Testing For SQL Injection Vulnerabilities using OWASP ZAP

Asaf Sahar · Follow
6 min read · Jun 16, 2023

In this blog post, I will explain SQL Injection (SQLi) and how to test for it using the OWASP ZAP tool. OWASP ZAP is one of the leading tools for testing web security vulnerabilities. It's free and open-source software.

SQL Injection at a glance

SQL Injection (SQLi) is the ability to interfere with queries that an application makes to its database. There are many types of SQLi, but I will explain the concept with a simple example.

Example:

On many websites, there is a login screen. When clicking the submit button, the username and password fields are sent to the server.

In our example, the server sends the following SQL query to its database:

```
SELECT * FROM USERS WHERE USERNAME = 'username' AND PASSWORD='password'
```

Do you think it would be interesting to log in as another user without a password? I believe you do :)

To change the query and perform a login as another user without a password, we can send a username as other_username' — and the SQL query will look like:

```
SELECT * FROM USERS WHERE USERNAME = 'other_username'-- AND PASSWORD='passwor
```

PASSWORD

' --' is a comment in SQL, so the AND PASSWORD='password' part will be commented out. If the website doesn't check the input that arrives from the client, the attack will succeed.

. . .

If you were looking for information about SQLi, I believe that you are already familiar with SQL commands. Being familiar with SQL queries can help you understand how this attack works.

Remarks:

- In the real world, it is unlikely to be implemented this way, but this example helps understand the concept of performing an SQLi attack.
- In some cases, we might need to find a way to interfere between the client (browser) and the server. ZAP has this capability, but if you are also familiar

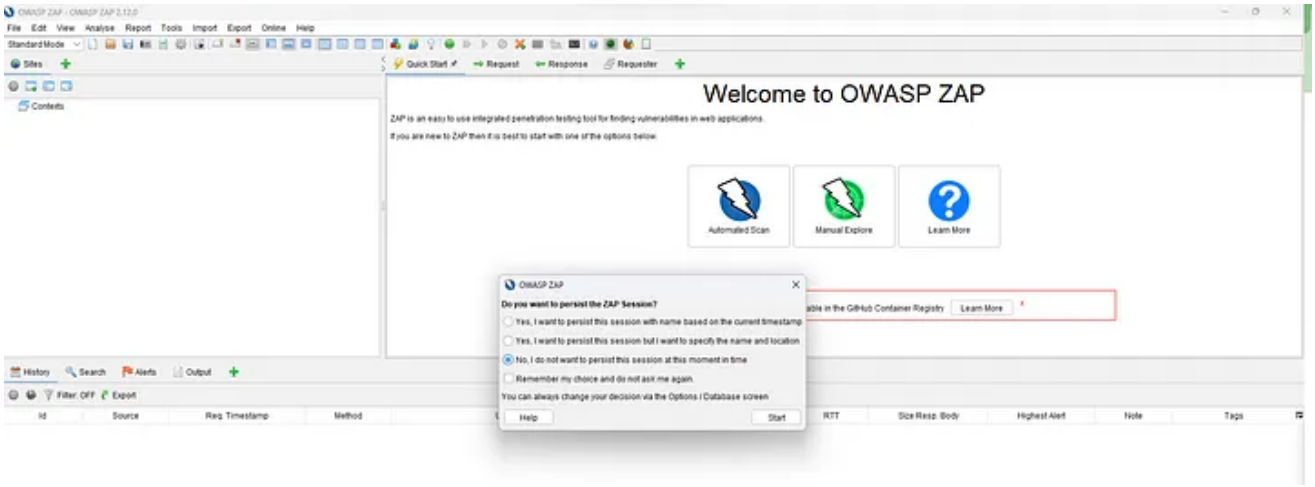
with the Burp Suite tool, you can find a detailed guideline for that in the blog post I wrote about Information Disclosure vulnerability at this link.

If you want to test your website for SQL Injection Vulnerabilities, keep reading...

