



Security Testing for Developers using OWASP ZAP

Simon Bennetts

OWASP ZAP Project Lead

Mozilla Security Team

psiinon@gmail.com

Overview

- Why you should be using ZAP
- Introduction to ZAP
- ZAP Use cases
- ZAP API
- ZAP Scripting
- Wrap up



My questions for you :)

- Who's heard of OWASP?
- Who's heard of ZAP?
- Who's used ZAP?
- Who does *any* security testing in development?
- Who thinks they do *enough* security testing in development?

“You cannot build secure web applications unless you know how they will be attacked”

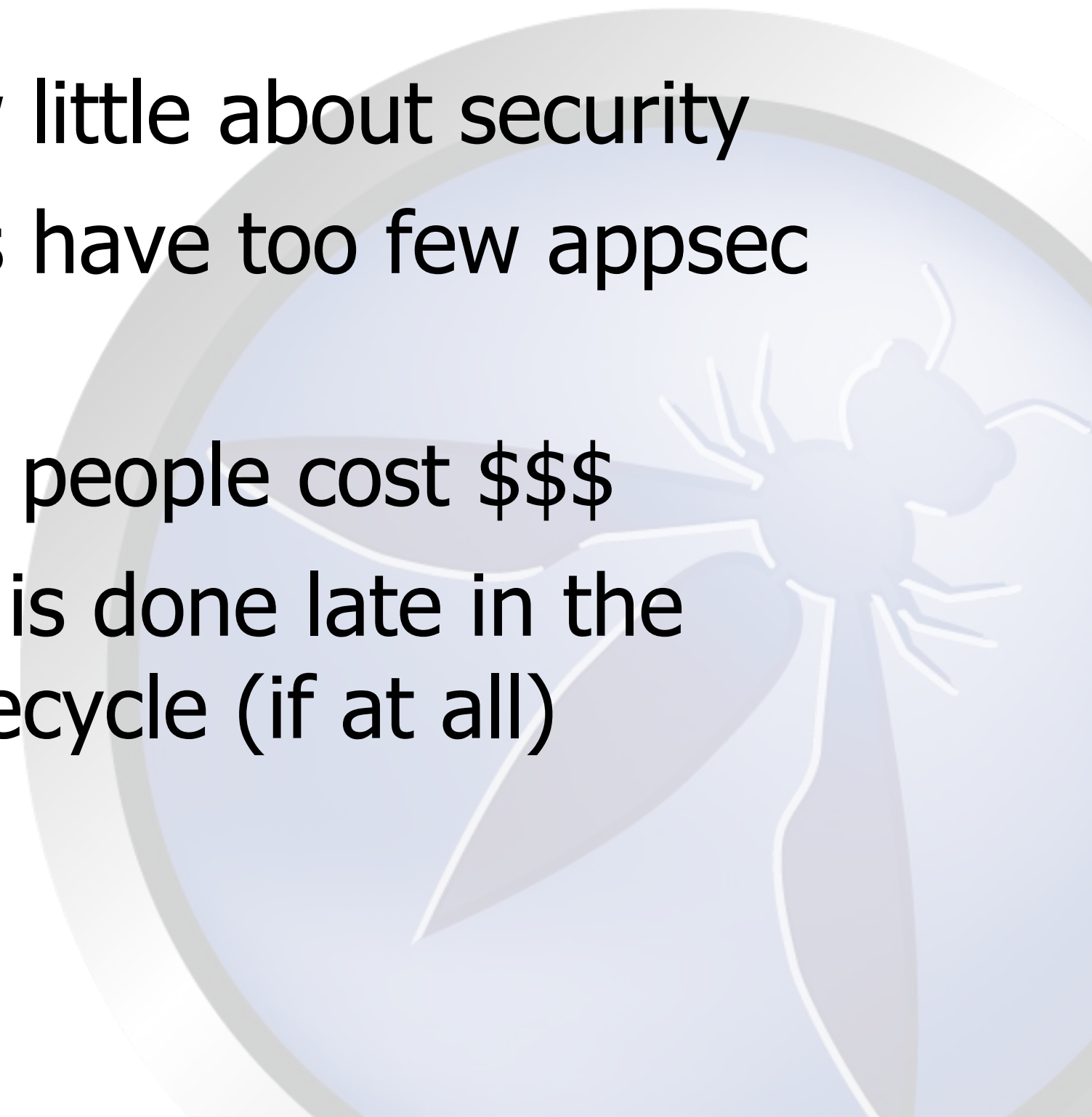


“This was fine for your nephew’s fifth, Sire, but I fear it is set for a sterner test.”

Thanks to Royston Robertson www.roystonrobertson.co.uk for permission to use his cartoon!

