LTIMindtree

WHITEPAPER

Open-Source Software in Retail & CPG: Minimizing Risk through Effective Security Management

Author

Visakh Sankaranarayanan | Atish Roy
Rajesh Singh | Abhishek Kaushik

Table of **Contents**

Introduction

Retail and consumer-packaged goods businesses have evolved unprecedentedly in the 21st century due to technological advancements, overall economic growth, and geo-demographic growth. An increase in purchasing power, expansion in new markets, innovation, and focus on experiential retail are also reasons for the exponential growth. The skyrocketing was fueled further by the pandemic and its impact on consumer behavior, with technology playing a pivotal role in this paradigm shift. Retail and CPG organizations have increased their IT budget spending for the last five years.

Modern IT applications rely increasingly on reusable open-source tools, libraries, and frameworks to accelerate innovations and time-to-market for new features. While these tools are helpful, they add several risks to the Application environment and IT network. The risks can be categorized as security, licensing, and operational risks.

This whitepaper examines the risks related to using open-source applications and how this can be monitored with Black Duck. It gives a high-level view of the Black Duck architecture and how it is configured within the LTIMindtree environment. Additionally, it will provide an implementation approach for working with the customer code base; and look at how governance can be instituted with the help of DevOps and Automation.

Open-source software in applications and its benefits

Open-source software can be freely used, modified, and distributed by anyone. These are covered under open-source licenses and come with some caveats. They are developed publicly and collaboratively and supported by robust peer review processes. Millions of applications use popular, open-source frameworks like Spring, servers like Node.js, and libraries like ReactJS. As per a Linux Foundation survey of Fortune 500 companies, most of them used open-source libraries and derived cost benefits in terms of productivity. These are distributed through public repositories.

For Java-based projects which leverage build tools like Maven, Gradle, Ivy, SBT, Grape, Buildr, etc., Maven Central is the popular open-source repository.

For JavaScript-based projects that leverage package managers like npm, NuGet, and yarn, NPMJS is a popular open-source repository.

Benefits of open-source software

The Linux Foundation survey evaluated the cost benefits across different organizations. Almost 67% of the respondents agreed that the benefits exceeded the cost of using open source, **which include -**

01
Better talent retention

02
Active community support

03
Accelerated go-to-market

04
Better interoperability

05
Independence from proprietary providers

06
Lower TCO

Risks in open-source libraries and the need for open source management

Most build and package tools leverage dependency management concepts to pull all the dependent libraries from central repositories automatically. **This leads to large applications having thousands of libraries which may include -**

01 Multiple copies of the same library

02 Multiple versions of the same artifact

03 Outdated versions

Some IT organizations invest in artifact repositories that can centralize dependency management and ensure the libraries are governed systematically.

Since open-source libraries are developed publicly and collaboratively, they may be subject to many **risks**. They are categorized into three types -

01 Security **Risk**

02 Licensing **Risk**

03 Operational **Risk**

Security risk

These vulnerabilities in open-source software make the user's code base vulnerable to threats. The risks are identified by agencies, individuals, end users, or developers. Several organizations maintain a repository of vulnerabilities. The most universally recognized vulnerability repository is the National Vulnerability Database (NVD), maintained by the National Institute of Standards and Technology under the US Department of Commerce. An example of a vulnerability in NVD is NVD - CVE-2022-22978.

NVD is integrated with a Common Vulnerability Exposure (CVE) Program, a community-driven program to identify and catalog vulnerabilities. The CVE number provides a reference to the vulnerability. The Common Vulnerability Scoring System (CVSS) provides a numerical representation of the vulnerability's severity, categorizing the risk as critical, high, medium, and low.

Other popular vulnerability repositories are VulnDB, Mend, OSV Dev, and Commercial repositories like Black Duck. Organizations and communities have also rolled out various security advisories.

