

Open Source SAST and DAST Tools for WebApp Pen Testing

Lenny Halseth



Funded by:

Department of Homeland Security
Science and Technology Directorate
Cyber Security Division

Definitions

Attack Surface – sum of all paths for data and commands into and out of the application, combined with the code that protects them and the data behind them

Burp Suite – tool developed by PortSwigger Security to test web application security

Endpoint – entry point to a service or process

JVM – Java Virtual Machine

Parameters – data passed to an endpoint

Penetration Testing – simulated attack on a computer system to evaluate security

Spider – collect, walkthrough, and follow linkages to data and other pages

Web Application Pen Testing

White Hats have plenty of disadvantages over their malicious counterparts

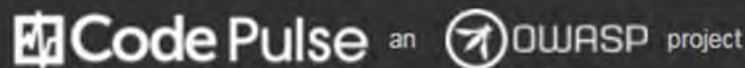
- Huge task of securing web app against all vulnerabilities
- Very limited time
- Hard to lock-step with dev team

There are a few advantages we can leverage with better penetration testing tools:

- Access to server binaries/bytecode
- Access to server-side source code

Open Source Tools for White Hats



OWASP Code Pulse – Provides insight into the real-time code coverage of black box testing activities by monitoring the execution of the web application



Attack Surface Detector – Performs static code analysis of a web application to detect hidden endpoints, optional parameters, and parameter datatypes, and makes that data available in Burp Suite and OWASP ZAP



Code Pulse

 Code Pulse an  OWASP project

Code Pulse Need

Coverage gaps – by definition, penetration testing is typically a purely black box perspective, which makes it almost impossible to ascertain the attack surface coverage gaps

Test tuning – DAST tools are tricky to configure, due to the complex variations in the target applications. Manual testers have challenges tying web requests to the underlying source code.

Coverage data communication – lack of coverage insight from the black box perspective makes this currently challenging, and comparing testing tools and techniques difficult

How Code Pulse Works

Leverages Java and .NET instrumentation libraries to provide real-time measurement of application method calls

- JVM Code Pulse agent runs in the same JVM as the target application
- .NET Code Pulse tracer based on OpenCover code coverage tool

Instruments server bytecode—no changes in source code are needed

Sends method coverage to Code Pulse client for real-time visualization

Code Pulse Benefit

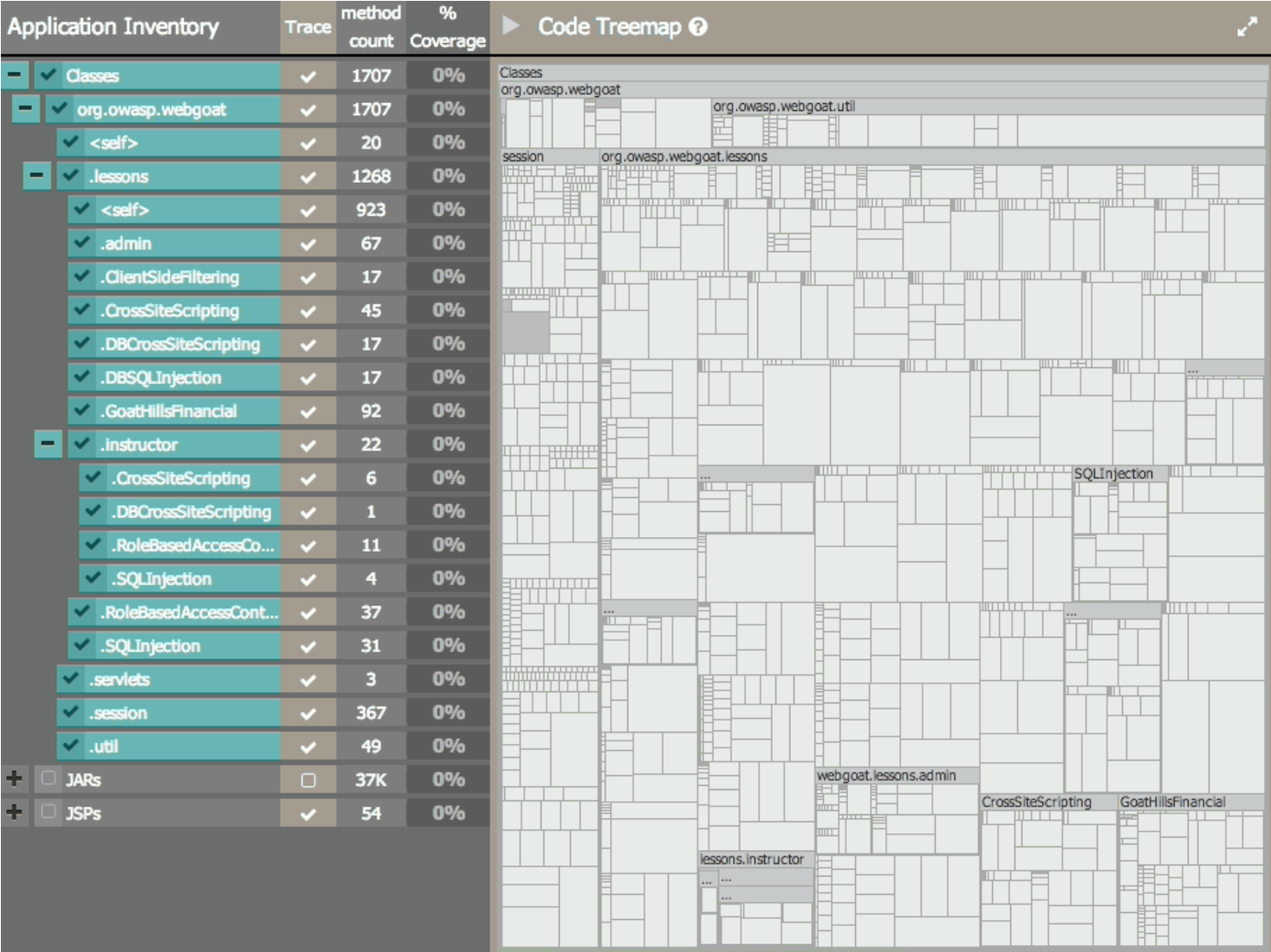
Helps web application testers associate the endpoints they interact with to the underlying classes and methods called in the application server

