HCL SOFTWARE | HCL AppScan

HCL

Empower Your Development and Security Teams with

Interactive Application Security Testing

Introduction

In Application Security testing, Dynamic Application Security Testing (DAST) and Static Application Security Testing (SAST) are prominent techniques. However, Interactive Application Security Testing (IAST) is a promising new entrant in AST, helping to dramatically reduce false positives.

One of the significant value propositions of IAST is the enablement of Shift-Left practices that allow AST to be integrated into the SDLC in its early stages, thereby reducing the security issues discovered in the late stages of the development process.

In this white paper, we will provide a brief overview of IAST, its functionality, use cases, advantages, and disadvantages.

What is IAST?

As the name suggests, IAST is a more interactive form of application security testing. Although DAST also interacts with an application, it is only from the end-user perspective and it tests only the application's exposed surface. An example of this would be how an end-user accesses an application, with results that are based on HTTP request and response.

SAST also interacts with the application, but it will miss the run time analysis, and mostly its analysis is based on data flow and pattern analysis. Also, SAST does not examine Frameworks.

To fill the gap, we have IAST instrumented within an application server to monitor everything from 'Source' to 'Sink'. IAST's location allows it to examine calls to the database, data flows, code execution in memory, file system access, input validations, etc. and analyze them to see if they result in any vulnerabilities. The results are then published to developers, along with identification of the source code that is vulnerable.

How is IAST implemented?

As mentioned above, IAST needs to be instrumented within the application server, which means the IAST tool/agent needs to be installed on the target application server. This agent then monitors the application whenever there are any requests.

Advantages of IAST

IAST can be easily integrated into the DevSecOps lifecycle. The user doesn't need special skills to install the agent in the application server, and the developer can continue his/her development. IAST leverages any interaction with the application into a security test, even if it is legitimate and doesn't contain any malicious properties.

Using IAST along with functional tests during Dev and QA:

- Gets more out of your existing development workflow.
- Solves security issues early in the development lifecycle.
- Makes things easier to fix (because of fewer context switches).
- Becomes more educational for your developers.
- Reduces noise in staging.

Using IAST along with DAST during the QA/deployment stage:

- Increases security scan coverage.
- Provides actionable information regarding your security issue.

During all of these processes, IAST will monitor applications and report vulnerabilities to end-users.

IAST is easy to implement and use, and IAST is a best fit for API testing, since most of its API testing is automated. The results obtained from IAST are more accurate, due to the reduction in false-positive findings. Hence, developers can fix code without being focused on eliminating false positives. And, IAST can be used to test any third-party application, API or framework, because IAST runs on the compiled code during run time.

Disadvantages of IAST

Although IAST is interactive, it does not cover the complete code base, but is instead limited to covered test cases.

An agent needs to be deployed, which requires your agreement.

In certain use-cases, performance can be impacted, as it takes time to analyze and determine vulnerabilities.

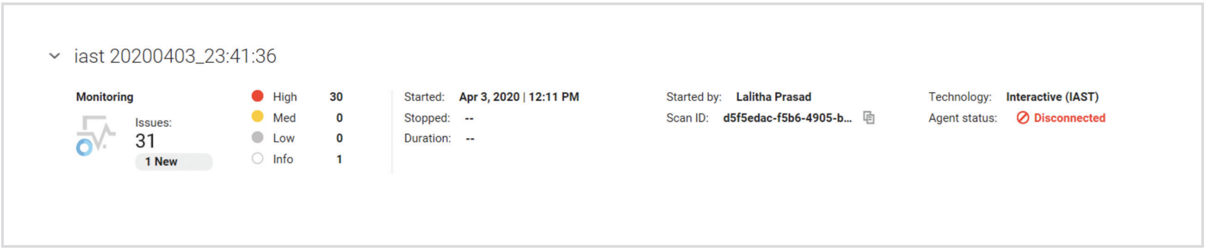
HCL IAST

HCL AppScan on Cloud (ASoC) has newly-deployed IAST capabilities.

Users need to create an IAST scan on ASoC and then just deploy the IAST agent called Secagent.war or access the jar file in the application server of the target Web application. Here's an example for your convenience: %system%Apache-tomcat\webapps

Once deployed, this agent will keep monitoring the application as the user interacts with it. The interaction can be a functional test script or DAST testing. The results are then pushed to IAST on the cloud.