License risks

Most open-source software fall into a generic group of licensing families as below -

01

Reciprocal

May apply to the overall body of work typically triggered by distribution

02

Permissive

No Restrictions

03

Unknown

Requires manual review of the risks

04

Affero General Public License

Highly reciprocal, based on exposure over a network, Distribution, and SAAS models

05

Weak Reciprocal

Reciprocity is mainly triggered only if the open source is modified

The table below provides the mapping between popular open-source licenses and licensing families. Licenses can have a risk based on reciprocity.

License Family	Affero General Public License (AGPL)	Reciprocal	Weak Reciprocal	Permissive	Unknown
Examples	GNU Affero General Public License v3 or later	GNU General Public License (GPL) 2.0 or 3.0 Sun GPL with Classpath Exception v2.0	Code Project Open License 1.02 Common Development and Distribution License (CDDL) 1.0 or 1.1 Eclipse Public License GNU Lesser General Public License (LGPL) 2.1 or 3.0 Microsoft Reciprocal License Mozilla	Apache 2.0 Artistic License BSD License 2.0 (2-clause Simplified, 3-clause, New, or Revised) Do What The F*ck You Want To Public License ISC License Microsoft Public License MIT License	N/A

The table below gives an overview of license risks associated with different types of projects.

License Family	Internal Projects	External Projects	SaaS Projects	Open Source Projects
Affero General Public License (AGPL)	None	High Risk	High Risk	None
Reciprocal	None	None	High Risk	Low Risk
Weak Reciprocal	None	None	Medium Risk	Low Risk
Permissive	None	None	None	None
Unknown	High Risk	High Risk	High Risk	High Risk

Operational risks

Over time, some open-source libraries may get outdated or become inactive. This makes them susceptible to risks regarding stability, compatibility with other dependencies, and support. These risks are evaluated by some commercial scanners like Black Duck based on active community support, number of commits, versions released, etc.

Need for open-source risk management

Organizations, regardless of size, may have in-house IT services to develop their products and applications or employ vendors. In any case, they must be cognizant of the extent of using open-source software and establish governance around their maintenance and use.

While security risks can expose applications and products to external threats, licensing risks involve legal and compliance issues. Operational risks tend to affect service reliance and application stability.

An effective open-source management and risk management policy can help minimize these risks. Typically, distributed applications or Android apps developed from the ground up use open-source software. Applications built on Commercial Off-the-Shelf Software (COTS) products like E-commerce Packages, SAP, Financial Packages, CMS Packages, or SAAS solutions need not be covered under this.

Architects and project delivery managers should evaluate the usage of third-party libraries.

Role of Black Duck and LTIMindtree in open-source risk management

As we execute extensive development and maintenance projects for our customers, we may extensively leverage open-source software. It is beneficial for us and our customers to address open-source risks by instituting an open-source risk management strategy as a part of the governance policy.

Black Duck from Synopsys is a Software Composition Analysis tool that scans the code base, identifies the third-party components, and generates reports related to vulnerabilities, licensing, and operations risks.

In partnership with LTIMindtree, Black Duck provides analysis and reporting as a value-added service to its customers. The LTIMindtree delivery team works with customers to identify and automate the governance of open-source risk management using Black Duck.

We ensure that Black Duck scanning, compliance to a mutually agreed risk threshold, and SLA-based mitigation are part of the software delivery.

