

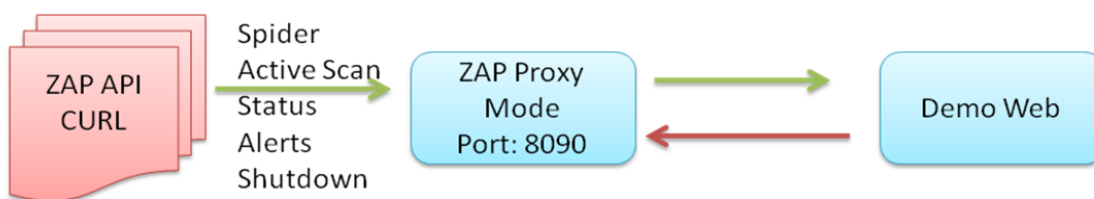
# Web Application Security Testing

In this lab, we will cover the following topics:

- Case 1 --- web security testing using the ZAP REST API
- Case 2 --- full automation with CURL and the ZAP daemon

## Case 1 -- web security testing using the ZAP REST API

In this case, ZAP will be running in proxy mode with port [8090]. Once ZAP is running, the ZAP web console can be reached at <http://localhost:8090/UI>. The demo website is the target website to be inspected by ZAP. We will use CURL to trigger the ZAP RESTful API to operate ZAP to do spider scans, active scans, review alerts, and shut down ZAP:



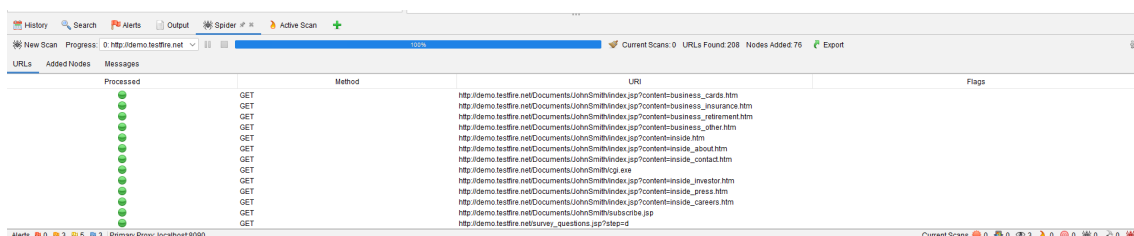
## Step 1 -- spider scanning the website

The purpose of the spider scan is to discover all resources and URLs for the target website. The spider visits these URLs and will try to identify all the hyperlinks in the responses. The scanning process continues recursively whenever new URLs are identified. All identified URLs can be used for further security inspection and active scans in the next step.

Sending the REST API request to ZAP will require the API key. To simplify the implementation, we will disable the API key in our demonstration. The API key can be disabled under the ZAP console menu, via [Tools] | [Options] | [API] | [Disable the API Key] checkbox.

Here is the command to execute the spider scan by [CURL]:

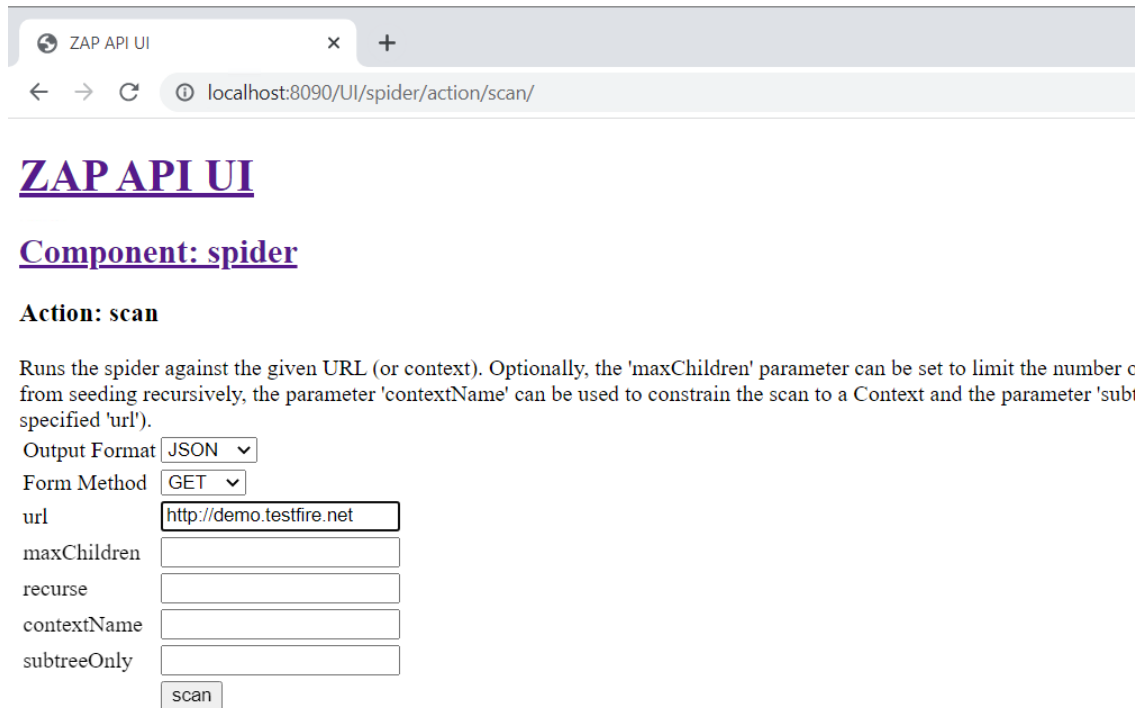
```
curl "http://localhost:8090/JSON/spider/action/scan/?zapapiformat=JSON&formMethod=GET&url=http://demo.testfire.net&maxChildren=&recurse=&cont
```



