CONTRAST SECURITY

EBOOK

# Authorization of IAST as an approved technique for application security testing

# TABLE OF CONTENTS

## Introduction

Standards organizations, industry groups and regulatory authorities have recognized that Interactive Application Security Testing (IAST) is an approved technique for Application Security Testing (AST).

Based on this authorization, a wide variety of organizations have selected IAST as their sole AST technology, and others have chosen to use a combination of other tools along with IAST. For organizations that want minimal effort, highly accurate results, instant feedback, contextual descriptions and massive scalability, IAST may be the right choice.

The references below specifically call out application security testing as a requirement to meet their cybersecurity standards and rules, that would apply across government and industry organizations. We are not aware of any standards that preclude the use of IAST for AST.

## Governance and compliance standards on IAST

Many compliance regulations and standards, such as EU DORA and NIST CSF 2.0, don't mention any specific technologies that could be leveraged to meet their requirements. Rather, they are designed to provide a framework for preparing organizations against the modern threat landscape. Here, IAST is particularly well-suited, since it has a very low false-positive rate and enables organizations to quickly discover and remediate vulnerabilities that actually pose a threat.

Still, many regulatory bodies, standards and authorities and industry groups specifically highlight the value that IAST provides, especially in comparison with Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST).

## NIST 800-53r5

Special Publication 800-53 Revision 5 from the National Institute of Standards and Technology (NIST), titled Security and Privacy Controls for Information Systems and Organizations, was created by NIST as part of its official mandate to develop "information security standards and guidelines, including minimum requirements for federal information systems."

Section SA-11 details approved alternative techniques for AST. IAST is detailed in SA-11(9). Organizations are free to choose whatever techniques they think are appropriate.

### (9) DEVELOPER TESTING AND EVALUATION
### INTERACTIVE APPLICATION SECURITY TESTING

**Require the developer of the system, system component or system service to employ interactive application security testing tools to identify flaws and document the results.**

**Discussion:** Interactive (also known as instrumentation-based) application security testing is a method of detecting vulnerabilities by observing applications as they run during testing. The use of instrumentation relies on direct measurements of the actual running applications and uses access to the code, user interaction, libraries, frameworks, backend connections and configurations to directly measure control effectiveness. When combined with analysis techniques, interactive application security testing can identify a broad range of potential vulnerabilities and confirm control effectiveness. Instrumentation-based testing works in real time and can be used continuously throughout the system development life cycle.

**Related Controls**: None.

**References:** [ISO 15408-3], [SP 800-30], [SP 800-53A], [SP 800-154], [SP 800-160-1].

## NIST minimum standard for application security testing

This document was prepared in response to the CyberSecurity Executive Order 14028 and recommends using both static and interactive testing to verify applications. They consider IAST a form of DAST, which is technically correct.

### 2.10 Web application scanning

If the software provides a web service, use a Dynamic Application Security Testing (DAST) tool, e.g., web application scanner, see Sec. 3.5, or Interactive Application Security Testing (IAST) tool to detect vulnerabilities.

## PCI Software Security Standard (SSS)

This widely used standard for Payment Cardholder Information (PCI) from the PCI Council explicitly recognizes IAST as an approved technique for testing security. IAST is specifically listed as an approved technique in the assessment procedures and under requirement 10.2.

**Assessment procedures and test requirements**

To facilitate validation of their software, software vendors must produce appropriate evidence that confirms they have satisfied the control objectives defined within this standard. The test requirements identified for each control objective describe the expected activities to be performed to validate whether the software and/or software vendor have met the objective. Where sub-bullets are specified in a control objective or test requirement, each bullet must be satisfied as part of the validation. In addition, where terms such as "periodic," "appropriate," and "reasonable" are used in the test requirement, it is the software vendor's responsibility to define and defend its decisions regarding the frequency, robustness, and maturity of the implemented controls or processes.

