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

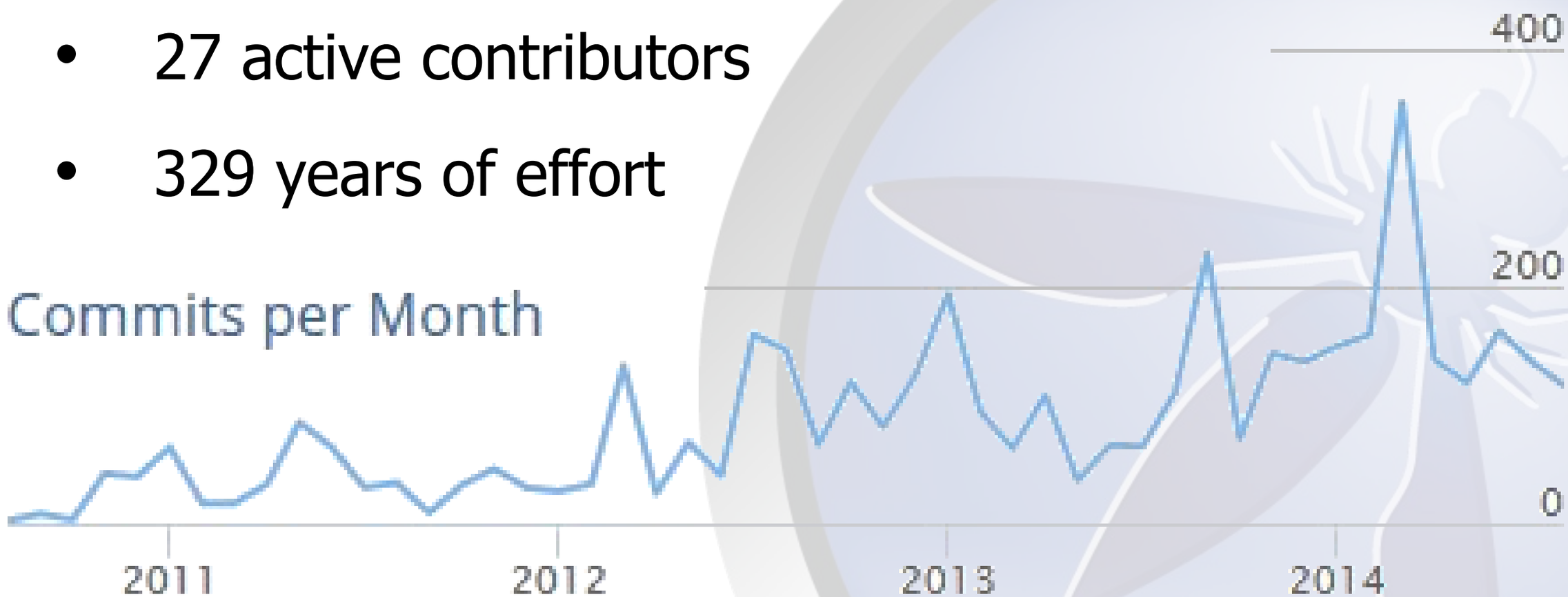
- Free, Open source
- Involvement actively encouraged
- Cross platform
- Easy to use
- Easy to install
- Internationalized
- Fully documented
- Work well with other tools
- Reuse well regarded components

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?

