WHITE PAPER

WHY YOU NEED DAST IN YOUR SDLC

netsparker

Executive Summary

This document demonstrates how you can use dynamic application security testing (DAST) in your software development lifecycle (SDLC) to continuously improve software security and streamline issue handling. By integrating a modern DAST solution like Netsparker into your development and testing workflows, you can feed accurate vulnerability scanning results directly into existing automation toolchains and processes. This allows you to catch and fix security issues early on in the development process, shifting security testing left to avoid the cost and disruption of addressing vulnerabilities at later stages.

Highlights from this technical white paper include:

- Why traditional pre-release security testing is no longer enough for modern web application development
- How modern DAST makes it possible to automate application security testing and integrate it into Agile software development lifecycles and DevOps workflows
- Why shifting left with accurate dynamic testing is the only real-life approach to building scalable web application security and moving towards DevSecOps
- Typical use cases for integrating Netsparker into the development pipeline at various levels of maturity both for the SDLC and security integration

The Challenge:
Building Security into the SDLC

The pace of modern web application development continues to accelerate and agile approaches are now the norm. Once a bold new strategy to streamline development and reduce costs, DevOps has become an everyday necessity not only to ensure that operations can keep up with frequent application changes but also to align development to increasingly dynamic cloud deployment models. As development and operations steamed ahead into this brave new world, application security was not a high priority – at least initially.

In the pre-cloud era, the focus of IT security was on systems, services, and endpoints. Application security took a back seat to functionality and performance because most software was only accessible from secured local systems and networks. With the rapid expansion of web applications and the shift to cloud platforms during the past decade, the threat landscape has changed dramatically. Now it is common for business-critical systems and data to be globally accessible, which has made application security a vital consideration.

“ DevOps has become a necessity to keep up with frequent changes and align development to dynamic cloud deployments.

Rapid Development, Cloud Deployment

Part of the web security challenge lies in the nature of modern web development. Application development is no longer a process with a clear start and end point but rather a continuous cycle of design, implementation, testing, and deployment. This allows feedback and new or changed business requirements to be incorporated into production in a matter of weeks or even days without going through the lengthy requirement definition process of waterfall development. Applications are now overwhelmingly built using premade frameworks, components, and libraries, which allows companies to easily adopt new technologies and design trends without breaking the cycle.

The price of this speed, convenience, and ease of development is that what you write is no longer what you run. This is because the code that is actually processed by the browser has little or nothing to do with the framework-based development code. Under the hood, frameworks and site generators automatically pull in dozens of external dependencies and combine them with templates, external function calls, and custom code. This poses unique challenges for security testing, as developer-controlled code is just a tiny part of the code base and much of it is not directly rendered by the browser but merely used to guide site generation. On top of that, you might have dynamic dependencies that are only loaded at runtime, usually from a content delivery network (CDN). A modern web application is a complex combination of changing parts where you have no control over the vast majority of the codebase. Testing just your own code for vulnerabilities can’t hope to provide complete coverage – you need to think much bigger.

This is all before we get to the minefield that is cloud deployment. With the monolithic applications and single-server deployments of traditional websites and applications, you had at least the theoretical possibility of checking all your code and securing access to the deployment environment and data. Modern web applications are often dynamic jigsaw puzzles glued together with a thin layer of custom code. Now imagine this patchwork being split into dozens of microservices, each running in one of thousands of containers in a public cloud environment. It's a similar story with data storage and access – no longer restricted to a single physical server, your business data can now be spread all over the world (and certainly all over your cloud region). All this creates a massive attack surface, potentially exposing the application with its myriad components and business-critical data to global cyberattackers.

> “A modern web application is a complex combination of changing parts where you have no control over the vast majority of the codebase.

The Challenges of Web Application Security Testing

In software testing, developers have traditionally focused on static code tests, using linting to catch simple errors and unit testing to check code logic and sanity. If an application or module passes all the code checks, compiles, and runs, it is no longer the developers' concern and dedicated QA testers take over to run dynamic testing and report any bugs they find. The same tried and tested approach was then applied to application security testing, with code-level tools used during development and runtime testing performed during QA and separate penetration testing. This is how we arrived at the two major categories of security testing products: static application security testing (SAST) and dynamic application security testing (DAST).

