OWASP ZAP – Zed Attack Proxy

By Russ McRee – ISSA member, Puget Sound (Seattle), USA Chapter

Join the Discussion
Connect

Prerequisites

Java Runtime Environment

ZAP runs on Linux, Mac OS X, and Windows

Happy Thanksgiving: "As we express our gratitude, we must never forget that the highest appreciation is not to utter words, but to live by them." - JFK

In November 2011's toolsmith is the 61st in the series for the *ISSA Journal*, thus marking five years of extensive tools analysis for information security practitioners. Thank you for coming along for the ride.

Fresh on the heels of a successful presentation on "OWASP Top 10 Tools and Tactics" at an even more successful ISSA International Conference in Baltimore, I was motivated to give full coverage this month to the OWASP Zed Attack Proxy, better known as ZAP. I had presented ZAP as a tool of choice when assessing OWASP Top Ten A1 – Injection, but, as so many of the tools discussed, ZAP delivers plenty of additional functionality worthy of in-depth discussion.

OWASP ZAP is a fork of the once favored Paros Proxy, which has not been updated since August 2006. As such, it should be noted with no small irony that we covered Paros in December 2006. This is an excellent opportunity to show you how far ZAP has come from the original project.

ZAP is the result of Simon Bennetts' (Psiinon) hard work, though he's got help from co-lead Axel Neumann (@a_c_neumann) and many contributors.[1]

As an official OWASP project, ZAP enjoys extensive use and development support as an "easy to use integrated penetration testing tool for finding vulnerabilities in web applications."[2]

Simon offered a veritable plethora of feedback for this article, as provided throughout the rest of the introduction. He indicated that he originally released ZAP specifically for developers and functional testers – a group which he believes is poorly represented in the security tools market.

Ease of use was a prime concern, as was documentation, and to his surprise it turned out that it was the security folk who took up ZAP the quickest, providing great feedback, reporting issues, and asking for lots of enhancements. Simon still wants ZAP to be ideal for people new to web application security, but it's also going to be enhanced with more and more advanced features aimed at profession penetration testers.

Simon also wanted ZAP to be a community project; there are many open source security tools that are tightly controlled by one individual or company. While he doesn't have a problem with that fact, he does believe that the real strength of open source comes when anyone can contribute to a project and take it in directions its initial developers never envisaged.

Anyone and everyone is welcome to contribute to ZAP, and not necessarily coding only; they welcome help with testing, documentation, localization, issues identification, and enhancement requests. Help spread the word as well via articles, reviews, videos, blogs, Twitter, etc.

ZAP is also one of the few open source security tools to be fully internationalized. It has been translated into 10 languages, and download statistics indicate that approximately half of the ZAP users worldwide are likely to be non-native English speakers.

ZAP is intended to provide everything that you need to perform a penetration test on a web application. If you are new to web application security, then it might be the only security tool you need. However, if you're an experienced penetration tester be sure to include it as one of the many tools in your toolbox. As such, the development team is trying to make it as easy as possible to integrate ZAP with other tools. They provide a way to invoke other applications from within ZAP, passing across the current context. In version 1.3 they introduced an API which allows the core ZAP functionality to be invoked by a REST API, and will be extended to cover even more of ZAP's features in future releases.

This is an ideal way for other applications to directly drive ZAP, and can be used when ZAP is running in "headless" mode (i.e. without the UI).

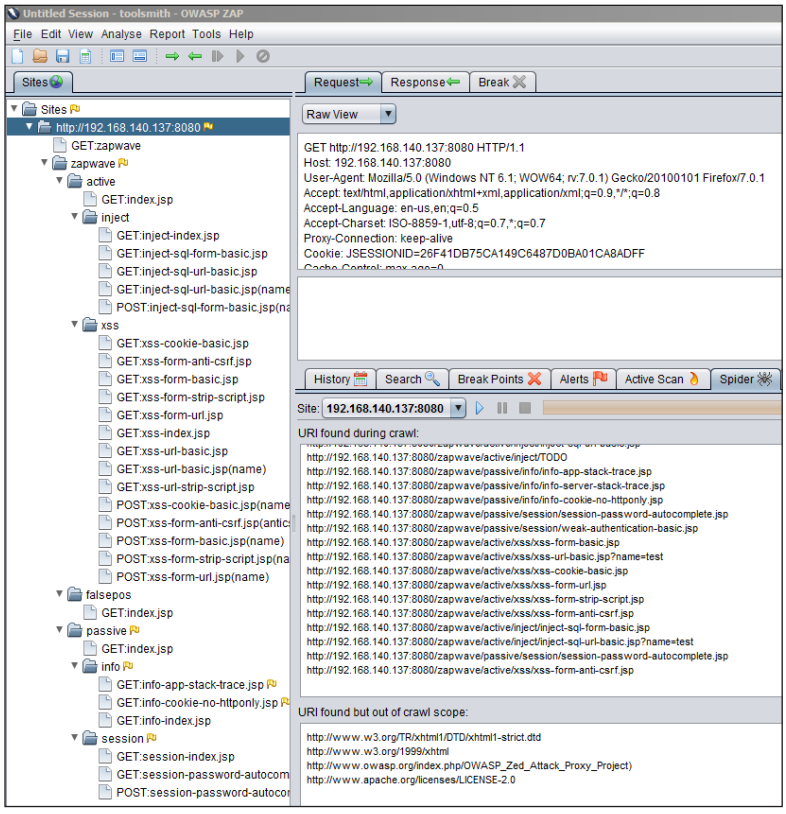
They've also put together a POC showing how ZAP can be used by developers to include basic security tests in their continuous integration framework and be alerted to potential security vulnerabilities within hours of checking code.[3]

