The Eastasouth Journal of Information System and Computer Science
Vol. 2, No. 03, April 2025, pp. 219-245
ISSN: 3025-566X, DOI: 10.58812/esiscs.v2i03

Shift Left Security

Gaurav Malik[1], Prashasti[2]
[1] The Goldman Sachs Group, Inc. Dallas, Texas, USA
[2] Application security engineer, The New York Times, Dallas, Unites States

Article Info

Article history:

Received Apr, 2025
Revised Apr, 2025
Accepted Apr, 2025

Keywords:

Continuous Security Testing;
DevOps and Agile;
Proactive Security;
Shift Left Security;
Software Development Lifecycle (SDLC);
Vulnerability Mitigation

ABSTRACT

Shift Left Security is a proactive approach to software development that aims to integrate security measures at the beginning of the software development lifecycle (SDLC) and at the design and development phases. In the past, security for software development has been reactive, looking for vulnerabilities at the test or deployment stages. However, this method has proven ineffective in the face of the complexity of these modern software systems and the frequency of cyber-attacks. Shift Left Security instead highlights embedding security practices from the beginning to capture vulnerabilities and detect and remediate them at the very beginning before they even hit the production stage. Continuous security testing, early risk assessment, and real-time feedback loops to rectify vulnerabilities immediately, given that solving them is critical during development, are a part of the proactive model. Shift Left Security integrates security into the SDLC, ensuring the security posture of the software applications is strengthened, post-release remediation is reduced, and time to market is accelerated. The significant advantage of it is that it lets organizations open up to a conversation around security as early as possible without the risk of it becoming an issue. Shift Left Security is an area of interest that this study explores in terms of its principles, benefits, challenges, and tools. This serves as a valuable offering that offers a hands-on approach to adopting this approach by organizations to achieve more secure, resilient software products through improved development efficiency and better protection against emerging cyber threats.

Corresponding Author:

Name: Gaurav Malik
Institution: The Goldman Sachs Group, Inc. Dallas, Texas, USA
Email: Gauravv.mmallik@gmail.com

1. INTRODUCTION TO SHIFT LEFT SECURITY

Shift Left Security is a means for security in the software development lifecycle (SDLC), and it believes that proactively addressing vulnerabilities early on in SDLC instead of being reactionary is more beneficial. Standardly, security in software development has been a reactive science that works around discovering and fixing vulnerabilities at the testing time or at some point after the product is deployed. However, insufficient as this approach has turned out to be in the current world, where cyberattacks have gotten more complicated and more sophisticated over time, the software systems they face have become increasingly complex. Given this, security threats associated with software applications have been ever-present

and ever more challenging to predict with the rapid pace of development enabled by methodologies such as Agile and DevOps. The reactive security models are ineffective and require a shift towards proactive security solutions.

As the complexity of modern software systems like cloud platforms, mobile applications, and IoT devices has grown, so has the potential attack surface for cyber threats. These are very closely interconnected systems that organizations are trying to innovate quickly, and with that, the need for information security mechanisms has never been greater. As an inevitable consequence of Agile and DevOps methodology, to scale speed, developers often compromise security by putting functionality and delivery over what may or may not be security risks. In this sense, Shift Left Security is not just a best practice but a necessity. This means embedding security in the development process and not just post-development. It allows organizations to identify and solve risks before they become critical vulnerabilities. By implementing a Shift Left strategy, organizations can put security as an agenda item on the development lifecycle to avoid threats to the products and services getting exploited when moving to the production environment.

As is the case today, major drivers for adopting Shift Left Security are the growing frequency and sophistication of cyberattacks. As continually seen, attackers keep refining their tactics to take advantage of weaknesses as early as possible in the SDLC, and traditional security approaches tend to take place later but cannot prevent weaknesses early enough. The delay in identifying and fixing the security flaws results in a high remediation cost after it gets deployed. Security breaches, ranging from financial losses to data theft and damage to an organization's reputation, can be devastating. Shift Left Security works to solve these problems at the beginning of the Security development lifecycle (Shift Left) by focusing on detecting and mitigating risks early in their life cycle. Integrating security practices from the initial stages of SDLC will help organizations detect vulnerabilities in advance and eliminate the requirement for resource time wasting regarding fixes after the application deployment. Early catching issues lower the costs of patching and remediating associated security flaws.

Shift Left Security provides the base of any security optimization and gives an organization a base to start with in terms of its security posture. Before any code is written, introducing security during design allows development teams to foresee and prevent publishing this type of vulnerability. Continuous security testing like static application security testing (SAST), dynamic application security testing (DAST), and software composition analysis (SCA), combined with constant monitoring of vulnerabilities in produced runtime, ensures no security vulnerability is introduced in the production runtime. Protecting against vulnerabilities in production reduces the chance of a security exploit and helps secure security outcomes through this proactive approach. If issues are detected early, resolutions can be made faster, remediation quicker, release cycles shorter, and the product's security improved. Shift Left Security helps create a culture of security within the organization. It ensures that security doesn't just come at the end of the development life cycle but is business as usual and integrated into every step.

This study explores the principles of Shift Left Security and its benefits, tools, and challenges. Next, it will present how this approach is successfully implemented in the SDLC space, discuss some of the limitations of traditional security models, and see it in actual real-world applications through a few case studies. By learning from the successful implementation of Shift Left Security, organizations will have the necessary tools and insights to address the significant security concerns of the modern digital environment.

2. THE EVOLUTION OF SECURITY IN THE SOFTWARE DEVELOPMENT LIFECYCLE

2.1 Traditional Security Models in Software Development

In the past, software security was considered an afterthought that was 'done after the code has been written or delivered'. The test amount had to be increased and focused on the final product. Typically, allowing this to happen over time results in identifying vulnerabilities that do not occur until after the testing phase or once the product has been deployed, adding to the overall remediation costs associated with it. Security has been regarded as a second-level concern behind functional testing and performance. However, this approach was antagonized because it was reactive and could not stop security flaws from entering the production environment.

Figure 1. Secure SDL practices

2.2 The Limitations of Traditional Security Approaches

Traditional security models have several shortcomings that make them unsuitable for developing modern software. Currently, security is rarely addressed until the last phases of development or even later after deployment, and no effort is made to fix vulnerabilities once they are discovered. However, in most cases, these flaws go unnoticed until the application is deployed, and then costly patching and remediation work must be conducted. Moreover, the traditional security approach overlooks the fast pace of modern development, where multiple updates are happening frequently, which increases the chances of security issues falling through the cracks. When IT security systems, particularly software applications, are involved in increasingly complex interconnected systems, and cyber threats are becoming more dynamic, direct and reactive security measures are no longer viable [1].

2.3 Emergence of DevOps and Agile Methodologies

DevOps and Agile methodologies came and marked the advent of a paradigm in software development. However, these approaches stressed collaboration, CD, and CI and were key in making the software release possible at a much greater pace than before. The need for security to be integrated throughout the development process derives from Agile's iterative cycles and DevOps' focus on collaboration between development, operations, and security teams. Security was no longer a 'tacked on' commodity in the tail of the development cycle, something that could be added at the end. The shift saw the birth of Shift Left Security, a practice that urges the adoption of security in the earlier stages of the SDLC [2].

2.4 The Shift towards Integrating Security Early in the Development Process

As more and more of the adoption of DevOps and Agile,

enterprises began to come to terms with the need to integrate security into each launch stage. The term shift left describes the movement of security practices earlier within the SDLC such that security activities occur during the design and development rather than hậu Actions. With the CI/CD pipeline, the developers do continuous tests on their code for vulnerabilities and provide immediate inputs to address the security issue before they make it a part of the final product, all by integrating security tools like SAST, DAST, and SCA. Similarly, by taking a proactive approach, security outcomes are improved and ultimately pay off in terms of time and cost spent late in the cycle patching rather than coordinating towards the same [3].