Find gaps in the test coverage

Allows comparison and tuning of dynamic testing tools and techniques

Percentage of code coverage is a useful metric for communicating testing activity

Code Pulse Screenshot



Future Code Pulse Plans

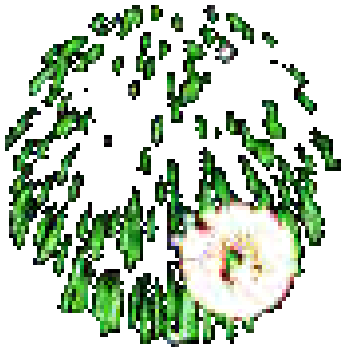
Provide line-level code display in Code Pulse

- Will allow more accurate measurement of code coverage
- Will simplify code review

Better integration of attack surface detection

- Display specific endpoints to access methods visualized in Code Pulse

Enough! Just show me already



LIVE DEMO



Attack Surface Detector



Attack Surface Detector Need

Attack surface gaps – black box testing by penetration testers can miss unlinked endpoints without extensive endpoint brute forcing

Parameter detection – Identifying optional parameters during a black box test can be time-consuming and often miss valid parameters that affect execution of the software

Enumeration effort – Manual penetration testing is costly, and the available time may not allow for thorough enumeration of an application's attack surface

How the Attack Surface Detector Works

Static code analysis identifies web application endpoints by parsing routes and identifying parameters in the supported languages and frameworks

Multiple parsers are needed in order to support different languages and frameworks

Supported Frameworks:

- C# / ASP.NET MVC
- C# / Web Forms
- Java / Spring MVC
- Java / Struts
- Java / JSP
- Python / Django
- Ruby / Rails

Pre-seeding in Burp Suite

Source Code Analysis

Use Attack Surface Detector to analyze the server side source code to detect endpoints and parameters and import them into Burp.
These results may include URL endpoints and optional parameters a spider may not find.

