English

- Russian
- Korean

- Support

positive technologies

English

- Russian
- Korean

- Solutions
  ICS/SCADA

  Critical infrastructure on the frontline

  Vulnerability Management

  Stop being an easy target

  Financial Services

  Can your security keep up with you?

  Protection from targeted attacks (anti-apt)

  Early detection, rapid investigation

  PT Industrial Cybersecurity Suite

  PT ICS is an integrated platform for cyberthreat detection and response in industrial systems

  Utilities

  Industrial-grade cybersecurity

  ERP Security

  Take control of your ERP security

  Security Compliance

  Turn policies into protection

  View all →
- Products
  MaxPatrol 8

  Vulnerability and compliance management system.

  MaxPatrol SIEM

  Knows your infrastructure, delivers pinpoint detection.

  PT Application Firewall

Intelligent protection of business applications.
PT Application Inspector

Source code analysis tool.

PT ISIM

Cyberthreat detection and incident response in ICS.
PT Network Attack Discovery

NDR system to detect attacks on the perimeter and inside the network.

PT Sandbox

Advanced sandbox with customizable virtual environments

XSpider

Vulnerability scanner.

MaxPatrol VM

Next-generation vulnerability management system.

MaxPatrol SIEM All-in-One

Full-featured SIEM for mid-sized IT infrastructures.

PT MultiScanner

Multilayered protection against malware attacks.

PT BlackBox

Dynamic application security testing tool

View all →

- Services
ICS/SCADA Security Assessment

Full Range of ICS-specific Security Services

ATM Security Assessments

Uncover Your Weaknesses

Web Application Security Services

Black Box and White Box Analysis

Mobile Application Security Services

Security Analysis and Compliance Audit

Custom Application Security Services

Independent Expert Analysis of Your Source Code

SAST, DAST, IAST, and RASP: how to choose?

Published on August 2, 2019

At the most basic level, application testing is aimed to rule out the possibility of malfunctioning code and to ensure the application runs smoothly after development. Squashing any bugs early on, preferably before they are baked into a final software release, is a challenge many developers face. Keeping the security level on a running application continuously tested will also further save organizations from financial and reputational damage.

A number of technologies have emerged on the long road to a well-designed and secure application. Employing static application security testing (SAST) allows the ability to catch defects early on in development. Dynamic application security testing (DAST) provides an outside perspective on the application before it goes live. Then, interactive application security testing (IAST) uses software instrumentation to analyze running applications. And finally, runtime application self-protection (RASP) can sense an attack happening and implement necessary measures.

If you need help deciding what method suits your needs, here are the benefits and drawbacks of each.

- Static application security testing (SAST)

Static application security testing (SAST)

SAST is also known as white-box testing, meaning it tests the internal structures or workings of an application, as opposed to its functionality. It operates at the same level as the source code in order to detect vulnerabilities. Since the SAST analysis is conducted before code compilation and without executing it, this tool can be applied early on in the software development life cycle (SDLC). Most SAST tools support the major web languages: PHP, Java, and .Net, and some form of C, C++, or C#.

The advantages of SAST include:

- SAST tools discover highly complex vulnerabilities during the first stages of development, which can be resolved quickly.
- Since it establishes the specifics of an issue, including the code line, it makes remediation simpler.
- It can be integrated into the existing environment at different points of the software development cycle.
- It takes little to examine code and compares favorably to manual audits.

The drawbacks to SAST are the following:

- Not all companies or individuals are willing to provide data for binary or byte-code and source code analysis.
- Deploying the technology at scale may be challenging.
- It tends to model code behavior inaccurately. Therefore, developers have to deal with many false positives and false negatives.
- Dynamically typed languages pose challenges; SAST tools need to semantically understand many moving pieces of the code that might be written in different programming languages.
- It can't test the application in the real environment, so vulnerabilities in application logic or insecure configuration are not detectable.

SAST tools are a very valuable technology but not a substitute for other methods. Developers would utilize a combination of techniques throughout the process to conduct assessments and catch flaws before going into production.

Dynamic application security testing (DAST)

DAST is a black-box testing method, meaning it is performed from the outside in. The principle revolves around introducing faults to test code paths on an application. For instance, it can use threat data feeds to detect malicious activity. DAST doesn't require source code or binaries since it analyzes by executing the application.

Other DAST benefits are:

- The analysis allows developers to spot the runtime issues, which isn't something SAST is capable of. These can be authentication and network configuration flaws or issues that arise only after the login.
- There are fewer cases of false positives.
- It supports off-the-shelf and customized programming languages and frameworks.
- It presents a less expensive and complex alternative to SAST.

However, the technology certainly has its own share of problems, such as:

- DAST tools provide no insight into the underlying causes of the vulnerabilities and also have difficulties maintaining coding standards.

- The analysis is not suited for earlier stages of development as it can only be done on a running application.
- It won't simulate potential attacks perfectly because exploits are often executed by a party with an internal knowledge base about the application.

The choice between adopting static or dynamic analysis tools mainly depends on what you are trying to achieve. SAST provides developers with educational feedback, while DAST gives security teams quickly delivered improvements. In most cases, you should run both, as the tools plug into the development process in different places. DAST should be used less frequently and only by a dedicated quality assurance team.

Interactive application security testing (IAST)

IAST uses software instrumentation to assess how an application performs and detect vulnerabilities. IAST has an "agent-like" approach, meaning agents and sensors are run to continually analyze the application workings during automated testing, manual testing, or a mix of the two.