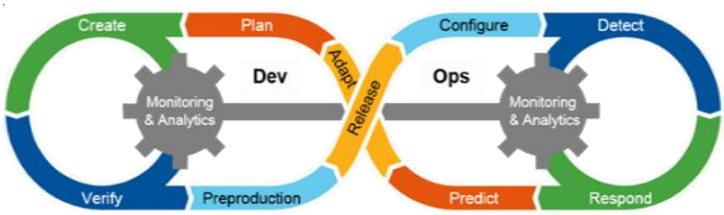

Figure 2. Understanding the Differences Between Agile & DevSecOps

3. KEY PRINCIPLES OF SHIFT LEFT SECURITY

3.1 Early Risk Assessment and Mitigation

Early Risk Assessments during the Software Development Lifecycle (SDLC) are a significant principle of Shift Left Security [4]. This proactive approach embeds security from the very beginning—something that is not done when it is added in the end. Using the early design phase to evaluate risks and threats ensures development teams will know what to watch for early before they crack their keyboards to write. This enables teams to anticipate security issues and tackle them in earlier stages that are cheaper and easier to resolve. In the early stage, threat modeling and risk assessment frameworks can be used to find possible attack vectors and determine the architecture's risks. With this approach, security becomes a part of the software design rather than a separate layer attached to the software. Therefore, organizations reduce any vulnerabilities by doing a lot of security-related work as early as possible during the development cycle, significantly reducing the chance of any crucial breach in production. Moreover, by making risk assessments early and as a core part of the SDLC, organizations can bypass the disruption from a reactive security fix typically required after the software is deployed.

3.2 Proactive Security Testing

Shift Left Security also maintains another cornerstone of proactive security testing, whereby security vulnerabilities are addressed at every step and throughout the SDLC, not in a final post-development stage. First of all, it's an approach of embedding security testing tools such as Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Software Composition Analysis (SCA) into the development pipeline. Applying these tools to CI/CD pipelines makes it possible to scan codebase on the iterations themselves as soon as the vulnerabilities come in rather than after the development cycle. This proactive approach is similar to the dynamic memory inference network model for identifying and addressing issues, as discussed, where systems continuously learn and adapt to

handle emerging challenges [5]. By continuously testing and scanning for vulnerabilities during development, organizations can identify and resolve security flaws at the earliest stages, ultimately enhancing the overall security posture of the software product.

Proactive security testing ensures that security does not happen once and is passed over—how it might get treated in manual development. Instead, it is something that is monitored and rewritten as a part of the development process. Those tools give immediate feedback on the code quality and security flaws so that developers can discover and fix them quickly. This way, the software development process does not delay the integration of such tools while ensuring the security integrity of the software. Consequently, vulnerabilities are discovered earlier, minimizing the time and resources necessary for this problematic fix and ensuring development performance. This way, continuous testing guarantees that security stays in the spotlight from Day 1 of development as issues are addressed when they emerge [6].

Figure 3. From Reactive to Proactive

3.3 Continuous Monitoring and Feedback Loops

Due to Shift Left Security, security is not isolated to one point in development; it is continuously monitored and fed back to be evaluated in a live state as the code grows [7]. Our approach here is dynamic, where the development process picks up some security constantly and provides feedback to the developer on the threats and vulnerabilities. With real-time monitoring tools, developers can be notified when a security problem arises and resolve the potential vulnerability before compromising the system. Because of this continuous feedback loop, security issues can be remediated quickly, and the possibility of their exploitation diminishes before reaching the production environment. Monitoring tools can be integrated into the development pipeline, keeping organizations proactive as the software is being developed in real time.

The development process needs to be agile but remain firm in security controls, which is essential because this feedback loop helps the development process stay effective. It also provides all development, security, and operations teams and opportunity to collaborate to quickly flag and fix security issues by the responsible stakeholders. It allows continuous risk management, identifying patterns of security incidents to review, and getting insights on what should be improved,

thus allowing the teams to anticipate future threats much more effectively. This approach is similar to the choice between eventual consistency and strong consistency in micro services, where the balance between flexibility and reliability is key to making informed decisions that improve system performance and reduce potential risks [8]. By adopting a similar mindset, development teams can enhance their security processes and proactively address vulnerabilities as they arise.

3.4 Collaboration between Developers, Security Teams, and Operations Teams

Shift Left Security advocates for collaboration between the developers and the security teams to ensure that security is a shared responsibility, not a siloed action. In the traditional approach to development, security was further separated from the rest of the process, where secure testers and auditors performed testing and audits after most of the code was already written [9]. These missed vulnerabilities, or late-stage discovery and remediation efforts, would often be costly. However, with Shift Left Security, all SDLC stakeholders, holder's developers, developers, security experts, and ops staff are jointly embedded with security measures throughout each development stage. Not only do security concerns transcend from design and coding through to deployment and operations, but the collusion of the two team's means that any issues that arise during any step of the cycle are eliminated quickly [10]. This allows developers to become more aware of the possibilities of vulnerabilities and practice writing secure code from the beginning. At the same time, security teams can provide advice and expertise in practice. Consequently, operations teams are crucial in making sure that secure deployment processes are enforced and that production systems are secure. Shift Left Security's predilection for a culture of continuous communication and shared responsibility between teams makes it easy to instantly spot and fix security risks in a smoother and more organized way. In the long run, this helps to bring more secure software because it fosters the collective work of all the teams involved in the SDLC [11].

Figure 4. Shift-Left Security

4. BENEFITS OF SHIFT LEFT SECURITY

Table 1. Benefits of Shift Left Security

Benefit	Description
Reduced Vulnerability Exposure	Early detection of vulnerabilities minimizes exposure in production.
Faster Detection & Remediation	Security issues are caught early, leading to quicker fixes and shorter release cycles.

Benefit	Description
Cost Efficiency	Identifying issues early reduces the costs of post-release remediation.
Enhanced Developer Responsibility	Developers take on more security responsibility, fostering a culture of security.
Improved Security Posture	Proactive security reduces vulnerabilities, creating more secure software.

4.1 Reduced Vulnerability Exposure

The listened vulnerability exposure is one of the most significant advantages of Shift Left Security [12]. As it addresses security concerns at the early stage of the development cycle, the organization can find and fix vulnerabilities before they become embedded into codebase, thereby reducing the likelihood of them being compromised in the online environment. Proactive testing and continuous monitoring help early risk assessments find security flaws in the design and production stages and minimize the possibility of leaving security flaws upon building the final product. The early identification of vulnerabilities provides a shorter window for attackers to exploit weaknesses. This proactive approach assures that security issues are resolved before they have any bearings on the production environment, which in turn means the software shipped has been made vulnerability exposure reduces the risk profile of the organization at large and the cost of such costly security breaches with subsequent damage to reputation and trust. Organizations reduce their exposure to vulnerabilities early, thus allowing the quality and security of their software to improve, and they can deliver more reliable products to their users.

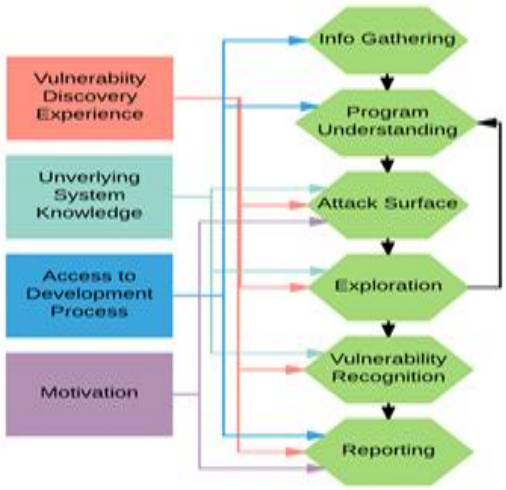

Figure 5. Vulnerability-finding process and influencing factors

4.2 Faster Detection and Remediation of Vulnerabilities

However, with Shift Left Security, it is possible to shorten the detection and remediation of vulnerabilities, ultimately shortening the release cycles and improving the development process. Using these early stages of development to perform security testing helps find vulnerabilities in codebase when they occur instead of when they are too late, at the end of the development cycle, or after deployment. This early detection keeps delays to a minimum as it allows developers to fix the issues in real time and saves a lot of time spent fixing things post-release, either in patches or hotfixes. Securing is a continuous problem when integrated into the CI/CD pipeline, which requires constant monitoring

and proactive security testing. Speeding up problems, fixing processes, and reducing chances of overlooking security issues. _Duplicates help organizations meet deadlines faster and with more integrity by allowing them to fix vulnerabilities faster._ In the end, faster detection and resolution of security flaws help simplify the development process and the time to market for delivering secure products [13].

