

## AIASST

Advanced Interactive Application Security Testing



### Overview

As the pioneering application security testing product of ZeroDay, it undertakes the initiative of our team to build a tool which simplifies the security work of DevOps team, while pinpointing the location of the vulnerabilities in the code as well as showing their detailed description and fix guidance, with high accuracy and low false positive rate.

### Instrumentation-based Test

By taking advantage of instrumentation technology in the platform where the target application runs, it can record all the HTTP flows to/from the application, track sensitive data flow, trace the stacks, and pinpoint the code lines where the vulnerabilities originate/fruit.

```
"username" in "spring-demos/sqlinjection/sql001.do" induced SQL Injection
Critical | First Detected: 2021-08-02 14:16 | Last Detected: 2021-08-02 14:16 | Risk: Reported | Vendor: Unsubscribed | Application: xxx

Execution | Code | HTTP | Fix | Comments

Input
  org.apache.catalina.connector.RequestFacade.getParameter(part)
  statementAddBatch(org.springframework.jdbc.datasource.DataSourceConnection.java:56)
  main
  Class Method: org.apache.catalina.connector.RequestFacade.getParameter(org.springframework)
  Object: org.apache.catalina.connector.RequestFacade
  Return: null
  Parameters: null
  Context: null
  Code Stack
  package org.aiaast.common.sqlinjection;
  public class SqlInjectionCommon {
    public void statementAddBatch() {
      56 org.apache.catalina.connector.RequestFacade.getParameter();
    }
  }

Inside
  org.apache.catalina.connector.RequestFacade.getParameter(part)
  statementAddBatch(org.springframework.jdbc.datasource.DataSourceConnection.java:56)
  main
  Show More

Output
  statementAddBatch(org.springframework.jdbc.datasource.DataSourceConnection.java:56)
  DetectFrom_app_user where username = 'username' AND payload = 'payload'
  main

End
```

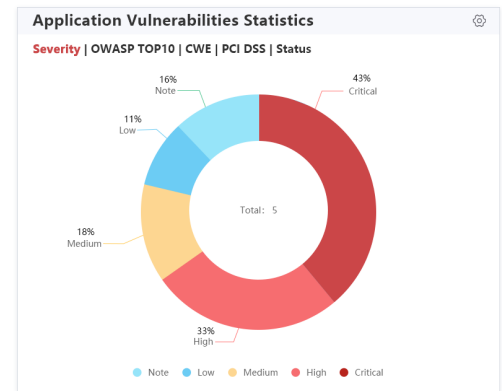
### Automatic Test Process

The test is automatically takes place during the normal functional test and the vulnerabilities would pop out spontaneously without manual work. It spares the development team, as well as the DevOps teams, of extra time spent on security test.



## Broad Vulnerability Detection and Assessment

It is capable of detecting all sorts of vulnerabilities publicly known by industry, which are categorized by industry standards such as OWASP top10 and CWE, as well as novel vulnerabilities discovered by our distinguished security experts. Furthermore, it tracks sensitive data and locates risks in the code against data protection regulations such as PCI DSS and GDPR.



## Comprehensive application security posture

For the first time in the industry, the interactive application security testing tool natively integrates the detection and analysis of all the open source dependencies involved in the application, so that it reveals a comprehensive risk view of the application, whose risk rating depends on the lower of the custom code and the open source dependencies.

Category	Total	Critical
Applications	8	2
Vulnerabilities	178	24
Open source dependencies	76	22
Common Vulnerabilities and Exposure	432	56