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

The main difference of IAST from both SAST and DAST is that it operates inside the application. Access to such a broad range of data makes IAST coverage bigger, compared to source code or HTTP scanning, as well as it allows for more accurate output.

Some of the reasons to apply IAST are:

- Potential issues are caught earlier so IAST minimizes costs and delays. This is due to the application of a Shift-left approach, meaning it is performed during the early stages of the project lifecycle.
- Similar to SAST, IAST analysis gives thorough data-containing lines of code, so security teams can pay immediate attention to a particular flaw.
- With the range of information the tool has access to, it can accurately identify the source of weaknesses.
- Unlike any other software dynamic testing, IAST can be integrated into CI/CD (continuous integration and deployment) pipelines with ease.

On the other hand:

- IAST tools can slow down the operation of the application. The agents essentially serve as added instrumentation, leading to the code not performing as well.
- Some of the issues may have not yet been uncovered as it is a relatively new technology.

Runtime application self-protection (RASP)

RASP is capable of inspecting application behavior, as well as the surrounding context. It captures all requests to ensure they are secure and then handles request validation inside the application. RASP can raise an alarm in diagnostic mode and prevent an attack in protection mode, which is done by either stopping the execution of a certain operation or terminating the session.

RASP technology possesses the following advantages:

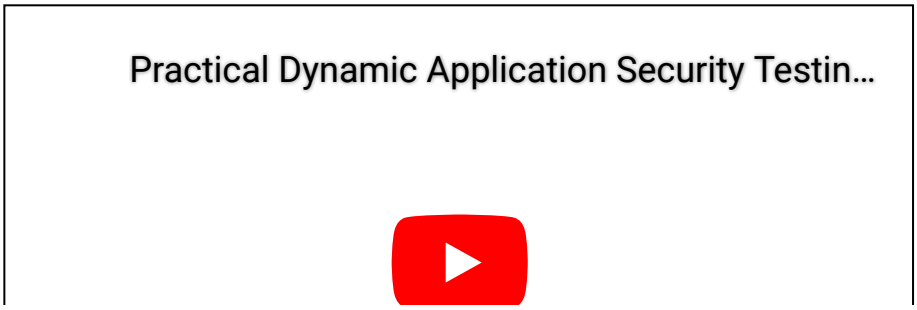
- RASP complements SAST and DAST by casting an extra layer of protection after the application has been set in motion (usually in production).
- It can be easily applied with faster development cycles.
- Unanticipated inputs will be inspected and controlled.
- It allows you to quickly respond to an attack by providing exhaustive analysis and weakness locations.

However, RASP tools come with certain drawbacks:

- By sitting on the application server, RASP tools may have a negative impact on application performance.
- The emerging technology may not be compatible with regulations or internal policies, which restrict installing other software, or locked-down services.
- You might get the feeling that your application is safer than it really is. Even if the tool has identified an issue, it still means taking your application offline while it is being fixed.
- RASP isn't a substitute for application security testing, as it is incapable of providing comprehensive protection.

While RASP and IAST have similar methods and use, RASP does not conduct comprehensive scans but instead runs as a part of the application inspecting its traffic and activity. They both report on attacks as they occur, but IAST does so at the time of testing, while RASP does so in production.

If you want to know more about the topic, watch these helpful and detailed videos:



Make the right choice

Every testing method serves a different purpose, and so they should be skillfully employed at a specific time. If you want to protect the very core of your business, experts from Positive Technologies will help you navigate every aspect of application security. We offer comprehensive and highly accurate tools and solutions to provide actionable reports for both the development process and operation.

PT Application Inspector will not only detect and patch vulnerabilities in your application but most importantly, will also prevent them from occurring in the first place.
Analytics articles

- June 17, 2022 Positive Research 2022
- June 7, 2021 Positive Research 2021
- December 7, 2020 Positive Research 2020

Editor’s Choice
February 13, 2020

Network traffic analysis: what is it, and why do we need NTA systems?

November 18, 2019

Securing Web Applications: OWASP Top 10 Vulnerabilities and what to do about them

August 2, 2019

How to prevent SQL injection attacks

All articles
Solutions

- ICS/SCADA
- Vulnerability Management
- Financial Services
- Protection from targeted attacks (anti-apt)
- PT Industrial Cybersecurity Suite
- Utilities
- ERP Security
- Security Compliance

Products

- MaxPatrol 8
- MaxPatrol SIEM
- PT Application Firewall
- PT Application Inspector
- PT ISIM
- PT Network Attack Discovery
- PT Sandbox
- XSpider
- MaxPatrol VM
- MaxPatrol SIEM All-in-One
- PT MultiScanner
- PT BlackBox

Services

- ICS/SCADA Security Assessment
- ATM Security Assessments
- Web Application Security Services
- Mobile Application Security Services
- Custom Application Security Services
- Penetration Testing
- Forensic Investigation Services
- Advanced Border Control

Analytics

- Threatscape
- PT ESC Threat Intelligence
- Cybersecurity glossary
- Knowledge base

Partners
About

- Clients
- Press
- News
- Events
- Contacts
- Documents and Materials

positive technologies

Find us:

-
-

- Report a vulnerability
- Help Portal
- Terms of Use
- Privacy Notice
- Cookie Notice
- Positive Coordinated Vulnerability Disclosure Policy
- Sitemap

Copyright © 2002—2023 Positive Technologies. All Rights Reserved.

- Report a vulnerability
- Help Portal
- Terms of Use
- Privacy Notice
- Cookie Notice
- Positive Coordinated Vulnerability Disclosure Policy
- Sitemap