Rocket® software

Vulnerability Scanning Methods:

From open systems to mainframe

Why vulnerability scanning matters across modern IT environments

Organizations today rely on a mix of distributed, cloud, web, and mainframe applications to deliver critical business services. As cyber threats continue to evolve, security teams require effective methods to identify vulnerabilities before they can be exploited. Vulnerability scanning has become a foundational component of modern cybersecurity programs, helping organizations proactively assess risk, prioritize remediation efforts, and support compliance requirements.

Over time, several approaches to application security testing have emerged, each designed to identify vulnerabilities at different stages of the software lifecycle. Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Interactive Application Security Testing (IAST) all provide unique perspectives on application security. Understanding how these methods work and where they fit within a broader security strategy is essential for building comprehensive protection across the enterprise.

For organizations running business-critical workloads on IBM Z® systems, vulnerability scanning presents additional challenges. Many traditional security testing tools were designed for open systems and web applications, leaving gaps when assessing vulnerabilities within the mainframe operating environment itself. This guide explores the most common vulnerability scanning methodologies and introduces Operating System Integrity Testing (OSIT), a purpose-built approach for identifying vulnerabilities within the mainframe operating system layer.

What is DAST?

Dynamic Application Security Testing (DAST) is an “outside-in” security approach that tests running web applications and APIs for vulnerabilities by simulating external attacks, such as SQL injection or cross-site scripting. It is a black-box testing method, meaning it does not require access to the source code.

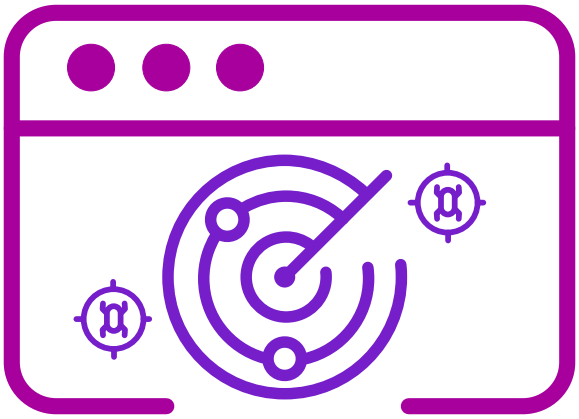
Key aspects of DAST

- **How it works:** DAST tools crawl the web application, identify inputs, and send malicious requests to detect vulnerabilities, like how a malicious user would interact with the application.
- **When to use:** DAST is typically used in the testing or staging phase of the software development life cycle (SDLC), or on production applications, to identify security issues that only appear at runtime.
- **What it finds:** DAST is effective at identifying server misconfigurations, authentication issues, and security flaws in application components.
- **DAST vs. SAST:** Unlike Static Application Security Testing (SAST), which analyzes code from the inside, DAST tests from the outside, making it language-independent.

Common DAST tools include commercial solutions and open-source scanners, such as Microsoft’s RESTler, which helps find security bugs during development, as well as Veracode. **Mythos is a DAST & SAST tool.**

How does DAST work?

DAST works by simulating automated attacks on an application, mimicking a malicious attacker. The goal is to find outcomes or results that were not expected and could therefore be used by attackers to compromise an application. Since DAST tools don’t have internal information about the application or the source code, they attack just as an external hacker would — with the same limited knowledge and information about the application.



What is SAST?

Static application security testing (SAST) and dynamic application security testing (DAST) are both methods of testing for security vulnerabilities, but they're used very differently.

Here are some key differences between the two:

SAST vs. DAST

SAST and DAST techniques complement each other. Both need to be carried out for comprehensive testing.

SAST

White box security testing

- The tester has access to the underlying framework, design, and implementation.
- The application is tested from the inside out.
- This type of testing represents the developer approach.

Requires source code

- SAST doesn't require a deployed application.
- It analyzes the source code or binary without executing the application.

Finds vulnerabilities earlier in the SDLC

- The scan can be executed as soon as code is deemed feature-complete

Less expensive to fix vulnerabilities

- Since vulnerabilities are found earlier in the SLC, it's easier and faster to remediate them.
- Findings can often be fixed before the code enters the QA cycle.

Can't discover run-time and environment-related issues

- Since the tool scans static code, it can't discover run-time vulnerabilities.

Typically supports all kinds of software

- Examples include web applications, web services, and thick clients.

DAST

Black box security testing

- The tester has no knowledge of the technologies or frameworks that the application is built on.
- The application is tested from the outside in.
- This type of testing represents the hacker approach.

Requires a running application

- DAST doesn't require source code or binaries.
- It analyzes by executing the application.

Finds vulnerabilities toward the end of the SLC

- DAST doesn't require source code or Vulnerabilities can be discovered after the development cycle is complete.

More expensive to fix vulnerabilities

- Since vulnerabilities are found toward the end of the SLC, remediation often gets pushed into the next cycle.
- Critical vulnerabilities may be fixed as an emergency release.

Can discover run-time and environment-related issues

- Since the tool uses dynamic analysis on an application, it is able to find run-time vulnerabilities.

Typically scans only apps like web applications and web services

- DAST is not useful for other types of software.

What is IAST?