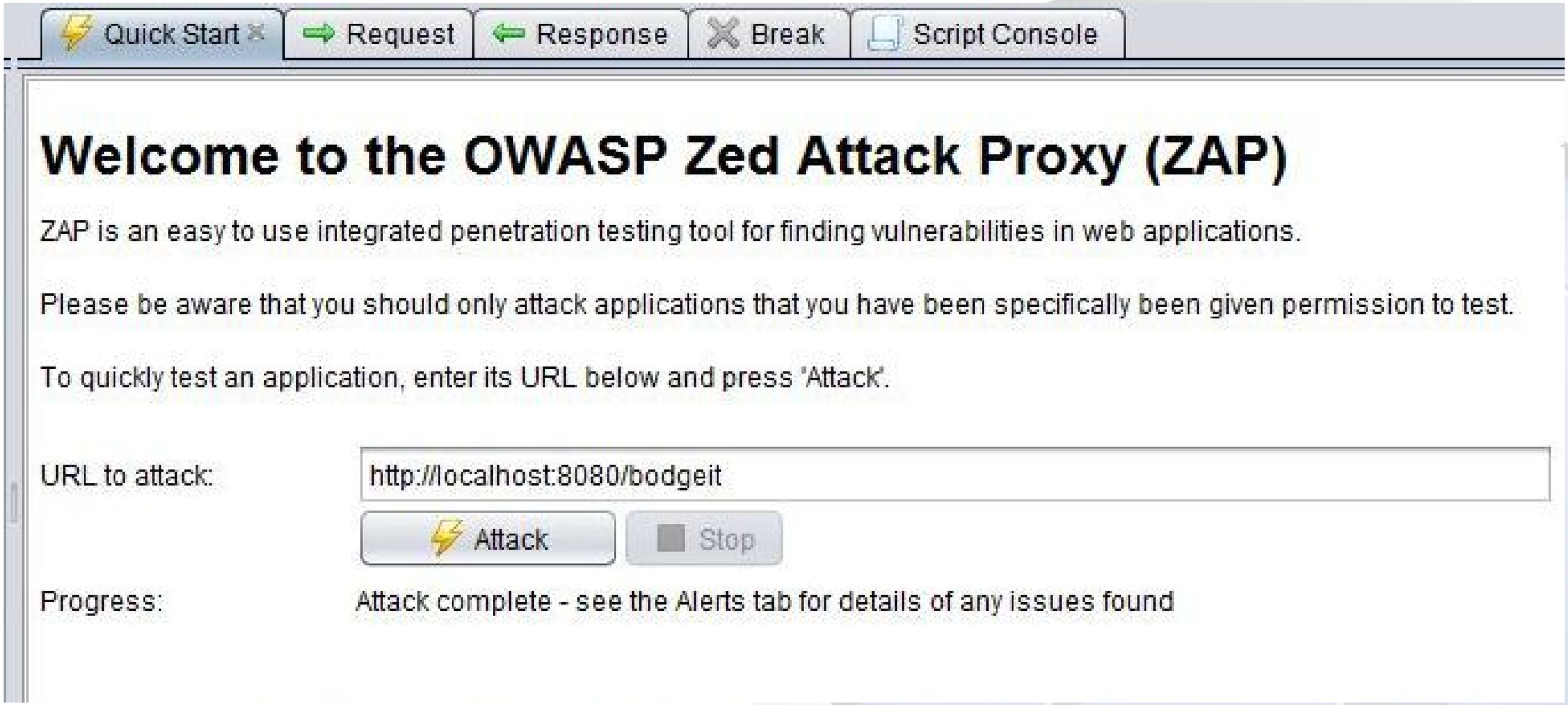
What to configure?