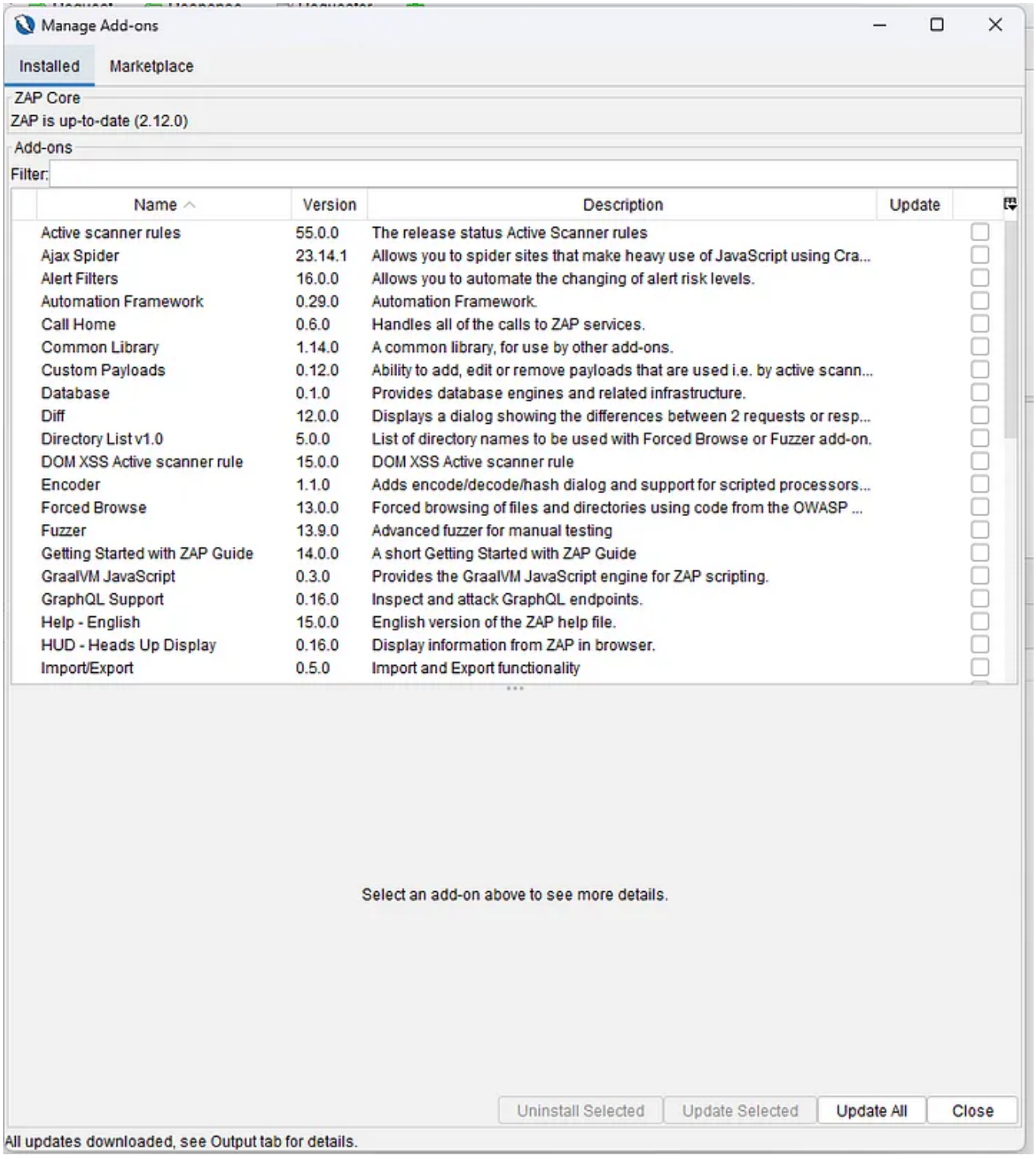
Install OWASP ZAP

First, download OWASP ZAP (from now on, I will refer to it as ZAP) from https://www.zaproxy.org/download/ according to your operating system.

After the installation is completed, launch ZAP. It will ask you if you want to save the current session or not; I usually don't save it.



Also, it will open the add-on installer; you can close it for now.