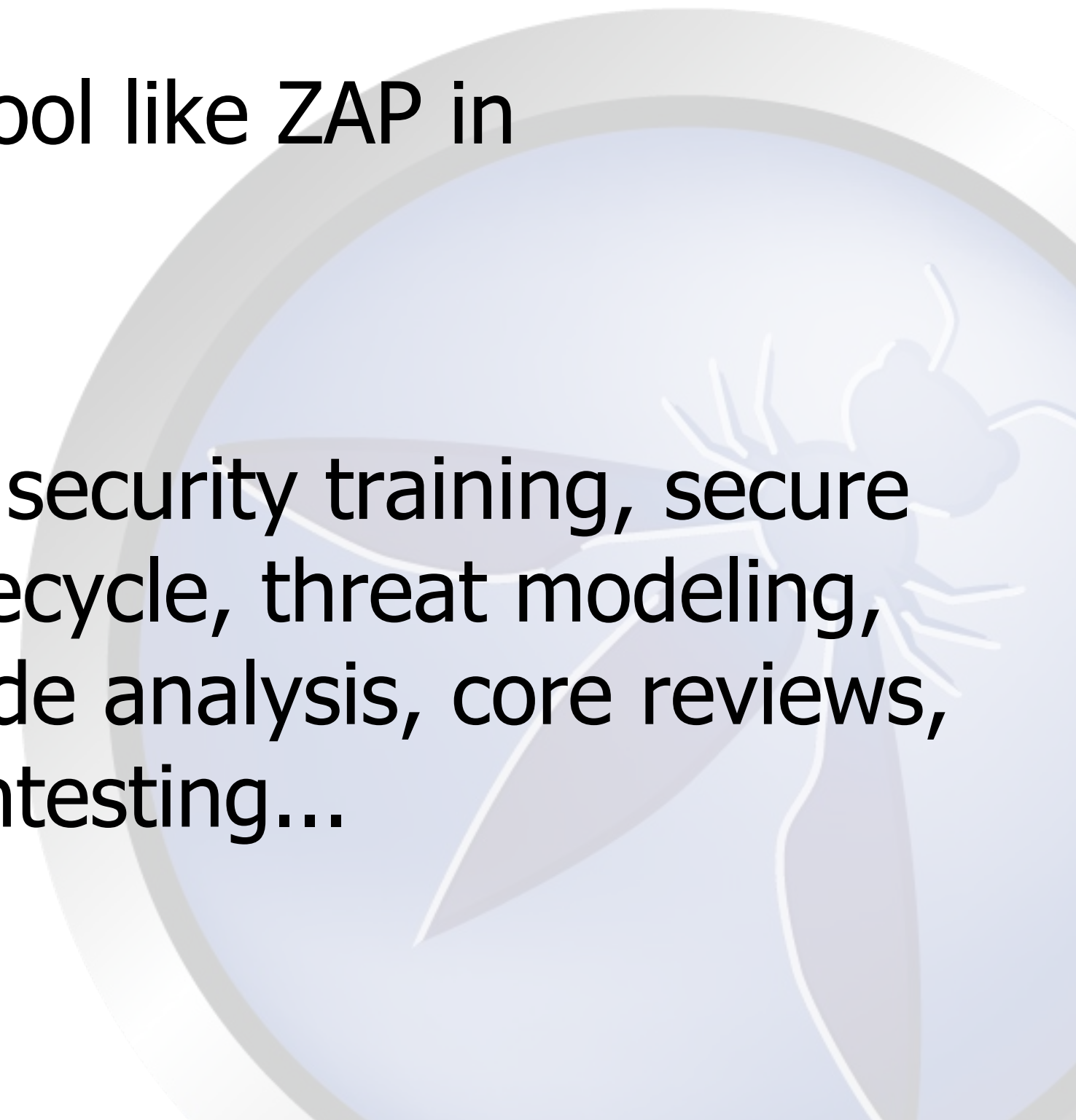


The problems

- Most devs know little about security
 - Most companies have too few appsec folk
 - External appsec people cost \$\$\$
 - Security testing is done late in the development lifecycle (if at all)
- 



Part of the Solution

- Use a security tool like ZAP in development :)
 - In addition to a security training, secure development lifecycle, threat modeling, static source code analysis, core reviews, professional pentesting...
- 

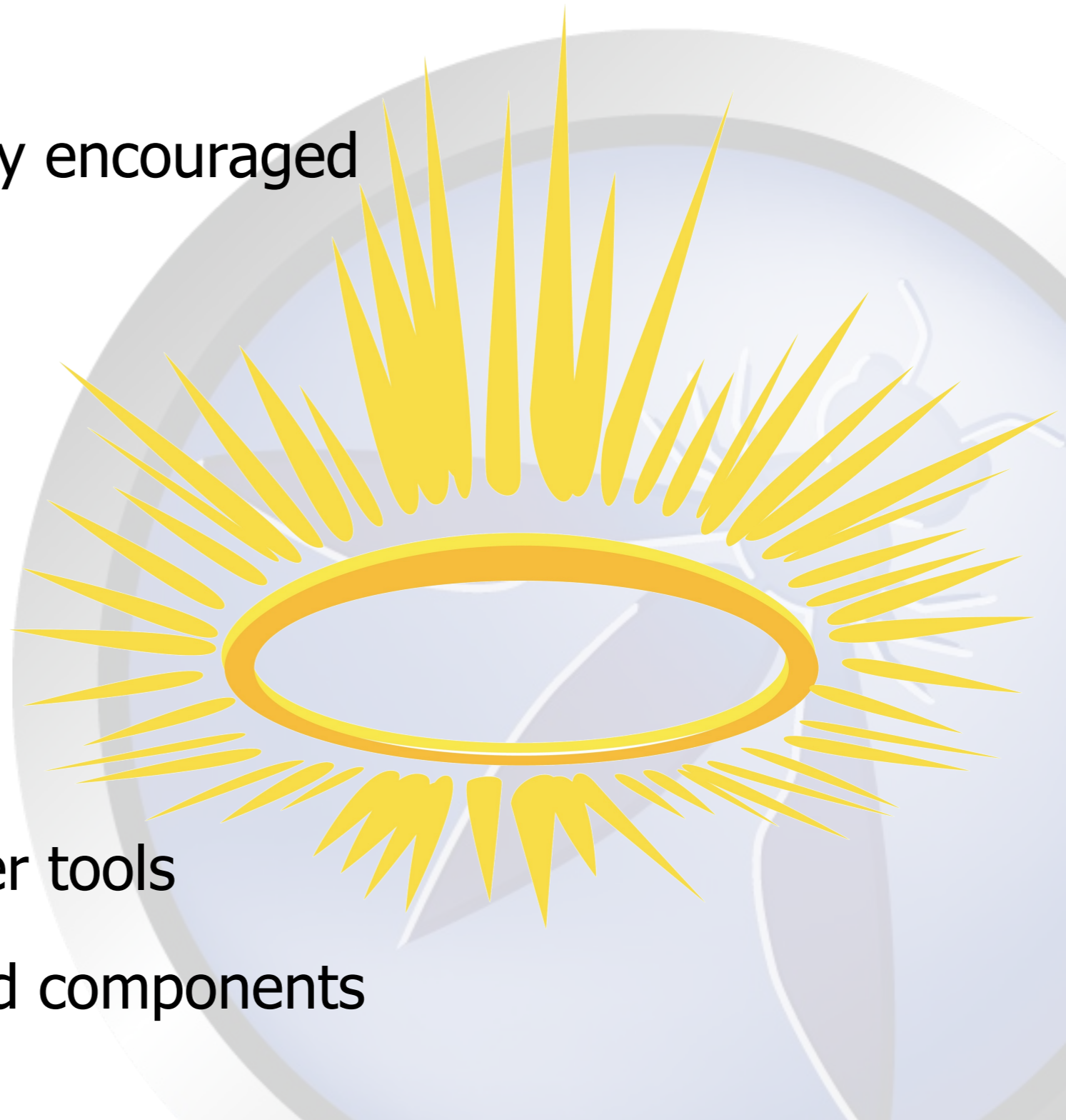
What is ZAP?

- An easy to use webapp pentest tool
- Completely free and open source
- Ideal for beginners
- But also used by professionals
- Ideal for devs, esp. for automated security tests
- Becoming a framework for advanced testing
- Included in all major security distributions
- ToolsWatch.org Top Security Tool of 2013
- Not a silver bullet!



ZAP Principles

- Free, Open source
- Involvement actively encouraged
- Cross platform
- Easy to use
- Easy to install
- Internationalized
- Fully documented
- Work well with other tools
- Reuse well regarded components



