LTIMindtree

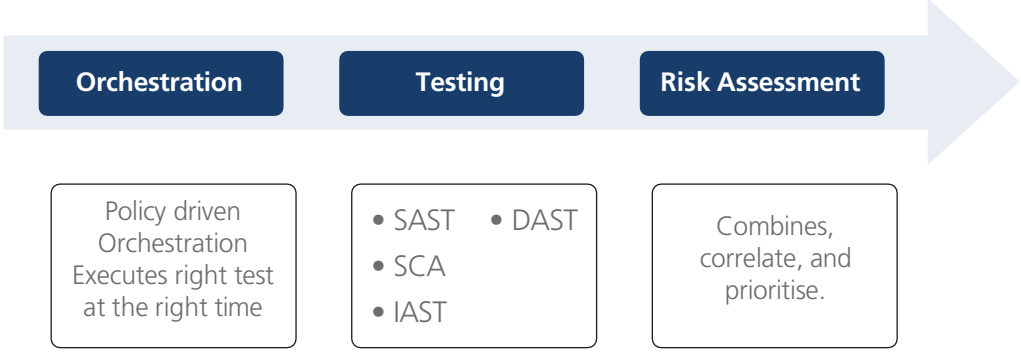
WHITEPAPER

The why, what, and how of application security testing

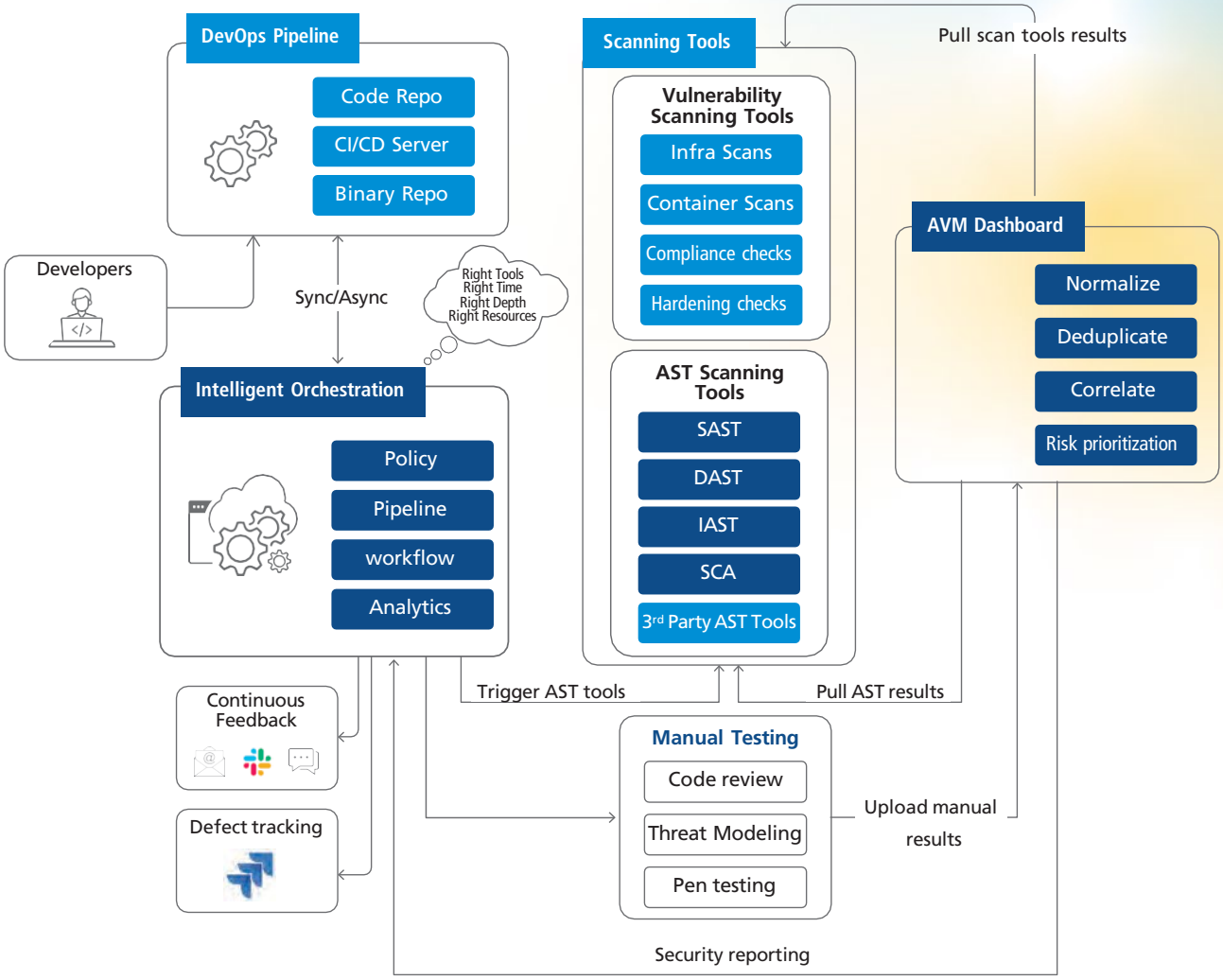
Cyberattacks affect both end users and business teams and continue to grow at a frightening scale. Run-of-the-mill security services are no longer adequate to fight the battle against these new-age cyber threats. Most modern applications are often available over various networks and are connected to the cloud. In a world rife with ever-increasing cyber-security threats, these applications need multiple layers of security, and that is where robust application security testing is an absolute must for organizations. Most successful security breaches target exploitable vulnerabilities residing in the application layer, indicating the need for enterprise IT departments to be extra vigilant about application security. If we don't care about vulnerabilities that come with dependencies during application development, the applications are more likely to get exposed to cyber-attacks later on.