1 http://code.google.com/p/zaproxy/people/list.

2 http://code.google.com/p/zaproxy/.

3 http://code.google.com/p/bodgeit/wiki/RegTests.

Figure 1 – ZAP spidering



Simon and team don't believe in reinventing the wheel, which is why they always seek high quality open source components to reuse before implementing a new feature from scratch. As such, the brute force/forced browsing support is provided via DirBuster[4] and fuzzing makes use of the JBroFuzz[5] libraries (both OWASP projects).

Amongst the more advanced features that users might not be aware of is that ZAP keeps track of all of the anti-CSRF tokens it finds. If fuzzing a form with an anti CSRF-token in it, ZAP can regenerate the token for each of the payloads you fuzz with. There's also an experimental option that allows this to be turned on when using the active scanner as well. I can say that quality CSRF testing is not commonplace among ZAP's web application testing contemporaries.

For ZAP version 1.4 the development team has decided to focus on:

- Improving the active and passive scanners
- Improving stability (especially for large sites)
- Session token analysis

In July 2011 ZAP was evaluated and designated as a "stable" OWASP project, the highest level currently available. Further, OWASP projects are now being restructured; ZAP has been designated as one of the small number of "flagship" projects.

Rightfully so; thank you, Simon.

Let's run ZAP through its paces.

ZAP installation and configuration

ZAP is installation is very simple. Once unpacked on your preferred platform, invoke ZAP from the application icon or at the command prompt via the appropriate executable. A current Java Runtime Environment is a requirement as all the executables (EXE, BAT, SH) invoke `java –jar zap.jar org.zaproxy.xap.ZAP`.

Most importantly ZAP, runs as a proxy. Configure your preferred browser to proxy via localhost and the default port of 8088. I change the port to 8088 to avoid conflict with other proxies and services. You can change the port under *Tools* → *Options* → *Local proxy* if you run multiple proxies that you bounce between during assessments. I do and as such I use the Firefox add-on FoxyProxy to quickly dial in my proxy of choice.

You must also generate an SSL certificate in order to use and test SSL-enabled sites. You will be prompted to do when running ZAP for the first time.

4 https://www.owasp.org/index.php/Category:OWASP_DirBuster_Project.

5 https://www.owasp.org/index.php/JBroFuzz.

ZAP use

In addition to the aforementioned Security Regression Tests for developers, the OWASP ZAP project offers ZAP Web Application Vulnerability Examples, or ZAP WAVE. Download it and drop `zap-wave.war` in the webapps directory of your favorite servlet engine. On Debian/Ubuntu systems `sudo apt-get install tomcat`[6] will get you in business with said servlet engine quickly. In addition to a LAMP stack on an Ubuntu 11.10 VM, I run Tomcat for just such occasions. OWASP WebGoat also runs as a standalone test bed or via a servlet engine.

Enable ZAP, with your browser configured to proxy through it, then navigate to the system (VM or real steel) hosting ZAP WAVE, usually on port 8080. As an example: `http://192.168.140.137:8080/zapwave/`.

ZAP WAVE includes "active" vulnerabilities such as cross-site scripting and SQL injection as well as "passive" vulnerabilities including three types of information leakage and two session vulnerabilities. There are also pending false positives that are not yet ready for primetime.

The developers recommend that you explore the target app with ZAP enabled as a proxy, and touch as much of it as possible before spidering. Doing so helps ZAP find more vulns as you may cross paths with error messages, etc.

I typically visit the root of the application hierarchy for a web application I wish to assess, right-click on it, select *Attack*, then *Spider site*. This crawls the entire site hierarchy and populates the tree view under the *Sites* tab in ZAP's left pane as seen in Figure 1.