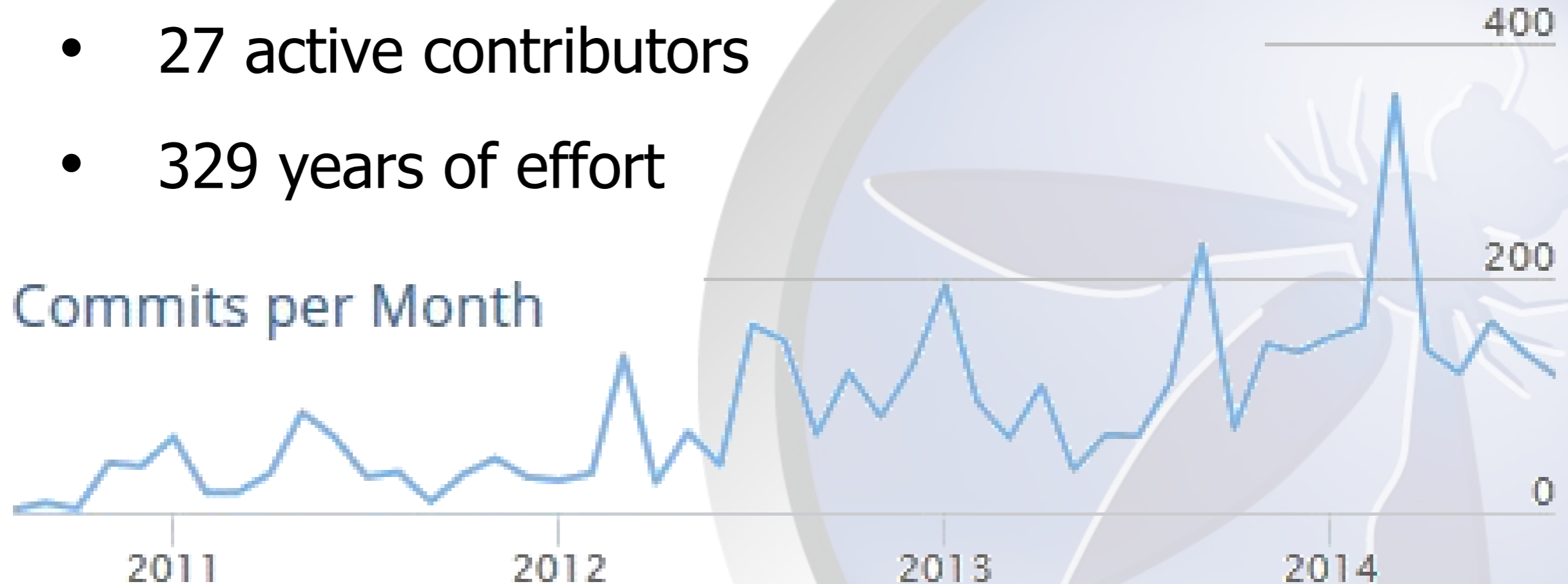
Statistics

- Released September 2010, fork of Paros
- V 2.3.1 released in May 2014
- V 2.3.1 downloaded > 70K times
- Translated into 20+ languages
- Over 100 translators
- Mostly used by Professional Pentesters?
- Paros code: ~20% ZAP Code: ~80%



Ohloh Statistics

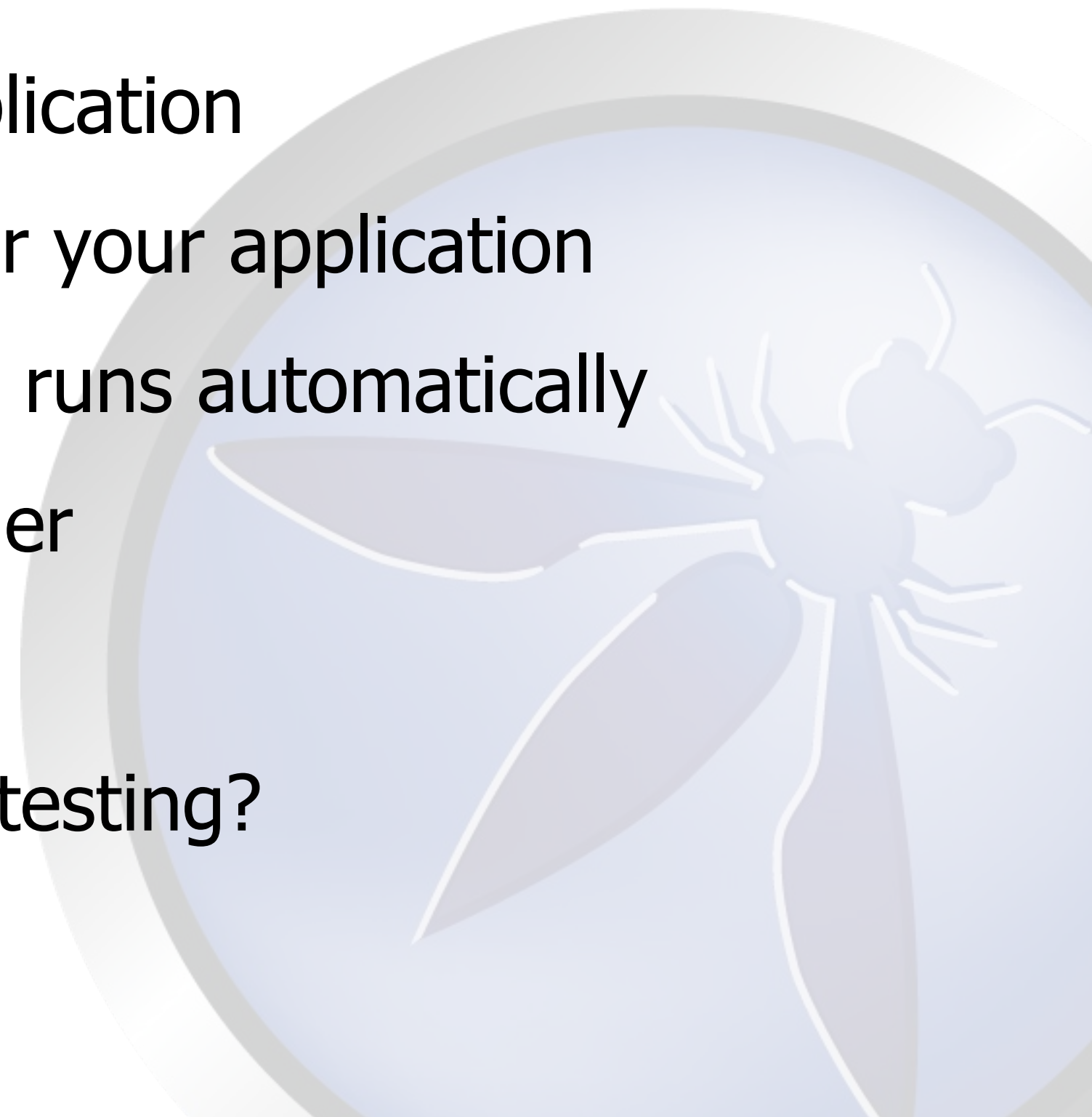
- 🏗️ Very High Activity
- The most active OWASP Project
- 27 active contributors
- 329 years of effort