Gartner defines ASOC tools as those that "streamline software vulnerability testing and remediation by automating workflows. These tools are used to automate application security testing through a range of methods. These include ingesting data from static, dynamic, and interactive sources through SAST/ DAST/IAST methods; software composition analysis [SCA]; vulnerability assessments, and other methods.

ASOC in a DevSecOps Pipelining



The Intelligent Orchestration and Code Dx integration



Source: [synopsys.com]

This is where security scanning tools come in.

To address these issues and make the applications secure, organizations are using a range of security scanning tools to identify and rectify vulnerabilities in applications before they are run in production environments.

In general, 4 categories of security scanning tools - SAST, DAST, IAST, and SCA - are used by developers to ensure that applications go through a robust security check in the development environment itself.

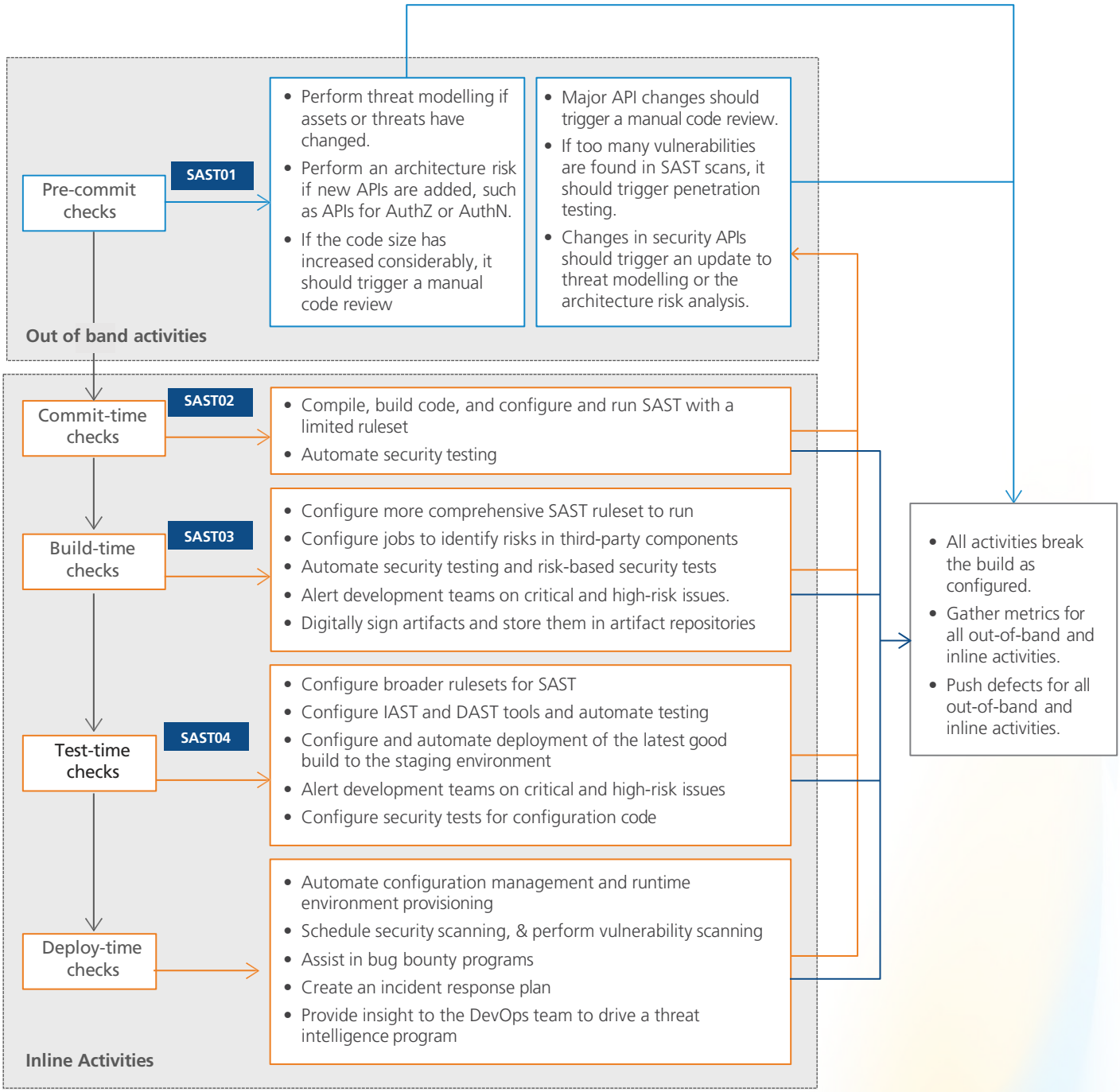
APPLICATION SECURITY TESTING								
		Coverage	Low False Positives	Exploitability	Code Visibility	Remediation Advice	SDLC Integration	Broad Platform Support
Security Scanning Tools	SAST	✓			✓	✓	✓	✓
	DAST		✓	✓				✓
	IAST		✓	✓	✓	✓	✓	
	SCA	✓		✓	✓	✓	✓	✓
Runtime Protection Tools	WAF	✓	*			✓		✓
	Bot Mngmt		*			✓		✓
	RASP	✓	✓	✓	✓	✓		

Source: mend.io

Static Application Security Testing (SAST)

SAST is a white-box testing method where the source code is analyzed from the inside out while the components are at rest. SAST analyses application source code, byte code, and binaries for coding and design flaws that suggest possible security vulnerabilities.