**Test requirements typically include the following activities:**

- **Examine:** The assessor critically evaluates data evidence. Common examples include software design and architecture documents (electronic or physical), source code, configuration and metadata files, and security-testing results.
- **Interview:** The assessor converses with individual personnel. The purposes of such interviews may include determining how an activity is performed, whether an activity is performed as defined, and whether personnel have particular knowledge or understanding of applicable policies, processes, responsibilities or concepts.
- **Test:** The assessor evaluates the software code or the operation of the software using a variety of security-testing tools and techniques. At a minimum, assessors must use the appropriate combination of static and dynamic analyses to validate each control objective. Examples of such tools and techniques might include the use of automated Static Analysis Security Testing (SAST), Dynamic Analysis Security Testing (DAST), Interactive Application Security Testing (IAST), and Software Composition Analysis (SCA) tools as well as manual techniques such as manual code reviews and penetration testing.

## Monetary Authority of Singapore Technology Risk Management Standard (TRM)

The Monetary Authority of Singapore (MAS) is both Singapore's integrated financial regulator and central bank. MAS, which regulates some of the world's largest financial institutions states in its MAS Tech Risk Management Guidelines section 6.1.6, that it is essential for the financial institution to establish comprehensive strategy to perform Application Security (AppSec) validation and testing. The financial institution may use a mixture of static, dynamic and interactive AST methods to validate the security of the software application. The software validation and testing rules should be reviewed periodically and kept current.

The TRM section 6.1.6 specifically authorizes the use of IAST and recognizes that IAST involves "a combination of SAST and DAST techniques to analyze application codes, runtime controls libraries, requests and responses, as well as data and control flows to identify vulnerabilities in an IT system."

**6.1.6**

It is essential for the FI to establish a comprehensive strategy to perform application security validation and testing. The FI may use a mixture of static, dynamic and interactive application security testing methods (refer to Annex A on Application Security Testing) to validate the security of the software application. The software validation and testing rules should be reviewed periodically and kept current.

**Interactive Application Security Testing**

Interactive Application Security Testing (IAST) involves a combination of SAST and DAST techniques to analyse application codes, run-time controls libraries, requests and responses, as well as data and control flows and identify vulnerabilities in an IT system.

## GSA Application Security Testing (AST) buyers guide

The General Services Administration (GSA) provides consumer guidance on a number of topics, including technology.

GSA recognizes IAST as an approved technique as an alternative to SAST that delivers "both static and dynamic visibility" and "offers a high degree of testing accuracy including low false negative rates (failing to detect risk that exists) and low false-positive rates (reporting a risk which does not actually exist)."

### 7.1.3 Interactive Application Security Testing (IAST)

IAST tools analyze code for security vulnerabilities while the application is run by an automated test, human tester, or any activity "interacting" with the application's functionality. It searches for known vulnerabilities inside the application's functions by simulating the various scenarios in which a user runs or interacts with the application. The analysis is conducted from the inside of the application, which provides an ideal vantage point to perform security testing. More specifically, the implementation relies on an agent that injects functionality in certain points of the execution of the application.

**GSA notes some benefits of SAST and DAST, but they also call out the downsides of both technologies:**

## Cons of **SAST:**

- Not capable of identifying vulnerabilities in dynamic environments.
- High risk of reporting false positives.
- Code analyzers must be tuned and configured.
- Lacks third-party component analysis.
- Since the report is static, the results are quickly outdated.
- Limited security coverage.
- The scanning process can be lengthy which slows down the development process.
- Not applicable for use in production environment.

## Cons of **DAST:**

- Does not find the exact location of a vulnerability in the code.
- Security knowledge is needed to interpret reports.
- Tests can be time consuming.
- Lack of zero-day vulnerability support.
- Finds vulnerabilities late in the SDLC, or after the development cycle is complete.
- Some DAST tools may not be well adapted to support DevSecOps.

