliver powerful results. They all come with different ideas and deliver quicker development and innovative solutions.

**Low TCO:** The adoption of open-source components brings the overall cost of the project down. Organizations do not need to invest in hiring engineers and programmers to develop something new that is already available as an open-source component.

**Transparency:** Open-source code means that code is available for everyone to explore and judge. The community around the open-source product keeps an eye on making open-source products and makes sure that the latest design and acceptable practices are followed for the popular open-source products. The community consists of developers having a varying degree of skills. Some of them may be sound designers and some good programmers. The code’s availability means that organizations using OSS can customize the component as per their needs. It is in complete contrast to proprietary software, which is available to a limited number of in-house developers.

**Reliability:** As mentioned in the points above, the popular open-source components have more eyes to review the code and advise the best development practices. It leads to a more reliable product. The final product tends to be extremely robust tried-and-tested code. As a matter of fact, OSS now powers about 90% of web-based applications and is being rapidly adopted across major enterprises.

**Merit-based:** With OSS, most of the time, community developers are self-motivated, and they take their decisions around the direction of the best and the most useful product possible. Organizations creating in-house products usually think of their ROI (return on investment) and
the bottom-line. It results in financial constraints, and that is not always ideal. The popular OSS projects attract the best of the breed and self-motivated software designers and engineers. So, most of the time, the decision about software construction is creative, and an agreement is reached to implement the best design after a lot of discussion in the community.

**Agility:** One may get multiple options in the OSS world when it comes to implementing the solution. The organization can do in-house security analysis and chose the best option available. The net result is a faster time to market. Agile and DevOps enable development teams to release new features multiple times a day, making software development a competitive differentiator. The demand for new and innovative software is brisk—64% of organizations report on application development backlogs (19% have more than ten applications queued).

**Freedom from Vendor Lock-in:** It is a well-known fact that commercially purchased & proprietary software increases dependency on a single vendor, and customers get locked in by the supplier or to a particular security technology. If this happens, the consuming enterprises end up staying at the vendor’s mercy and lose flexibility. Some vendors are in the business of choosing OSS and packaging the same to make the same more user friendly. The enterprise should be careful enough when they choose this kind of packaged software because this again may result into a lock-in kind of scenario. The organization should choose the software developed by open-source community and customize the same to make the same more user friendly. The enterprise should be careful enough when they choose this kind of packaged software because this again may result into a lock-in kind of scenario. The organization should choose the software developed by open-source community and customize that in-house if there is a need to do so. So far as support is concerned, there are numerous examples where enterprises can find support options available from multiple sources. There are both, free communities based, as well as, paid support options available. Linux OS is an excellent example where one does not have to pay the license fee, and multiple support options are available from different vendors.

**Freedom from developer lock-in:** Last but not the least, open-source software does not make the organizations dependent upon the proprietary developers. The organization is always on the risky side if the developers, developing the home-grown software, leave the organization. The open-source applications have long-term viability to outlast the in-house developers who come and go. Supportive communities are always there to continually implement and introduce new ideas. The communities make sure that OSS applications remain at the forefront in terms of the newer technologies, and the applications evolve with the needs of the market.
Risks with OSS?

Despite the benefits listed above, it is always risky to deploy the open-source software without doing proper due diligence. The most significant risk is the security risk. The developers supporting the open-source software have a varying degree of skills. If one happens to integrate the OSS version, which was modified by an inexperienced developer, it may contain all kinds of security flaws and other problems. In the following section, this paper is going to focus only on the security risks introduced by improper integration of OSS.

One vendor-conducted study revealed that 96% of codebases examined contained at least some open-source component, and 40% of those packages contained at least one high-risk vulnerability. In most modern DevOps development projects, most of the code used in an application is made up of open-source components—with the remaining code mainly serving as “glue” to assemble and invoke the various functions.

(Source link: https://blog.sonatype.com/gartner-the-crucial-role-of-oss-license-compliance)

Most of the developers are poorly trained in the secure development life cycle, so the inherent security vulnerabilities get introduced in the final OSS product.

