Contrast Assess Market-Defining Application Security Testing for Modern Agile and DevOps Teams

Table of contents

01 What is Interactive Application Security Testing?

- Serious vulnerabilities impact fewer applications, but a bigger share of vulnerabilities were serious
- The percentage of applications with multiple vulnerabilities stabilized
- Serious vulnerabilities impacted slightly more java and .Net applications

02 Contrast Architecture

- Contrast Agent
- Contrast TeamServer

03 Deep Security Differentiators

1. Ideal for Agile and DevOps
2. Highest Accuracy
3. Continuous Analysis
4. Scalable Architecture

04 Key Features of Contrast Assess

- Extensive Vulnerability Coverage
- Code-Level Remediation Advice
- Third-Party Code Analysis
- Application Inventory
- Live Application Architecture

05 Conclusion

Introduction

Contrast is a revolutionary application security solution that transforms an organization's ability to secure their software by making applications self-protecting.

Contrast Assess infuses software with vulnerability assessment capabilities that uniquely combine runtime data and control flow analyses with static, dynamic, configuration and software composition analysis techniques in a single solution. Leveraging a technique known as deep security instrumentation, the Contrast solution operates unobtrusively during normal use and testing of the target web application or API. This "passive" approach to security testing eliminates the need for time-wasting security "scans," and other out-of-band security testing activities.

So, as security-instrumented software runs, security flaws are continuously and automatically identified, without changing the application software stack, or how teams build, test, or deploy code. The result is accurate, continuous vulnerability assessment that integrates seamlessly with existing software development and security processes, scales across the software development lifecycle and the entire application portfolio, and easily outpaces traditional solutions.

This whitepaper covers how Contrast Assess' unique Application Security Testing solution, sometimes referred to as Interactive Application Security Testing (IAST), makes software capable of assessing itself continuously for vulnerabilities, while providing the highest accuracy, efficiency, and coverage

01 | What is Interactive Application Security Testing?

Analyst firm Gartner has defined the IAST category as follows:

"Interactive application security testing (IAST) uses instrumentation that combines dynamic application security testing (DAST) and static analysis security testing (SAST) techniques to increase the accuracy of application security testing. Instrumentation allows DAST-like confirmation of exploit success and SAST-like coverage of the application code, and in some cases, allows security self-testing during general application testing." [1]

HOW IS THE CONTRAST ASSESS APPROACH TO IAST UNIQUE?

Gartner's definition is relatively broad, allowing for a variety of tools to be classified as IAST products. In practical terms, the difference between IAST products is significant.

Specifically, only Contrast represents a fundamental shift in how application security is performed and enables vulnerability self-assessment during general testing, which eliminates the need for a separate security testing phase. Contrast replaces the point-in-time security scan "snapshot" of other faux-IAST solutions with a continuous flow of real-time telemetry leveraged for vulnerability assessment.

Competing solutions may conform to the broad Gartner definition of IAST, but only find vulnerabilities using DAST or DAST-like techniques that simulate attacks against a running application and depend on its successful exploitation. Organizations using those solutions must still dedicate human-expert resources to effectively configure and run the security testing activity. They then wait for a scan to complete, only to receive an incomplete, point-in-time snapshot of their application security status from that scan.

Unfortunately, this approach does not provide the continuous security feedback that Agile software development teams require.

Contrast Assess neither scans nor attacks applications to identify vulnerabilities, but uses patented stateof-the-art deep security instrumentation technology to combine modern runtime data and control flow driven analyses with a defined set of the most effective elements of traditional static, dynamic, software composition and configuration analysis techniques, and deliver them as a unified solution directly into applications.

Contrast Assess' passive testing approach, enables teams to leverage their existing functional testing to automatically perform vulnerability assessment of their web applications and APIs, eliminating the need for dedicated, explicit security testing activities (known as scans).

HOW IS CONTRAST ASSESS DIFFERENT FROM SAST?

SAST solutions attempt to build a model of an application and pseudo-execute it in order to guess what the application's runtime behavior might look like – but SAST is limited in recognizing how all the pieces of an application work together at runtime, resulting in extensive false negatives and false positives. Not only does SAST deliver an incomplete, inaccurate assessment, but it also involves a slow testing process, which makes it unfit for modern, high-velocity software development that is built around short feedback loops and quick turnaround.