Alternate solutions

Some customers may prefer alternate solutions due to several factors like -

01 Reports being generated and accessible inside the LTIMindtree (external) landscape

02 Black Duck portal not being directly accessible to the customer

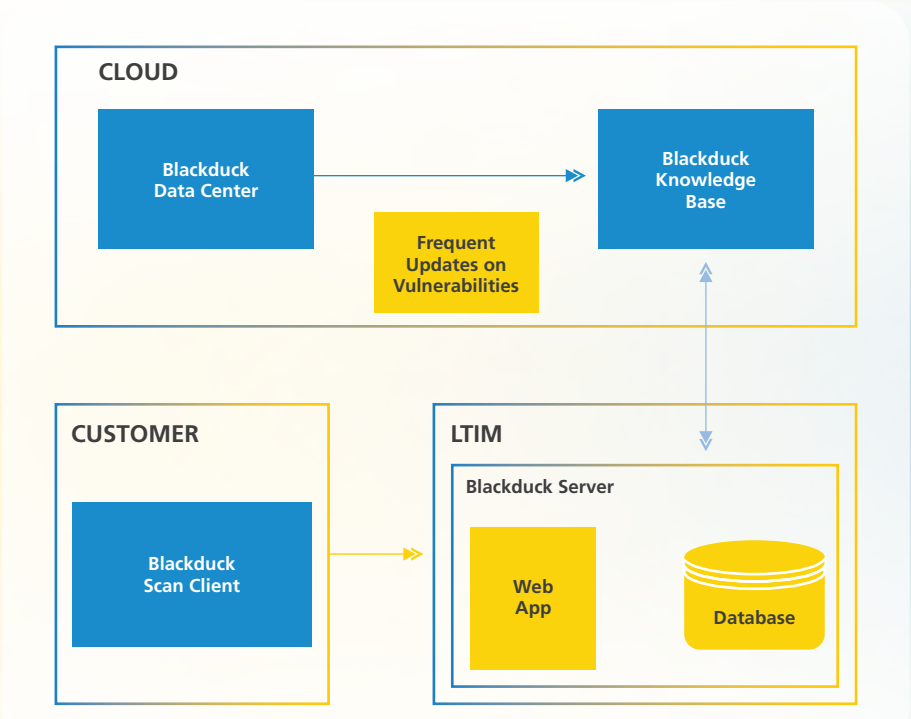
03 Already invested in artifact registries that come with these features

Customers with registries like Sonatype Nexus or JFrog can leverage add-on vulnerability scanners that come with those products. These are licensed and are available at an additional cost.

Other standalone scanners are White Source, GitLab, Fossa, DeBricked, etc. They work with vulnerability databases like Mend to report risks.

Black Duck architecture

Detailed documentation of Black Duck architecture is provided in Black Duck: A Technical Introduction. The below details are provided as a reference, and actual implementation should be done in consultation with the Black Duck team.



Black Duck has four components –

Black Duck Knowledgebase

This is the comprehensive database of open-source components and their risk scores.

Black Duck Data Center

This keeps the knowledge base up-to-date with the latest risks.

Black Duck Scan Client

This software scans the code base. It runs in standalone mode or as a pluggable component in artifact registries like Sonatype Nexus. It also scans in an offline mode so that the report can be manually uploaded to the Black Duck portal.

Black Duck Server

This is an on-premises (inside LTIMindtree) hosted web application, which stores the Customer Project information (Name of the project and artifact) and encrypted Bill of Materials of the open-source software found in the project.

The actual customer code or third-party libraries are not uploaded onto these servers. Only the software's signature is collected to be matched against the Black Duck Knowledgebase.

Implementation approach

01

Implementation considerations

02

Partially automated process due to limitations

03

Shift-Left approach for scanning

Implementation considerations

Multiple approaches can be adopted in the DevOps model based on the existing customer infrastructure and network considerations.

On a high level, there are two approaches –

01

If an enterprise uses an artifact registry for managing open-source artifacts, like Sonatype Nexus or JFrog, Black Duck plugin, integration is recommended to scan the registry and generate reports.

02

If enterprise applications directly pull dependencies from public repositories, the scanning and reporting process can be built into the Continuous Integration process of the application.

Since the customer code base is protected, there should be an alignment on the functional architecture of Black Duck and whether the customer's security team is comfortable with the data-sharing approach.

Black Duck has a policy management feature where Policy violations can be triggered in case of identification of vulnerabilities.

Some customer information, like the Name of the Customer, Project ID, Name of the Project, etc., will be available on the LTIMindtree Black Duck servers. The project name can be obfuscated if the customer is uncomfortable with this information being available on these servers.