Applying the traditional software testing mindset to web application security brings many problems because the two are very different. First off, each stage of software testing checks for correctness at some level: syntax, value sanity, business logic, business requirements, performance, data integrity, user acceptance, etc. In other words, you are checking if the software works as expected. Security testing is not about correct or incorrect – you are checking if an attacker can compromise the software.

Checking the source code for suspicious constructs and data flows can provide a starting point but comes with the inevitable problem of false positives. Static analysis tools don't know the developer's intent, so many of the warnings they generate will be irrelevant. In any case, source code checking by itself is not enough because real-life attackers will be trying to compromise the entire resulting application as built and executed. This makes dynamic testing a vital part of the security testing process as the method that most closely approximates the actions of malicious actors. The most reliable approach here is manual vulnerability assessment and penetration testing, but this is slow and costly – not something you can fit into a weekly release schedule.

“ Dynamic testing is a vital part of the security testing process as the method that most closely approximates the actions of real-life attackers.

The move to rapid development and deployment in a continuous software development lifecycle (SDLC) was only made possible by extensive automation across all stages of development and operations. To be effective without interfering with release schedules, web application security testing must be automated to the same level to keep up with the breakneck pace of development. In this context, automation encounters two major obstacles. First, you need to find tools that are accurate and advanced enough to run automated security testing and deliver useful results. Once you have all those results, you then need to automate processing and management in a way that meets all the requirements of a modern SDLC.

Why Traditional Pre-Release Security Testing Is Not Enough

With the growing importance of web application security, organizations have scrambled to add security to their agile development workflows. However, simply running occasional security testing or even adding manual vulnerability testing at the pre-release stage does little to measurably improve security. You might be able to find vulnerabilities and fix individual issues, but there is no realistic way of getting a large application environment to an acceptable level of security and consistently keeping it there.

With data breaches and other cyberattacks now acknowledged as a major business risk, sweeping security bugs under the carpet is no longer an option. If a critical vulnerability is only discovered during pre-release testing, the whole release has to be put on hold while the issue is verified, triaged, fixed, and retested – and that consumes time and money. The earlier vulnerabilities are found, the cheaper they are to fix. Shifting security left, i.e. to earlier stages of the development pipeline, has become a practical necessity to avoid the costs and delays associated with late-stage security testing.

THE SHORTCOMINGS OF ISOLATED SECURITY TESTING IN AGILE DEVELOPMENT

NOT AUTOMATED
Manual issue processing by small security teams becomes a major bottleneck to development and releases. A large organization can have thousands of developers but only a handful of web security staff.

NOT INTEGRATED
Security teams and developers often use separate tools and workflows. Security engineers need to manually process test results, create tickets, provide feedback, and follow up on fixes.

NOT SCALABLE
Agile development and operations environments can grow rapidly to respond to business requirements. Manual security testing and issue management can't scale and adapt to the same level.

NOT AGILE
When vulnerabilities are detected late in the development cycle, it takes far longer to isolate, assign, and fix issues. This makes fixing more laborious and less effective, with a high risk of vulnerabilities resurfacing in the future.

The Solution: Integrating DAST into Development Workflows

To be truly effective and agile, modern web application security testing must be fully integrated into the software development lifecycle and cover the entire attack surface of the application as deployed. This means that any application security program must include at least dynamic testing as that is the only feasible approach to testing everything that an attacker could possibly target.

Approaches to Integrating Security into the SDLC

In traditional waterfall-based development, there was a clear-cut division of labor in application security: static testing (SAST) and other code-level checks were done during development, dynamic testing (DAST) was done in QA and staging. When it came to building security into the software development lifecycle, SAST was initially the natural choice as it was relatively easy to add into existing build scripts and test suites and could (at least in theory) provide full source code coverage. However, actually implementing static testing in the context of modern web development can be challenging and the results frequently need a lot of tweaking to minimize false positives.

As we have already seen, checking your own source code is not enough to provide full coverage in modern web applications. Even if you add software composition analysis (SCA) to ensure that known vulnerable dependencies are not brought in, some form of dynamic testing is still essential to test the final application as built and deployed, including all modifications, component interactions, and dynamic dependencies. At the same time, leaving dynamic testing to be done by security specialists in the late stages of development is inefficient and doesn't scale. The obvious solution to both problems would be to introduce dynamic testing early on in the development cycle. Modern web development workflows up to and including CI/CD (Continuous Integration/Continuous Deployment) pipelines already make it technically possible to introduce DAST from the very start – all you need is a DAST solution that supports this.