Contrast Assess removes all these limitations in SAST, while keeping some sound elements of static analysis. Assess, analyzes every single line of code as it loads looking for flaws that are best identified statically, such as hardcoded passwords. It then continues to passively observe the actual data and control flow activity from within the running application to identify vulnerabilities that can only be discovered effectively through runtime analysis, such as injection flaws. This allows **Contrast Assess** to automatically identify a **much broader range of vulnerabilities – with greater accuracy and speed** – than SAST tools.

Additionally, Contrast Assess is fully distributed and infuses each application with a "self-assessment" capability that performs analysis continuously, in parallel, across an entire portfolio of applications. SAST solutions cannot effectively operate this way, and rely heavily on experts to analyze and triage results, which creates a significant bottleneck and slows teams down.

HOW IS CONTRAST ASSESS DIFFERENT THAN DYNAMIC APPLICATION SECURITY TESTING (DAST)?

DAST tools try to exploit the running application with specially-crafted HTTP inputs and simulated attacks, and then detect vulnerabilities by analyzing the corresponding HTTP responses – but DAST is blind to what occurs within the application, and provides only limited coverage. Like SAST, DAST tools are very slow, with a typical scanning activity taking hours, if not days, to complete.

Contrast Assess performs a complete runtime data and control flow analysis, combined with static analysis of all the code, as described above, while also analyzing all the inbound and outbound HTTP traffic generated during normal testing of the application. This allows Contrast to perform dynamic analysis similar to, but more effective than DAST, without requiring any dedicated security tests, exploitation of the target application, or security experts to be involved in the testing process.

Contrast Assess performs static analysis before the code starts running – this includes custom code as well as all code found in libraries, frameworks, application servers, and the runtime platform – and further leverages instrumentation to observe, analyze and report on the running code as it executes.

Because Contrast Assess works from within the application, it also provides more accurate analysis than traditional **Penetration (Pen) Testing** tools. And, unlike either SAST or DAST products, Contrast Assess performs **Software Composition Analysis (SCA)** to build an inventory of all third-party components (e.g. libraries, frameworks etc.), including open source software (OSS), that are used by the application. This lets Contrast identify known and unknown vulnerabilities across an entire application, including all those components.

02 | Contrast Architecture

Contrast Assess consists of two main components: 1.) an agent that runs alongside the application on the application server and performs vulnerability assessment, and 2.) a centralized management server (TeamServer) that collects and reports on vulnerabilities identified by the agents, and controls the deployment.

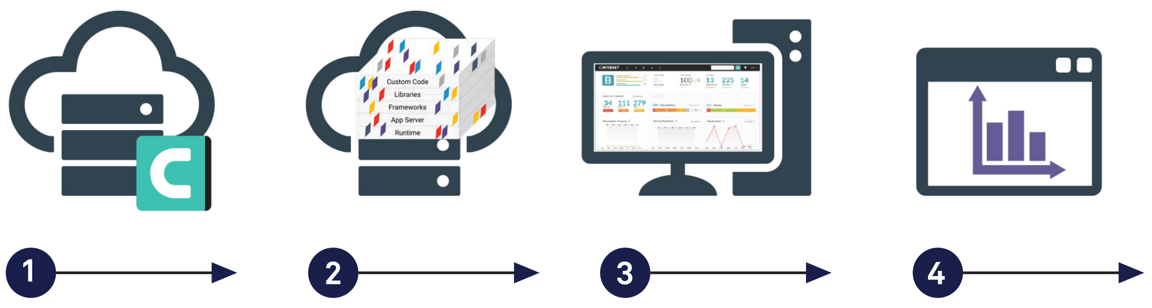
CONTRAST AGENT

Contrast Assess uses an **agent** on each application server to instrument applications with passive sensors to discover and report vulnerability data to the Contrast TeamServer. Agents instrument applications and perform their assessments automatically and in a "passive" manner, and therefore, require no changes to the build-test-deploy cycle. This allows the solution to seamlessly scale out across large portfolios.

CONTRAST TEAMSERVER

Contrast Assess operates under a command and control architecture. Contrast **TeamServer** is a centralized server with a modern user interface for managing the deployment and reporting on vulnerabilities. Contrast TeamServer delivers native integrations with popular software development tools (e.g. IDEs, CI/CD tools and chat-ops products) and supports a variety of notification, reporting, and API access methods, including a comprehensive RESTful API that can be used for custom integrations. TeamServer is available as a SaaS offering or can be deployed on-premises

Figure 3. Continuous Analysis



Add Contrast agent to application servers

Contrast agent instruments applications at runtime

Agents report to Contrast TeamServer

Also available on premises

Results delivered via:

- User interface
- Email
- Integrations
- (e.g. Jira, Slack, SIEM, etc)
- REST API