01

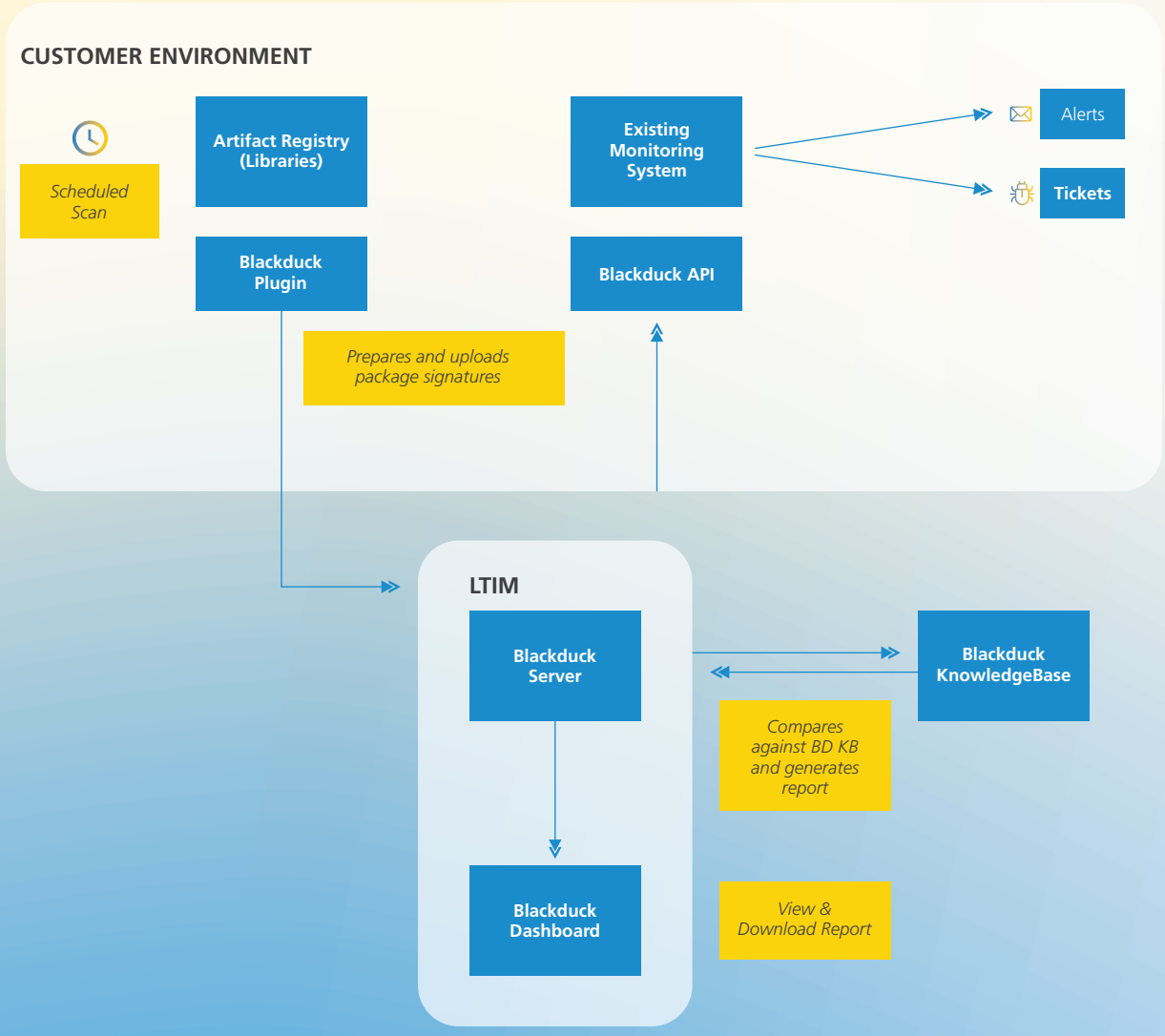
Integration with artifact registries

In case the customer enterprise has already centralized open-source governance with artifact registries, integration of scanning can be done with the artifact registry.