SAST tool integration in DevSecOps pipeline

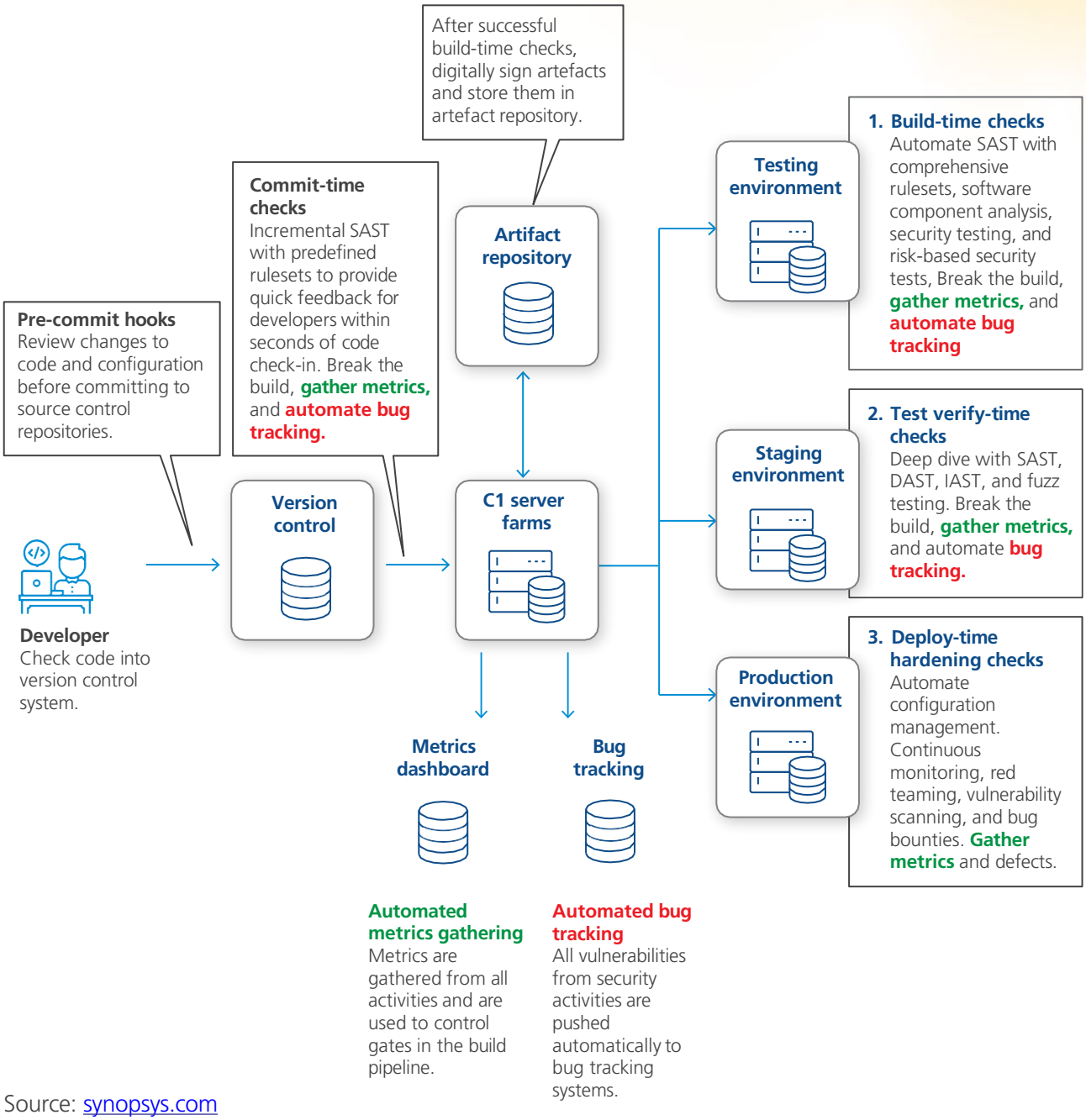


SAST01	SAST02	SAST03	SAST04
The SAST tool runs in the IDE as developers write code. The tool is configured to detect vulnerabilities that have zero false positives, including issues such as SQL injection and XSS. The scan should take seconds.	The SAST tool is automated on the CI server. The tool is configured for the client's top 10 issues, such as command injection and hard-coded keys. The tool also uses rules from SAST01. The scan should take 4-5 minutes so developers should get feedback.	The SAST tool uses rules for the OWASP Top 10 and any customized rulesets written for client-specific APIs. The scan can be run in parallel with other activities and should take 10-15 minutes.	The SAST tool uses comprehensive rulesets. All previous rulesets are excluded. The goal is to find issues before the code goes to production. The scan should take anywhere from an hour to 3-4 hours, depending on production velocity.