Total Endpoints Detected: 83

Detected Endpoints	Number of Detected Parameters	GET Method	POST Method
/redirect/uriTemplate	0	☐	☐
/async/callable/response-body	0	☐	☐
/async/callable/custom-timeout-handling	0	☐	☐
/messageconverters/string	0	☐	☐
/messageconverters/string	0	☐	☐
/async/deferred-result/exception	0	☐	☐
/form	8	☐	☐
/form	0	☐	☐
/mapping/path	0	☐	☐
/views/*/pathVariables/{foo}/{fruit}	2	☐	☐
/data/param	1	☐	☐
/data/group	0	☐	☐
/data/body	0	☐	☐
/data/path/{var}	1	☐	☐
/data/standard/response/writer	0	☐	☐

Selected Endpoint

URL:
/views/*/pathVariables/{foo}/{fruit}

Methods:
GET

Parameters and type:
fruit - String
foo - String

Contacts and Source Code

Code Pulse Website

<https://code-pulse.com/>

OWASP Code Pulse project site

[https://owasp.org/index.php/OWASP Code Pulse Project](https://owasp.org/index.php/OWASP_Code_Pulse_Project)

GITHUB LINKS

Attack Surface Detector plugin for Burp Suite

<https://github.com/secdec/attack-surface-detector-burp>

Attack Surface Detector plugin for OWASP ZAP

<https://github.com/secdec/attack-surface-detector-zap>

OWASP Code Pulse real-time code coverage monitor

<https://github.com/secdec/codepulse>

Lenny Halseth

LennyH@securedecisions.com

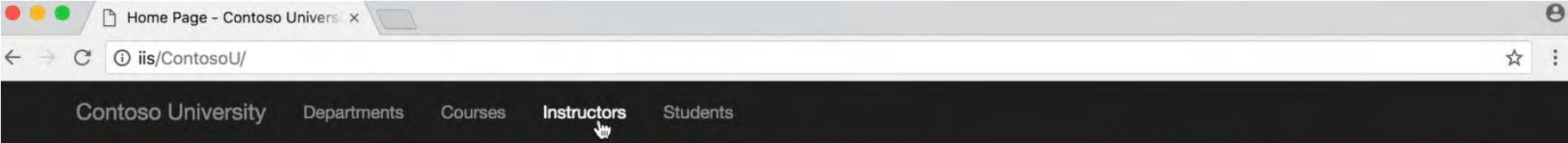
info@securedecisions.com



Plan B – Backup slides



Demo Screenshot – Contoso University Home Page



Welcome to Contoso University

Contoso University is a sample application that demonstrates how to use Entity Framework 6 in an ASP.NET MVC 5 web application.

© 2018 - Contoso University



















Demo Screenshot – Contoso University Courses

Contoso University Departments Courses Instructors Students

Courses

Create New

Select Department: All Filter

Number	Title	Credits	Department	
106	Computer Basics	3	Computer Science	  
107	Introduction to Programming	3	Computer Science	  
115	Computer Science 1	3	Computer Science	  
119	Computer Science 2	3	Computer Science	  
123	Precalculus	3	Mathematics	  
127	Calculus I	3	Mathematics	  

Demo Screenshot – Controller Code Treemap

The screenshot shows the Code Pulse CodeDx interface. On the left, the 'Application Inventory' table lists classes and their coverage. The main area displays a treemap of code, with a tooltip showing the details of a method in the CourseController. On the right, there are settings for the treemap legend and recordings.

Application Inventory	Trace	method count	% Coverage
Classes	0	1198	0%
Antlr.Runtime		898	0%
ContosoUniversity		300	0%
<self>		6	0%
.Controllers		54	0%
.DAL		34	0%
.Logging		13	0%
.Migrations		112	0%
.Models		65	0%
.ViewModels		16	0%

Code Treemap

Classes
ContosoUniversity
ContosoUniversity.Controllers

public ActionResult Details(Int32>)
ContosoUniversity.Controllers
CourseController
public ActionResult Details(Int32>)