This will ensure that the vulnerability scanning is independent of the continuous integration process of individual projects. This approach assumes that the LTIMindtree Black Duck server is accessible online. The diagram below depicts the integration of Black Duck with artifact registries. Integration is achieved via Black Duck Artifactory Plugin. There are specific implementations for popular registries like Nexus and JFrog. Once the scanning is done, the results are uploaded to the Black Duck server.

For reporting, Black Duck Software Development Kits (SDKs) can pull risk-related data to institute alerts, create tickets, etc.

DEVOPS MODEL 1 – INTEGRATION WITH REGISTRIES



02

Integration into the CI pipeline

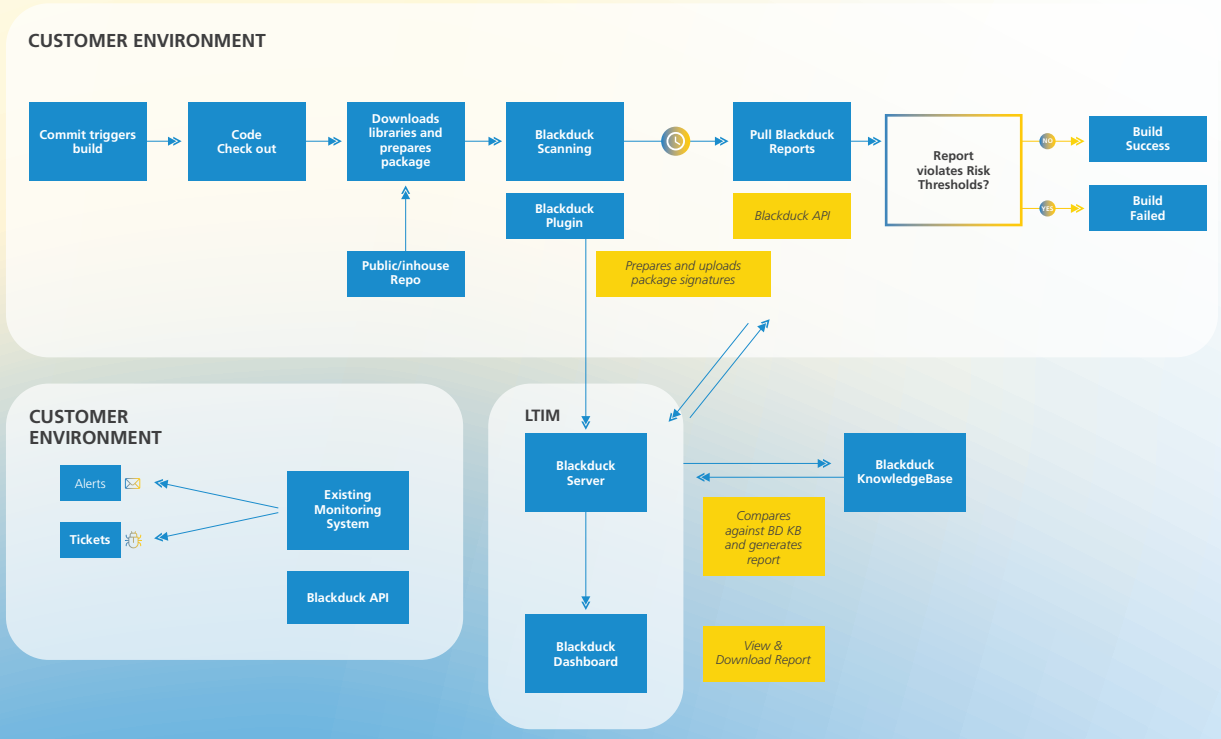
Scanning and reporting can be integrated into a typical CICD pipeline. Since a Black Duck server typically takes 3-5 minutes to process the uploaded scan results and provide a report, gating conditions based on the report processing will delay the build.

While Black Duck has plugins for popular CICD tools like Jenkins, its scanning client, Synopsys Detect, has Command Line Interface (CLI) options, which can be integrated into any build tool.

A typical CI process will get triggered on code commit, check out the code and start the build process. During this, it will resolve the dependencies and download packages from an artifact repository.