- 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



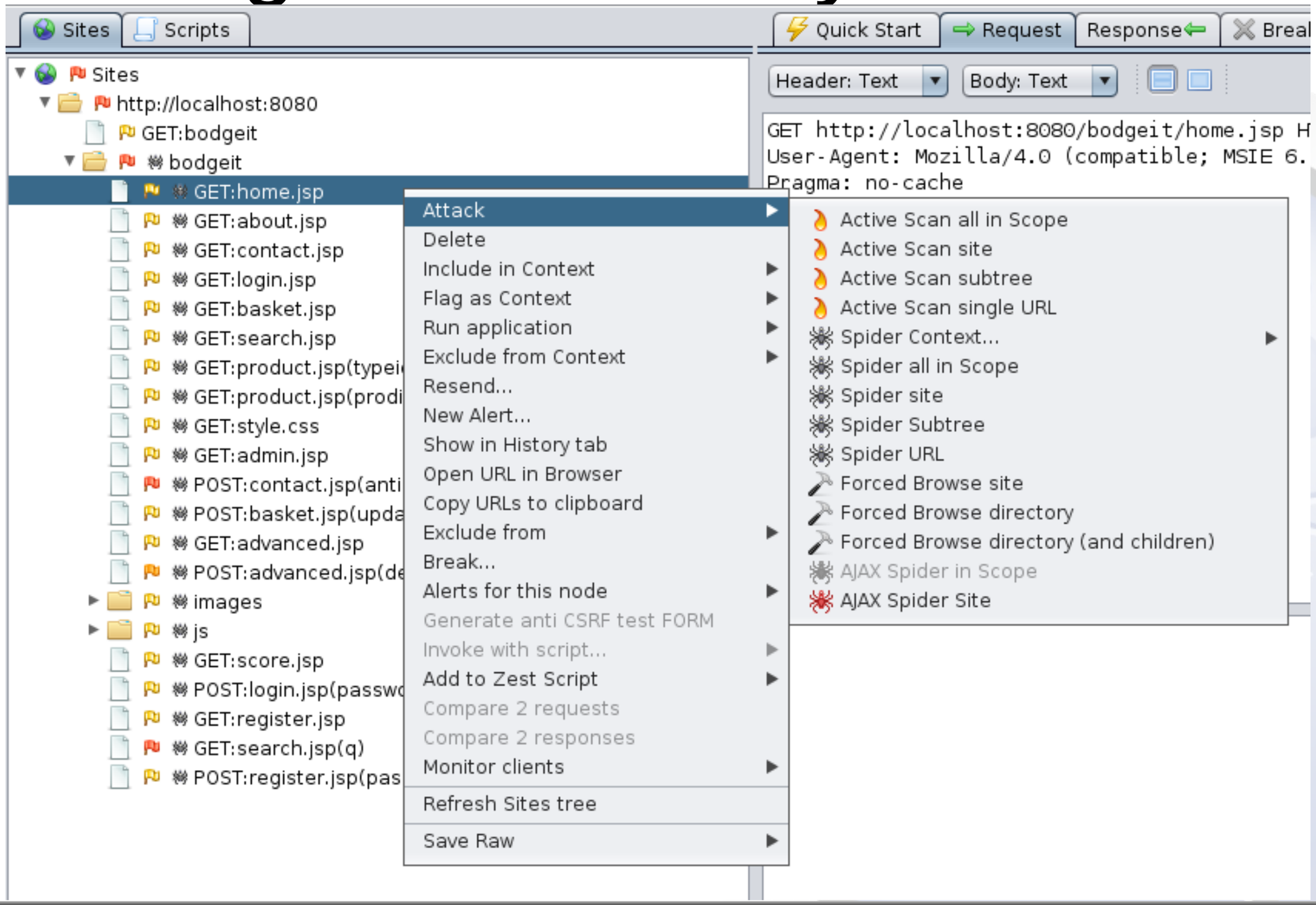
Proxying via ZAP

Options:

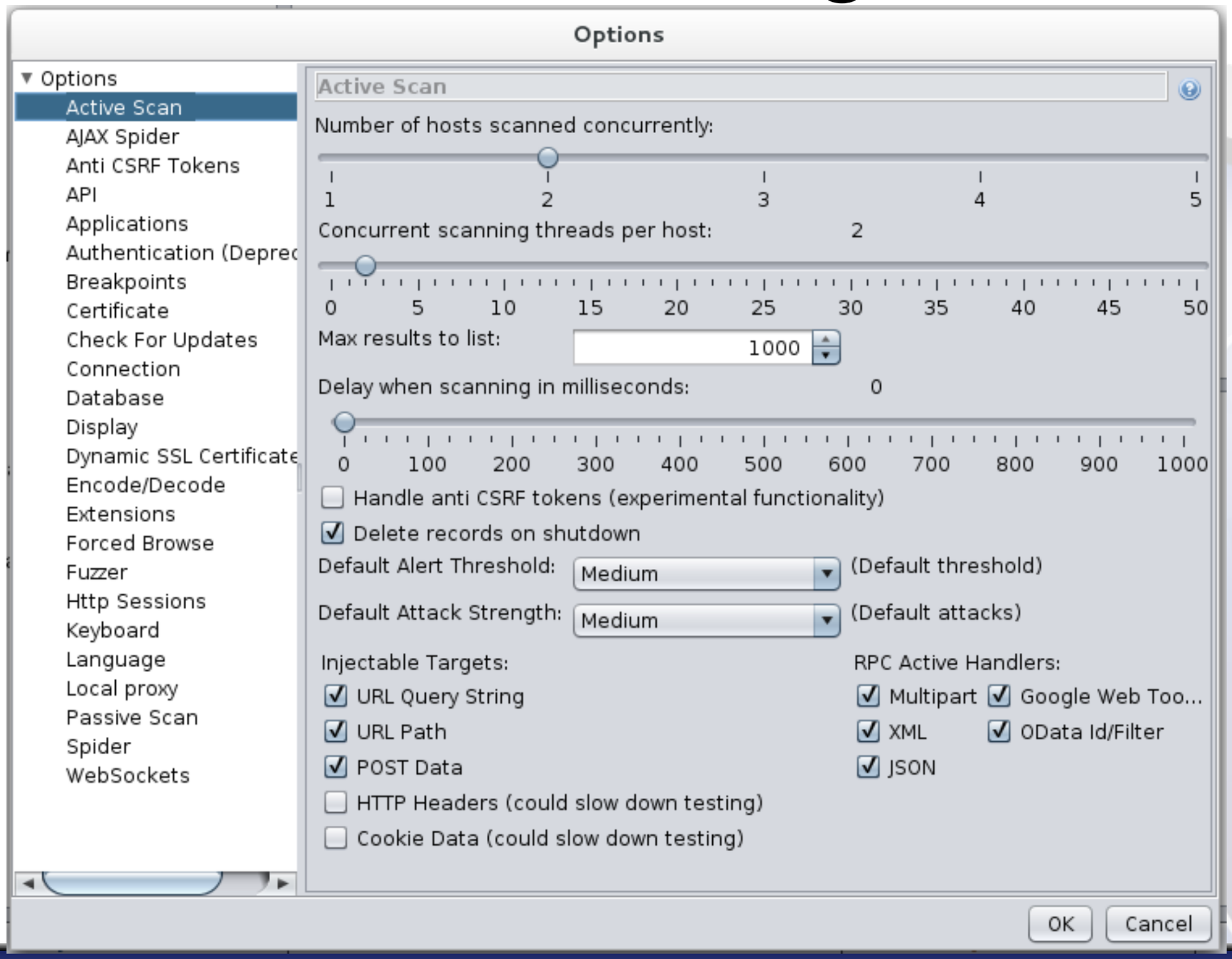
- Plug-n-Hack

- Configure your browser's