Source: synopsys.com

Dynamic Application Security Testing (DAST)

DAST is a black-box testing method. The key difference between the SAST and DAST methods is that while in SAST, application code is scanned line-by-line while the app itself is at rest; in DAST, developers simulate external attacks on applications when the apps are running to look for possible security vulnerabilities as well as find weaknesses in the app architecture. The DAST method cannot access the source code. Instead, this method attempts to penetrate an application from the outside to find potential vulnerabilities in its exposed interface. As its name suggests, the DAST method is applied in a dynamic environment (often a QA environment) instead of a production environment.



Source: synopsys.com

Best practices on DAST

What kind of scans should you run, and at which point?

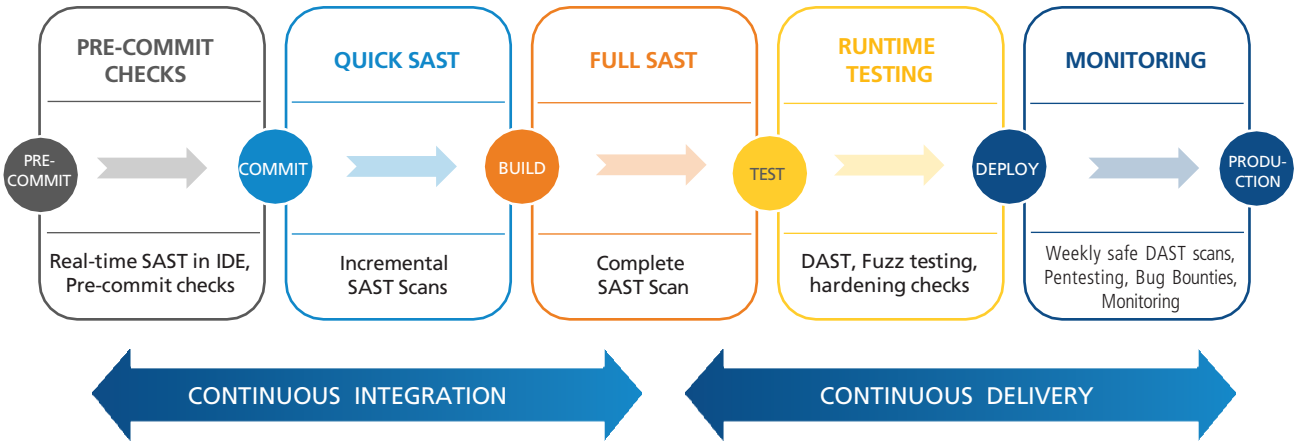
Depending on the environments and places in the pipeline where they are used, scans can be generally put under four categories - lightning, safe, normal, and full. For instance, as the name suggests, lightning scans are very fast scans (usually completed in under 60 seconds) and are hence used together with the unit and functional test so that the flow is not stalled. However, the lightning scans can only test relatively small vulnerabilities related to SSL/TLS, HTTP headers, and cookies attributes. However, these are still important since hackers often use loopholes in these to get started before mounting major attacks. The "safe scan," on the other hand, has a broader range as they cover all the vulnerabilities. However, these scans don't make POST, PUT, DELETE or UPDATE requests as they work with a relatively limited set of payloads and methods. The best scenario to use safe scans in the product pipeline is the stage close to the production environment. The "full scan" on the other hand, is, as the name suggests, a complete scan that uses the largest number of payloads. This is typically run at the staging/testing environments, and even though some of the payloads used in this are risky, the "full scan" is generally carried out in its entirety to ensure that nothing in the production stage is broken.

Which results are allowed to block the pipeline (or block a deploy)?