4.3 Enhanced Developer Responsibility and Security Awareness

Shift Left Security promotes security as a cultural phenomenon, allowing developers to take more responsibility for code security, thus being more and more needed. Typically, in the traditional development models, security was considered the job of a professional security team or testing team alone. But by pushing security into the earlier stages of the development lifecycle, ShiftLeft Security puts the responsibility of writing secure code into the developers ' hands and rushes developers to write secure code from the onset. It also encourages developers to follow the best coding security practices like proper input validation, secure data handling, and prevention of probable coding vulnerabilities like SQL injection or cross-site scripting (XSS). With security becoming a shared responsibility throughout the different stages of development, developers learn how to look for and eliminate security risks throughout the coding process, providing them with the knowledge and tools to do so without security professionals. In addition to reducing the risk of introducing vulnerabilities, this improved responsibility also makes the development team more aware of security in general, leading to a more security-conscious culture. This is how Shift Left Security takes control of security by giving developers the authority to get the bugs fixed as early as possible in the process — and that shifts the development of more secure software products.

4.4 Cost Efficiency in the Long Run

While the initial investment of tools, training, and process changes will be needed to adopt Shift Left Security, it will ultimately save a great deal in the long term [14]. Once the software is deployed, identifying and eliminating vulnerabilities in the software development process is much less expensive than remedying them afterward. Worst of all, security vulnerabilities that result in data breaches or even evidence of tampering can all cost significant amounts in post-release patches, hotfixes, and emergency remediation costs. On the contrary, if organizations take security testing and risk assessments into the early stages — such as during the architecture and design stage — the early detection and resolutions of issues will ensure no critical problems arise, and the fixes and potential resulting outlay of cash will be less costly compared to a security breach. In addition, early identification of such vulnerabilities helps reduce the dependency on rework, thus reducing development time and overhead costs. Implementing Shift Left Security may involve an up-front investment; however, long-term savings in reduced remediation cost, fewer security breaches, and faster time-to-market outweigh the cost, so it is a cost-effective strategy. Shift Left Security reduces the number of vulnerabilities in production, reducing the resulting high costs incurred due to post-release fixes/security incidents.

4.5 Improved Overall Security Posture

Shift Left Security embeds security into the entire development

lifecycle, leading to a more assertive, more resilient security posture [15]. Suppose organizations prioritize security early in the process and make security a fundamental aspect of the SDLC. In that case, they will be able to catch vulnerabilities in each stage of the development. It means an active approach towards security testing, monitoring, and collaboration to keep security at the core of the development process. This means that these applications and systems become more secure over time because they are part of an ingrained culture of development around security. Cyberattacks won't succeed as much due to early detection and resolution of vulnerabilities. The overall security covered at all ends continues, and they will rule out any damage it can cause before it occurs. This allows organizations to keep a fingernail-length distance from security posture and prove that they deliver secure and trusted products to their end users. Shift Left Security is not an afterthought or an end play but a continuous and integral part of the development process, strengthening the overall security of the organization's products.

Figure 6. ASPM (Application Security Posture Management)

5. IMPLEMENTING SHIFT LEFT SECURITY IN ORGANIZATION

Table 2. Common Security Tools Used in Shift Left Security

Tool Type	Description	Example Tools
Static Application Security Testing (SAST)	Analyzes source code to identify vulnerabilities before execution.	SonarQube, Checkmarx
Dynamic Application Security Testing (DAST)	Tests running applications for real-time vulnerabilities.	OWASP ZAP, Acunetix
Software Composition Analysis (SCA)	Scans third-party libraries and open-source components for vulnerabilities.	Black Duck, Snyk
CI/CD Integration	Automates security testing in the development pipeline.	Jenkins, GitLab CI, CircleCI

5.1 Identifying Security Gaps in Your Current Development Process

Organizations must also conduct an exhaustive audit of their present development process before implementing Shift Left Security to identify any gaps or vulnerabilities in security [16], It includes assessing the current workflows and tools used for development and practices to

ascertain where security may be nonexistent or insufficient. However, a thorough assessment should look into how security gets integrated (or dissolved) in each software development life cycle (SDLC) phase, from design to coding, testing, and deployment. This audit usually reviews past security incidents, finds patterns of overlooked vulnerabilities, and determines areas of security checks that are either rarely performed or never performed. For instance, security assessments may be done only at the final test stage or post-deployment, resulting in significant gaps in the majority of the earlier phases of development. It helps organizations identify which areas have gaps and, prior, criticize which shift left preshift-lifted be enacted first. In early-stage code review or while threat modeling, identifying these gaps gives a target for proactive security implementation into the CI/CD pipeline. Furthermore, an audit process that involves all stakeholders (development, security, and operations team) on security makes it a company responsibility. The idea is to create a baseline where security is constantly in the development cycle rather than being added as an afterthought.

5.2. Integrating Security into the CI/CD Pipeline

Necessary for Shift Left Security is integrating security into the Continuous Integration and Continuous Delivery (CI/CD) pipeline [17]. This integration enables its practices to be continuously automated within the death development process instead of isolated stages or manual checks at the end of the e-cycle. To allow development teams to detect vulnerabilities in real time as code is written and incorporated into the CI/CD pipeline, software security can be automated via Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and other controls directly in the CI/CD pipeline. By doing this, security testing is never an afterthought; it is always an integral part of every code change, and immediate feedback is provided to the team to allow the issue to be fixed as it is encountered. Automating security checks at every commit or push will enable teams to spot vulnerabilities as they are developed and not when they become hardwired into the system. To illustrate, an automated scan could check for such items as new code is pushed to a repository - when built code and data are transplanted into the stack for production. In CI/CD integration, security gets incorporated into the CI/CD lifecycle as an embedded part of the process, and developers cease functioning, believing that security is an extra burden that must be carried into their workflow. As a result, it speeds up software delivery while lowering the chances that security vulnerabilities will be discovered too late in the development process.

5.3. Tools and Technologies for Shift Left Security

To have Shift Left Security effective, organizations should have several tools and technologies to combine security measures throughout the SDLC. This technique helps automate and streamline the security testing process and makes security an essential part of the development workflow. Static Application Security Testing (SAST) tools, for example, would let developers scan the source code for security weaknesses like code injection or insecure data dealing with practices without executing the code. These tools detect flaws in the early stages of development, thus

reducing the chances of flaws reaching the final product. On the other hand, dynamic Application Security Test (DAST) tools verify that running applications for vulnerabilities are used during a running environment, such as cross-site scripting (XSS) or SQL injection. It allows this security to be tested in the static code and running applications and gets a broad picture of possible vulnerabilities. Such tools are also needed because many modern applications rely on third-party libraries or open-source components. These tools use these dependencies and scan for known vulnerabilities that help to facilitate the solving of security risk dependencies. Moreover, CI tools such as Jenkins, CircleCI, or GitLab allow integration of these security tools in the development pipeline and automatically perform continuous security validation. Apart from detecting vulnerabilities, these tools keep the organization compliant with regulatory requirements, enforcing industry standards, and strengthening the rest of the organization's security posture [18].

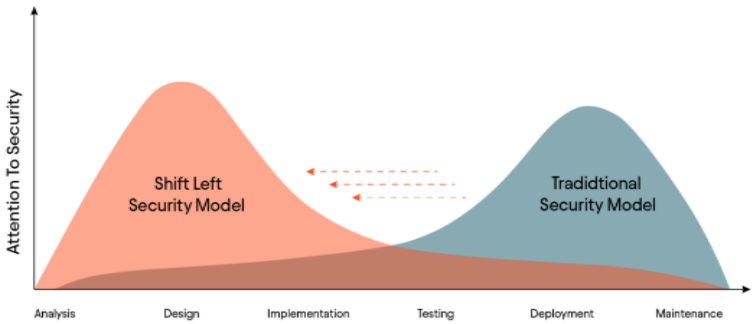

Figure 7. Shift Left Security: Transforming the Software Development Lifecycle

5.4. Building a Culture of Security in Development Teams

For Shift Left Security to truly work, organizations must create a security culture among their development teams [19]. This is because the security no longer belongs to the security professionals alone. Still, it has become a shared responsibility of every other personnel—operations staff or the developers. The way to start developing a culture of security is by training developers in secure coding practices and persuading developers to behave security aware all through development. Security should be a subject of conversation in team meetings, and developers should consider security an ongoing issue rather than something completed and done before shipping. Coverage includes ensuring secure coding by following a good user mindset, such as input validation, proper secure authentication, and error handling, as part of the daily routine. It also involves developing a security-first mindset by enabling developers to actively identify vulnerabilities before committing them. Security teams also need to work hand in hand with developers, being able to give insights and advice regarding how to minimize the threat factors at the design and coding stages. Such a security culture not only cuts down the risk of vulnerabilities slipping through the space between the cracks but also encourages collaboration between traditionally separated teams, leading to a more secure and efficient development cycle. Security involves embedding it in the organization's values and making it part of the routine [20]. All organization personnel should share

this responsibility, not as something to be outsourced [21].