Source: <http://www.ohloh.net/p/zaproxy>



Typical ZAP use

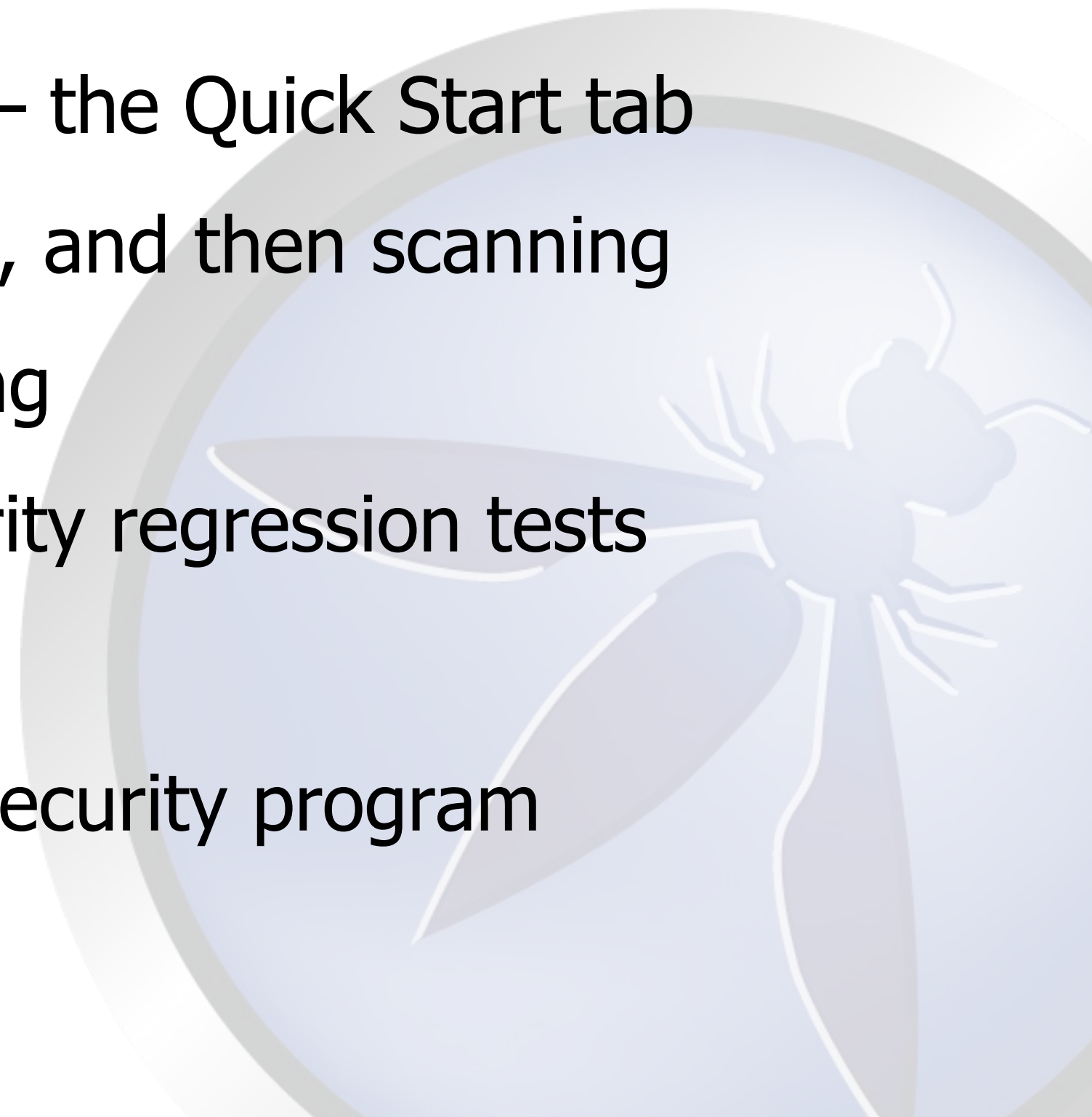
1. Explore your application
 2. Configure ZAP for your application
 3. Passive scanning runs automatically
 4. Run active scanner
 5. Fine tuning?
 6. Perform manual testing?
- 

What to configure?

- Pages to ignore (logout, duplicates)
- Anti CSRF tokens
- Session handling
- Authentication
- Users
- Structure (single page apps)
- 'Non standard' separators
e.g. aaa:bbb;ccc:ddd



Some ZAP use cases

- Point and shoot – the Quick Start tab
 - Proxying via ZAP, and then scanning
 - Manual pentesting
 - Automated security regression tests
 - Debugging
 - Part of a larger security program
- 

Quick Start Attack



The screenshot shows the OWASP Zed Attack Proxy (ZAP) interface. At the top, there is a navigation bar with five tabs: 'Quick Start' (selected), 'Request', 'Response', 'Break', and 'Script Console'. Below the navigation bar, the main content area displays a welcome message and instructions for performing a quick start attack.

Welcome to the OWASP Zed Attack Proxy (ZAP)

ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

Please be aware that you should only attack applications that you have been specifically given permission to test.

To quickly test an application, enter its URL below and press 'Attack'.

URL to attack:

Progress: Attack complete - see the Alerts tab for details of any issues found

Proxying via ZAP

Options:

- Plug-n-Hack

If you are using Firefox 24.0 or later you can use 'Plug-n-Hack' to configure your browser:

Configure your browser:



Or point your browser at:

<http://localhost:8090/pnh/>

- Configure your browser's proxy manually

Right click everywhere!

The screenshot displays a web application security tool interface. The top navigation bar includes tabs for 'Sites' and 'Scripts', and buttons for 'Quick Start', 'Request', 'Response', and 'Break'. The main area is divided into a left sidebar and a right pane. The sidebar shows a tree view of sites, with 'http://localhost:8080' expanded to show a folder named 'bodgeit'. Inside 'bodgeit', a list of files is shown, with 'GET:home.jsp' selected. A context menu is open over this file, listing various actions. The 'Attack' sub-menu is expanded, showing options like 'Active Scan all in Scope', 'Spider Context...', 'Forced Browse site', and 'AJAX Spider Site'. The right pane shows the request details for the selected file, including the URL and headers.