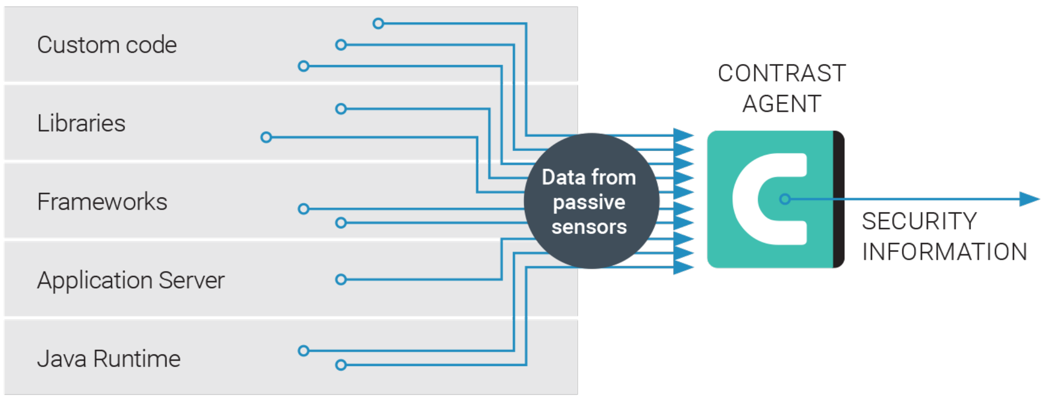
03 | Deep Security Instrumentation

Deep security instrumentation is a technique performed at runtime, as program code loads into memory.

Software instrumentation has been used for many years for application performance monitoring and logging. Companies like New Relic and AppDynamics deliver solutions that leverage instrumentation of running applications to improve visibility into their performance. They send findings to a server, and use that data to create useful reports and dashboards.

Similarly, when the Contrast agent is installed, it automatically and invisibly instruments applications with an analysis engine and a network of passive sensors that gathers information about the application's security, architecture, libraries, and frameworks, and reports its findings to the central Contrast TeamServer. Since the evidence needed to accurately identify and describe a vulnerability is usually scattered in several places within an application, Contrast's **deep security instrumentation** has numerous advantages over traditional static and dynamic analysis tools, which have no inside-the-running-application "context" to inform their analysis. Contrast's patented methodology provides developers with instant security feedback as soon as they write and run their code, instead of weeks or months later.

Figure 2. Deep Security Instrumentation



Instrumenting an application with passive sensors provides more access to information about theapplication and its execution, and delivers unprecedented levels of speed and accuracy in identifying vulnerabilities.

By instrumenting all layers across the entire application, Contrast Assess can trace data and execution through custom code, libraries, frameworks, and even the runtime platform.

For example, Contrast's instrumented analysis can identify:

- Credit card numbers extracted from a database and flag when they end up in a log file
- A weak encryption algorithm specified in a properties file
- Data flowing from within an encoded cookie, through a data bean, into a session store, into a JSF component, and finally to a browser – indicating cross-site scripting (XSS) weakness.

Contrast Assess instrumentation monitors and reports on the following information with an application:

HTTP Requests contain useful information about authentication, sessions, use of SSL, certain headers, CSRF tokens, and sitemap information. Contrast passively monitors HTTP requests with sensors, enabling testing in just about any environment. Other techniques like DAST use HTTP requests to carry out security tests, and require an expert to configure and run them.

HTTP Responses provide a wealth of security information. Security headers, CSRF tokens, SSL, autocomplete, access control, and session management are all easily revealed by analyzing responses.

Data Flow is the holy grail of security analysis. Most of the interesting injection problems can only be revealed with a strong data flow engine. Static analysis tools used to be the only way to analyze data flow, but they were difficult to use and suffered from false alarms. Newer instrumentation-based approaches can perform highly accurate "whole application" data flow analysis including frameworks, libraries, dynamic class loading, and more.

Control Flow shows whether the code executes the right steps in the right order, and analysis can reveal several types of vulnerabilities. For example, it will check if the proper hashing occurred during the authentication process, or if there is an access control check in every Struts action.

Libraries and Components are the building blocks of modern applications. Analyzing these components for known vulnerabilities is of critical importance. It is also important to know if the library has unknown vulnerabilities or hazards that an unsuspecting developer may have used improperly.

Frameworks are used by all modern web applications and web services, and many frameworks have a set of security controls built in. Instrumentation monitors and reports on things like whether these controls were used properly, or if the developers invoked them in all the right places. Understanding the framework pays off immensely in providing useful context-sensitive guidance to developers.