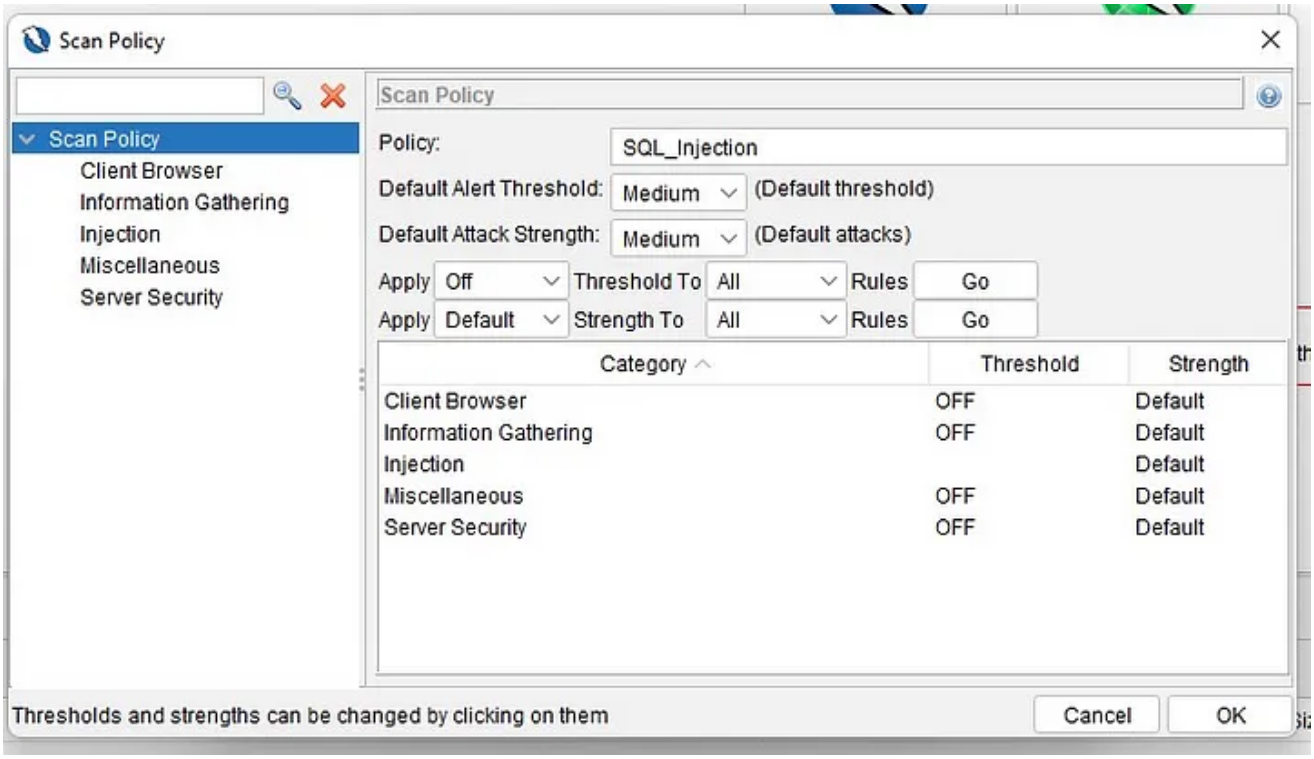


Define SQL Injection policy

After loading ZAP, the next step is to define a policy that focuses on SQLi. In a policy we can define which vulnerabilities we want to scan, their strength and what is our threshold to get alerts.

To define a policy in ZAP:

1. Go to Analyze -> Scan Policy Manager.
2. Click on Add and enter the name in the Policy field (I use SQL_Injection).
3. We want to disable everything and only keep SQLi available. To do that, we will set the Threshold to 'Off' and click on the Go button that appears to the right of the Rules label.
4. Go to the Injection menu on the left, select 'SQL Injection' as the Test Name, and change it to Default. Return to the Scan Policy; your policy should look like this:



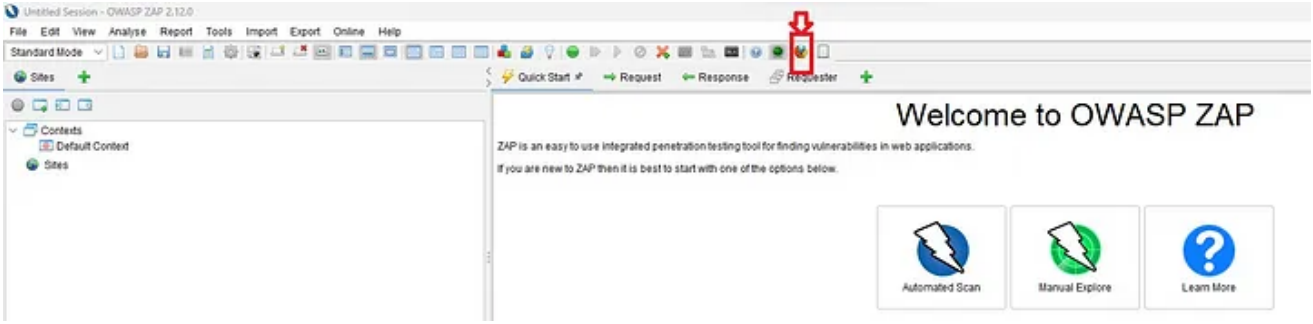
Click on OK. Now you have a SQL Injection policy defined. If you want to learn more about the Threshold and Strength, you can go here: https://www.zaproxy.org/docs/desktop/ui/dialogs/scanpolicy/ If you want to see the different SQL Injection alerts that ZAP suggests, you can go here: https://www.zaproxy.org/docs/alerts/ and search for alert = sql.