Blackbuck’s (now acquired by Synopsys) Center for Open-Source Research & Innovation (COSRI) analyzed 1,071 applications audited during 2016 [2].

Per that report, IoT applications had, on average, 77% of their code base comprising open-source components. The audit showed that, on average, 677 vulnerabilities per application were detected.

These numbers make it very clear that an enterprise using IoT needs to conduct due diligence from the security perspective on their software ecosystem. A device’s software part should be subjected to a secure development life cycle. This software might be consisting of internally developed and might contain open-source components and libraries as well.

The following table lists the common vulnerabilities detected in the application using open-source software. This list is published by the US Cybersecurity and Infrastructure Security Agency (CISA) https://us-cert.cisa.gov/. This list varies from year to year.

<table>
<thead>
<tr>
<th>Top 10 CVEs found</th>
<th>Percent of apps containing</th>
<th>Vulnerability type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2016-9878</td>
<td>13.49%</td>
<td>Path traversal</td>
</tr>
<tr>
<td>CVE-2016-7103</td>
<td>12.59%</td>
<td>Cross-site scripting</td>
</tr>
<tr>
<td>CVE-2014-3623</td>
<td>12.05%</td>
<td>Path traversal</td>
</tr>
<tr>
<td>CVE-2014-0050</td>
<td>11.51%</td>
<td>Permissions, privileges, and access control</td>
</tr>
<tr>
<td>CVE-2015-6420</td>
<td>11.33%</td>
<td>Deserialization of untrusted data</td>
</tr>
<tr>
<td>CVE-2014-3578</td>
<td>11.33%</td>
<td>Path traversal</td>
</tr>
<tr>
<td>CVE-2013-6429</td>
<td>11.15%</td>
<td>Permissions, privileges, and access control</td>
</tr>
<tr>
<td>CVE-2016-6303</td>
<td>10.97%</td>
<td>Out-of-bounds write</td>
</tr>
<tr>
<td>CVE-2009-1199</td>
<td>10.97%</td>
<td>Resource management errors</td>
</tr>
<tr>
<td>CVE-2016-5007</td>
<td>10.97%</td>
<td>Permissions, privileges, and access control</td>
</tr>
</tbody>
</table>

Table 2: Common Vulnerabilities Detected with Enterprise-Level Open-Source Usage
It is interesting to note some of the vulnerabilities even existed after four years of the disclosure when one analyzed the year-on-year reports. Strut vulnerability [3] is one such vulnerability which existed in 33% of components for quite long after disclosure. This particular vulnerability resulted in the infamous Equifax breach.

The Equifax data breach occurred between May and July 2017 at the American credit bureau Equifax. Private records of 147.9 million Americans, along with 15.2 million British citizens and about 19,000 Canadian citizens were compromised in the breach, making it one of the largest cybercrimes related to identity theft. In a settlement with the United States Federal Trade Commission, Equifax offered affected users settlement funds and free credit monitoring. Source– Wikipedia [4]
Security risks

Research has showed that 78 percent of audited codebases contained at least one open-source vulnerability, of which 54 percent were high-risk ones that hackers could exploit [1]. This paper tries to list the main security risks associated with open-source software in the coming sections:

**Available and open to all:** There are, of course, advantages of having code open for the components that one wants to use or plug-in in one’s software, but again open code is open to malicious programmers as well. As the source is available to all, a malicious user can inject malicious code into the component. If one uses that software without due diligence, then the software product with that piece of code can easily be hacked by malicious actors.

**Lack of verification:** There is no assurance for the open-source for the complete security development life cycle or even for security. The proprietary software code generally goes through a proper development cycle and is tested for vulnerabilities before code gets released in production.

**Lack of support:** There is community support available, but again, community support varies and depends upon many factors like the popularity of the chosen OSS component. Some communities might actively support the product, but sometimes, immediate help is not available. There are third-party companies available, providing active support, depending upon the adoption of the component. RedHat started providing support for Linux and later, IBM acquired this company in a massive deal [5].