After this step, a Black Duck scan can be triggered, as shown in the diagram below.

DEVOPS MODEL 2 – INTEGRATION WITH CICD PROCESS



Partially automated process due to limitations

Due to network restrictions on Customer networks, Black Duck Plugins may not be able to directly upload scan information to Black Duck servers. Synopsys Detect CLI is another option if the customer's code base cannot be imported into the LTIMindtree network. It is the command line interface of the Black Duck scanner. **The steps to generate a report on the LTIMindtree network are mentioned below.**

- Install the scanner CLI on the CICD server in the customer's network.
- Create a CI job to check out the code and build.
- Use a batch/shell script to invoke the CLI scanner offline and create a scan dump.
- Upload the scan report to a cloud location.
- Download the scan dump manually and upload it to the Black Duck Dashboard via a browser.
- The report will be generated in a few minutes.

Shift-Left approach for scanning

As developers introduce third-party libraries into the application environment, all developer workstations should be equipped with Black Duck client scanners. The code quality review process should have a mandatory step for developers to scan the code base and report zero risks or below the threshold. Plugins are available for popular integrated development environments (IDEs) like Eclipse and Visual Studio, but scans can also be done using CLI or a visual scanner.

A real-world case study

Introduction

The customer is a large retailer in North America with an annual revenue of over a billion dollars through multiple sales channels. They had several IT Applications which leveraged open-source libraries extensively. Many of them were developed and maintained by LTIMindtree. As part of our value-added security offering, LTIMindtree demonstrated the benefits of using Black Duck to help detect risks related to third-party libraries. After an enterprise-level implementation was approved, a pilot was launched for the e-commerce program.

The context

A Black Duck scan was instituted against Nexus Repository, which hosted the libraries using a plugin and was scheduled to run every week. There was no specific alerting mechanism or service operating procedure. It was agreed at a high level that the scan reports would be monitored frequently, and the application development team would take actions to mitigate the risks.

Scanning and mitigation continued for a year. Due to multiple issues like changes in network architecture which broke the scanning process, changes in personnel, absence of SOP, and other priorities, scanning and reporting stopped. This was not identified for a long period and impacted the go-live of the re-designed e-commerce program.

Action taken

After identifying the gaps, the following steps were taken to address the situation.

01 Established a timeline of the problem statement dating back to a year

02 Conducted RCA to identify the gaps in the process

03 Created a short-term mitigation plan, including manual scanning of the code base, a quick analysis, and rolling out measures to address the identified risks

04 Devised a long-term mitigation plan, including the establishment of governance with tri-party stakeholders – Customer Security Team, LTIMindtree Corporate Security, and LTIMindtree Service Reliability Team. RACI was established, and an automation strategy was put in place.

Lessons learned

- Invest in automation upfront and ensure complete automation of scanning, reporting, alerting, and integration with CICD and IT Operations processes.

- To shift left and prevent risk before it is introduced into the enterprise, you can also institute a sandbox repository where the library can be scanned before it can be transferred to the enterprise repository. This needs a sophisticated dev ops pipeline.

- Decide on a governance model and identify the RACI matrix with proper stakeholders. While delivery teams will own the process and governance, corporate security will audit this at the account level to remove dependency on delivery teams.

- Decide on the Risk thresholds and SLAs with customers during project inception. All risks may not be immediately mitigated; hence, zero-risk SLAs should not be committed without estimating time and effort.

- Risk severity and effort for mitigation are not correlated. Mitigation often involves research, POC, major upgrades, etc., and should be carefully analyzed for impact and estimated.

- Decide on a standardized definition of risk severity with the customer. In the above case, the customer favored CVE's definition of severity over Black Duck's, which resulted in many additional libraries coming under the scope of mitigation, which was not planned earlier.