To get further information for the HTTP [GET] request for the spider scan, we can use the browser to visit the ZAP URL: <http://localhost:8090/UI>. This provides an explanatory API document and operations. For example, we can trigger the spider scan by clicking on [spider] and [scan] ([url maxChildren recurse contextName subtreeOnly]). This will navigate us to <http://localhost:8090/UI/spider/view/scans/> (as shown in the following diagram), where we can

define some parameters and trigger the scan. After the spider scan is triggered, the URL we get is the final URL (the HTTP [GET] request) we need for the CURL automation.

The following diagram shows the spider scan UI operations in ZAP:



The screenshot shows a web browser window with the title 'ZAP API UI'. The address bar displays 'localhost:8090/UI/spider/action/scan/'. The main content area features the 'ZAP API UI' logo in purple, followed by the heading 'Component: spider'. Below this is the section '**Action: scan**'. A descriptive paragraph explains that the spider runs against a given URL or context, with optional parameters like 'maxChildren', 'contextName', and 'subtreeOnly'. The form includes two dropdown menus for 'Output Format' (set to 'JSON') and 'Form Method' (set to 'GET'). There are five text input fields: 'url' (containing 'http://demo.testfire.net'), 'maxChildren', 'recurse', 'contextName', and 'subtreeOnly'. A 'scan' button is located at the bottom of the form.

ZAP API UI

Component: spider

**Action: scan**

Runs the spider against the given URL (or context). Optionally, the 'maxChildren' parameter can be set to limit the number of from seeding recursively, the parameter 'contextName' can be used to constrain the scan to a Context and the parameter 'subtreeOnly' can be used to constrain the scan to a subtree.

Output Format

Form Method

url

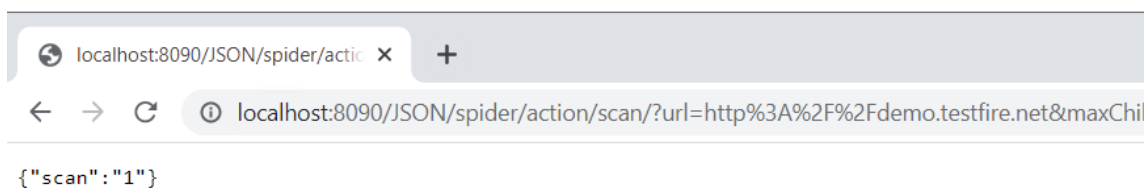
maxChildren

recurse

contextName

subtreeOnly

Click **Scan** to run new scan. You will get scan number on success (you might get different scan number):



The spider scan may take a long time since it will extensively and recursively search for any potential web URLs and resources. Besides, the time it takes also depends on the number of web pages, parameters, and the number of threads.

## Step 2 -- active scanning the website

Once we have done the spider scan, the active scan will find the security vulnerabilities by sending malicious requests, such as XSS or SQL injection, based on the scanning policies.

Here is the command to trigger the active scan with [CURL]:

```
curl "http://localhost:8090/JSON/ascan/action/scan/?
zapapiformat=JSON&formMethod=GET&url=http://demo.testfire.net&recurse=&inScopeOnly=&scar
```

The URL of the active scan is <http://localhost:8090/UI/ascan/action/scan/>.