**Exploits are public:** Because of their open-source code’s inherent nature, the community can flag potential vulnerabilities to the open-source project managers. Even before the vulnerabilities get fixed, the vulnerabilities get reported to the NVD (National Vulnerability Database) [6] or it may be reported to any kind of publicly available vulnerability database. Hackers take advantage of the reported vulnerability and can launch attacks to exploit the vulnerability.

**Missing version control:** As OSS is publicly available and community-driven, there is no single project manager or authority to ensure quality and maintenance. There is no proper version-control mechanism available in most of the project, though things have lately improved because of some freely available open-source system like GitHub [7]. The developers may inadvertently use buggy old versions. According to statistics available on the Open Web Application Security Project (OWASP) website, using versions of open-source components with publicly disclosed vulnerabilities is one of the main reasons for web application security risks on the website. [8].

**Operational Security Risks:** If a proper system is not put in place to manage and track the OSS components, and/or if these components are not adequately tracked, there is no way that one can patch the components correctly. These unpatched components are easy targets of exploitation as the patch often addresses critical security vulnerabilities. Equifax breach is a very serious example where unpatched components led to severe consequences.
Uncompleted open-source projects: More than 15% of the open-source projects are never completed, and those are abandoned, as per the authors of the research paper published in 2019 [9]-

We carefully selected 10,932 popular GitHub projects, recovered the abandoned and surviving projects, and conducted a survey with developers that have been instrumental in the projects’ survival. We found that 315 projects (16%) were abandoned and 128 of these projects (41%) survived because of new core developers who assumed the project development.

When the components from these open-source components make way for the enterprise apps, it is like a ticking time bomb if the corporate developers are unwilling to do due diligence and not take the ownership of fixing these components. This task comes with the additional burden for the organizations to hire the developers if they do not have technical skills that have been used to develop open-source components.

As per Gartner Research Members [10]

---

**Most Significant Challenges With OSS**

<table>
<thead>
<tr>
<th>Percentage of Respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rank 1</strong></td>
</tr>
<tr>
<td>Concerns related to long-term viability of OSS projects</td>
</tr>
<tr>
<td>Concerns about security vulnerabilities, such as use of untrusted third-party components, lack of vendor accountability, etc.</td>
</tr>
<tr>
<td>Difficulty deciding on when to seek out commercial support from a vendor</td>
</tr>
<tr>
<td>Concerns related to licensing, such as license compliance, loss of intellectual property control, etc.</td>
</tr>
<tr>
<td>The need to add additional technology resources</td>
</tr>
<tr>
<td>Concerns related to indemnification and other protections typically provided by proprietary vendor contracts</td>
</tr>
<tr>
<td>Others</td>
</tr>
</tbody>
</table>

n = 74 Gartner Research Circle members

Base: Excludes “unsure”

Source: Gartner Research Circle members examining the maturity of open-source management

Q. What are the most significant challenges your organization faces/ will face while using OSS [within its IT portfolio]? Rank the top three.

Note: Values less than 3% are not shown.

ID: 441693

Figure 1: Most Significant Challenges with OSS Usage [10]
Recommendations for remediation

There is little doubt that OSS components and libraries provide agility to the SDLC process, but the organization should address the challenge and security risks posed because of the usage of these open-source components. This paper tries to offer the following recommendations for the same.

Automation:
The first thing which comes to mind is security automation. Without doubt, one can conclude that, by adhering to a secure development life cycle [11], organizations can improve their processes to find and prioritize vulnerabilities in their portfolio of applications. It is one of the reasons that DevSecOps [12] is catching up a lot. DevSecOps is based on the concept that security tools can be integrated into the continuous integration and continuous delivery part of the software development life cycle.

Go for the inventory of all OSS footprints:
Suppose one does not have the process in place to collect the inventory of all the OSS components and libraries. In that case, it will be challenging to patch and fix the vulnerabilities in those components. The open-source frameworks and libraries across the enterprise should be kept in a single source-code repository. It will not only ensure visibility, but it would also avoid the significant problem of different version usage of the same component across the organization. It will make the periodical and continual security checks easier using the SCA (software composition analysis) and static analysis tools [14]. So, we must have a single repository of OSS components to be easier to track and automate the process of security checking of these components.