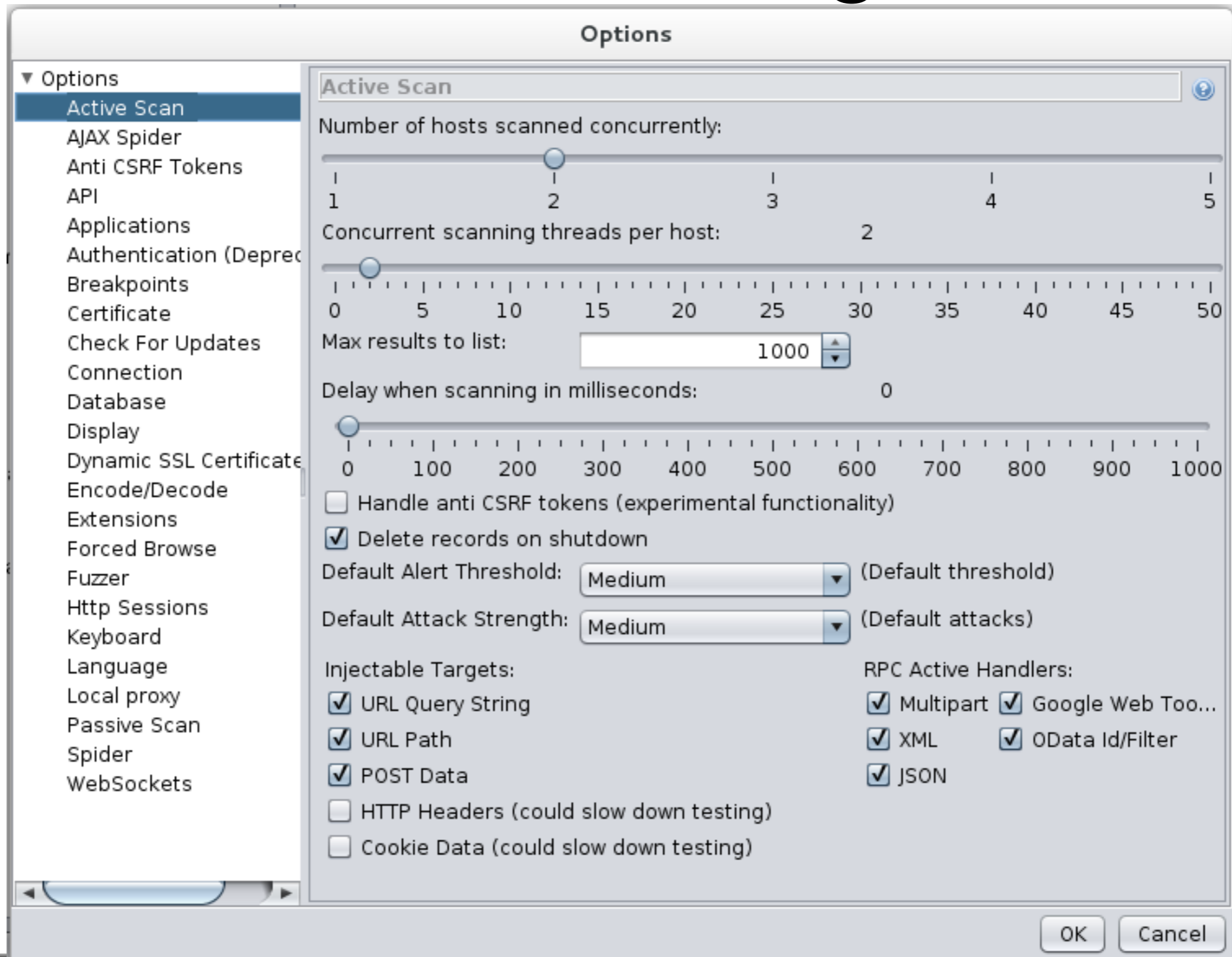
Header: Text Body: Text

GET http://localhost:8080/bodgeit/home.jsp H
User-Agent: Mozilla/4.0 (compatible; MSIE 6.
Pragma: no-cache

Attack

- Active Scan all in Scope
- Active Scan site
- Active Scan subtree
- Active Scan single URL
- Spider Context...
- Spider all in Scope
- Spider site
- Spider Subtree
- Spider URL
- Forced Browse site
- Forced Browse directory
- Forced Browse directory (and children)
- AJAX Spider in Scope
- AJAX Spider Site

Fine tuning



More fine tuning

Policy

▼ Plugin Category

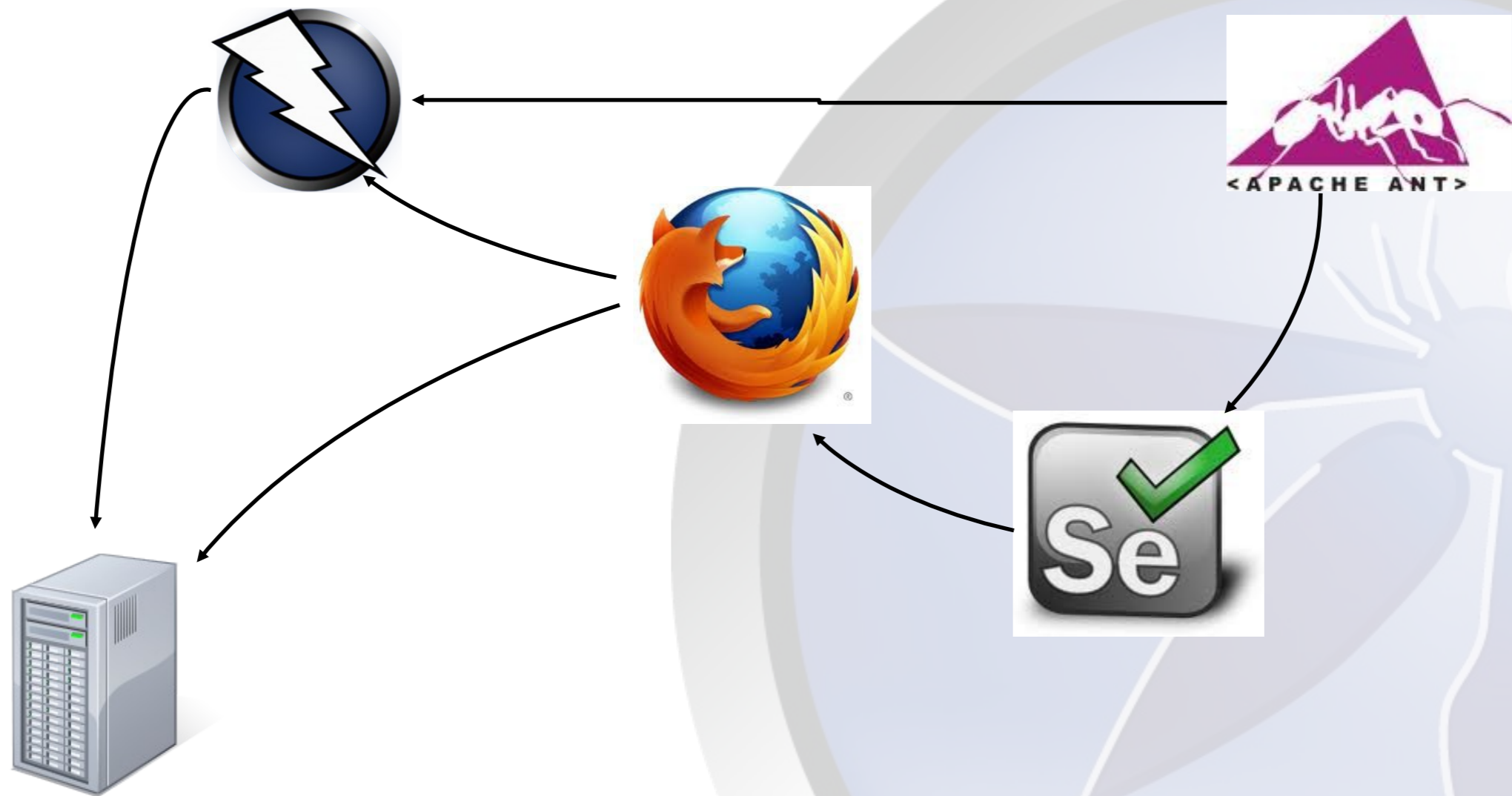
- Client browser
- Information gathering
- Injection**
- Miscellaneous
- Passive
- Server security

Test Name	Threshold	Strength
Server side include	Default	Default
Cross Site Scripting (Reflected)	Default	Default
Cross Site Scripting (Persistent)	Default	Default
LDAP Injection	Default	Default
SQL Injection	Default	Default
SQL Injection - MySQL	Default	Default
SQL Injection - Hypersonic SQL	OFF	Default
SQL Injection - Oracle	Default	Default
SQL Injection - PostgreSQL	Low	Default
Server Side Code Injection Plugin	Medium	Default
Remote OS Command Injection Plugin	High	Default
XPath Injection Plugin	Default	Default
CRLF injection	Default	Default
Parameter tampering	Default	Default
HTTP Parameter Pollution scanner	Default	Default
Cross Site Scripting (Persistent) - Prime	Default	Default