Governance model

A typical governance model includes the following stakeholders –

01	02	03
LTIM Corporate Security team	**LTIM Delivery/ SRE Team**	**Customer Security Team**

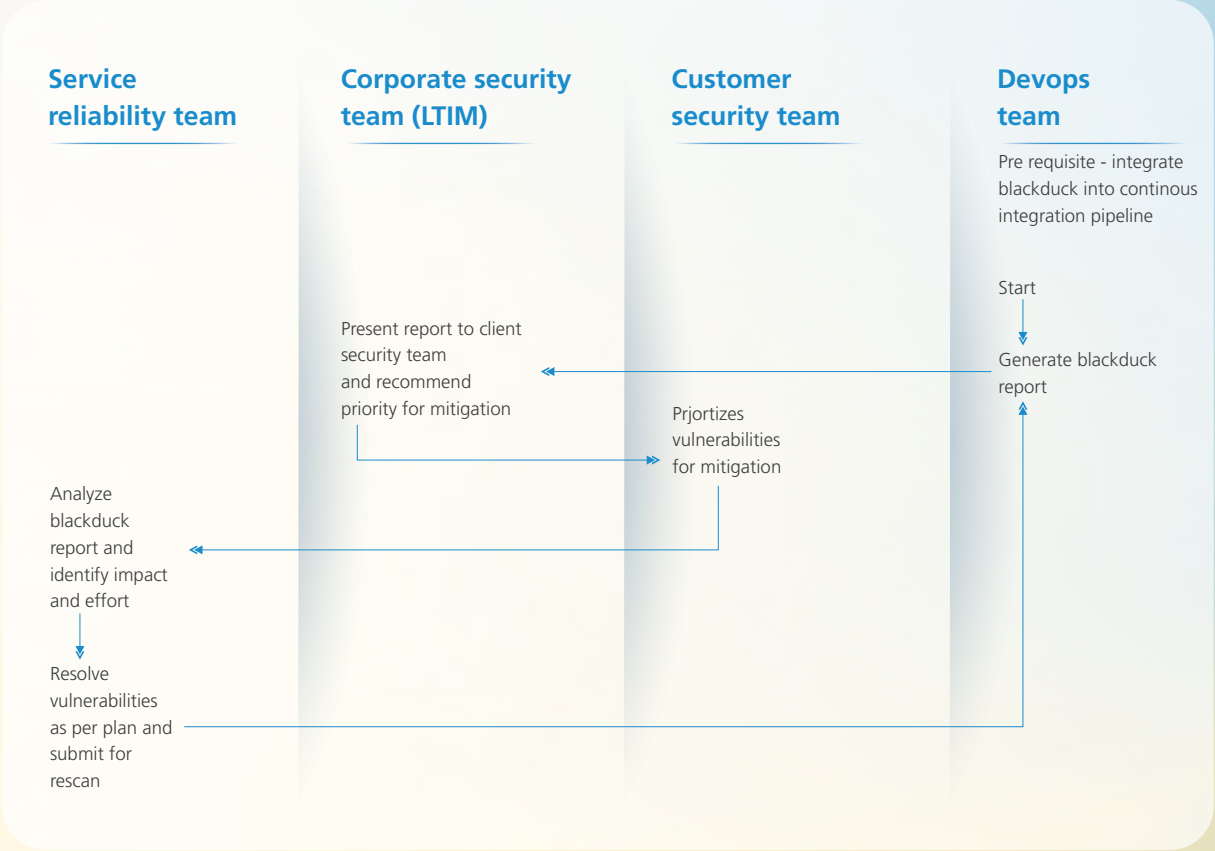
Governance should be owned by the delivery team and audited by the corporate security team at the account level, which will help to ensure standardization and people independent monitoring.

- Follow a shift-left approach and do checks at the developer and code merge stage.
- The Scanning, Reporting, and Ticketing processes should be automated as much as possible.
- The Corporate security team should collaborate with the Customer security team to explain the risks and help prioritize them.
- Customer Team hands over to the SRE team for planning and mitigation.

While licensing and security risks are under the security team's purview, Operational risks should be governed by individual delivery teams.