5.5. Training and Upskilling Developers on Secure Coding Practices

Shift Left Security is heavily predicated on the approach's success in terms of training and upskilling the developers in secure coding practices. The first line of defense is the developers, who are responsible for writing secure code and, therefore, are a key part of the overall security of the application. Secure coding training should cover input validation, proper dealing with proprietary info, preventing SQL injection, and securing authentication configuration. The development community should also consider this training to understand common security weaknesses like cross-site scripting (XSS) and buffer overflows so that the developers can recognize and protect against them before they are put into the code base. In addition, they should also educate developers on how to use security tools like SAST, DAST, and SCA so that they are aware of and familiar with these tools from the time of development. Upskilling is also extremely important as threats still evolve and hackers find ways to take advantage of developers' existing vulnerabilities and best practices. Now, organizations need to invest in educating developers through workshops and certifications, for example, external and internal training sessions, to keep developers more secure and up to date with recent security threats and to teach them how to protect themselves from these dangers. Organizations prioritizing developer training and upskilling embed that security in every development process meeting, making it much less likely for vulnerabilities to enter. Thus, their software products tend to be much more secure.

6. TOOLS AND TECHNOLOGIES FOR SHIFT LEFT SECURITY

6.1. Static Application Security Testing (SAST) Tools

One type of Static Application Security Testing (SAST) tool can analyze source code for security vulnerabilities without the developer having to execute the program to find out [22]. These tools take place in the codebase at an early stage of development and discover problems like insecure coding practices like hardcoded credentials, input validation errors, buffer overflow...etc. The main advantage of SAST is that the vulnerabilities can be detected before the code is executed or deployed, allowing them to get fixed early in the SDLC. Being incorporated into the development environment, SAST tools are very well suited for the design and programming phases of the development life cycle and readily point out potential issues to the developers. Therefore, organizations can ensure that testing for security happens continuously throughout the development lifecycle by integrating SAST tools into the CI / CD pipeline, where security testing is never optional and can be part of the automated process that happens whenever the code is changed. By doing proactive testing, vulnerabilities are caught before they become embedded in codebase, thus reducing the risk of later security breaches. In addition, SAST tools deliver a wide range of code analyses, allowing them to look deep into the source code and present an exact and prioritizable list of vulnerabilities that need to be fixed by developers [23].

6.2. Dynamic Application Security Testing (DAST) Tools

Dynamic Application Security Testing (DAST) tools are intended to test running applications for vulnerabilities that may be exploitable in a live environment. Unlike SAST, DAST tools test the application at runtime to find runtime vulnerabilities like cross-site scripting (XSS), SQL injection, insecure API calls, and other runtime security flaws. DAST tools understand how an application works exactly how a user would and simulate the attacks in real time to detect the vulnerabilities that even regular static code analysis may miss. Without these tools, the security is purely theoretical and cannot be validated in the actual running environment; this gives us an idea of what an application does under attack. This is similar to the method used in the detection and classification of arrhythmia, where real-time monitoring and analysis are applied to detect anomalies that might not be obvious in a static context, as discussed by [24]. By employing DAST tools, organizations can gain a clearer understanding of their application's vulnerabilities and address them before they are exploited in the production environment. DAST tools identify flaws that can only be found when the application is running and hence cannot be identified by static analysis. Integrating DAST into the CI/CD pipeline allows organizations to continuously monitor their applications for vulnerabilities and patch things before they negatively impact the production systems. Using DAST enables the security of an application to be tested against real-world attack scenarios.

6.3. Software Composition Analysis (SCA) Tools

Software Composition Analysis (SCA) tools are applied to scan for vulnerabilities within the third-party libraries and open-source components used to build an application. The more modern software development relies on the code reused from libraries and frameworks, open source, the more it becomes essential to mitigate the threats upon these external dependencies in terms of security. At compile time, SCA tools scan the dependencies within the codebase to look for known vulnerabilities, licensing issues, and old components, which could prove risky to the security of the entire application. Organizations can integrate SCA tools into their development process and verify that their software does not unknowingly bring in security vulnerabilities from other sources. SCA tools give such precise insight into which libraries or packages are vulnerable that there is no reason to waste time: Development teams can quickly update or replace them with secure alternatives. This proactive approach to managing dependencies prevents the application from integrating compromised components and keeps the application secure during its lifecycle. Therefore, it can be stated confidently that SCA tools are one of the essential components of any Shift Left Security strategy since their use enables organizations to keep their software both secure and compliant, safely limiting the risk of third-party vulnerabilities [25].

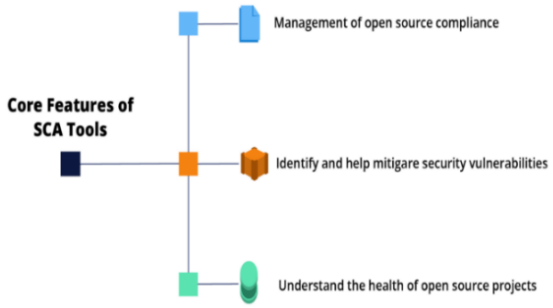

Figure 8. Three core features of an SCA tool

6.4. Continuous Integration Tools and Platforms

CI tools like Jenkins, GitLab, Circles, Travis CI, and others help to enable security practices throughout the development lifecycle. CI tools build and test the code automatically and secure it even before the release, allowing developers to do security testing at every stage of the development. Organizations can use CI tools to guarantee that these security checks will also be automatically run when new code is integrated into codebase. With this continuous, automated process, developers can catch vulnerabilities early and demonstrate on an ongoing basis that their security is being continually validated as part of the development lifecycle. With SAST, DAST, and SCA security testing tools, employing CI tools means a seamless workflow to integrate security testing tools into the development process, with no break in the development cycle and live detection of any vulnerabilities now. Moreover, CI tools also make the deployment process smooth to accelerate the production of secure code. With CI tools, security is an embedded continuous part of the development process that reduces the chances of vulnerabilities being introduced at the later stages of the lifecycle [26].

6.5. Example Tools and Platforms

Several tools and platforms are available to support the implementation of Shift Left Security, with capabilities that span continuous security testing and validation. For instance, in real-time, widely used SAST tools such as SonarQube can detect security vulnerabilities, bugs, and code quality issues. It doesn't struggle to fit in CI/CD pipelines and can provide continuous analysis as the code is written. One more popular SAST tool is Checkmarks, which automates security tests and offers extensive insights into security flaws in codebase. Also, GitLab has security integrated into it, provides a comprehensive system to manage the entire development lifecycle, and has automated security testing features as part of its CI/CD capabilities. Alongside Jenkins and GitHub Actions, these tools make it simpler for organizations to integrate security into the SDLC with a mix of ease, allowing ongoing security validation in the development cycle. If businesses employ those tools, they can remain on top of security engagement and detect vulnerabilities early in development.