Naming convention:
The enterprise needs to adhere to common vocabulary across the enterprise to understand the OSS component’s accurate deployment and collect the same inventory in its master repository. The organization can follow one of the following standards.

- Common Platform Enumeration (CPE) provides a standard, machine-readable format for encoding names of IT products and platforms (https://cpe.mitre.org/about/).
• The Software Package Data Exchange (SPDX) specification is a standard format for communicating the components, licenses, and copyrights associated with a software package. (http://spdx.org/)

**Get your developers out of the comfort zone:**

Most developers do not want to take the pain. They copy and paste from open-source websites on the internet. It brings some security issues:

• Vulnerabilities are being copied.

• It is challenging to keep track of these code snippets embedded in the final codebase. It introduces the risks in the final product, and it becomes a cumbersome task to patch the components as any tracking mechanism has not captured it.

Enterprises can avoid this issue by enforcing the policy that forbids the copying and pasting of code from the web. If the agility makes it inevitable to use open-source components, the organization should maintain a single repository, and the open-source components should be used from that monitored repository. Developers in an organization may use email to transfer the open-source components. It should be avoided, and a properly shared source code repository should be used. This repository can be an enterprise-wide source control system or a secure network location used from that monitored repository.

**Maintaining SBoM:**

It is crucial to maintain the list of all the software product components. This SBoM (Software Bill of Material) should contain all the open-source components and custom-developed components as part of it.

**According to Gartner:**

**One of the very important and the first steps to improving OSS security is to ensure that the SBoM (software bill of material) exists.**


After creating an automated SBoM, all open-source components should be mapped to an in-house maintained OSS repository. All custom code should map to the bespoke code repository. It is recommended to have one master repository of all the open-source components and custom-developed components. All reusable components should map to a single sub-repository within the master repository. Post-that centrally maintained repository should map to a reliable vulnerability database to get alerts about any vulnerability introduced in the components contained in the master repository. The two famous vulnerability databases are listed below.

| NVD (National Vulnerability Database) [16] | This is maintained by the NIST (National Institute of Standards and Technology. It is a public repository for information on all kinds of software vulnerabilities. It keeps on getting updated as the new vulnerabilities are discovered every day. This database is dependent upon CVE® (Common Vulnerabilities and Exposures) database for description and severity scoring of vulnerabilities. |
Both NVD and CVE are excellent resources, but it will be advisable to monitor other sources from proprietary vendors for patches and updates addressing security issues to be ahead of the curve. The advance reporting of vulnerabilities from vendors and other sources provides a more extensive understanding of the risks associated with the use of open source and propriety components, which helps address the vulnerabilities in an agile manner before too much damage is done.

**Embrace DevSecOps:** OSS components should be integrated as part of the DevSecOps pipeline. The organization should have some automated mechanism to check the components’ integration and their dependencies in the build procedure. The security analysis tool can be integrated as part of the build pipeline to check for vulnerable components. There should be dynamic security analysis to check against the master repository of the open-source components if any new vulnerability has been introduced in the integrated components. Subsequently, a ticket should automatically be logged in the project’s issue-tracking system to report on the subsequent steps taken to remediate the vulnerability.

**Production monitoring:** We should continuously monitor the application at runtime, and controls should be put in place to avoid any unwanted incident and subsequent damages. We can take following preventive actions.

**RASP (Runtime application self-protection):** The application should be designed for RASP (runtime application self-protection), which can act as a compensating control.

As per Wikipedia [18]

**Runtime application self-protection (RASP) is a security technology that uses runtime instrumentation to detect and block computer attacks by taking advantage of information from inside the running software.** The technology differs from perimeter-based protection such as firewalls, that can only detect and block attacks by using network information without contextual awareness.