Application Server is like a framework, in that it also has numerous security controls. Contrast's instrumentation checks if the application is using security controls. Tools can do better analysis when they understand the application server in use and what protection it provides.

Configuration Data can be stored in XML files, property files, databases, and the like. Most security controls can be configured on or off. Many of them have detailed configuration that allows their behavior to be tailored. This is a direct and very accurate way to measure application protection.

Backend Connections provide information about where sensitive data is stored, which is key to being able to perform great security analysis. Tools that can understand backend connections are much more likely to be able to see critical vulnerabilities and describe them to developers.

04 | Contrast Assess Differentiators

IDEAL FOR AGILE AND DEVOPS

Today's high velocity software development processes need application security testing that is efficient, non-intrusive, and integrated into development environments. Contrast Assess is purpose-built to work interactively with developers as they write and test web applications and APIs. A comprehensive RESTful service and native integrations with popular tools, such as Slack, HipChat, JIRA, Visual Studio Team Services, Eclipse, IntelliJ and many more, ensure developers get automatically notified when they've introduced a vulnerability, and can consume all relevant data, including remediation guidance, within the tools they already use. This allows for security vulnerabilities to be handled as any other defects, and limit context switching for developers.

DevOps teams can add the Contrast agent into their standard build tools, automated provisioning systems and containers, such as Maven, Gradle, Ansible and Docker, to discover and report on vulnerabilities within an application. Contrast Assess automatically enables security analysis while any manual or automated tests are executed, and can be integrated with a CI/CD automation server, such as Jenkins, to fail a build that has excessive vulnerabilities.

HIGHEST ACCURACY

Unlike legacy application security testing solutions, such as SAST, DAST or faux-IAST, Contrast Assess produces accurate results that developers can immediately act on without dependence on application security experts.

This is due to the deep and rich visibility Contrast Assess has into the application and its runtime environment, combined with its ability to fuse together the most effective elements of Interactive (IAST), Static (SAST) and Dynamic (DAST) application security testing approaches, with configuration and opensource security analyses, and deliver them directly into applications.

This unique approach to modern application security produces the intelligence and evidence necessary to detect vulnerabilities with virtually no false positives and no false negatives.[2]

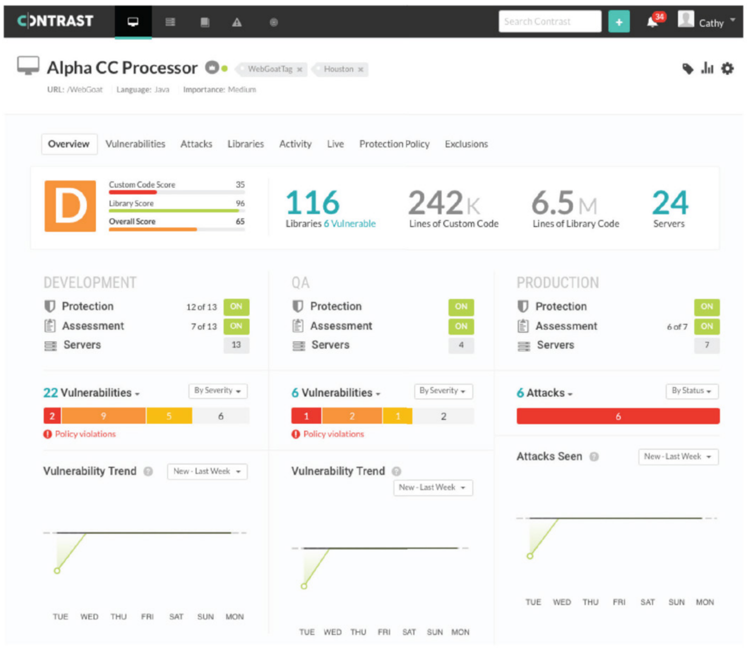
CONTINUOUS ANALYSIS

Unlike legacy approaches that require time-consuming, late-stage scans that disrupt the development process, there is no separate security testing phase with Contrast Assess. Contrast Assess uses its deep security instrumentation to produce a continuous stream of accurate vulnerability analysis whenever and wherever software is run. Development, QA, DevOps, and security teams get results as they develop and test software, enabling them to find and fix security flaws early in the software lifecycle, when they are easiest and cheapest to remediate.

Contrast Assess displays security analysis results in a real-time dashboard of critical security information, vulnerabilities, and remediation advice across all applications. Each application receives an easy-tounderstand security letter grade.