7. REAL-WORLD SUCCESSFUL CASE STUDY OF SHIFT LEFT SECURITY

Table 3. Real-World Benefits of Shift Left Security at GitLab

Key Metric	Pre-Shift Left Security	Post-Shift Left Security	Percentage Improvement
Security-Related Defects	High	Reduced	40% decrease
Emergency Patches & Hotfixes	Frequent	Reduced	30% decrease
Time to Release	Slow	Faster	Accelerated (X%)
Security Confidence	Low	High	Increased

7.1. Case Study Overview: Company GitLab's J Journey with Shift Left Security

As the provider of the DevOps platform, GitLab had to deal with serious security issues in its software offerings' scaling. Given an ever-growing number of security breaches where vulnerabilities were disclosed in GitLab's products, it was time to rethink security. Security was traditionally considered a reactive process rather than a part of the development cycle, being addressed only at the end of the development lifecycle and leading to the lateness of vulnerability detection and the corresponding costly post-release fixes. These issues made the company shift security as a part of its broader approach to better its software security across the board and its product portfolio. The idea was to build security into the development lifecycle in the early stages of development, design, and coding, not only in wrap-up patches and final stages of testing. To mitigate security risks proactively while vulnerabilities have yet to affect customers, GitLab decided to move left by raising the bar for security. This was key to increasing the security posture of the products themselves, reducing risk, and improving their overall software quality before deployments [27].

7.2. Challenges Faced Before Implementing Shift Left Security

If unfamiliar with GitLab, here is how they faced some challenges that showed the need for a Shift Left Security before implementing it. Lack of discovery of security vulnerabilities as late as final testing or after deployment was one of the significant problems. As a result, costly and time-consuming remediation efforts, such as emergency patches, hotfixes, and, in some cases, rollbacks of product updates, were needed to get the product back to a stable state. This also proved reactive, and with a large backlog of unresolved vulnerabilities, the product delivery process slowed, and customer satisfaction suffered. In addition, security breaches had started to of the company's reputation, and customers had begun complaining about the safety of their data. This repetition of challenges indicated that security must stop being an afterthought and be built throughout the development lifecycle to create secure software products. However, the company realized that a proactive and more integrated approach toward security was needed to stay in stride with fast development cycles [28].

7.3. How the Shift Left Approach Was Introduced and Integrated

GitLab's gradual shift to Shift Left Security started by embedding Static Application

Security Testing (SAST) and Dynamic Application Security Testing (DAST) tools in the Continuous Integration and Continuous Deployment (CI/CD) pipeline. With its help, codebase could be scanned continuously as it grew, detecting vulnerabilities in real time. Adopting the approach of embedding security tools intrinsically within the development process meant that security was no longer a solitary task when there were deployment teams, DevOps, and product managers to interact with. It became a part of the job of everyone working on the issue: it was the priority, and security had to be an integral part of it; otherwise, there would be no passing of the application into production. Developers were trained in secure coding practices and turned to be responsible for writing secure code from the outset of the project. In addition, security professionals collaborated with developers to resolve security issues early in the design, coding, and process during testing. With the collaboration of developers, security teams, and operations, this approach helped shift the company's security practices into a culture where security was part of every step in the process of SDLC. Adding security into the development lifecycle enabled GitLab to find and fix vulnerabilities earlier, streamlining remediation time and releasing faster and more securely.

7.4. The Results: Achievements and Outcomes

Shift Left Security was adopted because it resulted in massive improvements to Gitlab's security posture [29]. A good outcome was a 40% reduction in security-related defects and fewer emergency patches and hotfixes post-release. By identifying and addressing security issues early on in the SDLC, GitLab was able to increase its speed of release and have more confidence in its product launches. The company could concentrate more on innovation and meeting the demands of their clients since the time taken up by post-deployment remediation experienced a 30% decrease. Security, therefore, was deeply embedded within the development process, and risks were continuously improved and proactively managed. Instead, real-time security testing tool integrations were used to input the CI/CD pipeline, providing immediate feedback for GitLab to react quickly to avoid delays. Thus, GitLab was able to develop more secure software in time and satisfy customers. In addition, the successful integration of security into the development process also improved communication between teams, thus ensuring security concerns have been consistently addressed across the organization.

7.5. Key Takeaways for Other Organizations

From GitLab's perspective, Shift Left Security offers their experience that others can learn from if they want to implement this approach. The main point is that security needs to be integrated earlier in the SDLC, and security items should be moved to the code design and development parts of the cycle and not left as an afterthought at the end of the cycle. Security embedded throughout development will eliminate vulnerabilities and solve risks in advance. Moreover, cultivating a culture of security among development teams is imperative to make developers hold it in place throughout the process. Continuous security testing running as part of a more streamlined CI/CD system feeds real-time vulnerability

detection and helps teams catch up on the necessary actions toward security. In addition, development, security, and operations parties must work together to set up a unified security framework and ensure that everyone wins over the common objective of producing secure software. Improving the organization's security posture by reducing vulnerabilities and delivering safe, high-quality software will require organizations practicing Shift Left security to take to heart these practices.

8. COMMON CHALLENGES IN IMPLEMENTING SHIFT LEFT SECURITY

Table 4. Common Challenges in Implementing Shift Left Security

Challenge	Description
Resistance to Change	Legacy systems and developers' resistance to adopting new workflows.
Balancing Speed with Security	Concern that embedding security may slow down development cycles.
Skill Gaps and Expertise	Lack of trained developers and security professionals familiar with Shift Left Security tools and practices.
Tool Overload	Managing and integrating multiple security tools can become complex.
Ensuring Continuous Integration Without Compromising Security	Ensuring security is integrated without slowing down development cycles.

8.1. Resistance to Change and Legacy Systems

A significant challenge organizations face when pursuing Shift Left Security is change resistance, especially when dealing with legacy systems not created with security in mind [30]. A legacy system implies a complex web of old code, architectures, and technologies that often cannot support modern security policies. Upgrading or changing these systems to align with Shift Left principles can be expensive and time-consuming, requiring large amounts of resources, tools, and expertise. In addition, many developers are used to classical development methods. They are not inclined to use new workflows or tools for security, especially if they understand new techniques or tools will add complexity and slow the development process. However, to overcome this resistance, strong leadership and a clear vision regarding how Shift Left Security can help the organization in the long run. To encourage organizations to adopt proactive security, they must explain the benefits to their customers to help them see that it can reduce their risks, increase their product quality, and save time and money by identifying and resolving the vulnerabilities earlier in their SDLC. When it comes to Shift Left Security, there are many more stakeholders to engage, from leadership to developers, and showing the tangible benefit of Shift Left Security will help foster their buy-in and make it easier to adopt Shift Left Security [31].

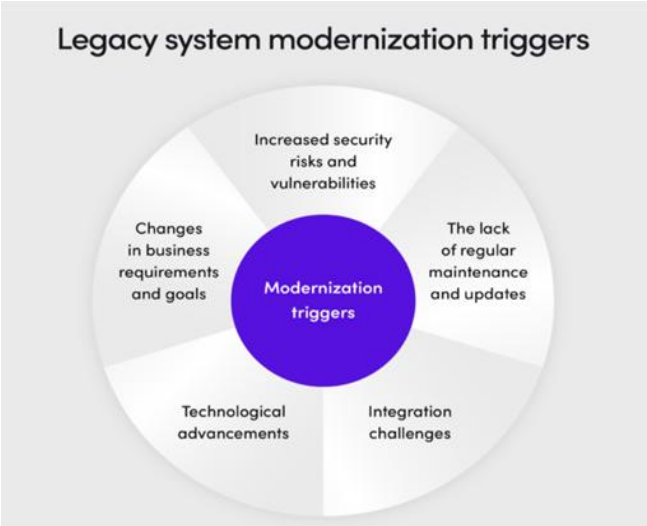

Figure 9. Challenges of Legacy Systems and How to Address Them

8.2. Balancing Speed with Security

In today's high-speed development environment, speed in releasing the product entails the engagement of security at the risk of the latter not being prioritized. Sometimes, Shift Left Security can even slow developers' progress, mainly when used in the CI/CD process. If development teams are unfamiliar with the new tools or additional workload, they may even be concerned that implementing security testing into the pipeline would result in delays. This said, if security testing is done right, it should speed development up and not slow down, thanks to catching issues early before they require expensive rework. They help to ensure that any vulnerabilities are caught and addressed in real-time without significant delays using automated security tests that can be integrated into the CI/CD pipeline. They need to strive for the automation of all they can so that it almost becomes part of what the organization does naturally. It, therefore, makes sense for organizations to strike a balance between security and speed by embedding security as part of their development process rather than treating it as a separate, final-stage task. Doing this will allow them to release software and features faster without compromising quality and safety.