proxy manually

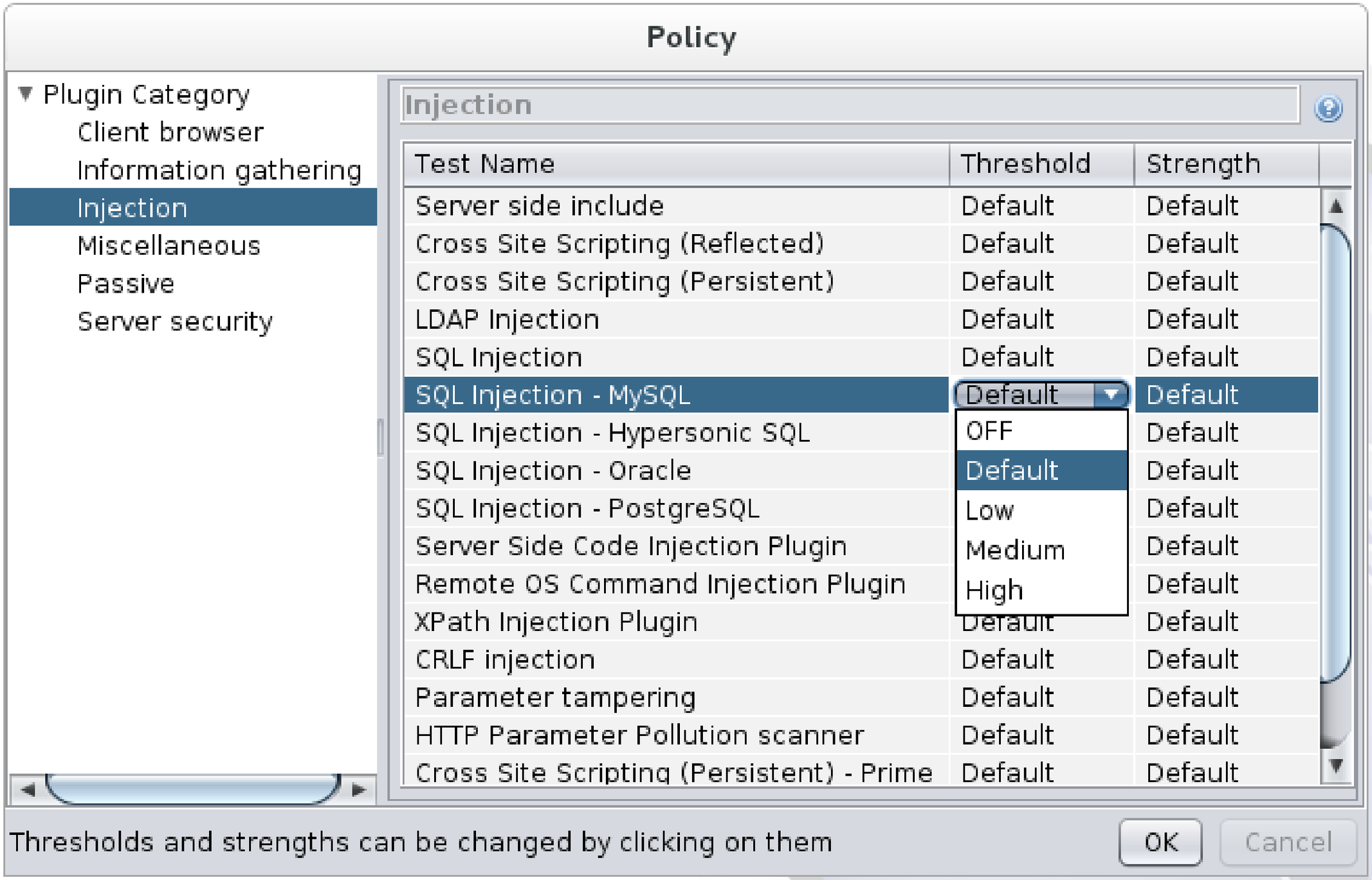
Right click everywhere!