“ Leaving dynamic testing to be done by security specialists in the late stages of development is inefficient and doesn't scale.

The Importance of Dynamic Testing

Apart from its core purpose of catching runtime issues, dynamic security testing is essential for web applications because it reveals and probes the actual attack surface of the application as it is visible to attackers. This includes not only your own application code but also all external dependencies and components brought into the application, whether directly in your code or in the underlying framework. You can test security without worrying about source code access or even knowing all your code repositories. In fact, you can even test dynamic dependencies that are only brought in at runtime and legacy or third-party components that you simply don't have the source code for.

All this gives you maximum visibility and also makes DAST extremely fast and easy to set up, run, and maintain – you just point the tool at a URL and start scanning, regardless of the application architecture or language. Even if the underlying language changes, the same dynamic scanner will still work without having to buy and configure new tools. Apart from uncovering exploitable vulnerabilities, modern dynamic scanners can also discover web assets to scan, return information about security misconfigurations, detect outdated software versions, and recommend security best practices to build a defense in depth.

Debunking DAST Myths

Early dynamic scanners were not standalone solutions but rather simple utilities to support the work of penetration testers. The first DAST products based on these simple tools had many limitations, partly due to the simplicity of the mostly static websites they were designed to scan. As web technologies advanced and web applications became ever more complex, the limitations of legacy DAST became obvious, leading to the lingering misconception that DAST can only handle simple static sites and find trivial issues.

Fortunately, modern DAST has come a long way from these rudimentary tools. A quality DAST solution can accurately scan any modern web application, including JavaScript-heavy single page applications (SPAs). It can handle automated authentication to ensure that high-value pages hidden behind login forms are also tested. It provides advanced crawling and discovery to detect and scan as many web assets as possible. And perhaps most importantly, especially in enterprise environments, it can be integrated into the SDLC to automate and streamline application security testing.

> “ A quality DAST solution can accurately scan any modern web application, including JavaScript-heavy SPAs.

PREREQUISITES FOR INTEGRATING DAST INTO THE SDLC

TRUSWORTHY
To automate security testing, DAST needs to give you confidence that you are not automating false positive results.

ACCURATE
To get measurable security improvements, DAST must be advanced enough to find actionable issues and provide fix guidance.

COMPREHENSIVE
To ensure maximum test coverage and quick detection, DAST needs to scan every corner of a modern web application (including authenticated scanning).

WORKFLOW-READY
To minimize security testing overhead and maximize adoption, DAST needs to integrate with existing development and collaboration tools.

FLEXIBLE
To be effective and adaptable in modern web environments, DAST needs to work with multiple web technologies and languages out-of-the-box.

The Benefits of Using Netsparker in Your SDLC

As an industry-leading DAST solution, Netsparker provides many options for integrating into modern web development workflows. Fueled by over a decade of relentless security research and development, Netsparker uses Proof-Based Scanning™ to automatically detect a wide array of web vulnerabilities and prove many common issues by safely extracting sample data as proof. This is combined with deep runtime insights from the additional IAST module to isolate issues with pinpoint accuracy and extend testing to assets that are inaccessible to crawlers. To ensure that all these capabilities bring real value, Netsparker also provides out-of-the-box integration with popular issue trackers, automation tools, and collaboration platforms to minimize the time and effort required to deploy a working solution.

Effective and Scalable Application Security Testing

To be an effective part of an agile DevOps pipeline, security testing must be automated as far as possible. In order to feed automated vulnerability scanning results into development workflows, you have to be confident that you are not raising false alarms. This is the biggest benefit of Netsparker's proprietary Proof-Based Scanning technology: proven results are 100% real issues that can go straight into the developers' issue tracker without burdening the security team with confirmations and triaging. In many cases, it is even possible to assign fix tasks directly to the developer who made the vulnerable commit and go from detection to bug ticket in a matter of minutes. This is a massive time-saver that also eliminates the inefficient (and annoying) practice of having to fix other people's code.

“ With Netsparker's proprietary Proof-Based Scanning technology, proven results are 100% real issues that can go straight into the developers' bug tracker.

All too often, new tools are bought, deployed, and then not used to their full potential (or at all) because they don't fit into existing workflows and toolsets. Netsparker integrates with popular systems that development and testing teams already use, so you can provide developers with actionable tickets directly in their current working environment. Combined with detailed technical information and issue remediation guidelines, this is vital to streamline communication and trim off the unnecessary back-and-forth that often plagues manual security testing processes.