8.3. Overcoming Skill Gaps and Lack of Expertise

To implement Shift Left Security, one must be skilled in secure coding practices, security testing tools, and the latest threat trends [32]. Up-skilling organization developers and security teams is complex for many organizations, and it is difficult to address these gaps. Developers may not be well aware of the latest security vulnerabilities and knowledgeable about the best secure coding practices, nor know how to utilize security testing tools properly. Security teams may also have to be trained in the new development practices, such as Agile and DevOps, to work in concert with the development teams. Continuous training and education are key to ensuring the teams have the security know-how to shift left security successfully. Therefore, organizations must invest in making their team resources, such as workshops, certifications, and internal training sessions, available to keep their teams updated on the newest security threats and best practices. Continuous learning and improvement must be built as a

culture to match the evolving threats and maintain security as a top priority in the development process.

8.4. Managing Tool Overload and Complexity

Shift Left Security is another challenge that organizations face when they implement it. They have many security tools to manage and use, but each has its purpose. Having the appropriate tools is essential for secure testing. However, enterprises often have problems integrating and managing multiple tools properly. Two primary issues may arise after this— tool overload and overreliance on external tools. Too many tools can cause confusion, inefficiencies, and missing vulnerabilities, or the teams may not be using the right tool at the right time or correctly interpreting the results. To address this problem, organizations must identify the most powerful tool to aid their needs and integrate them into a streamlined and consistent workflow. The selection of such tools should complement each other and be compatible with the existing development environments to make the security testing process as effective and smooth as possible. Reducing tool overload and making security testing more manageable can be helped by consolidating tools and using platforms that supply integrated security testing across the SDLC.

8.5. Ensuring Continuous Integration without Compromising Security

One of the key challenges of implementing Shift Left Security is ensuring that security can always be integrated into the development workflow without slowing the development cycle. Ultimately, security testing can be an added burden in fast-paced development environments, where the security tests or findings require manual intervention. The answer is, however, to use automation to enable continuous security validation without manual intervention. Organizations can automate security checks to run every time new code is integrated by incorporating security testing into the CI/CD pipeline, which would allow them to detect the vulnerabilities as early as possible and promptly fix them. Also, automation helps monitor security risks continuously so that they are not compromised at the cost of speed during development. Additionally, tools such as continuous integration that help automatically run security tests and identify security vulnerabilities will add less burden on the domains, equating the need for manual security audits and allowing developers to concentrate on their job of coding with security in mind throughout the entire development lifecycle. Security must be seamlessly integrated into the CI/CD pipeline so that security can be ensured in all stages of development without slowing down the pace of innovation [33].

9. BEST PRACTICES FOR SHIFT LEFT SECURITY

Table 5. Best Practices for Shift Left Security

Best Practice	Description
Prioritize Security Threat Modeling Early	Identify potential security threats during the design phase to mitigate risks upfront.
Automate Security Testing	Integrate automated security tools in the CI/CD pipeline for continuous security testing.
Enable Secure Coding Standards	Ensure all developers adhere to secure coding practices to minimize vulnerabilities.
Regularly Update and Patch Dependencies	Continuously monitor and update third-party libraries and components to prevent vulnerabilities.

Best Practice	Description
Collaboration Between Teams	Ensure security is a shared responsibility among developers, security, and operations teams.

9.1 Prioritize Security Threat Modeling Early

Threat modeling is one of the first things that should be accomplished in the development process. Early detection of potential security risks helps teams design mitigation strategies to meet potential vulnerabilities without introducing the vulnerability into codebase. It determines the system's architecture, deducts possible attack vectors, and learns how elements and their paths relate. This approach allows for anticipating security challenges and designing systems to prevent them. Early threat modeling can prevent many serious security flaws from contaminating the development team later in the cycle. It additionally allows for allocating resources to deal with those areas of the system having the most significant risk to address, ensuring that security efforts are made in the places where they are most required. Regularly returning to the threat model during development also reveals new risks as the system changes, strengthening security awareness and performing ongoing proactive risk management [34].

9.2. Automate Security Testing at Every Stage

Shift Left Security is a best practice that automates security testing throughout the SDLC. Security testing is, therefore, automatically run at every stage of development when tools, Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Software Composition Analysis (SCA) are integrated into the CI/CD pipeline. With automation, vulnerabilities are found early in the process, thus less likely to slip through the cracks. Furthermore, automated testing provides consistency and the ability to conduct a thorough security check without the threat of human mistakes. Developers can have honest feedback about security issues and be able to fix them immediately since it's automated, so there is less time for post-release fixes. Between an interactive pen test and a complete application scan, making testing and scanning part of development momentum can guarantee that security is continuously validated through the software lifecycle.

9.3. Enable Secure Coding Standards and Practices

Developers should follow secure coding standards from the beginning of the development. Standards include secure code guidelines to prevent the most common vulnerabilities like SQL injection, cross-site scripting (XSS), and buffer overflow. Cadence and culture of secure coding practices must be adopted at all team levels, and developers should be trained and aware of these. Aside from preventing vulnerabilities, practicing security coding guarantees that security is factored in throughout the development cycle, from the design's beginning to finishing deployment. The secure coding practices should also be reviewed and updated continually, using the most recent threat intelligence and industry best practices [35]. Implementing secure coding standards into the core development process improves software earlier in the SDLC and makes it more secure.

Figure 10. Secure Coding

9.4. Regularly Update and Patch Dependencies

Maintaining third-party libraries and dependencies up to date is of top importance when securing software [36]. In many modern applications, external code, such as open-source libraries, is used to speed up development. But, if not kept updated or patched regularly, these components can be quite a security risk. If left unaddressed, the vulnerabilities in third-party dependencies can become entry points for the attacker to compromise the complete application. Therefore, organizations should have a process for frequently checking for updates of these dependencies and using tools such as Software Composition Analysis (SCA) to track the known vulnerabilities in third-party code. Keeping dependencies up to date also allows organizations to prevent introducing security flaws in the application without knowing. Keeping dependencies patched will significantly reduce the risk of exploitation, but failing to patch them will undoubtedly result in vulnerability at some point.

9.5. Collaboration is Key: Developers, Security, and Operations

Implementing Shift Left Security is effective only when there is a collaboration between development, security, and operations teams. Traditionally, development models saw security slack away from the development process itself and separate security teams' work from those of developers and operations staff. This being said, Shift Left Security highlights the importance of security as a team's responsibility, not as a specific team within the SDLC. Because of this collaboration, embedded security is propagated throughout the entire development, from design to deployment. Developers must collaborate with security experts to discover potential risks and develop a secure coding practice. At the same time, the operation teams need to make sure that security policies are adhered to while pushing the code and while utilizing it. Organizations can build a culture of security in which everyone is rowing toward building secure software by encouraging open communication and teamwork.

9.6. Incorporating Secure Configuration Management

Another essential practice of Shift Left Security is secure configuration management. It considers ensuring that all systems, applications, and infrastructure are securely configured from the get-go. A common cause of security vulnerabilities is misconfigurations of the software or the infrastructure underneath. There should be secure configuration management practices

to ensure that security policies are also practiced in all environments. This involves securely setting up servers, databases, network devices, and all other cloud services, as well as those of any third-party software used to meet security standards. Security settings should be regularly checked to keep configurations up to date and identify potential misconfigurations early. Incorporating secure configuration management practices will help reduce the risk of vulnerabilities introduced by the misconfiguration of systems, thereby enhancing the overall security posture of applications and infrastructure.

10. FUTURE CONSIDERATIONS FOR SHIFT LEFT SECURITY

10.1. The Role of Artificial Intelligence in Enhancing Shift Left Security

Artificial intelligence (AI) will play a transformational factor in making Shift Left Security happen [37]. With the assistance of AI-powered tools, as compared to human testers, the tools can easily understand the extensive code base faster and better and discover vulnerabilities quickly. These tools use machine learning algorithms to identify better patterns in the code that could hint at potential security problems [38]. Furthermore, AI can also be employed to forecast future possible security risks based on historical data combined with real-time analysis, all of which could help development teams to fix at the break of dawn when risks have not yet escalated into critical ones. For instance, AI might examine codebase and immediately identify areas with frequently pre-sent vulnerabilities, like SQL injection, cross-site scripting (XSS), or insecure API utilization. Furthermore, AI can work through new threats quickly, making it easier to keep up with an ever-evolving cybersecurity party line. Introducing AI into the Shift Left process makes security testing proactive and efficient since potential vulnerabilities are found and mitigated early in development. Thus, in the long run, AI can transform Shift Left Security into an autonomous, intelligent system that evolves by itself, becomes capable of learning from previous attempts, and continuously becomes better at identifying security bugs without omitting any.