Interactive Application Security Testing (IAST) is a **runtime** security testing method that identifies vulnerabilities in web applications by analyzing code behavior **from the inside while the application is running.** It combines the strengths of SAST (scanning code) and DAST (testing in browser) to **detect issues with high accuracy and low false positives** during functional testing, making it ideal for CI/CD pipelines.

Key aspects of IAST

IAST uses sensors or agents deployed within the application server/runtime environment. These sensors monitor code execution, data flow, and HTTP requests in real-time.

- **Real-Time feedback:** It provides immediate, actionable feedback to developers during the QA/testing phase.
- **Accuracy:** Because IAST acts from inside the application, it can pinpoint the exact line of code responsible for a vulnerability, significantly reducing false positives.
- **Coverage:** It checks custom code, libraries, and frameworks

Benefits of IAST

IAST uses sensors or agents deployed within the application server/runtime environment. These sensors monitor code execution, data flow, and HTTP requests in real-time.

- **High precision:** It offers fewer false positives/negatives compared to traditional methods.
- **DevSecOps friendly:** It fits seamlessly into modern CI/CD pipelines, enabling automated security checks.
- **Reduced remediation cost:** By catching bugs early in testing rather than in production, it lowers the cost of fixing vulnerabilities.
- **Comprehensive insight:** It provides detailed information on vulnerabilities, including data flow, code location, and context, allowing for quick fixes.

Common IAST Tools & Vendors

- **Contrast Security:** Known for pioneering IAST with “sensors”.
- **Black Duck (Synopsys):** Focuses on comprehensive web application security testing.
- **CrowdStrike:** Offers solutions to protect against modern application threats.
- **OWASP DevSecOps Guideline:** Provides best practices for implementing IAST.

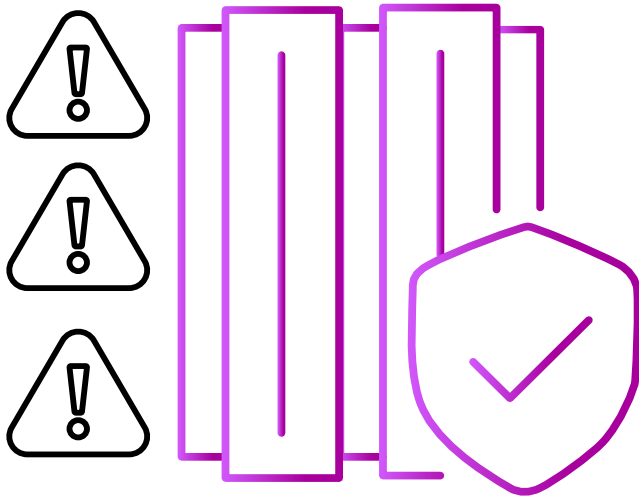
What is OSIT™?

VAP is OSIT

Operating System Security Testing (OSIT) is a **runtime** security testing method that identifies vulnerabilities in the mainframe operating system layer by analyzing code behavior **from the inside while the software is running.** It combines the strengths of SAST (scanning code) and DAST (testing in browser) to **detect issues with high accuracy and low false positives** during functional testing, making it ideal for mainframe integrity testing.

Benefits of OSIT - VAP

- **High precision:** It offers fewer false positives/negatives compared to traditional methods.
- **DevSecOps friendly:** It fits seamlessly into modern CI/CD pipelines, enabling automated security checks. CI/CD stands for Continuous Integration and Continuous Delivery (or Continuous Deployment). It enables teams to frequently run automated tests, and release software safely and quickly.
- **Reduced remediation cost:** By catching bugs early in testing rather than in production, it lowers the cost of fixing vulnerabilities.
- **Comprehensive insight:** It provides detailed information on vulnerabilities, including data flow, code location, and context, allowing for quick fixes.



Choosing the right vulnerability testing strategy

No single security testing methodology provides complete coverage of every potential vulnerability. SAST helps identify issues early in the development lifecycle, DAST evaluates applications from an external attacker's perspective, and IAST combines runtime analysis with application awareness to improve accuracy and remediation efforts. Together, these approaches help organizations strengthen application security and reduce risk.

However, organizations that depend on mainframe systems must also consider vulnerabilities that exist beyond the application layer. Traditional scanning methods may not provide sufficient visibility into operating system integrity, privileged access exposure, or platform-specific weaknesses that can affect critical business workloads.

Operating System Integrity Testing (OSIT), powered by Rocket's Vulnerability Assessment Program (VAP), extends vulnerability scanning into the mainframe operating environment, helping organizations identify security risks with greater precision and context. By combining traditional application security testing techniques with platform-specific integrity testing, organizations can build a more comprehensive approach to cybersecurity across both open systems and the mainframe.

About Rocket Software

Rocket Software is a global technology leader in modernization and a partner of choice that empowers the world's leading businesses on their modernization journeys, spanning core systems to the cloud. Trusted by over 12,500 customers and 750 partners, and with more than 3,200 global employees, Rocket Software enables customers to maximize their data, applications, and infrastructure to deliver critical services that power our modern world. Rocket Software is a privately held U.S. corporation headquartered in the Boston area with centers of excellence strategically located throughout North America, Europe, Asia and Australia. Rocket Software is a portfolio company of Bain Capital Private Equity.

Rocket® software

Modernization. **Without Disruption.**™

Visit RocketSoftware.com

MAR-18417_Broch_MFVulerabilityScanArchGuide_V3