Thresholds and strengths can be changed by clicking on them

OK Cancel

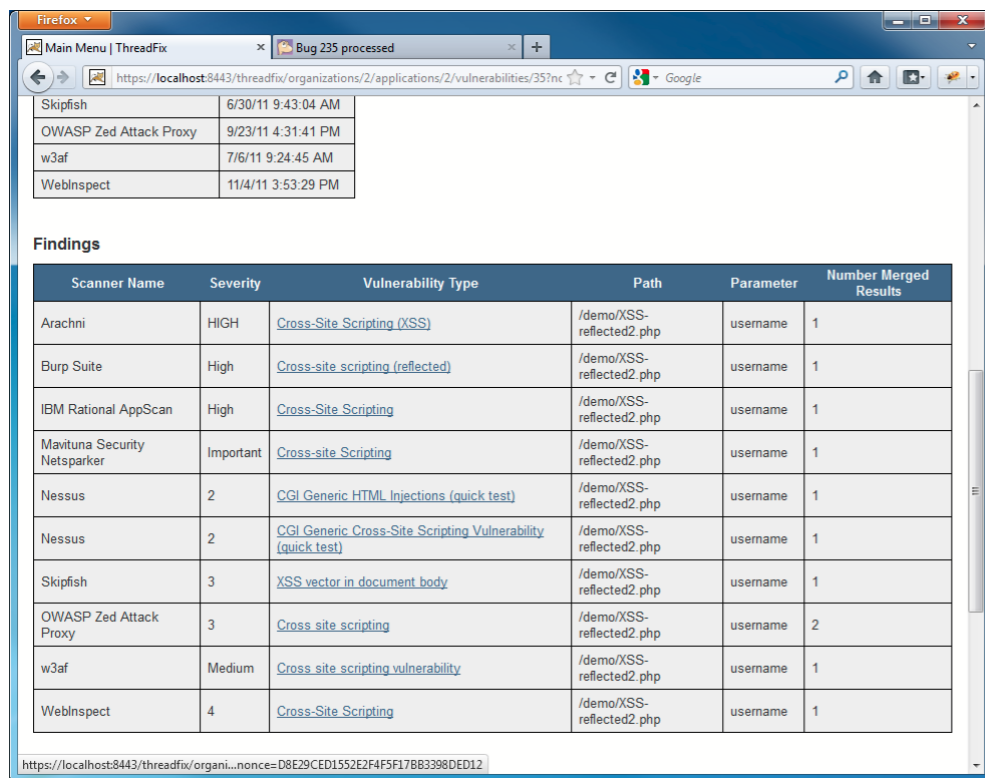
Security Regression Tests



<http://code.google.com/p/zaproxy/wiki/SecRegTests>

ZAP – Embedded

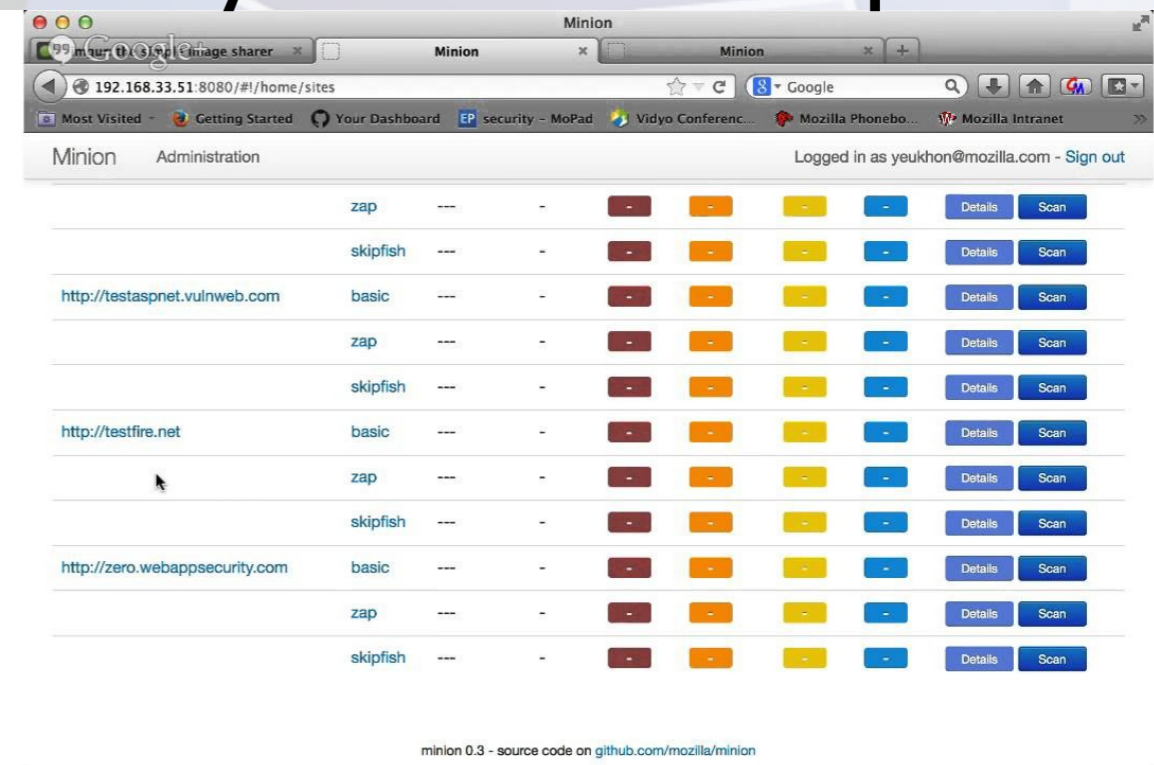
- ThreadFix – Denim Group
Software vulnerability aggregation and management system



The screenshot shows the ThreadFix web interface. At the top, there's a navigation bar with 'Main Menu | ThreadFix' and a tab for 'Bug 235 processed'. The main content area displays a table of findings. The table has columns for Scanner Name, Severity, Vulnerability Type, Path, Parameter, and Number Merged Results. Below the table, there's a 'Findings' section with a detailed table of results.

Scanner Name	Severity	Vulnerability Type	Path	Parameter	Number Merged Results
Arachni	HIGH	Cross-Site Scripting (XSS)	/demo/XSS-reflected2.php	username	1
Burp Suite	High	Cross-site scripting (reflected)	/demo/XSS-reflected2.php	username	1
IBM Rational AppScan	High	Cross-Site Scripting	/demo/XSS-reflected2.php	username	1
Mavituna Security Netsparker	Important	Cross-site Scripting	/demo/XSS-reflected2.php	username	1
Nessus	2	CGI Generic HTML Injections (quick test)	/demo/XSS-reflected2.php	username	1
Nessus	2	CGI Generic Cross-Site Scripting Vulnerability (quick test)	/demo/XSS-reflected2.php	username	1
Skipfish	3	XSS vector in document body	/demo/XSS-reflected2.php	username	1
OWASP Zed Attack Proxy	3	Cross site scripting	/demo/XSS-reflected2.php	username	2
w3af	Medium	Cross site scripting vulnerability	/demo/XSS-reflected2.php	username	1
Weblnspect	4	Cross-Site Scripting	/demo/XSS-reflected2.php	username	1