Most modern DSAT scanners have the capability to map vulnerabilities accurately to possible risks. This allows the users to gauge the severity of each result and customize how each of these results can affect either the pipeline or the deploy process. Based on these, there are three possible different levels of risk - high, medium, and low. Keeping this in mind, we can use the API and set up a process where the pipeline or a deploy is blocked only if there is a medium or high vulnerability. For example, in order to push a change to production, there should be a process in place that checks if there's a medium or high vulnerability in the last DAST scan in staging (and if there's no scan currently running). If that test passes, then the change is secure enough to be pushed into production. For example, we can block a deploy only for certain vulnerabilities, like SQL injections (because of the impact).

Scheduled Tests

Another way to use a DAST scanner is to schedule periodic tests. Based on our schedule, weekly, bi-weekly, or even nightly scans. Since DAST scans can take a while, the recommendation is to scan after a day of work, at night. We need to make these scans a requirement for pushing to production. To successfully run scheduled tests and not block the pipeline, we can create a 'separate' pipeline rather than integrating them into your current staging one, but this needs to be linked



Source: blog.probely.com

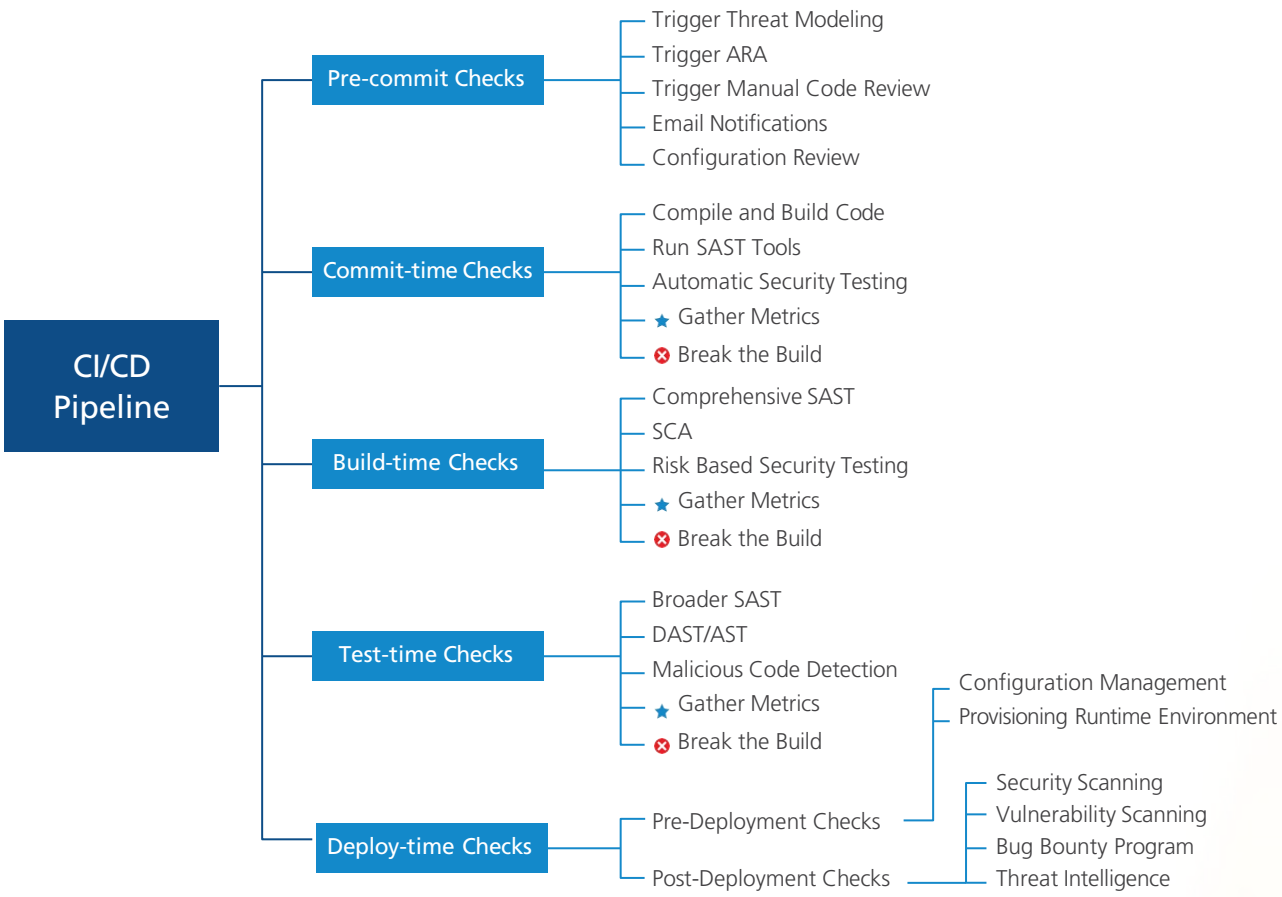
Interactive Application Security Testing (IAST)