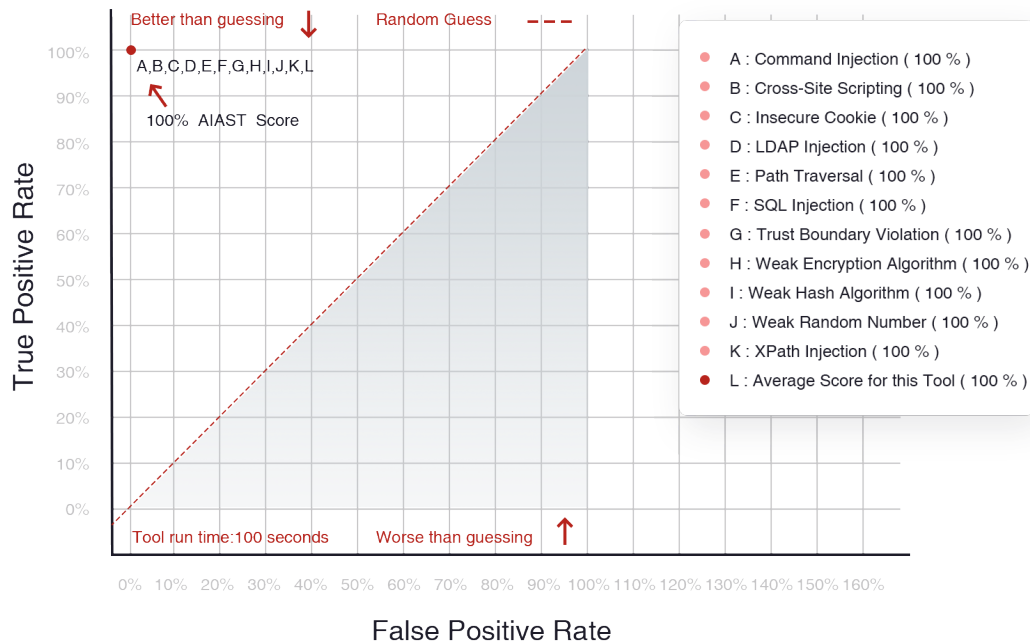
## High accuracy and low false-positive rate

The AIAST adopts a series of measures to boost accuracy and reduce false-positive rate.

- Users can define custom rules, including sanitizer, input validator, RegEx, and class filter, which users may use for input validation and other processing to suppress vulnerabilities of certain types if any, so that those vulnerabilities would not be reported in selected applications.
- Users can define a URL whitelist from which vulnerabilities of specific types would not be reported in selected applications.
- Token/key whitelist

- Users can customize RegEx to detect sensitive data of certain types for tracking and report of vulnerabilities related to the exposure of them.
- Active verification: Upon detection of certain types of potential vulnerabilities, the AIAST is capable of fabricating payloads to send to the URLs to confirm the veracity of the vulnerabilities. After being verified, the vulnerabilities would have a verified icon next to their names in the vulnerabilities list.

**Benchmark v1.2 Scorecard for AIAST**



## Ready to go & Quick start

- SaaS-based solution, no need to set up a server or install a bunch of software. It also spares your IT staff commissioning or maintaining in your own premise. Just a normal web browser and here you go.
- With just a few steps, you can download the Agent on your application's web server and start a new test.
- User-friendly interface, intuitive work flow, abundant and concise elaboration of technical terms, and various dashboards, enable any users with rudimentary security knowledge to complete complex security test tasks with ease .

## Seamless integration with DevOps tools

- Synchronization with Jira so that vulnerabilities detected in the AIAST can be sent to a configured Jira project and there is a status match between Jira issues and AIAST vulnerabilities.
- An AIAST plug-in can be uploaded to Jenkins to configure an automatic security test after build.
- Provide REST API for other services to get test data from AIAST, e.g. application info, vulnerabilities, open source dependencies, test report, etc.

## Detailed and customizable remediation guidance

For each detected vulnerability, there is a corresponding remediation guidance detailing the specific measures or suggestions to overcome or workaround the vulnerability, along with some code examples. Moreover, it provides users with the option of customizing the remediation guidance for each type of vulnerabilities.

