MEND

Achieving DevSecOps with SAST

Introduction

Secure software is a foundational requirement of any modern organization. Eclectic developers growing as one team mean that security engineers need to work with the business team to enforce security testing for the entire application portfolio. Without security testing, some organizations are still either releasing code to production containing known vulnerabilities or waiting until they are ready to deploy to address security issues. Years ago, organizations found this an acceptable risk. But that is no longer the case. Organizations need a way to embed static application security testing (SAST) into their dev pipelines as seamlessly as possible, allowing their developers to test their code more frequently and more accurately than before.

In particular, SAST integration and automation into existing dev tooling is imperative since it improves application security and reduces testing delays. In modern application development, it's crucial to remove anything slowing or stopping development teams from meeting their deadlines. This requires SAST tools that build into the developer experience and enable them to create secure applications with the tools they are already familiar with.

In this paper, we will explore how to create a remarkably seamless developer experience in static application security testing by integrating security tools in the developers' native environment.

> *"Traditional application security approaches rely on heavyweight, one-time gating inspections, typically performed late in testing, taking days (if not weeks) and requiring security professionals to perform them. This won't scale for DevSecOps. DevOps emphasizes continuous feedback throughout the process and improved automation. Security needs to adopt and support a mindset where security starts at the very beginning of service creation and throughout the DevOps processes, and is continuous, automated and improves with each subsequent iteration."*
>
> Source: Gartner "Integrating Security Into the DevSecOps Toolchain" Refreshed 4 March 2021, Published 15 November 2019 - ID G00377293 - Mark Horvath, Neil MacDonald

Ease of Setup

Building a developer-centric experience starts at the very beginning- the setup and prerequisites of the scanning tool. Tools that are easy to set up will enjoy far greater usage across a wide range of developers. This generates a better return on investment for your expensive and important security tool.

Traditional SAST tools require many steps for custom configurations to get scans operational. For example, a typical SAST tool might need to be configured for specific languages like Python or Java. These kinds of configurations take time and effort, which leaves the developer less interested in using the tool.

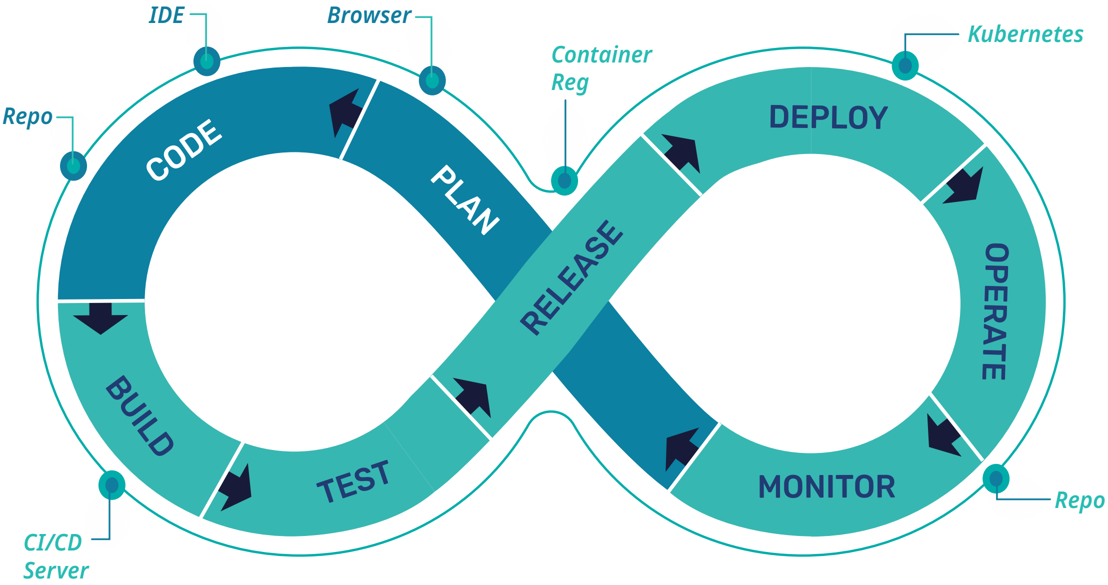
Mend SAST does not need to be configured in advance. The tool is equipped with auto language recognition and only requires access to the source code that you wish to be scanned. This dramatically reduces the time spent on set up. The time to set up from start to running your first scan can be as short as 3 minutes. Developers appreciate the lack of need to configure.

Ease of Pipeline Integration

Pipeline integration is a key component of a seamless developer experience. SAST solutions need to integrate into a developer's existing toolset in order to increase developer adoption. Integrations with popular build systems, such as Jenkins, Bamboo, or Azure DevOps, can serve this purpose. Providing these seamless integrations will increase developer efficiency because they will not be forced out of their day-to-day workflows to run an application security test.