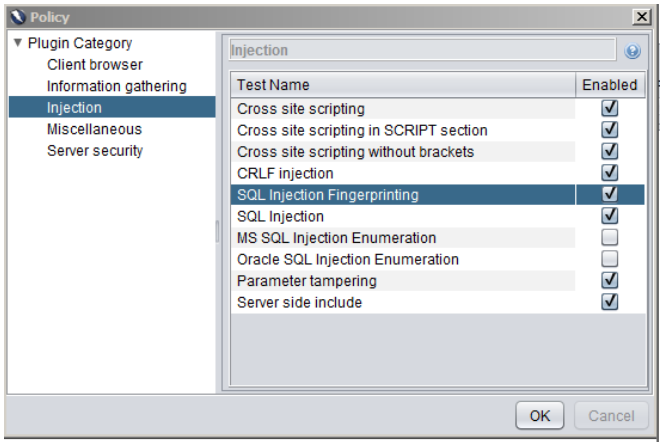


Figure 2 – ZAP scan policy

Crawling/spidering can have unintended side-effects on an application, even adding or deleting records in a database, so be advised.

A good crawl ensures a better active scan, but before beginning a scan, set your Scan Policy via *Analyze* → *Scan Policy* as seen in Figure 2. You may wish to more narrowly scope your scan activity to just the likes of information gathering or SQL injection as seen in Figure 2.

Spidering and scan policy configuration complete, right click the root, or a specific node you wish to assess as you can choose *Attack* → *Active scan site* or *Attack* → *Active scan node*. You can also exclude a site from the scope in a similar fashion.

A full scan of the ZAP WAVE instance completed in very short order; results were immediate as seen in Figure 3.

ZAP includes the expected Encode/Decode/Hash functionality via *Edit* → *Encode/Decode/Hash* or *Tools* → *Encode/Decode/Hash* along with a manual editor for generating manual requests. I'll often run ZAP for nothing more than encoding, decoding, and hashing; it's a great utility.

The Port Scan feature is also useful. It will select the in-scope host by default; just click the *Port Scan* tab then the start button.

The Brute Force tab is a function of the above-mentioned DirBuster component and includes seven dictionary lists to choose from. I ran this against my full host VM rather just the servlet element and included the dictionary-list-1.0 dictionary for a simple, quick test (Figure 4).

One of my favorite ZAP features (there are many) is the Fuzzer. Per the Fuzzer component guidance:

- Select a request in the Sites or History tab
- Highlight the string you wish to fuzz in the Request tab
- Right click in the Request tab and select *Fuzz...*
- Select the Fuzz Category and one or more Fuzzers
- Press the *Fuzz* button

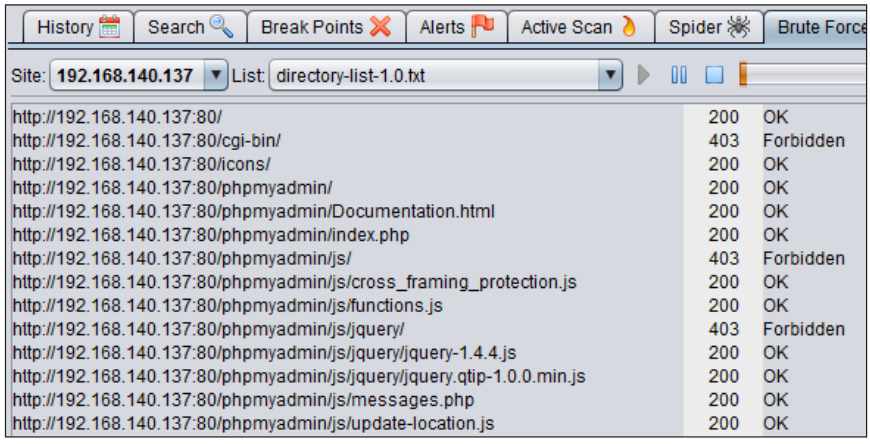


Figure 4 – ZAP DirBuster at work

The results are listed in the Fuzzer tab; select them to see the full requests and responses.

The fuzzer, like the scanner, includes functionality which causes ZAP to automatically regenerate the tokens when required. I ran Fuzzer against http://192.168.140.137:8080/za-