Vulnerabilities / Details

"username" in "/spring-demo/sqlInjection/sql001.do" induced SQL injection

**CRITICAL** First Detected: 2021-08-02 14:16 Last Detected: 2021-08-02 14:16 Status: Reported Handler: Unallocated Application: xxxx

Elaboration Code HTTP **Fix** Comments

**Remediation guidance**

- The most effective way to prevent SQL injection is to use the ORM(Object-relational Mapping) framework, such as Hibernate or MyBatis, to use PreparedStatement
- When you need to stitch SQL statements in your code, you also use precompiled SQL statements (PreparedStatement) and parameterized queries, the following are safe code fragment:

```
String name = request.getParameter("customername");
String query = "SELECT account_balance FROM user_data WHERE user_name = ?"; PreparedStatement pstmt =
connection.prepareStatement(query);
ResultSet results = pstmt.executeQuery();
```

Avoid using statement, and do not dynamically stitch SQL statements yourself, following this unsafe code fragment:

```
String query=ELECT account balance FROM user data WHERE user name
request.getParameter("customername") try
Statement statement=connection.createStatement();
ResultSet results = statement.executeQuery(query)
```

- the inability to do the above precompiled SQL statements and parameterized query usage scenarios you restrict use put by doing rigorous input validation and of course input validation is not the best practice for so ing QL
- If none of the above methods are applicable you can also protect the users input by encoding it this method needs to use different coding methods according to different databases, generally do not recommend that you write these coding functions, recommend the use of WASP ESAPI library functions The following is an example of encoding an Oracle special character:

```
Codec ORACLE_CODEC new Oraclecodec()
tring query = "SELECT user id FROM user data WHERE user name
SAFPIncoder) encodeforsql( ORACLE_CODEC, req.getParameter(userid))+and user password
ESAPIEncoder) encodeforsql( ORACLE_CODEC, req.getParameter ("pwd"))
```

**CWE:** <http://cwe.mitre.org/data/definitions/89.html>

**OWASP:** [https://www.owasp.org/index.php/SQL\\_injection\\_Prevention\\_Cheat\\_Sheet](https://www.owasp.org/index.php/SQL_injection_Prevention_Cheat_Sheet)

## Fix check

- There is an "Automatic Fix Check" option in the application configuration. When selected, for those vulnerabilities in the application which support automatic fix check, the AIAST would automatically resend payloads every a few minutes to the specified URLs where the vulnerabilities were detected to check whether they have been fixed or not.
- There is also a Fix Check option in the application lists. Upon click, the AIAST would resend payloads to the specified URLs where the vulnerabilities (fix-checkable) were detected to check whether they have been fixed or not. And such request would be recorded in the comments section of the vulnerability details page.

Vulnerabilities / Details

"username" in "/spring-demo/sqlInjection/sql001.do" induced SQL injection

**CRITICAL** First Detected: 2021-08-02 14:16 Last Detected: 2021-08-02 14:16 Status: Reported Handler: Unallocated Application: xxxx

Elaboration Code HTTP **Fix** **Comments**

Comments...

**OK**

**Records**

- name** 2021-08-12 18:12  
Fix Check : Request sent successfully.
- name** 2021-08-12 18:12  
**Vulnerability status changed** From "Reported" to "Closed"  
Fix Check : The vulnerability has been fixed so it is closed automatically.
- name** 2021-08-12 18:12  
Fix Check : Request sending failed.



# AIASST | Technical Specifications

## Supported language

Java  
JavaScript (coming soon)  
C# (coming soon)  
PHP (coming soon)  
Python (coming soon)

## Supported platform

Java  
Node.js (coming soon)  
.Net (coming soon)

## Supported web server/application framework

### Java

- Tomcat
- WebLogic
- Spring Boot
- Jetty
- WebSphere
- JBoss
- WildFly
- Resin
- WebSphere Liberty

### Node.js (coming soon)

- Express

### .Net (coming soon)

- Framework

### .PHP (coming soon)

- Nginx
- Apache



## About ZeroDay

Founded in 2016, ZeroDay aims to make software development more secure and application security work simpler, saving your precious time and work force.

Spearheaded by AIASST, our product portfolio will include software component analysis, static application security testing, dynamic application security testing, fuzz testing, and more in the coming days.

For more information about ZeroDay, visit us online at [www.zeroday.co.uk](http://www.zeroday.co.uk)

160 The Edge, Clowes Street Salford, Manchester, M3 5NE U.K.

Sales: + 44-16-1350-8028