This opens the way to true scalability, even across hundreds or thousands of websites, applications, and services. Enterprise-scale web application security requires you to automate everything that can possibly be automated, simply because otherwise you will end up with an eternal backlog of issues that just seems to keep growing. Netsparker combines efficient and accurate vulnerability scanning with confident automation and deep SDLC integration so your security testing can keep pace with development.

Improved Security in the Long Run

Web applications keep growing in size and complexity and multiplying at an astounding rate. If you are serious about security, you need to face not only the daunting task of finding and eliminating vulnerabilities in your existing application environment but also the challenge of constantly maintaining a good security posture in the future. To do this, you need to build application security directly into your SDLC and shift as much work as possible away from small security teams and towards large developer teams.

Netsparker's unique combination of integration, automation, and fully trusted results with Proof-Based Scanning allows you to build DAST into your development pipeline to eliminate vulnerabilities before they can reach production. Because developers get accurate feedback in real-time about the security bugs they introduce, they can not only fix them quickly but also avoid making the same mistakes in the future. This fosters a security-focused mindset among developers to improve application security in the long run.

Automated application security testing also alleviates another major drawback of pre-release security testing by a separate team: internal friction and inefficiencies related to reporting and resolving security issues. When developers get proven security bug reports from an accurate automated tool delivered directly into their favorite ticketing system, the often adversarial attitude to working with security teams changes to one of efficient collaboration. In more mature organizations, it may even be possible to handle all application security issues at the development team level, with the core security team now responsible for high-level vulnerability management and policy.

“ You need to build application security directly into your SDLC and shift as much work as possible away from small security teams and towards large developer teams.

Real Value and Tangible Savings

The reason for investing in security testing is not merely to run tests, but to improve security. The time-to-value calculation for an application security product starts from the moment of purchase to the first measurable security improvements. Netsparker combines the advantages inherent to DAST technology, such as ease of deployment and broad scope of testing, with its own unique benefits and IAST capabilities to deliver value within days of deployment – and continue to deliver even more value as deeper customization and integration proceeds.

As will be demonstrated in the next section, the flexibility of Netsparker means that you can get measurable security improvements at any level of SDLC maturity. Of course, the more mature your process, the deeper the benefits, but you can be sure that you are always getting maximum value from your current security workflow. If and when you decide to automate your existing processes, Netsparker's trusted vulnerability reports combined with integration and customization options help you make the transition.

Because you will be able to automate many manual processes and eliminate many inefficiencies related to team communication, you are likely to find that deploying Netsparker will reduce the cost of your application security program while increasing its effectiveness. Each vulnerability that is automatically confirmed with Proof-Based Scanning means one less manual verification task for a security engineer, allowing your security staff to focus on vulnerability management, security education, and analyzing complex vulnerabilities that really need human expertise. All this translates to fewer man-hours spent on tasks that could be automated, improved security, and more satisfied employees.