The interactive application security method is used to scan an app's source code in the post-build, dynamic environment. The advantage of the ISAT method is that it enables the tester to identify the problematic line of code and notify the developer for remediation real-time.

In both the SAST and IAST methods, the application code is scanned directly. However, the key difference between them is that IAST scans the code in the post-build, dynamic environment through the instrumentation of the code. In this method, agents and sensors deployed in the app are used to analyze the code and identify potential vulnerabilities. The key advantage of the IAST method is that it can easily integrate with the CI/CD pipeline. In addition, it's highly scalable and can be both automated as well as used manually by a human tester.

Software Composition Analysis (SCA)

SCA performs automated scans of an application's code base to provide visibility into open-source software usage. This includes identifying all open-source components, their license compliance data, and security vulnerabilities. In addition to providing visibility into open-source software use, SCA tools also prioritize open-source vulnerabilities and ideally provide insights and auto-remediation to resolve security threats.



Source: synopsys.com

Security Testing in CICD

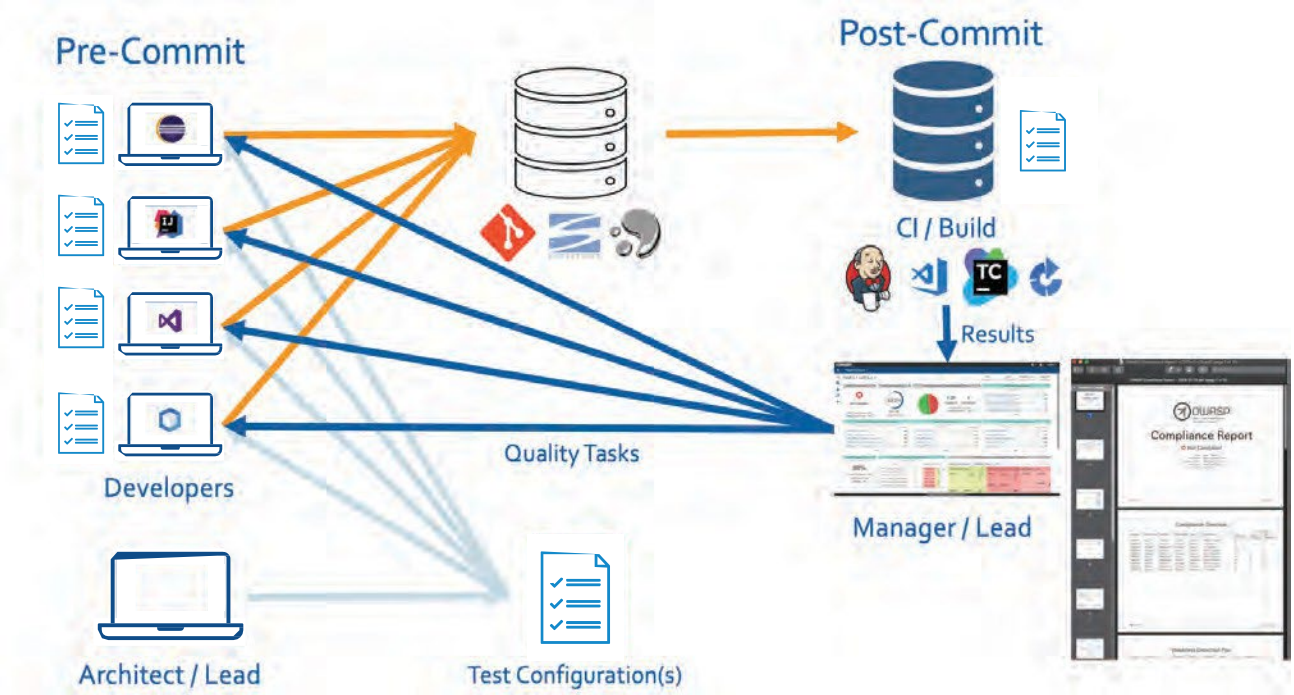
It all comes down to how security becomes an integral part of the Software Development Lifecycle of the application and the right way to do that is via CICD. All of the above security testing strategies are to be integrated into the CICD workflow to ensure relevant guard rails get automatically integrated into the system. This way we need to manually check the applications at different stages of the SDLC.