- Minion – Mozilla
Security automation platform

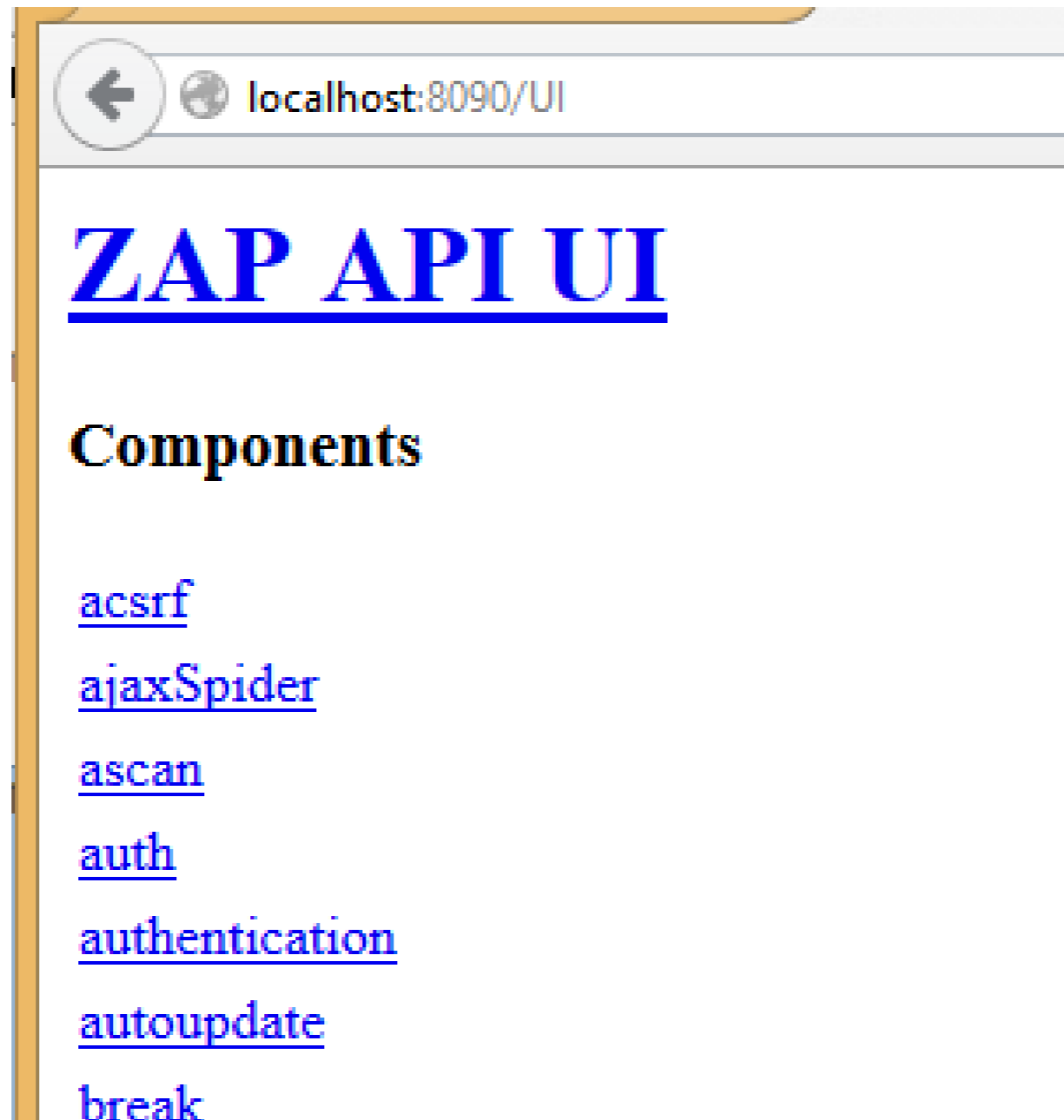


The screenshot shows the Minion web interface. At the top, there's a navigation bar with 'Minion Administration' and 'Logged in as yeukhon@mozilla.com - Sign out'. The main content area displays a table of scan configurations. The table has columns for URL, Scanner Name, and various status indicators (red, orange, yellow, blue). Each row has 'Details' and 'Scan' buttons.

URL	Scanner Name	Status 1	Status 2	Status 3	Status 4	Status 5	Buttons
	zap	---	-	-	-	-	Details Scan
	skipfish	---	-	-	-	-	Details Scan
http://testaspnet.vulnweb.com	basic	---	-	-	-	-	Details Scan
	zap	---	-	-	-	-	Details Scan
	skipfish	---	-	-	-	-	Details Scan
http://testfire.net	basic	---	-	-	-	-	Details Scan
	zap	---	-	-	-	-	Details Scan
	skipfish	---	-	-	-	-	Details Scan
http://zero.webappsecurity.com	basic	---	-	-	-	-	Details Scan
	zap	---	-	-	-	-	Details Scan
	skipfish	---	-	-	-	-	Details Scan

minion 0.3 - source code on github.com/mozilla/minion

The ZAP API



The ZAP API

- Direct access via:
 - `http://zap/` (if proxying through ZAP)
 - `http://<ip address>:<port>`
- API Clients:
 - Java
 - Python
 - Node.js
 - PHP
- <https://code.google.com/p/zaproxy/wiki/ApiDetails>

Scripting

- Full access to ZAP internals
- Support all JSR 223 languages, inc
 - JavaScript
 - Jython
 - JRuby
 - Zest :)