## Step 3 -- reviewing the status of the active scan

To review the status of the active scan, try one of the following commands. It will output the percentage of completeness as a status value. Depending on the output format you need, you may change JSON to HTML:

```
curl "http://localhost:8090/JSON/ascan/view/status/"
```

The following command will generate the active scan status in JSON format:

```
curl "http://localhost:8090/JSON/ascan/view/status/?
zapapiformat=JSON&formMethod=GET&scanId="
```

## Step 4 -- reviewing the security assessments

To review the security assessments made by OWASP ZAP, we may use one of the REST APIs, as follows:

```
CURL http://localhost:8090/HTML/core/view/alerts/
```

Alternatively, the HTML report can be generated by exporting to [ZAP\_Report.HTML] via the REST API, as follows:

```
curl "http://127.0.0.1:8090/OTHER/core/other/htmlreport/?formMethod=GET" >
ZAP_Report.HTML
```

## Case 2 -- full automation with CURL and the ZAP daemon

In this case study, we will further extend the case to execute ZAP in daemon (headless) mode. We will automate the web security tests with OWASP ZAP in the following order for a complete testing cycle:

1. Launch ZAP in daemon mode
2. Spider scan the whole website
3. Active scan all the scanned URLs
4. Check status and wait for the active scan to finish
5. Shut down the ZAP daemon

## Step 1 -- executing ZAP in daemon (headless) mode

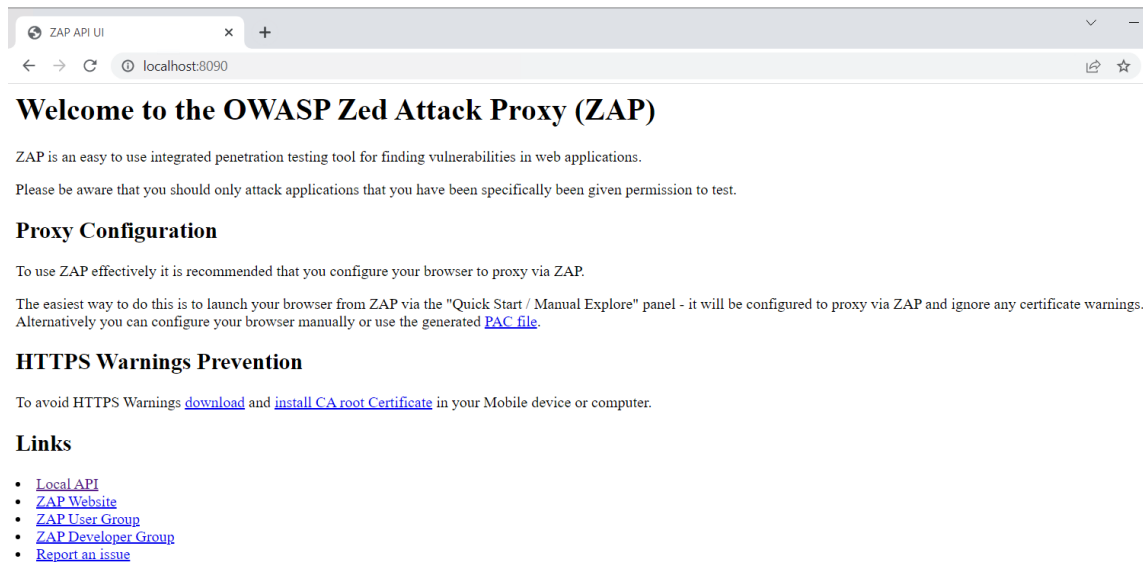
To launch ZAP in daemon mode, close `ZAP GUI` and then execute the following commands in the terminal:

```
cd C:\Program Files\OWASP\Zed Attack Proxy
ZAP.bat -daemon
```

```
5140 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing HTTP Panel Query Table View Extension
5140 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing HTTP Panel Syntax Highlighter View Extension
5140 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Adds support for configurable keyboard shortcuts for all of the ZAP menus.
5140 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Active and passive rule configuration
5140 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Statistics
5140 [ZAP-daemon] INFO org.zaproxy.zap.extension.stats.ExtensionStats - Start recording in memory stats
5140 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Custom Pages Definition
5140 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Context alert rules filter
5140 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Alert Filters Automation Framework Integration
5140 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing DOM XSS Active Scan Rule
5218 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing This extension allows a user to change the default values used by ZAP Spiders.
5234 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing The ZAP Getting Started Guide
5234 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Provides the GraalVM JavaScript engine for ZAP scripting.
5656 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Heads Up Display
5734 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing ExtensionHUDLaunch
5734 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing The Online menu links
5734 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Easy way to replace strings in requests and responses
5734 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing The Retest add-on allows to verify the presence/absence of certain alerts.
5734 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Ajax Spider Automation Framework Integration
5734 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Tips and Tricks
5734 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Active Scan Rules
5734 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Automation Framework
5750 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Handles all of the calls to ZAP services
5750 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing org.zaproxy.addon.commonlib.ExtensionCommonlib
5750 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Import and Export functionality supporting multiple formats.
5750 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Import/Export Automation Framework Integration
5750 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Provides the foundation for concrete message types (for example, HTTP, WebSockets) expose fuzzer implementations.
5750 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Allows to fuzz HTTP messages.
5750 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Allows you to inspect and attack GraphQL endpoints.
5750 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing GraphQL Automation Framework Integration
5765 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Provides core networking capabilities.
5828 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Adds OAST scripts.
5828 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Allows you to spider and import OpenAPI (Swagger) definitions
5843 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing OpenAPI Automation Framework Integration
5843 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Passive Scan Rules
5843 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Adds the Quick Start panel for scanning and exploring applications
5843 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Add the option to use the Ajax Spider in the Quick Start scan
5843 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Launch browsers proxying through ZAP
5843 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Launch browsers proxying through ZAP
5843 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Report Generation
5843 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Report Generation Automation Integration
5859 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing ExtensionSaveRawHttpRequestMessage
5859 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing ExtensionSaveXMLHttpRequestMessage
5859 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Scripts Automation
5875 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing SOAP Automation Framework Integration
5890 [ZAP-daemon] INFO org.parosproxy.paros.extension.ExtensionLoader - Initializing Allows to fuzz WebSocket messages.
6812 [ZAP-daemon] INFO org.zaproxy.addon.oast.services.callback.CallbackService - Started callback service on 0.0.0.0:1551
6837 [ZAP-daemon] INFO org.zaproxy.zap.DaemonBootstrap - ZAP is now listening on localhost:8090
```