A compensating control for a medical device has been explained on Wikia.org [19] as follow-

A safeguard or countermeasure, external to the device, employed by a user in lieu of, or in the absence of sufficient controls that were designed by a device manufacturer, and that provides supplementary or comparable cyber protection for a medical device.
How can HCL help?

HCL has developed the expertise around the usage of the industry leading SCA (software composition and analysis) tools. In addition to that, we can help our customers design the processes for smooth and risk-free integration of open-source components in their bespoke software. HCL can offer consulting and support services in the following areas:

• SBOM and inventory updates leveraging industry-leading SCA tools
• Policy development for enforcing the open-source usage rules and regulations across the organization
• DevSecOps support to build time checking and reporting of security vulnerabilities (both publicly known vulnerabilities and unknown vulnerabilities).
• License’s usage monitoring and reporting
• Run time vulnerabilities monitoring and patches

Conclusion

Security is always dynamic and requires continuous monitoring. The new software application component vulnerabilities are introduced every day even if the application was completely secure at the time of production roll-out. There should be effective monitoring of the applications in the production environment. A process should be put in place to address the threats.

It becomes imperative that all newly disclosed vulnerabilities be mapped to SBoMs maintained in the organization’s master source repository. This is achieved by creating a dynamic inventory of software components in the enterprise’s IT environment and then cross-referencing the same with a prestigious database like NVD or CVE. This cross-referencing helps determine the severity and probability of the risk because of the vulnerable components. It also helps in taking appropriate remediation steps like patching or bug fixing immediately without allowing a malicious actor to cause immense financial/reputational damage to the organization’s assets. The security solutions must have inbuilt policies across the release pipeline. This serves a dual purpose—raise the alerts on vulnerable components and stream feedback back to development and security teams continuously.
References

5. IBM closes its $34 billion acquisition of Red Hat (cnbc.com)
6. NVD - Home (nist.gov)
8. https://owasp.org/
13. Cost of a Data Breach Study - India | IBM
17. https://cve.mitre.org/
20. https://www.xfive.co/blog/5-open-source-security-risks/
Sachin Kumar
ERS-CET-Technology Office-Security COE

Sachin Kumar has more than 25 years of experience in IT industry. He works as Solution Director for HCL ERS Engineering & Technology office – Security COE. He has diverse experience in security and worked for long time in Fintech domain for a big Swiss bank in Europe. He holds bachelor in Electrical engineering. He is an alumnus of IIM Kozhikode from where he completed his executive MBA with specialization in IT and Finance. He hold multiple certificates like PMP, PMP-ACP and TOGAF.
HCL Technologies (HCL) empowers global enterprises with technology for the next decade today. HCL’s Mode 1-2-3 strategy, through its deep-domain industry expertise, customer-centricity and entrepreneurial culture of ideapreneurship™ enables businesses to transform into next-gen enterprises.

HCL offers its services and products through three lines of business - IT and Business Services (ITBS), Engineering and R&D Services (ERS), and Products & Platforms (P&P). ITBS enables global enterprises to transform their businesses through offerings in areas of Applications, Infrastructure, Digital Process Operations, and next generation digital transformation solutions. ERS offers engineering services and solutions in all aspects of product development and platform engineering while under P&P. HCL provides modernized software products to global clients for their technology and industry specific requirements. Through its cutting-edge co-innovation labs, global delivery capabilities, and broad global network, HCL delivers holistic services in various industry verticals, categorized under Financial Services, Manufacturing, Technology & Services, Telecom & Media, Retail & CPG, Life Sciences, and Healthcare and Public Services.

As a leading global technology company, HCL takes pride in its diversity, social responsibility, sustainability, and education initiatives. As of 12 months ending on June 30, 2021, HCL has a consolidated revenue of US$ 10.5 billion and its 176,000 ideapreneurs operate out of 50 countries.

For more details contact: ers.info@hcl.com
Follow us on twitter: http://twitter.com/hclers and our blog http://ers.hclblogs.com/
Visit our website: http://www.hcltech.com/engineering-services/