The results in Figure 3 were captured by Contrast after only a few minutes of using a vulnerable open source software application.

Figure 3. Continuous Analysis



All applications are presented in a clear, understandable dashboard in the Contrast TeamServer user interface. Each application also has its own dashboard with a letter grade for both security and coverage.

Contrast Assess has no separate testing phase. It produces results continuously so development teams can release software as fast as they want, knowing it is secure.

SCALABLE ARCHITECTURE

Contrast Assess scales because it instruments application security into each application, delivering and distributing vulnerability assessment across an entire application portfolio. Every running application continuously produces results in parallel.

Contrast differs from legacy approaches that require application security experts – a human element that does not scale.

Even with hundreds or thousands of applications, Contrast Assess provides an always up-to-date dashboard for each application with vulnerability, library, architecture, and other security details.

05 | Key Features of Contrast Assess

EXTENSIVE VULNERABILITY COVERAGE

Contrast provides extensive coverage over most common vulnerabilities, including the OWASP Top Ten.[3] Some examples include:

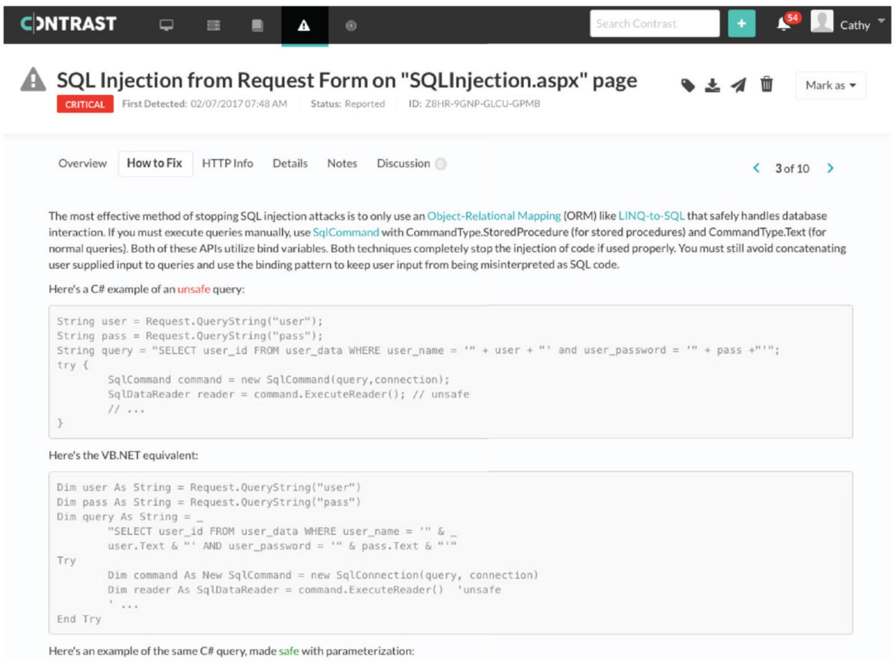
- SQL injection (A1)
- Blind SQL injection (A1)
- Command injection (A1)
- Reflected XSS (A7)
- Stored XSS (A7)
- Session ID disclosure (A2)
- XML External Entities (XXE) (A4)
- Insecure deserialization (A8)
- Authorization missing (A5)
- Weak hash algorithm (A3)
- Weak encryption algorithm (A3)
- Under-protected APIs (A10)
- Arbitrary forward
- Unchecked redirect
- No size limit on data read
- File download injection
- HTTP header injection
- And more

CODE-LEVEL REMEDIATION ADVICE

The Contrast user interface also explains vulnerabilities to those that need to understand and fix them. Contrast's innovative Security Trace format pinpoints exactly where a vulnerability appears in the code, and how it works.

The SQL Injection example illustrated in Figure 4 explains to the developer exactly how untrusted data flows through the application and gets embedded in SQL query without either validation or parameterization. Contrast "speaks the developer's language," and provides remediation guidance that is easy to understand and implement.

Figure 4. SQL Injection Remediation



SQL Injection from Request Form on "SQLInjection.aspx" page

CRITICAL First Detected: 02/07/2017 07:48 AM Status: Reported ID: Z8HR-9GNP-GLCU-GPMB

The most effective method of stopping SQL injection attacks is to only use an Object-Relational Mapping (ORM) like LINQ-to-SQL that safely handles database interaction. If you must execute queries manually, use SqlCommand with CommandType.StoredProcedure (for stored procedures) and CommandType.Text (for normal queries). Both of these APIs utilize bind variables. Both techniques completely stop the injection of code if used properly. You must still avoid concatenating user supplied input to queries and use the binding pattern to keep user input from being misinterpreted as SQL code.