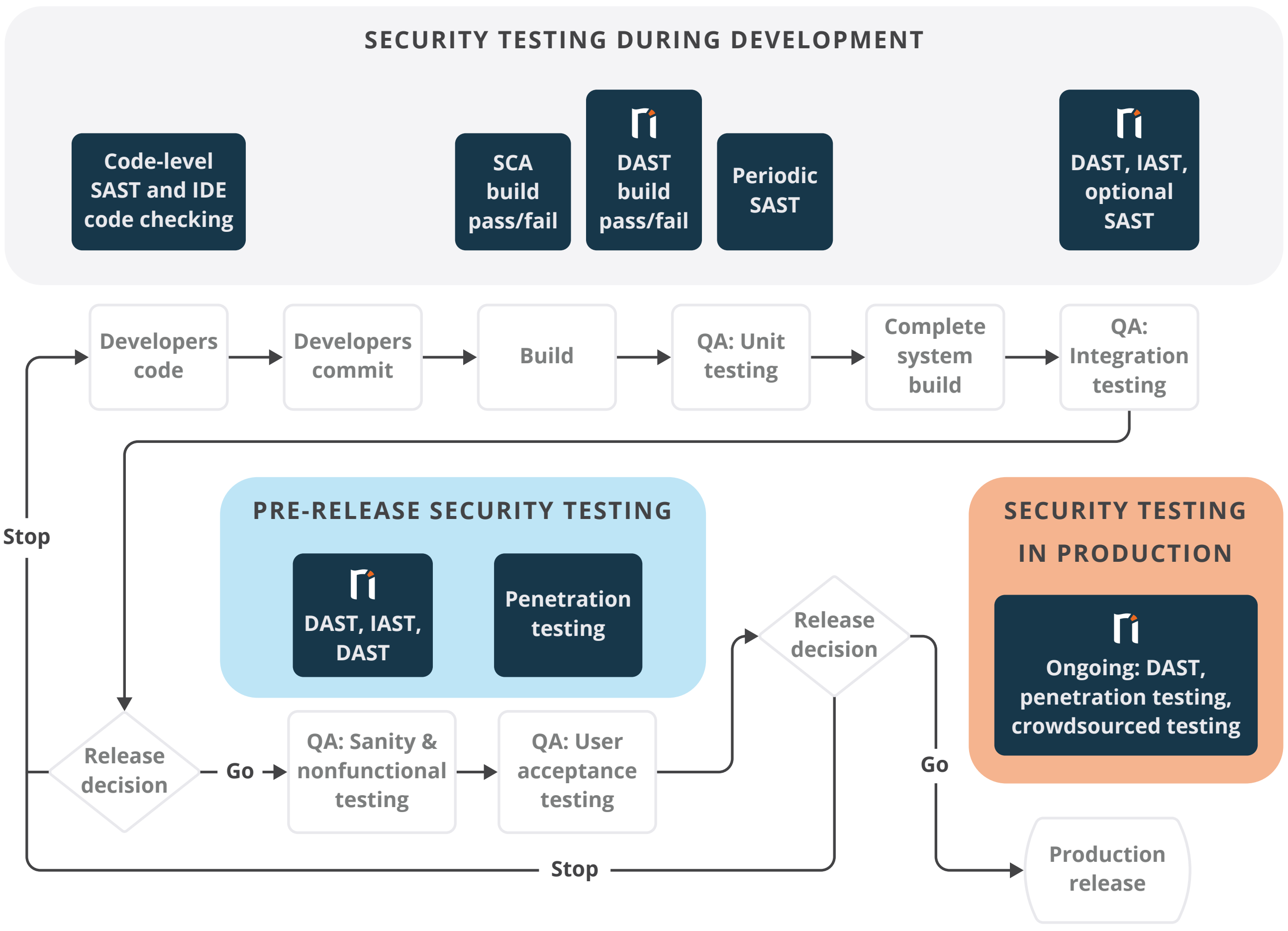
“ Deploying Netsparker will reduce the cost of your application security program while increasing its effectiveness.

Selected Scenarios for Using Netsparker in the SDLC

The versatility of dynamic testing coupled with the integration capabilities of modern tools makes it possible to use DAST both for automated penetration testing and at one or several stages of the software development lifecycle, depending on the scale, resources, requirements, and workflow maturity of the organization. Netsparker, in particular, has been designed with scalability and automation in mind, with its Proof-Based Scanning technology opening the way to confidently integrating scan results into development and testing workflows.

In this section, we will look at several options for building application security testing with Netsparker into modern web development processes.

WHERE YOU CAN USE NETSPARKER FOR SECURITY TESTING IN THE SDLC



SECURITY TESTING DURING DEVELOPMENT
Code-level SAST and IDE code checking
SCA build pass/fail
DAST build pass/fail
Periodic SAST
DAST, IAST, optional SAST
Developers code
Developers commit
Build
QA: Unit testing
Complete system build
QA: Integration testing
Stop
PRE-RELEASE SECURITY TESTING
DAST, IAST, DAST
Penetration testing
SECURITY TESTING IN PRODUCTION
Ongoing: DAST, penetration testing, crowdsourced testing
Release decision
Go
QA: Sanity & nonfunctional testing
QA: User acceptance testing
Release decision
Go
Stop
Production release