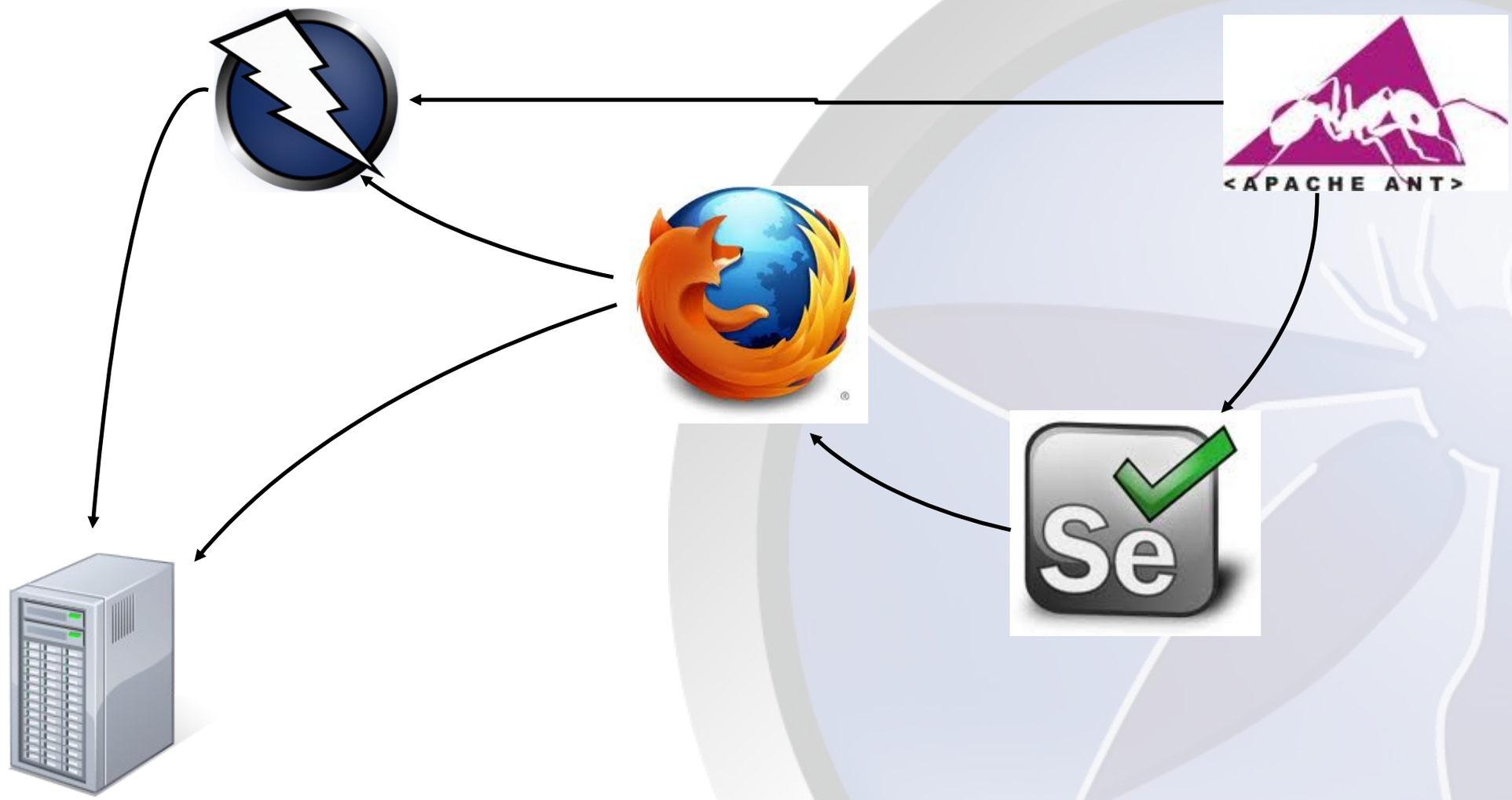
Fine tuning


Options
Options
Active Scan
AJAX Spider
Anti CSRF Tokens
API
Applications
Authentication (Deprec
Breakpoints
Certificate
Check For Updates
Connection
Database
Display
Dynamic SSL Certificate
Encode/Decode
Extensions
Forced Browse
Fuzzer
Http Sessions
Keyboard
Language
Local proxy
Passive Scan
Spider
WebSockets
Active Scan
Number of hosts scanned concurrently:
1 2 3 4 5
Concurrent scanning threads per host: 2
0 5 10 15 20 25 30 35 40 45 50
Max results to list: 1000
Delay when scanning in milliseconds: 0
0 100 200 300 400 500 600 700 800 900 1000
Handle anti CSRF tokens (experimental functionality)
Delete records on shutdown
Default Alert Threshold: Medium (Default threshold)
Default Attack Strength: Medium (Default attacks)
Injectable Targets:
URL Query String
URL Path
POST Data
HTTP Headers (could slow down testing)
Cookie Data (could slow down testing)
RPC Active Handlers:
Multipart
Google Web Too...
XML
OData Id/Filter
JSON
OK
Cancel