## GSA perspective of IAST

### Pros of **IAST:**

- Effectively pushes testing toward the early stages of software development (test shifting left), so problems are caught earlier in the development cycle, reducing remediation costs and delays.
- Provides detailed information (including lines of code) to help development and security teams triage test results.
- Performs analysis from within applications and has access to application code, runtime control and dataflow information, memory and stack trace information, Hypertext Transfer Protocol (HTTP) requests and responses, and libraries, frameworks and other components (via an SCA tool). This analysis allows developers to pinpoint the source of an identified vulnerability and fix it quickly.
- Integrates easily into Continuous Integration (CI) and Continuous Development (CD) tools.
- Offers a high degree of testing accuracy including low false negative rates (failing to detect risk that exists) and low false-positive rates (reporting a risk which does not actually exist).
- Allows for earlier, less costly fixes.
- Delivers both static and dynamic visibility.
- Useful during all phases of the SDLC.

### Pros of **IAST:**

- Limited to the discovery of different flaw types in comparison to DAST and SAST.
- Compatible with only major programming languages.
- Contains non-blocking functionality, meaning that even when a risk is detected, the execution flow continues in the server.
- IAST based scanners cannot operate on their own and almost always require an additional external testing component in the form of a DAST scanner.

## Industry guidelines around IAST

**Gartner — "Streamline Your DevSecOps Profile" — Mark Horvath, Dale Gardner and Aaron Lord (9/17/2024)**

Recent DevSecOps guidance from Gartner analysts acknowledges the hybrid nature of IAST, while also noting its better efficacy. This is why they recommend using IAST instead of SAST and DAST for modern software development.

> As an alternative to traditional SAST and DAST, use Interactive Application Security Testing (IAST) where possible. IAST incorporates attributes of both SAST and DAST, leveraging instrumentation of the application during testing. This provides both outside-in and inside-out visibility. This combination enables IAST approaches to provide a better balance of efficacy — the reduced false positives of DAST with the precise line of code and code coverage visibility of SAST. However, this approach remains limited in platform support and requires that the application be assembled in a running state.

This OWASP DSO guide lists IAST as one of the approved vulnerability testing techniques. The standard recommends IAST technology as being more flexible, as it is more accurate, doesn't require access to source code and provides results in seconds.

## OWASP DevSecOps guide v. 0.2

### Interactive Application Security Testing

**IAST** is an application security testing method that tests the application while the app is run by an automated test, human tester or any activity "interacting" with the application functionality.

The core of an IAST tool is sensor modules, software libraries included in the application code. These sensor modules keep track of application behavior while the interactive tests are running. If a vulnerability is detected, an alert will be sent.

The process and feedback are done in real time in your Integrated Development Environment (IDE), Continuous Integration (CI) environment, or quality assurance, or while in production. The sensors have access to:

- Entire code
- Dataflow and control flow
- System configuration data
- Web components
- Back-end connection data

Examples of such vulnerabilities could be hardcoding API keys in cleartext, not sanitizing your users inputs or using connections without SSL encryption.

## IAST vs **SAST**

**Static application security testing** method examine source code in a non-runtime environment early in the SDLC. They look for suspicious code patterns that indicate security risks. Even though they are easy to deploy, SASTs throw too many false positives because SASTs do not take into account the presence of other security countermeasures and they lack visibility during runtime. SAST tools normally run inside the IDE as part of the compilation phase and introduce delays as the scan process takes time to finish. IASTS are more flexible than SASTs because they are applicable in production runtime environments (SASTs require direct access to the source code).

## IAST vs **DAST**

**Dynamic application security testing** method works like a black box scanner that executes requests against the application to find security issues. DASTs look at the applications from the exterior and determine the presence of risks by looking at the response (including body and headers) of the server to a battery of tests, but DASTs have no visibility of the internal workings of the app. Furthermore, DAST tests are hard to automate, because DASTs must be operated by experienced appsec teams, such as penetration testers, to be truly useful. Forrester estimates that the duration of a DAST scan can take around 5 to 7 days, while testing with IAST is a real-time (zero minutes) operation.