Traditional application security testing is not integrated into a developer's existing toolset. Previously, security was a step at the end of the development lifecycle where a security professional runs tests to identify vulnerabilities, assesses each vulnerability for false positives, prioritizes them by risk, and triages for remediation. With security acting as a gate for code deployment, it's sometimes perceived as a roadblock to innovation by developers and engineers. The intent is to test all of the code but often the volume of code is too high, the tests take a long time to run, and there's a shortage of security analysts to triage, prioritize, and validate results. Because the effort is siloed from the development workflow, it may be some time before the developer learns of the needed remediation. By then, they have moved on to another project and must take some time to re-engage. This separation and delay creates friction in the software development life cycle.

Mend SAST can work with any build system because it utilizes a lightweight command line interface (CLI). The code is being scanned locally that is triggered by the command line, which means the source code does not get potentially compromised from uploading it into the cloud. Scans can be done before an application is built, which frees up your pipeline to operate without adding another tool that could accidentally break the build or deplete more resources. There is no need to install anything other than the executor on the scanning machine itself, resulting in little overhead costs and up to 10x faster scanning speeds.



Ease of Repository Integration

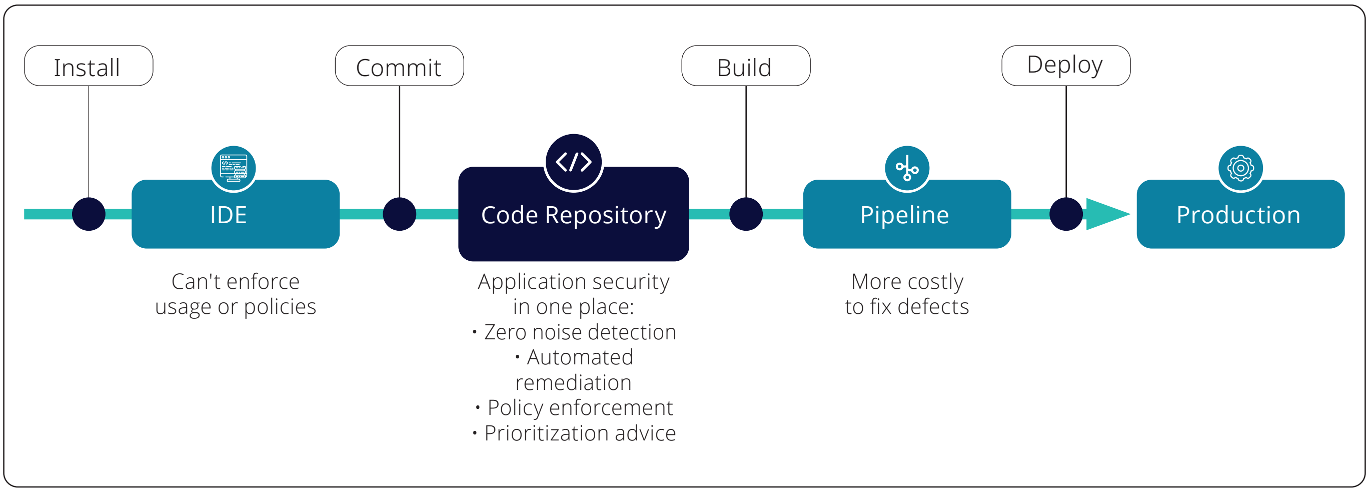
Repository integration is as important as pipeline integration in providing a seamless developer experience. In selecting a SAST tool, an important criteria should be ease of integration with the repository, one of the main tools in a developer's toolkit; otherwise day-to-day workflows are disrupted from toggling in and out of disparate tools.

Traditional application security testing did not need to operate within native development environments because the security tests were run by a security professional at the end of the software development lifecycle. Traditional application security approaches were cumbersome and rigorous, providing a technical report that developers had to go back to fix, outside of the developer's original context.

Mend SAST scans in the repository natively. By scanning in the repository, developers receive feedback in their native development environment at the exact moment they are asking for information and before they have moved on to new coding tasks. By giving instant feedback when a pull request is made, developers are given the ability to fix any security issues before they are merged. Furthermore, by giving results within the repository, you avoid the context switching of moving to a different UI, which saves time and resources and reduces the friction between developers and security teams. In fact, scanning in the repository has several benefits:

- **Shift left** - Scanning at the repository is the furthest left you can shift while still enforcing policies and requiring all developers to scan their code.
- **Feedback on demand** – Developers receive feedback on their code when it is fresh in their minds, making it easier to remediate vulnerabilities.
- **No context switching** – Developers don't need to leave their native environment and don't have to learn a new UI, making it easier to consume and act upon scan results.
- **Differential results** – Developers are notified only if a pull request introduces new errors. Positive feedback is given to developers when a pull request resolves vulnerabilities. This differential view that focuses on feature branches – not mainline – prevents interruptions to workflows.
- **Automated remediation** – Security vulnerabilities can be automatically prioritized and remediated.

By scanning and providing instant feedback in the repository, organizations are able to implement and fine-tune policies that help automate the security process. Detailed prioritization and remediation advice at this point in development also helps lessen the burden placed on developers.



Conclusion

Modern applications require securing, and that starts with the developers. Developer-centered security tools are a far departure from traditional security tooling. To drive developer adoption of security, the security tools must be integrated with the developers' native application development environments. This drives more usage of the tool, which results in more secure applications.