Treemap Legend

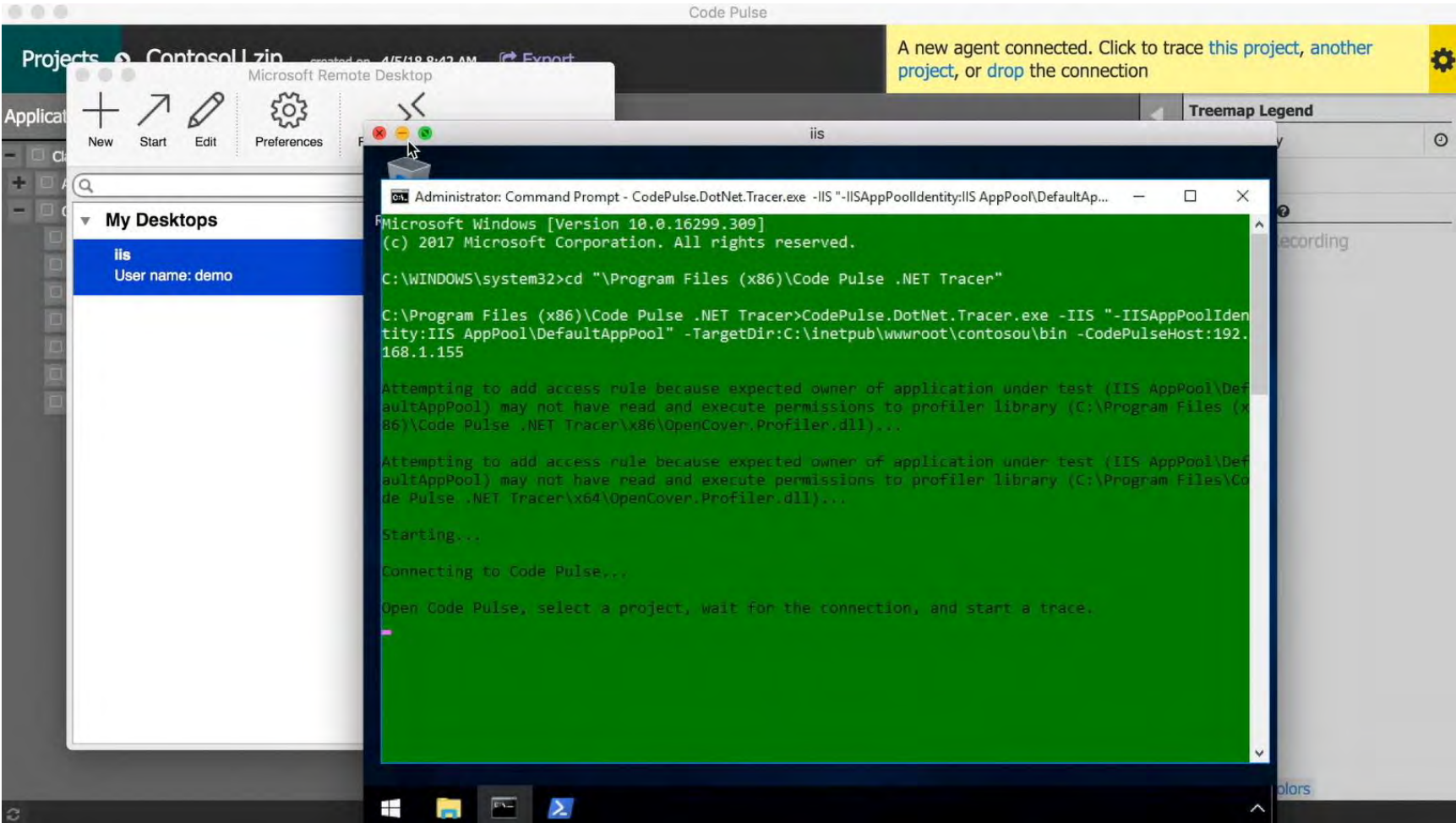
- Latest 5 minutes
- Overlaps

Recordings

- + Start a Recording

Reset colors

Demo Screenshot – Code Pulse .NET Tracer



Demo Screenshot – Home Page Code Coverage

The screenshot shows a web browser window with the URL `iis/ContosoU/Course`. The page displays a list of courses from Contoso University. A Code Pulse overlay is visible on the right side, showing a treemap of code coverage for the project `ContosoU.zip`, created on 4/5/18 8:42 AM. The treemap shows a large dark area representing code coverage, with a smaller dark area in the center. The treemap is titled 'Code Treemap' and shows a hierarchy of classes: `ContosoUniversity` and `ContosoUniversity.Controllers`.

Number	Title	Credits	Department	
106	Computer Basics	3	Computer Science	
107	Introduction to Programming	3	Computer Science	
115	Computer Science 1	3	Computer Science	
119	Computer Science 2	3	Computer Science	
123	Precalculus	3	Mathematics	
127	Calculus I	3	Mathematics	
128	Calculus II	3	Mathematics	

Demo Screenshot – Courses Code Coverage (Partial)

The image shows a side-by-side comparison of a web application and its code coverage analysis. On the left is a browser window displaying the 'Courses - Contoso University' page. On the right is the 'Code Pulse' interface for the 'ContosoU.zip' project, showing a 'Code Treemap' view.