## Open Software Application Maturity Model (OpenSAMM)

The verification/security testing section of OpenSAMM recognizes IAST as a hybrid technique that combines the strengths of both SAST and DAST approaches, delivering better accuracy.

Application security testing can be performed statically, by inspecting an application's source code without running it, or dynamically by simply observing the application's behavior in response to various input conditions. The former approach is often referred to as Static Application Security Testing (SAST), the latter as Dynamic Application Security Testing (DAST). A hybrid approach, known as Interactive Application Security Testing (IAST), combines the strengths of both approaches (at the cost of additional overhead) by dynamically testing automatically instrumented applications, allowing accurate monitoring of the application's internal state in response to external input.

## OWASP Top Ten 2021

The venerable Top 10, most recently released in 2021, specifically lists IAST as a recommended approach for security testing applications and APIs.

Some of the more common injections are SQL, NoSQL, OS command, Object Relational Mapping (ORM), LDAP and Expression Language (EL) or Object Graph Navigation Library (OGNL) injection. The concept is identical among all interpreters. Source code review is the best method of detecting if applications are vulnerable to injections. Automated testing of all parameters, headers, URL, cookies, JSON, SOAP and XML data inputs is strongly encouraged. Organizations can include Static (SAST), Dynamic (DAST), and Interactive (IAST) Application Security Testing tools into the CI/CD pipeline to identify introduced injection flaws before production deployment.

## Taking the wide view: Understanding the true value of IAST

- IAST is a proactive application analysis technique that continuously monitors applications during runtime in both development and in-production environments and leverages advanced analytical capabilities to instantly identify and report security weaknesses found in custom code and third-party libraries.

- IAST is a runtime testing approach that combines the best-of-breed from both static and dynamic analysis techniques with intelligent sensors that can directly monitor and observe application behavior during execution.

- IAST tools operate during the execution of an application. They are deployed within the application to continuously monitor its behavior, data flows and interactions with external systems. This dynamic approach provides real-time feedback on security vulnerabilities, including issues that SAST often misses due to its static nature.

## When to use IAST

IAST's real-time analysis and comprehensive coverage make it a compelling solution for application security testing. Here's a breakdown of when it should be considered:

### Pre-production testing

IAST is essential for pre-production testing. Its dynamic analysis uncovers a wide range of technical vulnerabilities, including runtime vulnerabilities that SAST often misses.

### Augmented automated testing

IAST can leverage existing automated testing pipelines to turn any runtime tests — such as integration tests and end-to-end tests — into security tests to gain further security insights.

### Security monitoring

IAST tools empower robust security monitoring in all runtime environments, allowing real-time detection and response to security incidents.

### Comprehensive security strategy

For organizations seeking the most thorough protection, IAST can be used throughout the Software Development Life Cycle (SDLC), identifying potential security issues from the earliest stages to production.

## When to use IAST (cont.)

### Live production environments

IAST should be conducted in both development and live production environments to ensure comprehensive protection. In development, IAST helps identify vulnerabilities early, reducing the cost and complexity of fixing security flaws before deployment. This proactive approach improves code security without slowing development. In production, IAST provides real-time monitoring, detecting threats that may arise from new attack vectors or evolving code changes. Continuous testing in both environments ensures ongoing security, compliance and resilience against cyber threats. By integrating IAST throughout the software lifecycle, organizations minimize risks and maintain a strong security posture.

IAST dynamic analysis and real-time capabilities make it the ideal tool for pre-production application security testing. IAST's ability to identify runtime vulnerabilities and to provide in-depth monitoring in production environments give AppSec teams unparalleled application protection. While SAST can be useful for initial code scans during early development, organizations seeking the most robust security posture should prioritize IAST.



6800 Koll Center Parkway
Ste. 235
Pleasanton, CA 94566
Phone: 888.371.1333