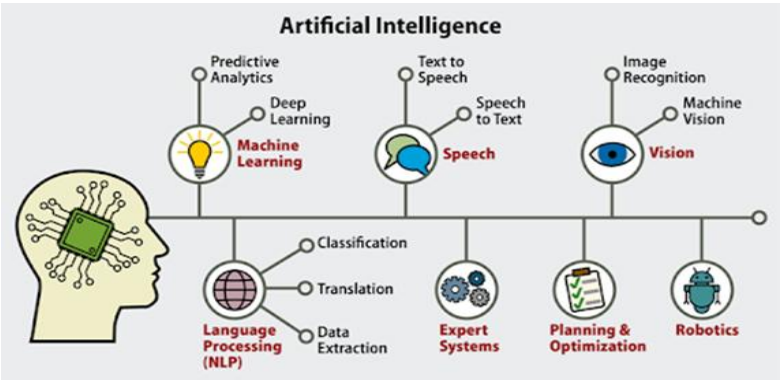

Figure 11. Power of Machine Learning and AI for every business

10.2. Cloud-Native Applications and Security Challenges

The new complexities and challenges of cloud-native applications must be addressed using Shift Left Security practices [39]. Cloud-native architectures, which mainly contain micro services, serverless, container environments, and so on, are the new software development and deployment formats. However, these changes bring about distinct security issues like the requirement for secure

communication among micro services, control of access across diverse cloud surroundings, and the holes that arise due to the dynamic scaling of resources. APIs and external services play a vital role in cloud-native applications, and any insecure APIs can also introduce additional attack vectors. Furthermore, misconfigurations in a cloud environment can leak sensitive data or provide access to unauthorized users, one of the most remarkable security concerns. Moving Security left needs to be considered for the unique cloud-native app risk. This necessitates robust identity and access management (IAM), secure API development, and automated security tools that integrate with the cloud environments seamlessly. To minimize risk, Security should be baked into the early development lifecycle of cloud-native applications using micro services, containers, and cloud infrastructure. Security is core to the development process, not something done at the very end. With ongoing cloud technology advancements, Shift Left Security must be able to balance the required security code coverage changes to address the new challenges while ensuring continuous security validation from the application lifecycle throughout its lifecycle in the cloud.

10.3. The Integration of Threat Intelligence in Development Processes

Threat Intelligence incorporated within the development process is a necessary hurdle to hurdle through to develop ahead of ongoing security threats. Known vulnerabilities, attack vectors, and trends in the cybersecurity landscape at various sourcing points, as well as the best practices to keep in mind, are all valuable software that real-time threat intelligence provides and helps proactively address potential risks. Integrating threat intelligence into the Shift Left Security framework enables the ability to make educated decisions on the Security of applications while being aware of the current threat actors and tactics involved. Development teams can use this information to customize their security measures to deal with the threats they are most likely to face. For instance, to keep threat intelligence feeds from feeding so many agencies, the information can also help developers know of a new malware strain, a new zero-day vulnerability, or even a new attack method so they can change their security protocols, patch for the bugs, and include protection of that vulnerability [40]. Moreover, real-time threat intelligence could also assist in utilizing automated security testing tools more efficiently to uncover the latest or new threats. Not only does this promote Security explicitly, but it likewise means organizations can resort to Security in light of the most current accessible information, so vulnerabilities can be tended to before executioners can benefit them. Threat intelligence is a dynamic resource for improving a software application's security posture because the development process must be guided by the most current security trends and risks [41].

10.4. Security as Code: Moving Towards a Unified Security Framework

Security as Code is a relatively new concept of defining security policies as Code and automatically enforcing it across all the stages of the software development lifecycle. This is to guard the application from security vulnerabilities at the source code and early stage. Treat security

configurations, policies, and controls as Code allows to provide the same security rules to all environments and automate the enforcement of security rules. This also makes scaling easier since security policies can be versioned and applied to all software deployments as they evolve. For example, security rules can be coded inside the Code to be always validated during the build, deployment, and test cycle. Thus, it integrates security policies into the development pipeline and thereby ensures that security is always maintained. Still, at, the same time, it enables an agile and flexible response to changes in security needs. The organizations can move towards a unified security framework, i.e., Towards the security as part of Code where security is treated as part of the Code and integrated seamlessly with the Continuous Integration / Continuous Deployment (or simply CI/CD) pipelines to make security as an intrinsic part of the software development life cycle instead of an isolated practice. Security as Code concept also promotes collaboration among development, reliability, and operation teams by ensuring that both are on the same page of how security should be applied and that security should be maintained throughout the software lifecycle.

10.5. How Regulations and Compliance Will Shape Future Practices

With the rapid changes in data privacy and cybersecurity regulations, Shift Left Security will also have to conform to the new legal requirements and industry standards [42]. Their software must comply with rules such as the General Data Protection Regulation (GDPR) or the California Consumer Privacy Act (CCPA) and other regional or industry-specific regulations. Typically, these regulations lay down strict requirements for how personal data should be handled, stored, and transmitted. Companies have to integrate such requirements into their development processes to avoid penalties and loss of reputation. Given this, Shift Left Security will require compliance with regulatory checks to take place as early as possible in the SDLC, which may encompass ensuring that security measures and data privacy protocols are considered from design early. Organizations can guarantee that their software continues to comply with ever-evolving regulations by automating compliance checks and continuously checking those through the development cycle. For example, security tools that are part of the CI/CD pipeline can do vulnerability checks for data encryption, access control, and data storage and identify errors early so the software doesn't get released and reach production. Taking a proactive stance toward compliance will aid organizations in avoiding huge fines and legal risks and building customer trust, proving that the organization is dedicated to protecting the user's data and conforms to the industry's best security standards.

11. CONCLUSION

Shift Left Security is a principle to adopt security practices earlier in the Software Development Lifecycle (SDLC). Addressing security risks in the design and development phases leads to far fewer vulnerabilities, better security posture, and lower costs for post-release remediation efforts. This proactive strategy is suitable for continuous security testing, provides honest time feedback, and is ideal for all teams involved, from development to security to operations. It helps ensure that security has to be considered at every phase of SDLC and not as a one-last

check as in traditional software development. Shifting security to the left helps organizations identify and mitigate vulnerabilities before they become problems, reducing the chances of software products being less secure and safer. Nowadays, it is no longer a best practice but a necessity to adopt Shift Left Security in today's fast-paced digital world in which development cycles are becoming faster, and the complexity of the systems on our hands is ever increasing, for which the traditional security measures are no longer capable of dealing with.

With the rapid advancement of the software development field, security will become increasingly important to integrate in every phase of the SDLC. Software security needs to be fully integrated into the software development process. Shift Left Security is building upon itself for more advanced security methodologies with a way to adapt to modern development environments. This also includes using AI-driven security tools, tackling the uniqueness of cloud-native applications, and meeting the increasing importance of data privacy and compliance. It's prudent to prioritize security early on whilst developing software to build more resilient and robust software capable of standing strains from the fast-changing and constantly spreading cyber threat landscape. It also helps bring applications to market faster and with increased customer satisfaction by ensuring that security does not slow down the pace of development or reduce product quality.

Adopting Shift Left Security as an integral part of the software development process for organizations is encouraged. Placing security practices, starting from the beginning of the development cycle, organizations can build a better security stance without much investment after release and create a security culture among their teams. By acting early, the vulnerabilities are found and taken care of before they become significant security weaknesses or result in financial loss. In addition, Shift Left Security promotes closer interaction between developer, security, and operations teams, leading to more secure application development practices and earning additional customers' confidence. With the rise of cyber threats, businesses that begin by emphasizing security will be best able to manage these issues and achieve sustained success in their software products. In the era of digital data, proactive security practices like Shift Left Security are required to ensure that the software application is secure when development begins. Every phase of the SDLC needs to be secured as organizations face ever more sophisticated threats of cyber-attacks. Shift Left Security offers a wide range of proactive means for early risk identification and mitigation in favor of more secure applications and a more amicable development process. Organizations that put security first from the beginning can keep their products and users safe, build a security-first culture, and create successful software in a fast and growing environment where they are vulnerable.