Web Application (Left):

- URL: `iis/ContosoU/Course`
- Page Title: Contoso University
- Section: Courses
- Buttons: Create New
- Filter: Select Department: All
- Table:

Number	Title	Credits	Department
106	Computer Basics	3	Computer Science
107	Introduction to Programming	3	Computer Science
115	Computer Science 1	3	Computer Science
119	Computer Science 2	3	Computer Science
123	Precalculus	3	Mathematics
127	Calculus I	3	Mathematics
129	Calculus II	3	Mathematics

Code Pulse (Right):

- Project: ContosoU.zip (created on 4/5/18 8:42 AM)
- View: Code Treemap
- Classes: ContosoUniversity, ContosoUniversity.Controllers
- Highlighted Code:

```
public ActionResult UpdateCourseCredits()  
ContosoUniversity.Controllers  
CourseController  
public ActionResult UpdateCourseCredits()
```

Demo Screenshot – Courses Code Coverage (Full)

The screenshot is split into two main sections. On the left, a web browser window displays the 'Update Course Credits' page for 'Contoso University'. The browser's address bar shows the URL 'iis/ContosoU/Course/UpdateCourseCredits'. The page header includes the 'Contoso University' logo and a 'Courses' link. The main content area features the title 'Update Course Credits', a status message 'Number of rows updated: 29', and a 'Back to List' link. The footer contains the copyright notice '© 2018 - Contoso University'.

On the right, the 'Code Pulse' interface is shown for a project named 'ContosoU.zip', created on '4/5/18 8:42 AM'. The 'Code Treemap' view displays a hierarchical tree of code coverage. The root node is 'Classes', which branches into 'ContosoUniversity' and 'ContosoUniversity.Controllers'. The 'ContosoUniversity.Controllers' node is expanded, showing a large, dark grey rectangular area representing high code coverage. A mouse cursor is positioned over this area. The treemap also shows other nodes with varying levels of coverage, represented by different shades of grey.

Demo Screenshot – DAST Tool 1

The screenshot displays the Code Pulse interface for a project named 'ContosoU.zip'. The interface is divided into several sections:

- Application Inventory:** A table listing application components with their method counts and coverage percentages.
- Code Treemap:** A visualization of code coverage across the application's structure.
- Terminal:** A PowerShell window showing the execution of a simulation script.
- Right Panel:** Controls for the treemap legend and recording status.

Application Inventory	Trace	method count	% Coverage
Classes	0	1198	5%
Antlr.Runtime		898	0%
ContosoUniversity		300	23%
<self>		6	33%
.Controllers		54	29%
.DAL		34	55%
.Logging		13	15%
.Migrations		112	<1%
.Models		65	40%

```
ContosoU — powershell ./SimulateRunOfTool1.ps1...  
demo$ powershell ./SimulateRunOfTool1.ps1
```

Demo Screenshot – DAST Tool 2

The screenshot displays the Code Pulse CodeDx interface for a project named 'ContosoU.zip'. The interface is divided into several sections:

- Application Inventory:** A table listing application components with their method counts and coverage percentages.
- Code Treemap:** A central visualization showing the distribution of code activity across different classes and methods, with orange and grey blocks representing different levels of activity.
- Terminal:** A blue terminal window at the bottom left showing the execution of simulation scripts.
- Right Panel:** Contains a 'Treemap Legend' with options for 'All Activity' and 'Overlaps', and a 'Recordings' section with 'Start a Recording' and two recording tools (Tool 1 and Tool 2).

Application Inventory	Trace	method count	% Coverage
Classes	✖ 0	1198	10%
Antlr.Runtime	✓	898	0%
ContosoUniversity	✓	300	41%
<self>	✓	6	33%
.Controllers	✓	54	66%
.DAL	✓	34	61%
.Logging	✓	13	15%
.Migrations	✓	112	<1%
.Models	✓	65	76%

```
demo$ powershell ./SimulateRunOfTool1.ps1
demo$ powershell ./SimulateRunOfTool2.ps1
demo$
```

Demo Screenshot – DAST Tool Overlap