## Step 2 -- checking the status of the ZAP daemon

In our testing environment, our ZAP proxy is configured using port `8090` . Open following URL in browser to check if it's running normally:



## Step 3 -- fully automating the ZAP API

The whole scanning process can be fully automated in one script file. Here, we use the Windows BAT script as an example. The fully automated ZAP security testing script for the website is named `AutoZAP.BAT` :

```
echo the status of ZAP

CURL http://localhost:8090

echo spider scan for the web site

CURL "http://localhost:8090/JSON/spider/action/scan/?
zapapiformat=JSON&formMethod=GET&url=http://demo.testfire.net"

echo Active Scan for the website

CURL "http://localhost:8090/JSON/ascan/action/scan/?
zapapiformat=JSON&formMethod=GET&url=http://demo.testfire.net&recurse=&inScopeOnly=&scan"
```

```

echo Wait for 20 sec to complete the ActiveScan before generating the testing report

echo The timeout is for Windows command. For running in Linux, please change it to
sleep.

timeout 20

echo List the security assessments results (alerts), and output the report to
ZAP_Report.HTML

CURL "http://localhost:8090/JSON/ascan/view/status/"

CURL "http://localhost:8090/HTML/core/view/alerts/"

CURL "http://127.0.0.1:8090/OTHER/core/other/htmlreport/?formMethod=GET" >
ZAP_Report.HTML

echo shutdown the ZAP

CURL "http://localhost:8090/JSON/core/action/shutdown/?
zapapiformat=JSON&formMethod=GET"

```

Run the following commands in the cmd terminal to run automated scan. Make sure zap daemon is running first.

```

cd C:\Users\fenago\Desktop\DevSecOps-course\lab06

AutoZAP.BAT

```

## Summary

In this lab, we performed web security testing using ZAP. ZAP was used for web security scanning, which was automated by a REST API or CLI.

The purpose of case 1 was to demonstrate how to automate the ZAP spider scan by using a REST API and CURL.

The objective of case 2 was to run ZAP in daemon mode and to execute a full security scan cycle in one script. The automation steps of ZAP scanning include the following:

1. Launch ZAP in daemon mode
2. Spider scan the whole website
3. Active scan all the scanned URLs
4. Check status and wait for the active scan to finish
5. Shut down the ZAP daemon

These cases demonstrated different automation approaches to web security scanning. In the next lab, we will discuss different automation approaches to Android security testing.