Run active scan

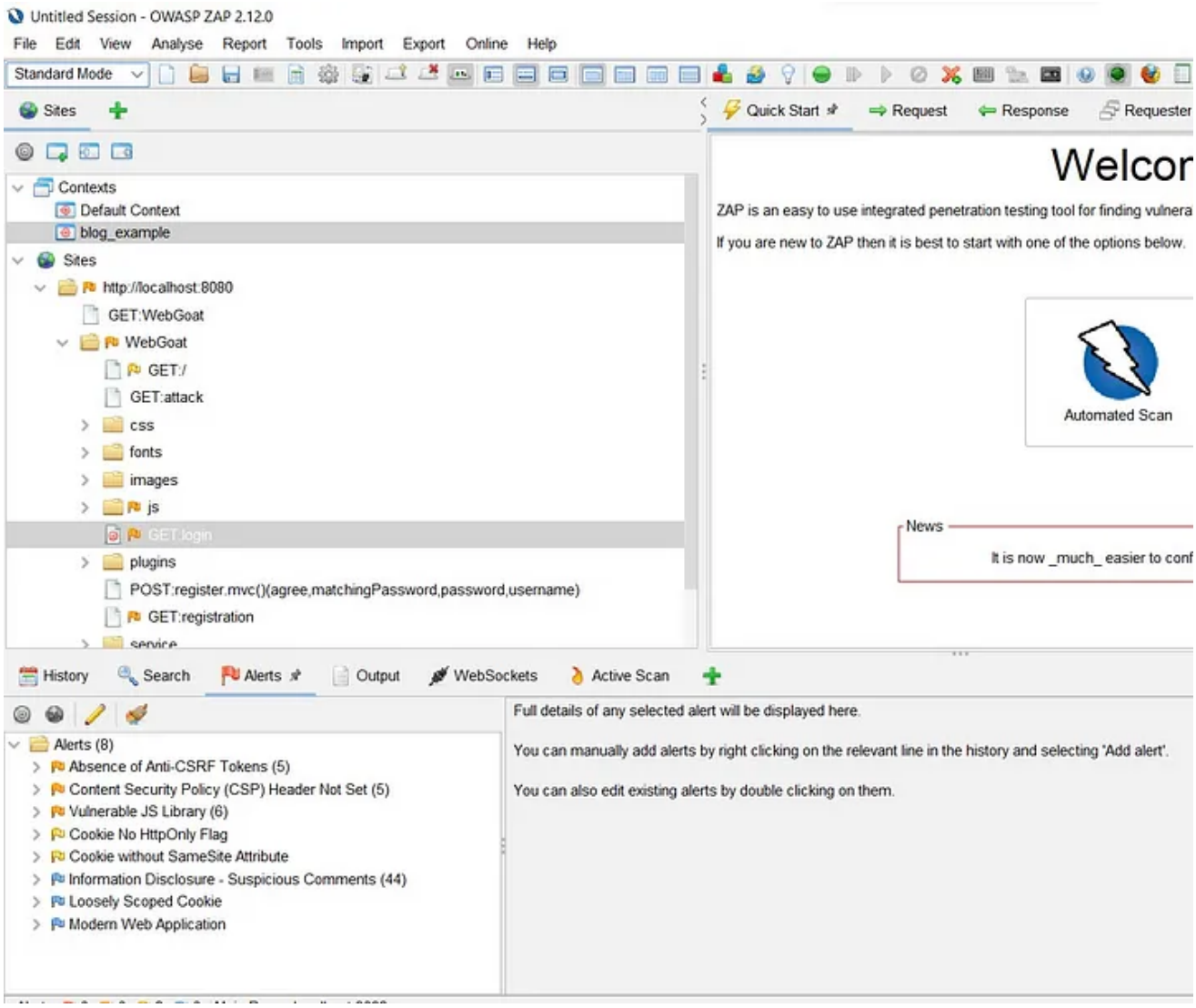
There are two ways to run an active scan using ZAP: through the API or the UI. In this post, I will explain the UI method, which is easier.