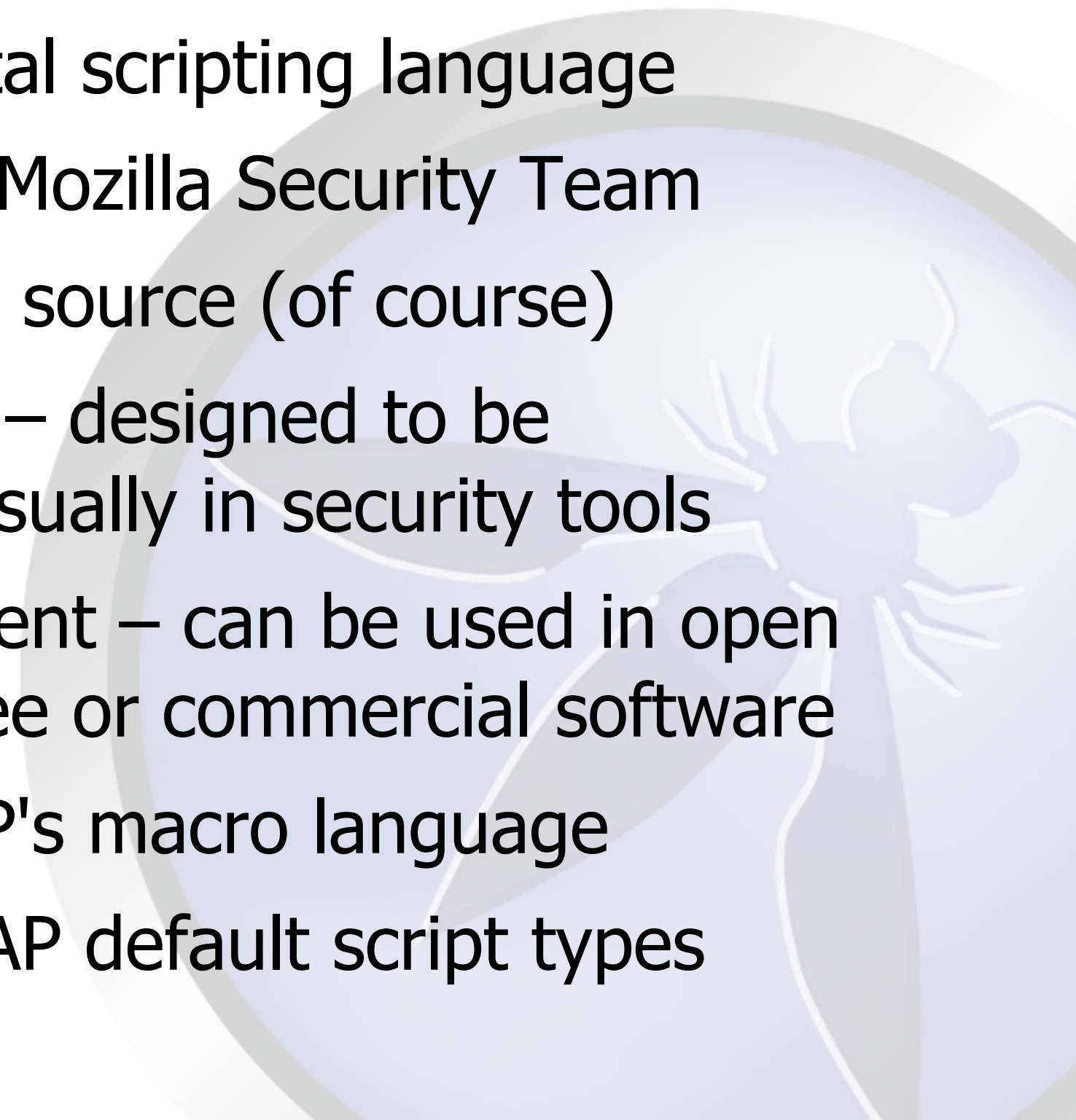


Scripting

- Different types of scripts
 - Stand alone Run when you say
 - Targeted Specify URLs to run against
 - Active Run in Active scanner
 - Passive Run in Passive scanner
 - Proxy Run 'inline'
 - Authentication Complex logins
 - Input Vector Define what to attack



Zest - Overview

- An experimental scripting language
 - Developed by Mozilla Security Team
 - Free and open source (of course)
 - Format: JSON – designed to be represented visually in security tools
 - Tool independent – can be used in open and closed, free or commercial software
 - Essentially ZAP's macro language
 - Supports all ZAP default script types
- 

Zest Scripts

The screenshot displays the Zest GUI interface. The top navigation bar includes 'Sites' and 'Scripts' tabs, along with control buttons for 'Quick Start', 'Request', 'Response', 'Break', and 'Script Console'. The left pane shows a tree view of the script configuration:

- Scripting
 - Scripts
 - Passive Rules
 - 301-302 body
 - IF :OR
 - OR
 - Status Code (301)
 - Status Code (302)
 - THEN
 - IF :Length
 - Length (response.body = 0 +/- 0%)
 - THEN
 - ELSE
 - Action - Fail (Redirect contains a body)
 - ELSE

The right pane shows the JSON configuration for the selected rule:

```
1 {  
2   "about":  
3     "This is a Zest script. For more details about Zest visit https://de.org/en-US/docs/Zest",  
4     "zestVersion": "0.3",  
5     "title": "301-302 body",  
6     "description": "Redirect (301-302) contains a body",  
7     "prefix": "",  
8     "type": "Passive",  
9     "parameters": {  
10      "tokenStart": "{{",  
11      "tokenEnd": "}}",  
12      "tokens": /
```

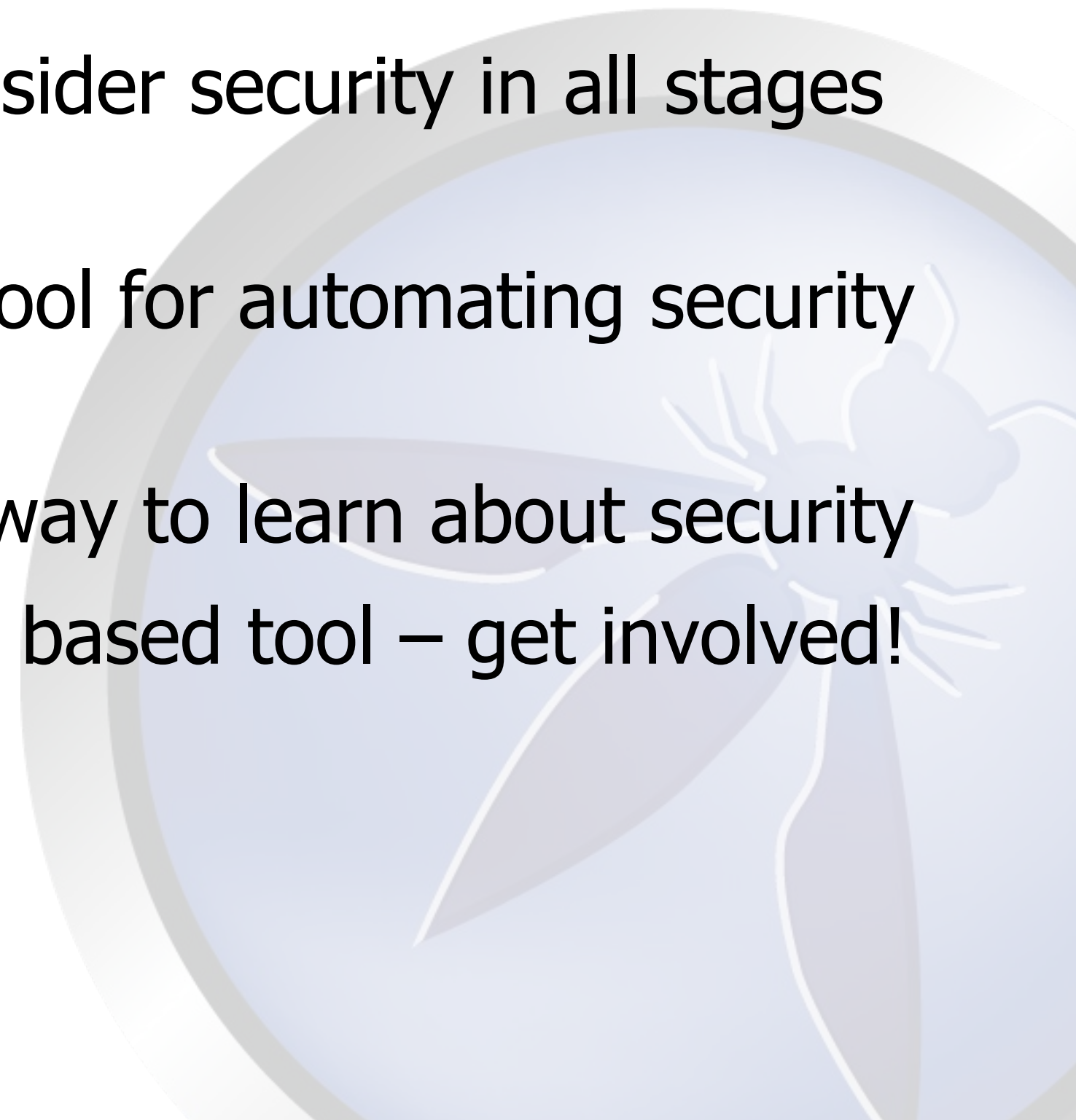
Below the JSON editor, a message states: "This is a graphical script that can only be edited via the Scripts tab on the left hand side."

The Source Code

- Currently on Google Code
- Will probably move to GitHub when time allows
- Hacking ZAP blog series:
<https://code.google.com/p/zaproxy/wiki/Development>
- ZAP Internals:
<https://code.google.com/p/zaproxy/wiki/InternalDetails>
- ZAP Dev Group:
<http://groups.google.com/group/zaproxy-develop>



Conclusion

- You need to consider security in all stages of development
 - ZAP is an ideal tool for automating security tests
 - Its also a great way to learn about security
 - Its a community based tool – get involved!
- 



Questions?

<http://www.owasp.org/index.php/ZAP>