As seen in the diagram below, IAST has begun its monitoring process, and it has found 30 high-severity issues and 1 new issue that's informational in nature:



iast 20200403_23:41:36

Monitoring

Issues: 31

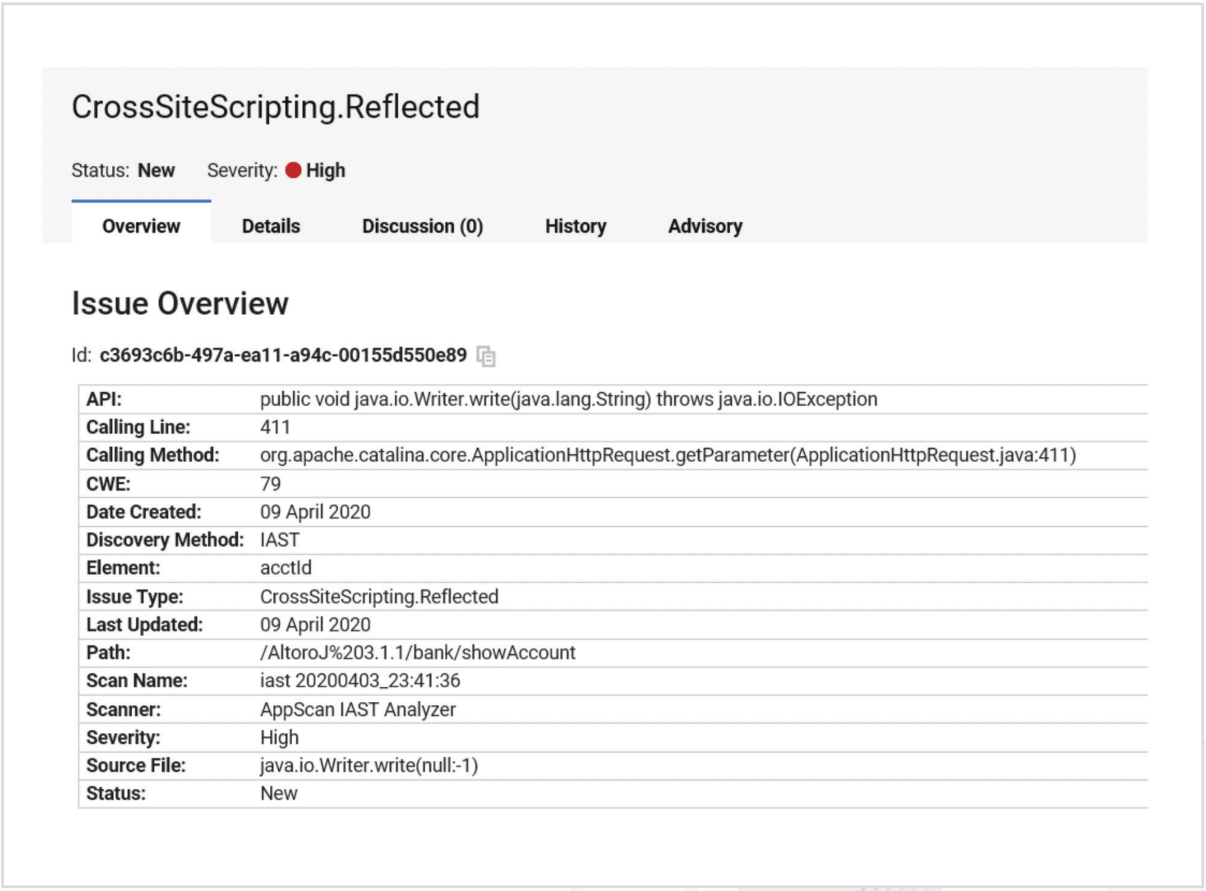
1 New

● High	30
● Med	0
● Low	0
○ Info	1

Started:	**Apr 3, 2020 | 12:11 PM**
Stopped:	--
Duration:	--
Started by:	**Lalitha Prasad**
Scan ID:	**d5f5edac-f5b6-4905-b...**
Technology:	**Interactive (IAST)**
Agent status:	**Disconnected**

Utilize IAST to Detect a Cross-Site Scripting Vulnerability

Now, let's have a look at one of AppScan on Cloud's high-severity findings, a Cross-Site Scripting (XSS) vulnerability:



CrossSiteScripting.Reflected

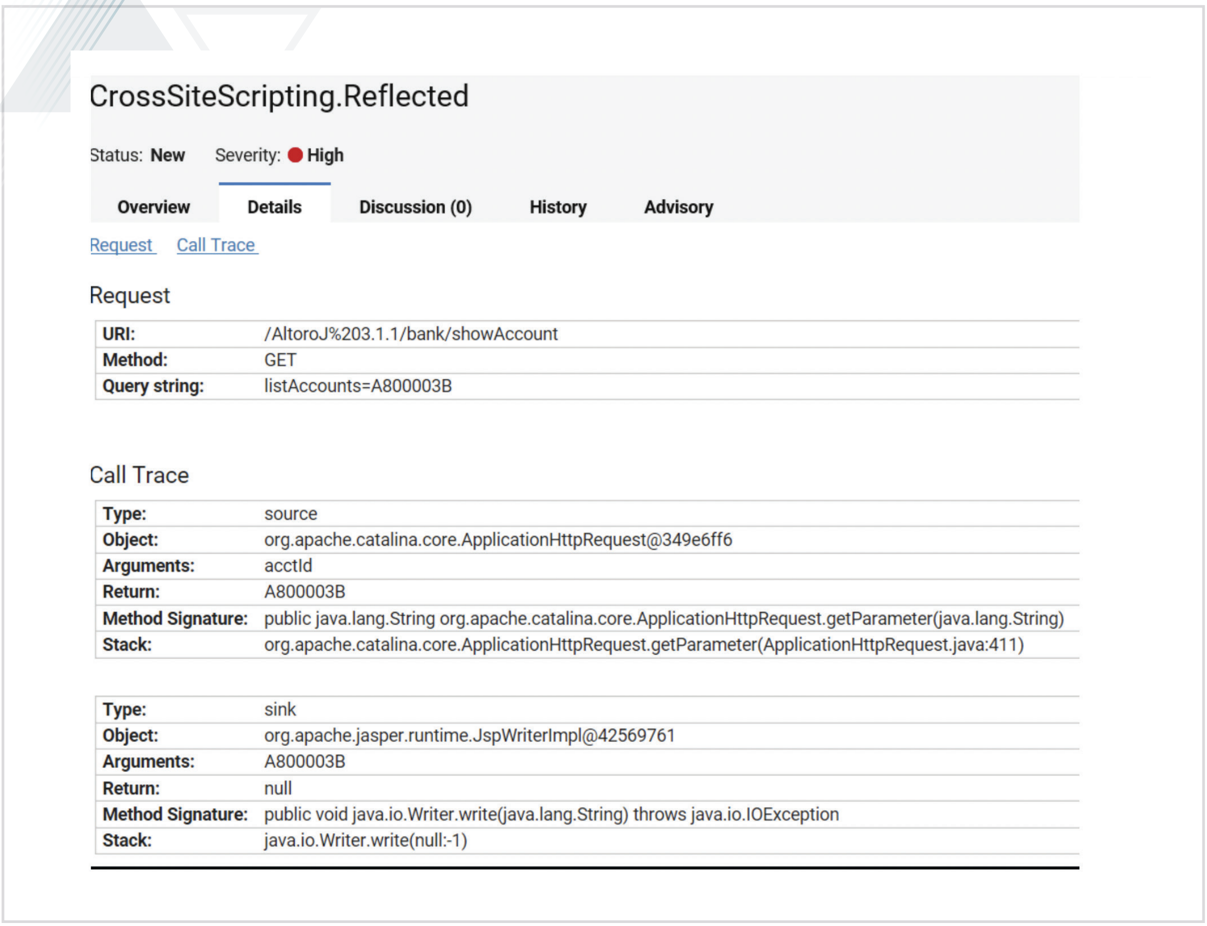
Status: **New** Severity: ● **High**

Issue Overview

Id: **c3693c6b-497a-ea11-a94c-00155d550e89**

API:	public void java.io.Writer.write(java.lang.String) throws java.io.IOException
Calling Line:	411
Calling Method:	org.apache.catalina.core.ApplicationHttpRequest.getParameter(ApplicationHttpRequest.java:411)
CWE:	79
Date Created:	09 April 2020
Discovery Method:	IAST
Element:	acctId
Issue Type:	CrossSiteScripting.Reflected
Last Updated:	09 April 2020
Path:	/AltoroJ%203.1.1/bank/showAccount
Scan Name:	iast 20200403_23:41:36
Scanner:	AppScan IAST Analyzer
Severity:	High
Source File:	java.io.Writer.write(null:-1)
Status:	New

And, here are additional details about the XSS finding:



CrossSiteScripting.Reflected

Status: New   Severity: ● High

Request

URI:	/AltoroJ%203.1.1/bank/showAccount
Method:	GET
Query string:	listAccounts=A800003B

Call Trace

Type:	source
Object:	org.apache.catalina.core.ApplicationHttpRequest@349e6ff6
Arguments:	acctId
Return:	A800003B
Method Signature:	public java.lang.String org.apache.catalina.core.ApplicationHttpRequest.getParameter(java.lang.String)
Stack:	org.apache.catalina.core.ApplicationHttpRequest.getParameter(ApplicationHttpRequest.java:411)

Type:	sink
Object:	org.apache.jasper.runtime.JspWriterImpl@42569761
Arguments:	A800003B
Return:	null
Method Signature:	public void java.io.Writer.write(java.lang.String) throws java.io.IOException
Stack:	java.io.Writer.write(null:-1)

As you can see in the diagrams above, IAST provides an extremely detailed view of potential vulnerabilities, including their requests and call trace information. Hence, we can conclude this finding is definitive.

Conclusion

HCL IAST is an optimal solution for DevSecOps, in that it can be implemented easily and can ease the work of developers and security professionals. It is an excellent tool for Dev teams that offers the advantage of being able to resolve security issues directly in the development environment. This is a more effective approach than fixing issues in a staging environment. And, it has the added advantage of allowing you to leverage existing QA processes.

Basically, IAST helps organizations who want to adopt a Shift-Left strategy to reduce their security vulnerabilities, thereby reducing the cost of fixing bugs.

About HCL Software

HCL Software is a division of HCL Technologies (HCL) that operates its primary software business. It develops, markets, sells, and supports over 20 product families in the areas of DevOps, Automation, Digital Solutions, Data Management, and Mainframes. HCL Software has offices and labs around the world to serve thousands of customers. Its mission is to drive ultimate customer success with their IT investments through relentless innovation of its products. For more information, please visit www.hclfederal.com.