More fine tuning



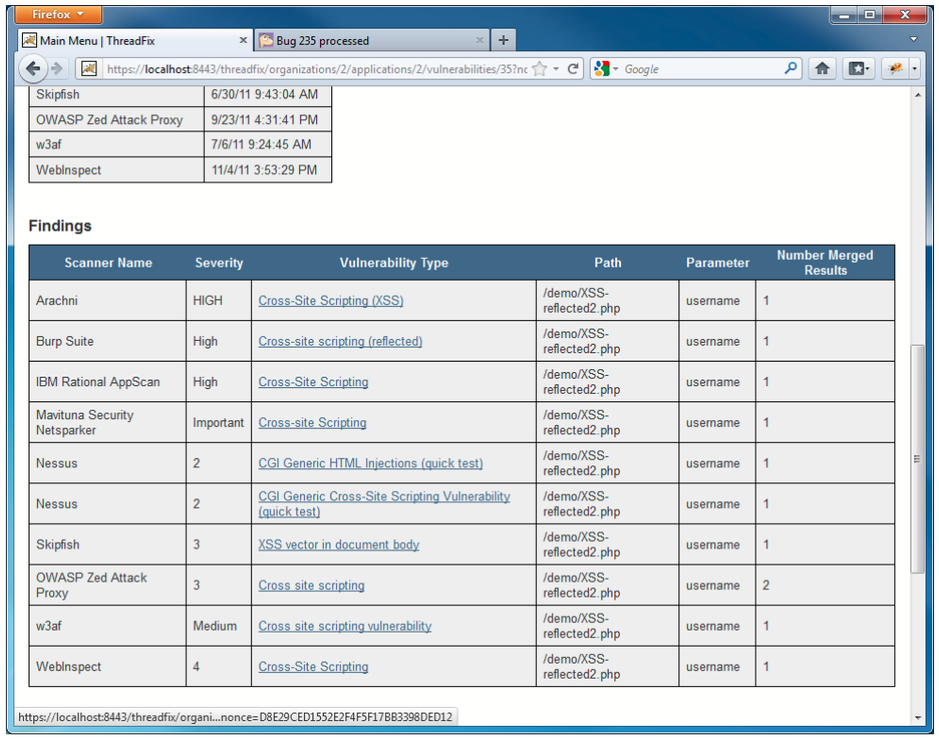
Security Regression Tests



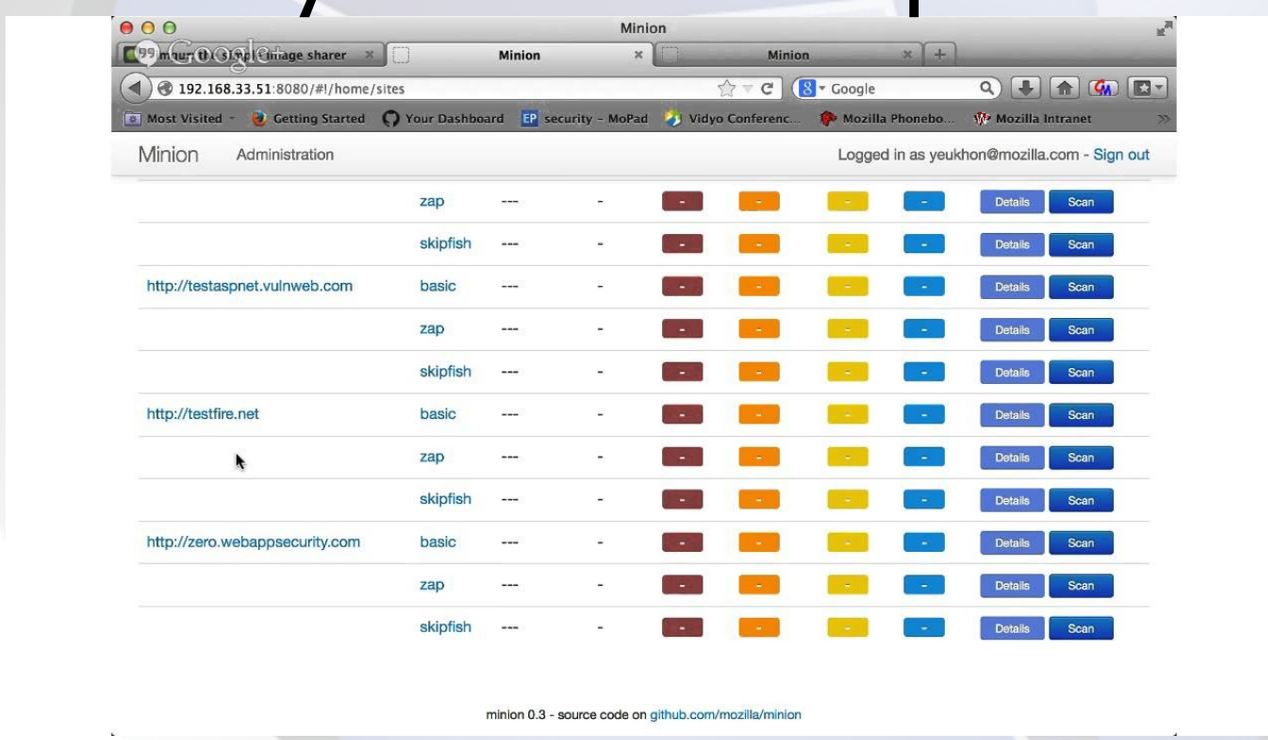
http://code.google.com/p/zaproxy/wiki/SecRegTests

