NOPSEC

Pen Testing Toolkit: White Hat Tools to Improve Web Application Penetration Testing

June 13, 2018

Drew Kirkpatrick

Many of our clients at NopSec have mature web application security programs with their own internal penetration testing personnel. Performing penetration testing in coordination with an agile software development team presents unique challenges as the speed of feature development can make thorough testing of the application difficult to achieve. The more time spent assessing the security of the application, the more out of sync with the development team the tester becomes. There are however some interesting advantages a penetration tester has when they're part of the development team, such as access to the application source code and the application hosting server. Unfortunately few penetration testing tools actually take advantage of access to source code and application servers.

I recently gave a BSides talk demonstrating two such penetration testing tools. These are open source projects of my prior employer Secure Decisions (https://securedecisions.com (https://securedecisions.com)) who has a long history of developing unique and powerful application security tools. These two tools improve white box penetration testing by better enumerating the attack surface of the application and by tracking the real-time code coverage to identify testing gaps.

Attack Surface Detector

The Attack Surface Detector tool performs static code analysis to detect the application endpoints, parameters, and parameter data types. The benefit of performing this analysis on the server side source code is that unlinked endpoints are discovered that a spider or brute force guessing wouldn't find, and optional parameters not seen in the client side code are readily discovered. The findings of static code analysis are then imported into Burp Suite and OWASP ZAP and added to the target sitemaps to make the

findings easy to test. The plugins also support analyzing two different versions of the web applications source code, and it will highlight the endpoints and parameters that are new or modified in the latest version of the application to help prioritize penetration testing efforts.

In my own use of the Attack Surface Detector I've found that I can discover significantly more endpoints and parameters in a web application over spidering and brute force guessing alone. Starting your enumeration phase with the attack surface detector and then proceeding to manual exploring, spidering, and brute forcing maximizes the enumerated attack surface in minimal time. If your application is implemented in a supported framework and you have access to the source code, I highly encourage you to try this tool out. You could be surprised how many endpoints and parameters have been going untested in your application.

The static code analysis in the Attack Surface Detector currently works for the following languages and frameworks:

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

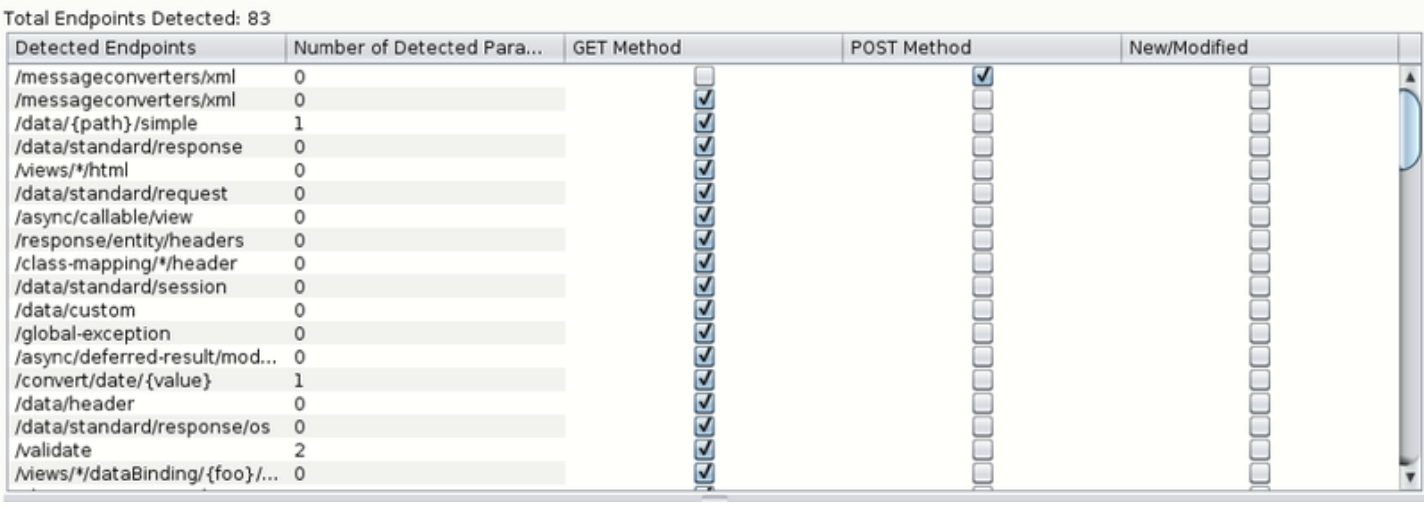
You can get the Burp and ZAP plugins on github:

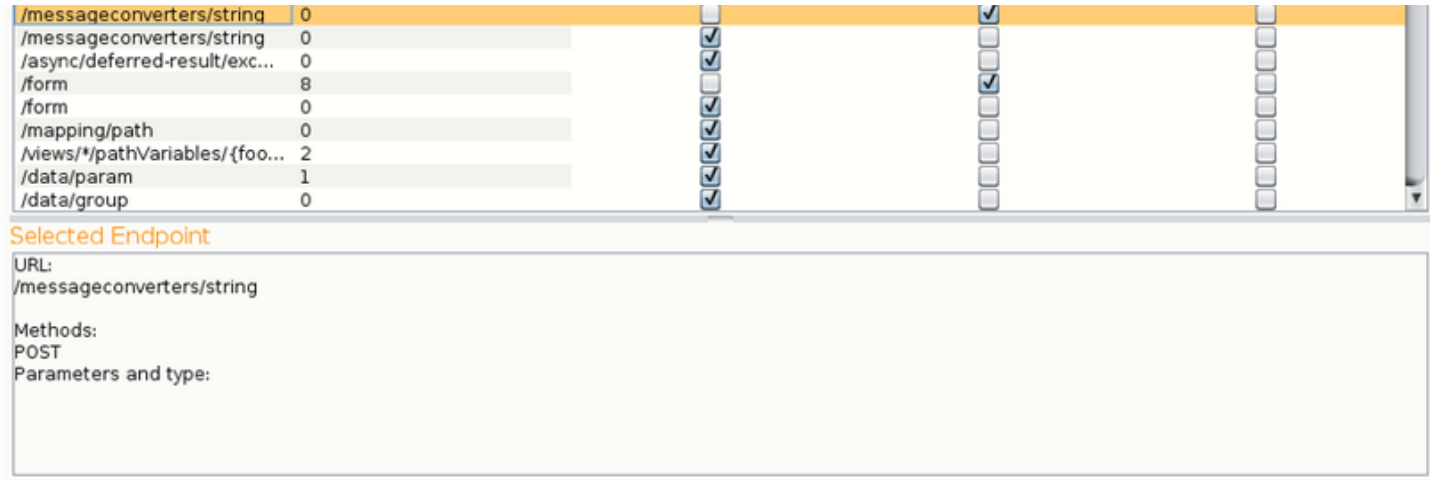
https://github.com/secdec/attack-surface-detector-burp

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

A demo video of an early version of the software can be seen on youtube:

https://youtu.be/jUUJNRcmqwI



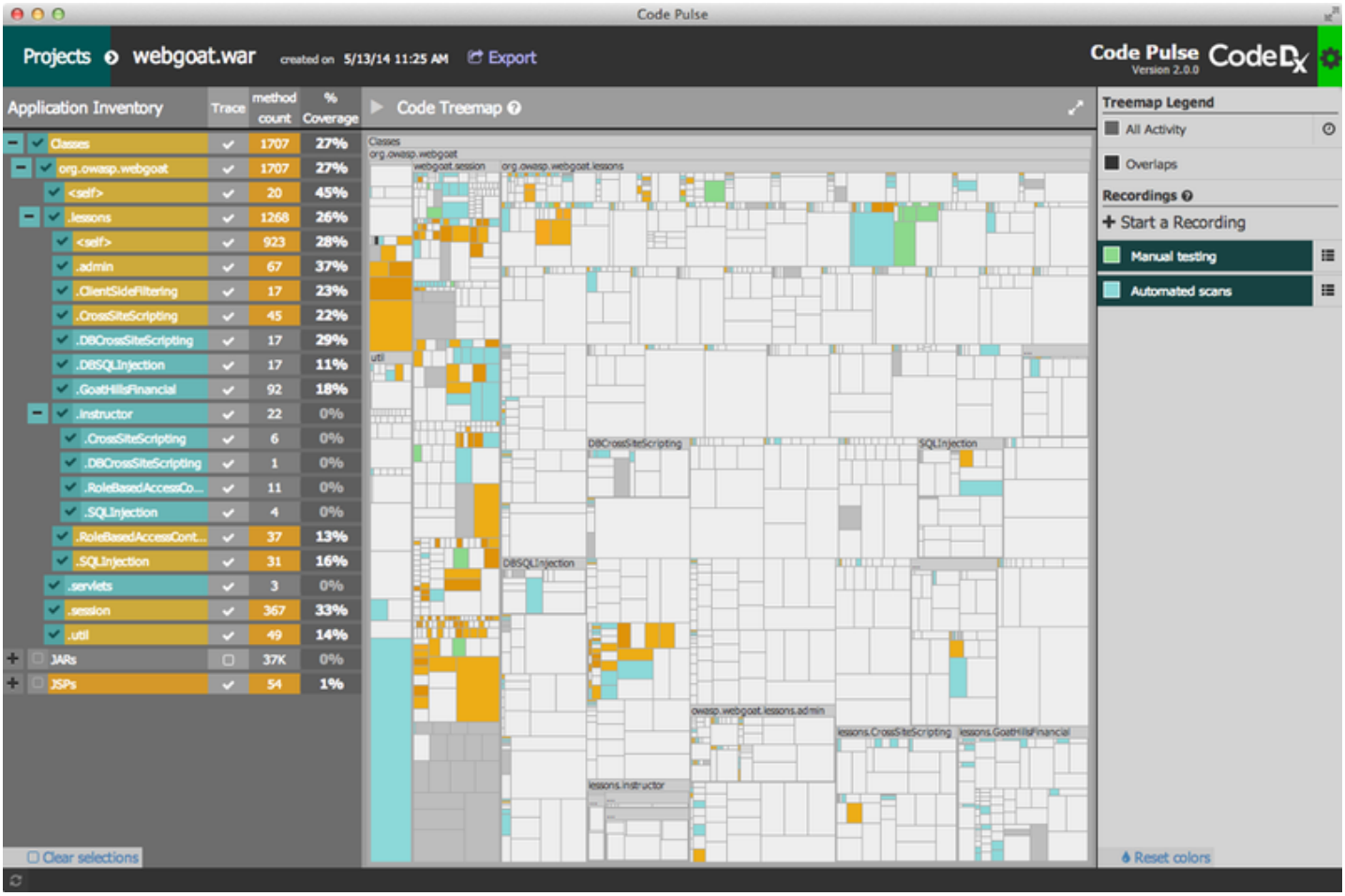


OWASP Code Pulse

The second tool, OWASP Code Pulse, instruments the web application server bytecode to provide real-time code coverage to help identify gaps in testing, help tune and compare testing tools, as well as provide a useful metric for communicating testing activities. This tool has been available for instrumenting Java web applications for a few years, but has recently been updated to add support for instrumenting .NET web applications.

By instrumenting the bytecode of the web application itself, Code Pulse provides direct insight into the behavior of the application server while the penetration tester is interacting with or scanning the application. Instead of attempting to infer the behavior of the application server based on server responses, the tester can directly see the method execution by monitoring the desktop Code Pulse application visualization. This visualization makes it easy to see methods that have and have not been executed, and the sequence of their execution. The visualization is particularly useful to penetration testing when set to visualize the application controllers methods.

Code Pulse provides another novel method to identify unlinked endpoints in a web application that would often be missed by traditional spidering and brute force guessing methods. Code Pulse also calculates the percentage of methods called as a useful metric. These metrics can be used for easily communicating high level testing activity to stakeholders and to compare the performance of different scanning tools or different configurations for scanning tools.



You can find OWASP Code Pulse on github here:

https://github.com/codedx/codepulse

And you can find additional information on Code Pulse here:

https://www.owasp.org/index.php/OWASP_Code_Pulse_Project

http://code-pulse.com/

If you are a penetration tester who has access to source code or have access to the web application server, I encourage you to try these tools out to see if they can help improve your testing. I've found them useful in the past and hope you do as well.