01 Clear RACI between SI Vendor and Client teams	02 Security report cadence to be built in Governance cadence	03 Security Scan sign-off included as a checklist as a part of the release sign-off

GOVERNANCE MODEL



Conclusion

The customer landscape and buying patterns are changing. Leveraging technology helps enhance customer experiences and supports businesses to run the enterprise more effectively. Most enterprise applications depend on open-source libraries, tools, and frameworks. Monitoring and maintaining their security, reliability, and legality is critical for the overall health of applications.

A robust Software Development Life Cycle (SDLC) process integrated with Black Duck scanning and SLA-based resolution will ensure that Applications can be released without exposure to risks due to external software and keep the customer and brand safe in digital disruption.

References

Detailed documentation of Black Duck architecture is provided in Black Duck: A Technical Introduction. The below details are provided as a reference, and actual implementation should be done in consultation with the Black Duck team.

https://www.synopsys.com/

Maven Repository https://mvnrepository.com

NPM Registry https://npmjs.com

Henry Chesbrough, "Measuring the Economic Value of Open Source: A Survey and a Preliminary Analysis," foreword by Irving Wladawsky-Berger, The Linux Foundation, March 2023
https://project.linuxfoundation.org/hubfs/LF%20Research/Measuring%20the%20Economic%20Value%20of%20Open%20Source%20-%20Report.pdf?hsLang=en

A sample NVD Vulnerability - NVD - CVE-2022-22978 (nist.gov)

CVE - CVE (mitre.org)

Black Duck Artifactory Plugin - Integrations Documentation - Confluence (atlassian.net)

Black Duck: A Technical Introduction (synopsys.com)

About Authors

Visakh Sankaranarayanan
Senior Principal – Architecture

Visakh has over 22 years of experience implementing large digital and eCommerce programs in RCG. As a certified commerce tools functional architect, he has deep expertise in eCommerce packages like HCL Commerce and commerce tools. Visakh has worked extensively on customizations for B2C and B2B domain models. He has also architected mobility and web solutions for B2B Sales channel integrations.

Atish Roy
Principal – Architecture

Atish has over 17 years of experience implementing large digital and eCommerce programs in RCG. He is a certified commerce tools developer with deep expertise in eCommerce packages like Commerce tools, HCL Commerce, and MACH architecture. He has also worked extensively on cloud architectures, DevSecOps, and CICD strategies, specializing in Google Cloud Platform.

Dr. Rajesh Singh
Associate Vice President, RCG

Rajesh is an accomplishment-driven Ph.D. from the Indian Institute of Management (Big Data & Finance) and a PGDBM from the Indian Institute of Management, K. He has over 26 years of rich experience spearheading Digital Transformation Programs in RCG & CMT sectors. His expertise lies in Digital Technologies, Cloud, AI/ML/NLP, and Data Science.

Abhishek Kaushik
Senior Director, Retail and CPG Industry

Abhishek leads the digital transformation strategic accounts in LTIMindtree. He partners with clients to keep them ahead of competition and solve business challenges. He leads large-scale accounts with cross functional, industry and technology teams to drive Innovative technology solutions for clients and drive business, revenue and people growth for organization. He has experience in Products and IT services and helped clients in BSFI, Telecom, Manufacturing, Retails & CPG industries.

LTIMindtree

LTIMindtree is a global technology consulting and digital solutions company that enables enterprises across industries to reimagine business models, accelerate innovation, and maximize growth by harnessing digital technologies. As a digital transformation partner to more than 700 clients, LTIMindtree brings extensive domain and technology expertise to help drive superior competitive differentiation, customer experiences, and business outcomes in a converging world. Powered by 82,000+ talented and entrepreneurial professionals across more than 30 countries, LTIMindtree — a Larsen & Toubro Group company — combines the industry-acclaimed strengths of erstwhile Larsen and Toubro Infotech and Mindtree in solving the most complex business challenges and delivering transformation at scale. For more information, please visit **https://www.ltimindtree.com/**

LTIMindtree Limited is a subsidiary of Larsen & Toubro Limited