Here's a C# example of an unsafe query:

```
String user = Request.QueryString("user");
String pass = Request.QueryString("pass");
String query = "SELECT user_id FROM user_data WHERE user_name = '" + user + "' and user_password = '" + pass +"'";
try {
        SqlCommand command = new SqlCommand(query,connection);
        SqlDataReader reader = command.ExecuteReader(); // unsafe
        // ...
}
```

Here's the VB.NET equivalent:

```
Dim user As String = Request.QueryString("user")
Dim pass As String = Request.QueryString("pass")
Dim query As String = _
        "SELECT user_id FROM user_data WHERE user_name = '" & _
        user.Text & "' AND user_password = '" & pass.Text & "'"
Try
        Dim command As New SqlCommand = new SqlConnection(query, connection)
        Dim reader As SqlDataReader = command.ExecuteReader()  'unsafe
        ' ...
End Try
```

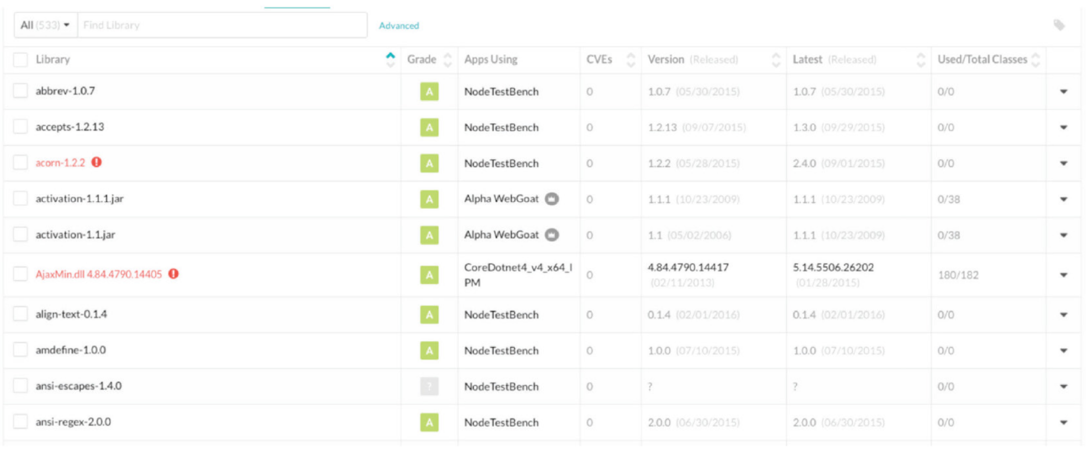
Here's an example of the same C# query, made safe with parameterization:

Contrast vulnerability reports include all the details needed to understand the problem, find it in the code, and how to fix it correctly.

THIRD-PARTY CODE ANALYSIS

Like icebergs, 80 percent of the code in modern applications is “beneath the surface,” lurking in libraries, frameworks, and other components. Applications often have 50 or more of these libraries, comprising millions of lines of potentially vulnerable code. Contrast Assess automatically analyzes these libraries and provides a detailed dashboard that lists the actual libraries, which library versions are in use, and how many classes of each library are in use for each application, as well as the number of CVEs associated with each library.