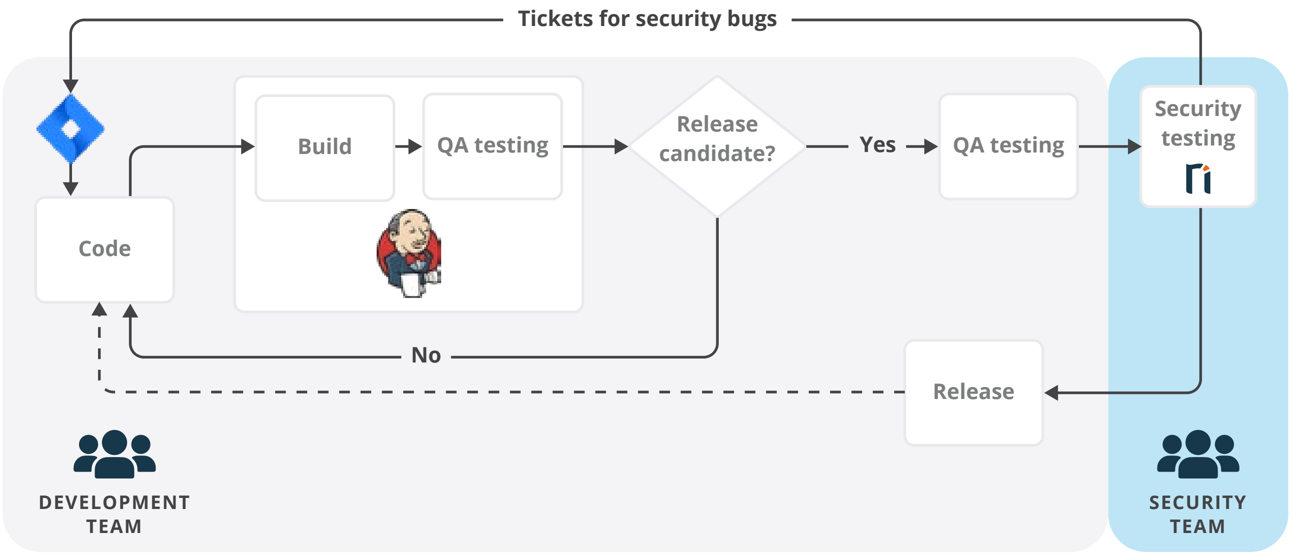
Scenario 1: Small to Medium-scale Agile Development

SDLC MATURITY LEVEL:	Agile development
DAST INTEGRATION LEVEL:	Low
BENEFITS FROM NETSPARKER:	Streamlined manual vulnerability scanning and management. Quickly add application security testing to any development setup.

Agile development relies heavily on automation to enable frequent or even continuous deployment. This means that most or all of the testing is also automated. When you need to add application security testing to an existing web development workflow but don't have the resources or the need for security automation, your security team can manually run Netsparker during pre-release testing as a valuable addition to penetration testing. To eliminate some of the manual work, you can also trigger scanning from build scripts, for example, for release candidate builds.

In this setup, the security team still handles all scanning, vulnerability management, and ticket creation. Netsparker streamlines this work by delivering accurate results and detailed vulnerability reports, so security engineers can have a picture of the current security posture and quickly create developer tickets for actionable vulnerabilities. Many high-severity issues are accompanied by proof that a vulnerability is exploitable, so security staff don't need to verify them manually before creating a ticket.

SCENARIO 1: ADDING APPLICATION SECURITY TESTING TO AGILE DEVELOPMENT



Scenario 2: Integrating Application Security into DevOps

SDLC MATURITY LEVEL:	DevOps
DAST INTEGRATION LEVEL:	Medium
BENEFITS FROM NETSPARKER:	Automatically confirmed issues are handled without security team interaction. Build fails provide rapid feedback for quick fixes and improved code quality.