REFERENCES

[1] A. Ahmad, I., Namal, S., Ylianttila, M., & Gurtov, "Security in software defined networks: A survey. IEEE Communications Surveys & Tutorials," 17(4), 2317–2346, 2015.

[2] D. A. Arrey, "Exploring the integration of security into software development life cycle (SDLC) methodology (Doctoral dissertation, Colorado Technical University)," 2019.

[3] N. Dissanayake, M. Zahedi, A. Jayatilaka, and M. A. Babar, "A grounded theory of the role of coordination in software security patch management," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021, pp. 793–805.

[4] M. Deschene, "Embracing security in all phases of the software development life cycle: A Delphi study. Capella University," 2016.

[5] R. K. Raju, "Dynamic memory inference network for natural language inference. International Journal of Science and Research (IJSR)," 6(2), 2017.

[6] M. Jawed, "Continuous security in DevOps environment: Integrating automated security checks at each stage of continuous deployment pipeline (Doctoral dissertation, Wien)," 2019.

[7] T. Xu, M., Song, C., Ji, Y., Shih, M. W., Lu, K., Zheng, C., ... & Kim, "Toward engineering a secure android ecosystem:

A survey of existing techniques. ACM Computing Surveys (CSUR),” 49(2), 1–47, 2016.
[8] A. Chavan, “Eventual consistency vs. strong consistency: Making the right choice in microservices. International Journal of Software and Applications,” 14(3), 45–56, 2021.
[9] B. Ransome, J., & Schoenfield, “Building in Security at Agile Speed. Auerbach Publications,” 2021.
[10] A. CHAVAN, “Exploring event-driven architecture in microservices: Patterns, pitfalls, and best practices. International Journal of Software and Research Analysis,” 2021.
[11] V. Boppana, “Secure Practices in Software Development. Global Research Review in Business and Economics [GRRBE],” 10(05), 2019.
[12] F. Alaba, F. A., Othman, M., Hashem, I. A. T., & Alotaibi, “Internet of Things security: A survey. Journal of Network and Computer Applications,” 88, 10–28, 2017.
[13] N. Hassija, V., Chamola, V., Gupta, V., Jain, S., & Guizani, “A survey on supply chain security: Application areas, security threats, and solution architectures. IEEE Internet of Things Journal,” 8(8), 6222–6246, 2020.
[14] M. A. Bell, S. C., & Orzen, “Lean IT: Enabling and sustaining your lean transformation. CRC Press,” 2016.
[15] R. Ross, R., Pillitteri, V., Graubart, R., Bodeau, D., & McQuaid, “Developing cyber resilient systems: a systems security engineering approach (No. NIST Special Publication (SP) 800-160 Vol. 2 (Draft)). National Institute of Standards and Technology,” 2019.
[16] R. Pompon, “IT Security Risk Control Management: An Audit Preparation Plan. Apress,” 2016.
[17] C. Deegan, “Continuous Security; Investigation of the DevOps Approach to Security (Doctoral dissertation, Dublin, National College of Ireland),” 2020.
[18] D. Landoll, “The security risk assessment handbook: A complete guide for performing security risk assessments. CRC press,” 2021.
[19] L. L. Kegan, R., & Lahey, “An everyone culture: Becoming a deliberately developmental organization. Harvard Business Review Press,” 2016.
[20] S. Nyati, “Revolutionizing LTL carrier operations: A comprehensive analysis of an algorithm-driven pickup and delivery dispatching solution. International Journal of Science and Research (IJSR), 7(2), 1659-1666. Retrieved from,” 2018.
[21] P. Sandhu, M. A., Shamsuzzoha, A., & Helo, “Does outsourcing always work? A critical evaluation for project business success. Benchmarking: An International Journal,” 25(7), 2198–2215, 2018.
[22] M. I. Mateo Tudela, F., Bermejo Higuera, J. R., Bermejo Higuera, J., Sicilia Montalvo, J. A., & Argyros, “On combining static, dynamic and interactive analysis security testing tools to improve owasp top ten security vulnerability detection in web applications. Applied Sciences,” 10(24), 9119, 2020.
[23] S. Graham Linck, E. J., Richmond, P. A., Tarailo-Graovac, M., Engelke, U., Kluijtmans, L. A., Coene, K. L., ... & Mostafavi, “metPropagate: network-guided propagation of metabolomic information for prioritization of metabolic disease genes. NPJ genomic medicine,” 5(1), 25, 2020.
[24] P. Singh, V., Murarka, Y., Jaiswal, A., & Kanani, “Detection and classification of arrhythmia. International Journal of Grid and Distributed Computing,” 13(6), 2020.
[25] A. V Ali, M., Khan, S. U., & Vasilakos, “Security in cloud computing: Opportunities and challenges. Information sciences,” 305, 357–383, 2015.
[26] R. Aljawarneh, S. A., Alawneh, A., & Jaradat, “Cloud security engineering: Early stages of SDLC. Future Generation Computer Systems,” 74, 385–392, 2017.
[27] L. Bass, L., Weber, I., & Zhu, “DevOps: A software architect’s perspective. Addison-Wesley Professional,” 2015.
[28] S. Assal, H., & Chiasson, “Security in the software development lifecycle. In Fourteenth symposium on usable privacy and security (SOUPS 2018) (pp. 281-296),” 2018.
[29] A. Shajadi, “Automating security tests for web applications in continuous integration and deployment environment,” 2019.
[30] J. D. Bailey, J. R., & Raelin, “Organizations don’t resist change, people do: Modeling individual reactions to organizational change through loss and terror management. Organization management journal,” 12(3), 125–138, 2015.
[31] S. Sharma, “The DevOps adoption playbook: a guide to adopting DevOps in a multi-speed IT enterprise. John Wiley & Sons,” 2017.
[32] A. Takanen, A., Demott, J. D., Miller, C., & Kettunen, “Fuzzing for software security testing and quality assurance. Artech House,” 2018.
[33] V. V. R. Boda, “Balancing Speed and Safety: CI/CD in the World of Healthcare. Journal of Innovative Technologies,” 3(1), 2020.
[34] F. E. Settembre-Blundo, D., González-Sánchez, R., Medina-Salgado, S., & García-Muiña, “Flexibility and resilience in corporate decision making: a new sustainability-based risk management system in uncertain times. Global Journal of Flexible Systems Management, 22(Suppl 2),” 107–132, 2021.
[35] A. Kumar, “The convergence of predictive analytics in driving business intelligence and enhancing DevOps efficiency. International Journal of Computational Engineering and Management, 6(6), 118-142. Retrieved from,” 2019.
[36] M. Derr, E., Bugiel, S., Fahl, S., Acar, Y., & Backes, “Keep me updated: An empirical study of third-party library updatability on android. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 2187-2200).”
[37] K. Shibuya, “Digital transformation of identity in the age of artificial intelligence,” Springer, 2020.

[38] S. NYATI, "Transforming telematics in fleet management: Innovations in asset tracking, efficiency, and communication. International Journal of Science and Research (IJSR), 7(10), 1804-1810. Retrieved from," 2018.

[39] P. Laszewski, T., Arora, K., Farr, E., & Zonooz, "Cloud Native Architectures: Design high-availability and cost-effective applications for the cloud. Packt Publishing Ltd," 2018.

[40] P. Singh, V., Oza, M., Vaghela, H., & Kanani, "Auto-encoding progressive generative adversarial networks for 3D multi object scenes. In 2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT) (pp. 481-485). IEEE."

[41] J. F. Samtani, S., Chinn, R., Chen, H., & Nunamaker Jr, "Exploring emerging hacker assets and key hackers for proactive cyber threat intelligence. Journal of Management Information Systems," 34(4), 1023–1053, 2017.

[42] C. A. Tschider, "Regulating the internet of things: discrimination, privacy, and cybersecurity in the artificial intelligence age. Denv. L. Rev., 96, 87," 2018.