Figure 5. Library Analysis



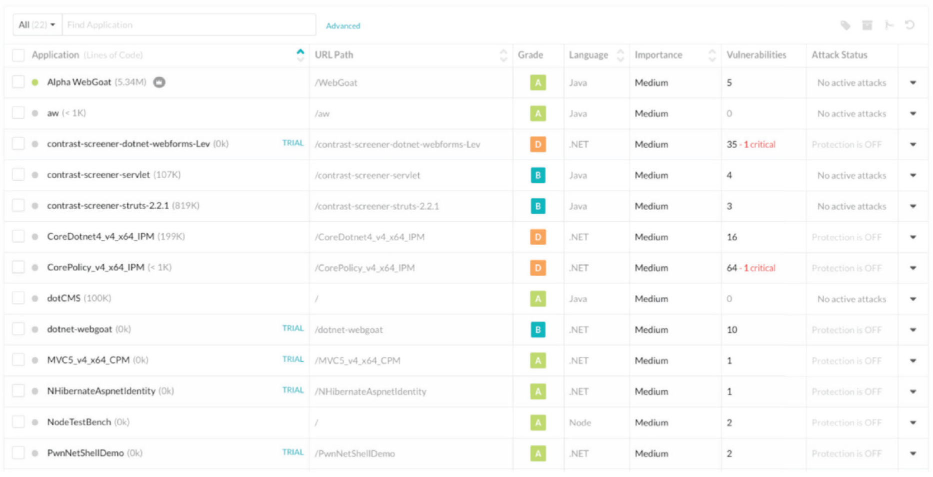
Library	Grade	Apps Using	CVEs	Version (Released)	Latest (Released)	Used/Total Classes
abbrev-1.0.7	A	NodeTestBench	0	1.0.7 (05/30/2015)	1.0.7 (05/30/2015)	0/0
accepts-1.2.13	A	NodeTestBench	0	1.2.13 (09/07/2015)	1.3.0 (09/29/2015)	0/0
acorn-1.2.2	A	NodeTestBench	0	1.2.2 (05/28/2015)	2.4.0 (09/01/2015)	0/0
activation-1.1.1.jar	A	Alpha WebGoat	0	1.1.1 (10/23/2009)	1.1.1 (10/23/2009)	0/38
activation-1.1.jar	A	Alpha WebGoat	0	1.1 (05/02/2006)	1.1.1 (10/23/2009)	0/38
AjaxMin.dll 4.84.4790.14405	A	CoreDotnet4_v4_x64_I PM	0	4.84.4790.14417 (02/11/2013)	5.14.5506.26202 (01/28/2015)	180/182
align-text-0.1.4	A	NodeTestBench	0	0.1.4 (02/01/2016)	0.1.4 (02/01/2016)	0/0
amdefine-1.0.0	A	NodeTestBench	0	1.0.0 (07/10/2015)	1.0.0 (07/10/2015)	0/0
ansi-escapes-1.4.0	?	NodeTestBench	0	?	?	0/0
ansi-regex-2.0.0	A	NodeTestBench	0	2.0.0 (06/30/2015)	2.0.0 (06/30/2015)	0/0

Contrast automatically discovers third-party libraries, alerts to the known (and unknown) risks they may bring with them, and provides critical versioning and usage information that helps remediate risks.

APPLICATION INVENTORY

Though it may seem simple, application inventory may be the hardest problem to solve for application security teams. Organizations may have hundreds or thousands of applications, Microservices, APIs – each with multiple instances of different versions installed across development and QA – and they're all constantly changing. Contrast tracks and continuously feeds that information about internal and external web services, and their relationships across an application into a unified security inventory and bill of materials that's always up-to-date.