Pre-Commit Checks: The objective of pre-commit checks is to enable activities such as updating a threat model when controls or new assets are added to the application. Pre-commit checks are also used to enable manual code reviews if a large change in the code base is detected. These checks can also trigger risk analysis while identifying security vulnerabilities.

Next, we can create hooks to trigger activities such as threat modelling and architecture risk analysis, and manual code review. We can also create additional hooks to review your configuration files for hard-coded credentials.

- **Commit time Checks:** In this method, the top 10 vulnerabilities of apps are identified manually by a quick, incremental scan designed to provide feedback to developers within minutes. For instance, using a typical SAST tool, we can identify common vulnerabilities like SQL injections and XSS (cross-site scripting).
- **Build time checks:** In this type of check, a range of methods are used. The build time checks can include anything from open-source management to risk-based security tests, processes like signing binary releases with PGP signatures, storing artifacts in repositories, and even a deeper level of SAST.
- **Test Time checks:** The test time checks are usually done after a SAST method has already been used and can be configured to run on DAST tools. The test time checks are generally used to test both common critical and high-severity issues. This method typically uses a tool's full set of security rules.
- **Deploy time checks:** The deploy time checks are used in a post-deployment phase to periodically trigger security tests and ensure that the changes in the production environment didn't trigger any further security issues.

From Security perspective:



Source: parasoft.com

Key benefits of having robust application security testing

Organizations need robust application security testing for both to minimize business disruptions and cultivate a range of technical and business benefits.

- **Protecting applications against cyber-attacks** by identifying potential vulnerabilities.
- **Fixing vulnerabilities** in the early stage of development and thus **minimizing the cost** by reducing the regression of development cycles
- **Helping development, security, and DevOps** teams understand the process of making applications secure.
- Enabling organizations **to build customer confidenc**e through creating apps that are capable of protecting user data.

Conclusion

To tackle the ever-looming threats of cyber-attacks, organizations are looking for wholistic cloud security and resiliency solutions that go beyond the scope of security for the sake of compliance. In that context, our well-defined application testing security maturity models help organizations with a model that brings together all the aspects of application security testing in an well-orchestrated manner. Head over here to explore more about LTIMindtree's cyber security practice. https://www.ltimindtree.com/services/cyber-security/

Authors

Santosh Malimath

Sr Project Manager/Technical Architect

Over the last 16 years, Santosh has worked with several large enterprises. His experience spans as a Full-stack development, DevSecOps engineering and IOT solutioning. During the last 6 years he has engaged in analyzing and creating project scope and milestones for several technical company initiatives. Has been contributing to help define key partnerships to targets, establishing technical relationships, and managing the day-to-day interactions to build long-term business and marketing opportunities.

About 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/.