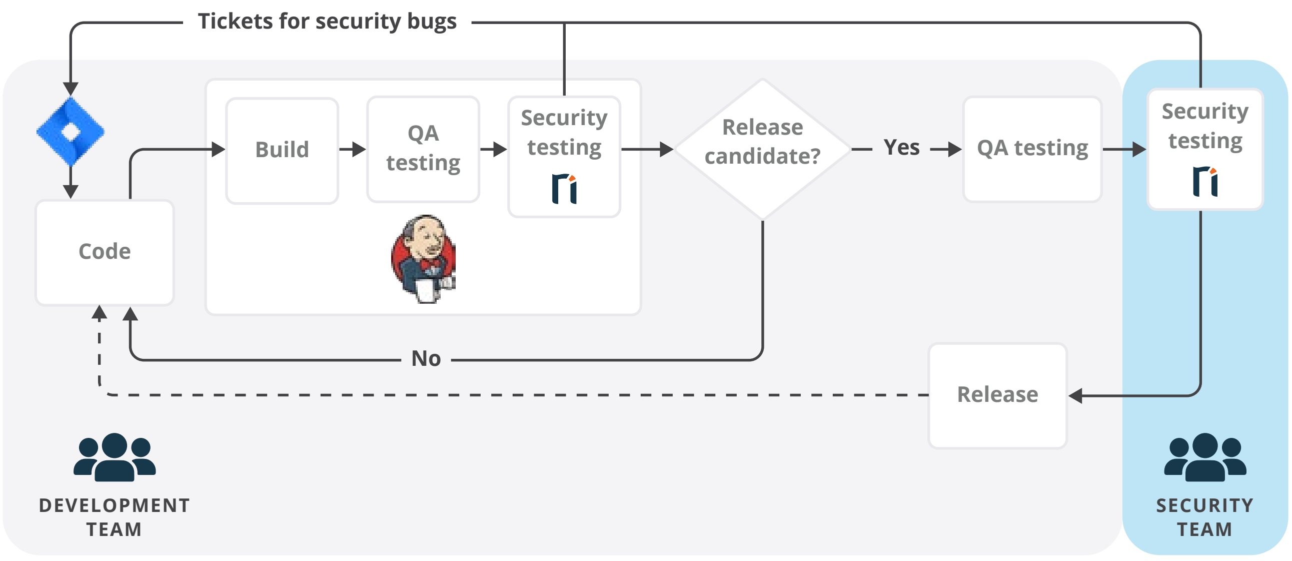
In many modern organizations, developing and maintaining internal web applications has been merged into a DevOps cycle, with heavy reliance on automation tools like Jenkins and issue trackers such as Jira. To be effective in this dynamic setup, application security has to be integrated with existing tools and automated as much as possible.

Netsparker features out-of-the-box integration with both Jenkins and Jira, allowing you to trigger scanning at build level and automatically fail builds if security vulnerabilities are found (with configurable fail thresholds based on severity). Jira integration makes it possible to submit bug tickets directly from Netsparker and automatically retest security bug fixes.

For vulnerabilities that are automatically verified using Proof-Based Scanning, Netsparker can automatically add tickets for the right developer team, providing detailed information and proof that an issue is real and not a false positive.

With verified vulnerabilities going directly to the developers, the security team can focus on efficient vulnerability management and analyzing complex issues that need human expertise. Even though security is still separated from development and the small security team can be a potential bottleneck, accurate and confident automation makes this scenario workable for many large organizations today.

SCENARIO 2: DEVOPS WITH INTEGRATED APPLICATION SECURITY TESTING



Scenario 3: Fully Integrated Application Security

SDLC MATURITY LEVEL:	CI/CD with DevSecOps
DAST INTEGRATION LEVEL:	High
BENEFITS FROM NETSPARKER:	Enables application security as a service for development teams. Automation and integration not only at the level of tooling but also internal structures.