Figure 6. Application Inventory



All (22) | Find Application | Advanced

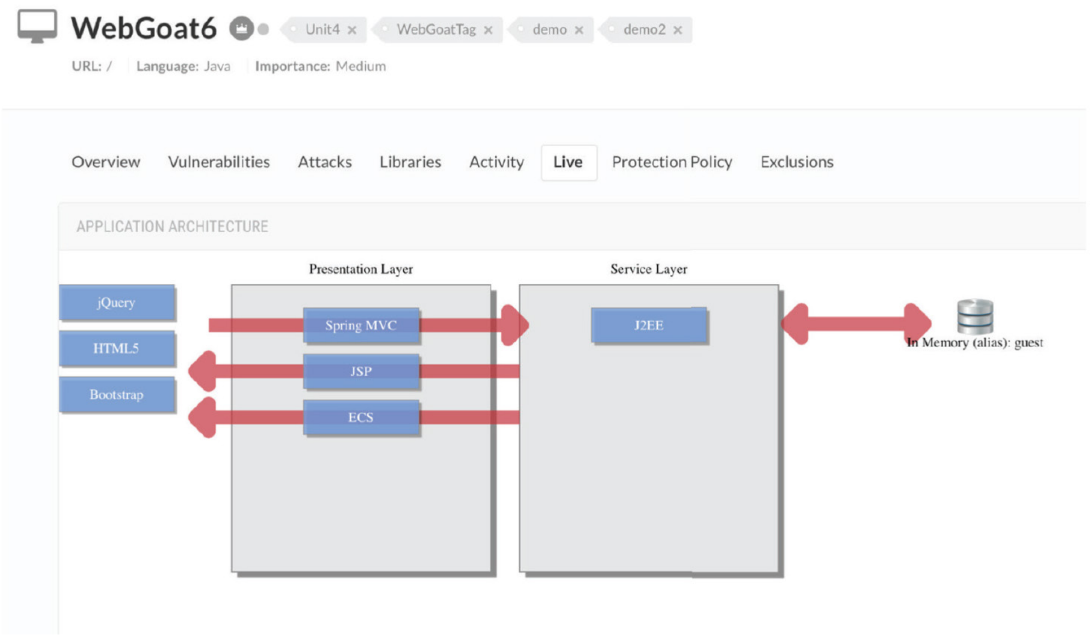
Application (Lines of Code)		URL Path	Grade	Language	Importance	Vulnerabilities	Attack Status
Alpha WebGoat (5.34M)		/WebGoat	A	Java	Medium	5	No active attacks
aw (< 1K)		/aw	A	Java	Medium	0	No active attacks
contrast-screener-dotnet-webforms-Lev (0k)	TRIAL	/contrast-screener-dotnet-webforms-Lev	D	.NET	Medium	35 - 1 critical	Protection is OFF
contrast-screener-servlet (107K)		/contrast-screener-servlet	B	Java	Medium	4	No active attacks
contrast-screener-struts-2.2.1 (819K)		/contrast-screener-struts-2.2.1	B	Java	Medium	3	No active attacks
CoreDotnet4_v4_x64_IPM (199K)		/CoreDotnet4_v4_x64_IPM	D	.NET	Medium	16	Protection is OFF
CorePolicy_v4_x64_IPM (< 1K)		/CorePolicy_v4_x64_IPM	D	.NET	Medium	64 - 1 critical	Protection is OFF
dotCMS (100K)		/	A	Java	Medium	0	No active attacks
dotnet-webgoat (0k)	TRIAL	/dotnet-webgoat	B	.NET	Medium	10	Protection is OFF
MVC5_v4_x64_CPM (0k)	TRIAL	/MVC5_v4_x64_CPM	A	.NET	Medium	1	Protection is OFF
NHibernateAspnetIdentity (0k)	TRIAL	/NHibernateAspnetIdentity	A	.NET	Medium	1	Protection is OFF
NodeTestBench (0k)		/	A	Node	Medium	2	Protection is OFF
PwnNetShellDemo (0k)	TRIAL	/PwnNetShellDemo	A	.NET	Medium	2	Protection is OFF

Contrast automatically displays up-to-date intelligence on which applications are in use, and application metadata including lines of code, libraries in use, component technologies, architecture, and backend connections.

LIVE APPLICATION ARCHITECTURE

Understanding an application's architecture is extremely helpful when performing security analysis. Contrast automatically generates simple diagrams that illustrate the application's major architectural components. This information helps the developer quickly identify the meaning of a vulnerability that Contrast pinpoints. In Figure 7, Contrast has correctly identified that the WebGoat6 application has the following backend connections: web services, frameworks being used within the application (Spring, JSP and ECS), a Java virtual machine, and a database. Imagine the benefit of having up-to-date architectural information available, on demand, for every application across the entire portfolio.

Figure 7. Live Application Architecture



Contrast Assess displays the full application architecture under the Live tab.

06 | Conclusion

Organizations looking for a modern application security solution, should consider interactive application security testing technologies, but look beyond the Gartner definition of IAST and evaluate solutions based on how well they provide application security testing that is accurate, continuous, integrated, scalable, and works in modern agile and DevOps environments.

Contrast Assess' implementation of IAST uses a highly scalable architecture – employing distributed agents with a central management service and dashboard – to empower every application to analyze, enforce and communicate about application security across the software development lifecycle. Contrast Assess analyzes every line of code – including third-party components – for vulnerabilities, and provides instant and accurate feedback to developers. This built-in and continuous analysis means that developers can find and fix vulnerabilities as part of their development process. With Contrast Assess, application security experts retain oversight, yet can remove themselves from the critical path of software development, so they can spend more time on strategic security initiatives.

Contrast Assess enables organizations to prioritize their development and operations teams to remedy application security risks immediately, and reduce friction throughout the entire software lifecycle. Visit **www.contrastsecurity.com** and click "**get a demo**" button for a personalized demo of Contrast with one of our experts.

[1] Source: Tirosh, Ayal. Gartner, "Hype Cycle for Application Security, 2016"

[2] For information on Contrast Assess accuracy, read the Technical Brief, "Accurately Assessing AppSec with the OWASP Benchmark" at https://www.contrastsecurity.com/owaspbenchtechbrief

[3] https://www.owasp.org/index.php/Top_1

240 3rd Street
2nd Floor
Los Altos, CA 94022
Phone: 888.371.1333
Fax: 650.397.4133

Contrast SECURITY

contrastsecurity.com