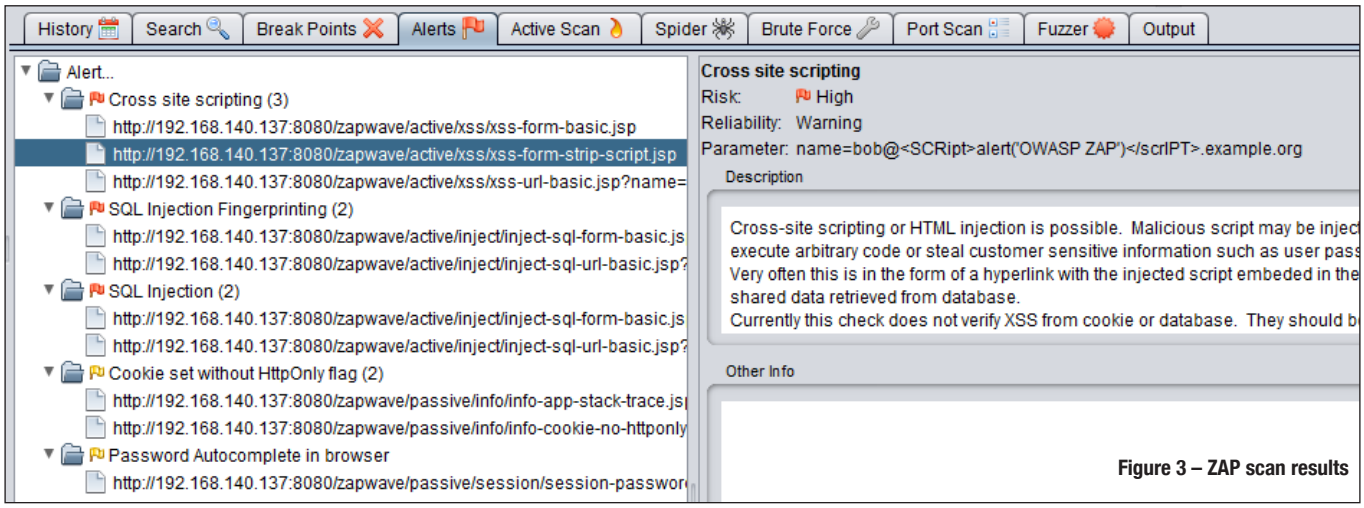


Figure 3 – ZAP scan results

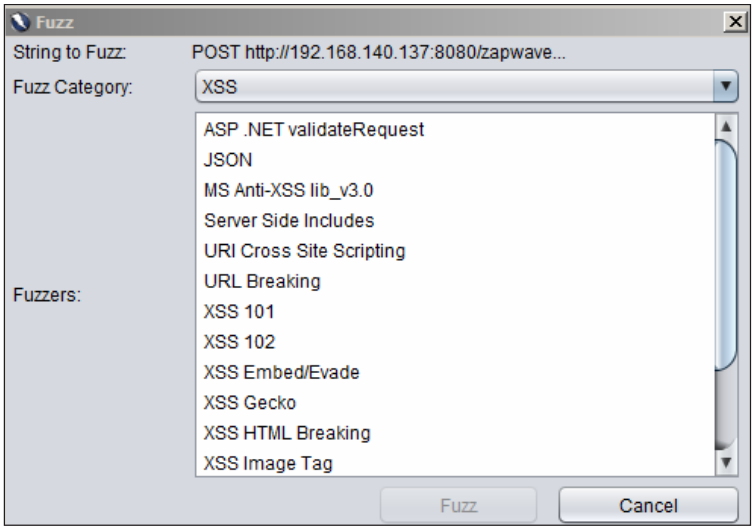


Figure 5 – ZAP fuzzer config

pwave/active/xss/xss-form-anti-csrf.jsp and fuzzed the `anticsrf` and name variables as it is a recent addition per the ZAP WAVE download site.

As seen in Figure 5, the fuzzer offers a wider array fuzzers within a given category.

In the understanding that fuzzing is the art of submitting a great deal of invalid or unexpected data to a target, you can look for variations in results such as response code (200 OK) and response times. Where normal response times per request average between 2ms and 4ms for ZAP WAVE hosted on a local VM, one request in particular stood out at a 402ms response time. I checked for the string passed and cracked up.

```
%3CIMG+SRC%3D%60javascript%3Aalert%28%22RSnake+says%23%23%23+%27XSS%27%22%29%60%3E
```

Or, courtesy of the handy ZAP decoder: `<IMG SRC=`javascript:alert(“RSnake says### ‘XSS’”)`>`

Mr. Slowloris HTTP DoS himself causing grind even here. ;-)

In conclusion

ZAP deserves its status as an OWASP flagship project. Whether you’re a seasoned veteran or new to the web application security game, make the Zed Attack Proxy part of your arsenal. I’d go so far as to say, as 2011 is winding down, that ZAP feels like a likely front runner for *2011 Toolsmith Tool of the Year*. But that is for you to decide, dear reader. Let me know if you agree.

Ping me via email if you have questions (russ at holisticinfosec dot org).

Cheers…until next month.

Acknowledgements

—Simon Bennetts (Psiinon) for project feedback and details

—Axel Neumann (@a_c_neumann) for draft review

About the Author

Russ McRee, GCIH, GCFA, GPEN, CISSP, is team leader and senior security analyst for Microsoft’s Online Services Security Incident Management team. As an advocate of a holistic approach to information security, Russ’ website is holisticinfosec.org. Contact him at russ at holisticinfosec dot org.