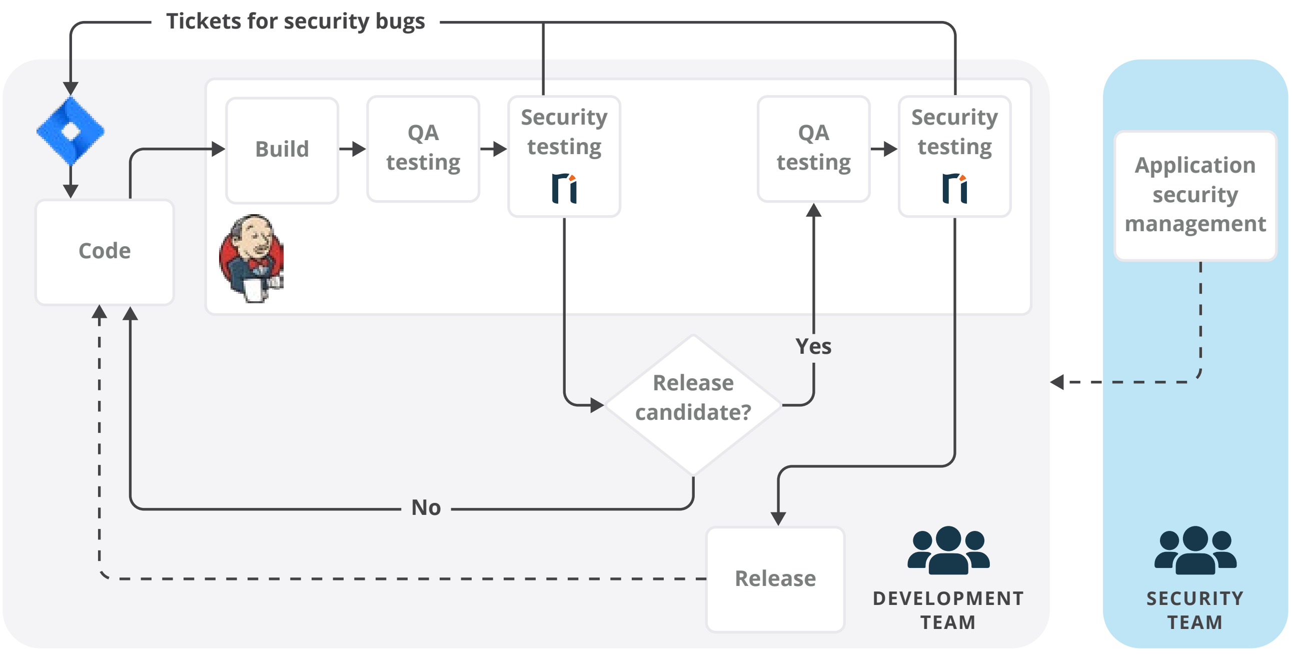
Large web development operations can have thousands of websites and applications being developed by hundreds of developers in dozens of teams. In such environments, application security can only be enforced effectively if moved to the development team level. To do this, the core security team might set up Netsparker as an automated and auto-provisioned self-service platform for development teams and development-level AppSec engineers.

In this setup, scanning is triggered automatically for each build and proven vulnerabilities are immediately added as Jira issues for the developer who made the relevant commit. Scan automation using Jenkins can include build pass/fail (also supported for GitLab) and fix submissions can also be automatically retested.

At later stages, vulnerabilities that Netsparker has identified with a high level of confidence but cannot prove are automatically submitted to development-level security engineers for verification.

For this integration scenario, Netsparker creates the technical environment to take full advantage of the internal transformation to a security-first culture and mindset. Security engineers are now distributed among development teams and work side-by-side with developers to determine the most effective fixes for a specific codebase and avoid similar issues in the future. The central security team no longer deals with specific issues but focuses on managing the overall security posture, defining policies, and maintaining the tool platform based on Netsparker.

SCENARIO 3: FULLY AUTOMATED APPLICATION SECURITY TESTING



Summary

As web applications grow ever more complex and opaque, it is becoming clear that comprehensive dynamic testing is an indispensable part of any application security program. At the same time, effective security testing must be fully integrated with agile development workflows that are now the norm in web application development – and legacy DAST products intended for manual scanning just won't cut it. To combine these two pressing needs, you need a quality DAST solution that can integrate deeply into the SDLC, such as Netsparker.

Fueled by over a decade of relentless security research and development dedicated to dynamic testing, Netsparker combines accurate and actionable vulnerability scanning results with an embedded interactive testing module and extensive integration and automation capabilities. The result is a powerful and flexible DAST+IAST solution that can streamline and automate application security testing across the development and testing workflow.

With a modern solution like Netsparker, you absolutely can (and should) use DAST in your SDLC. The only decision you need to make is how you want to deploy it – and what to do with all the time that your security engineers and developers are going to save.

About

Netsparker is a comprehensive automated web security solution that includes web vulnerability scanning, vulnerability assessment, embedded interactive security testing, and vulnerability management. Its strongest points are scanning and crawling accuracy, rapid asset discovery technology, and integration with leading issue management and CI/CD solutions.

The Netsparker scanner can find vulnerabilities in all types of modern and custom web applications, regardless of the architectures or platforms that they are based on. For major classes of vulnerabilities, the scanner can safely exploit the identified flaw and deliver proof that the issue is real and not a false positive, greatly improving automation and scalability.

Netsparker is designed for organizations that require a customizable and scalable solution for complex environments. Depending on the customer's deployment needs, Netsparker can be implemented as desktop software, a managed service, or an on-premise solution.

Netsparker is part of Invicti Security, a leader in dynamic and interactive application security testing.