ZAP – Embedded

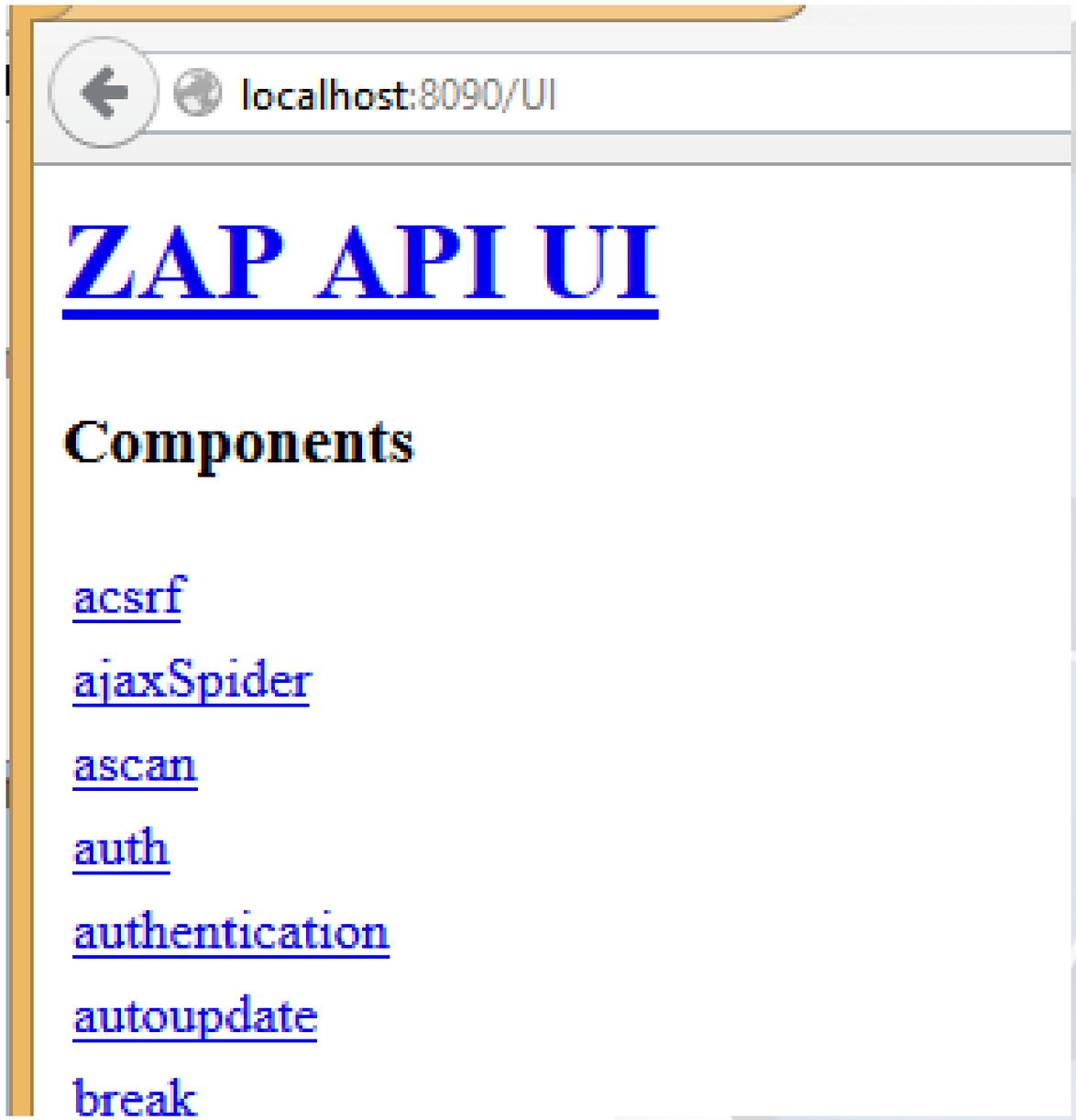
- ThreadFix – Denim Group
  Software vulnerability aggregation and management system



- Minion – Mozilla
  Security automation platform



The ZAP API


localhost:8090/UI
ZAP API UI
Components
acsrf
ajaxSpider
ascan
auth
authentication
autoupdate
break

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

Sites | Scripts | Quick Start | Request | Response | Break | Script Console

Run | Zest : 301-302 body

- 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
    - Active Rules
    - Proxy
    - Stand Alone
    - Targeted
  - Templates
    - Passive Rules
      - Passive default template.js

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

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