Very Important: *DO NOT PERFORM A SCAN ON ANY WEBSITE THAT YOU DON'T HAVE PERMISSION TO RUN SECURITY TESTS ON.*

On one of the options on the top bar, you will find a small icon of the Firefox browser (it must be installed on your computer). Click on it, and a built-in browser of ZAP will open.

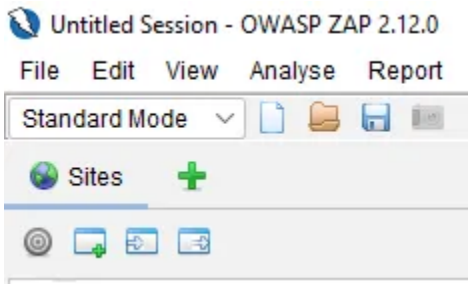


Navigate to the website you want to test. ZAP will start scanning the website and will build a tree of sites. Then navigate to the page or URL that you want to test using the ZAP browser.



We should define a new Context. In the context, we can define many things related to the website or URL that we want to test. We can define which users (username and password) are required for a login to our website, which URLs we want to check, which we don't, and much more.

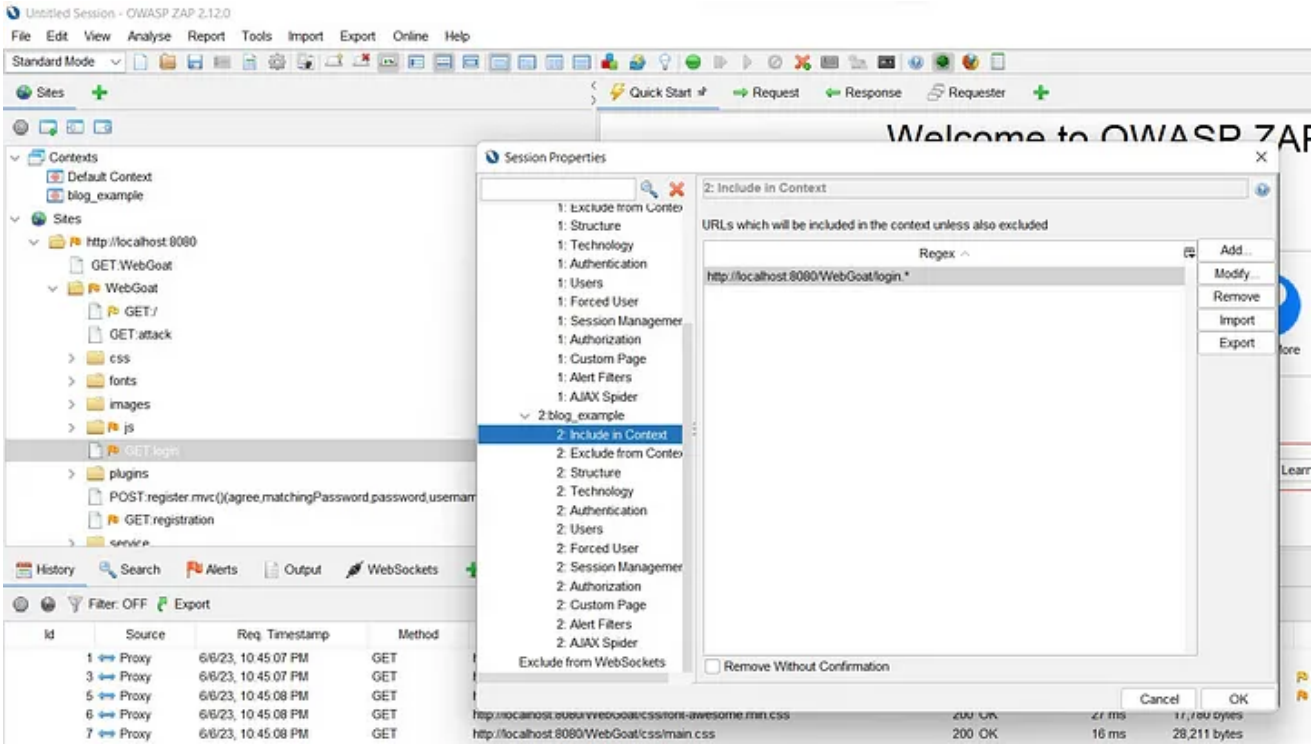
We create a Context by clicking the small window with a green plus icon:



Give a name to the context. I recommend focusing on a specific endpoint that you want to test and click on Save. To do that:

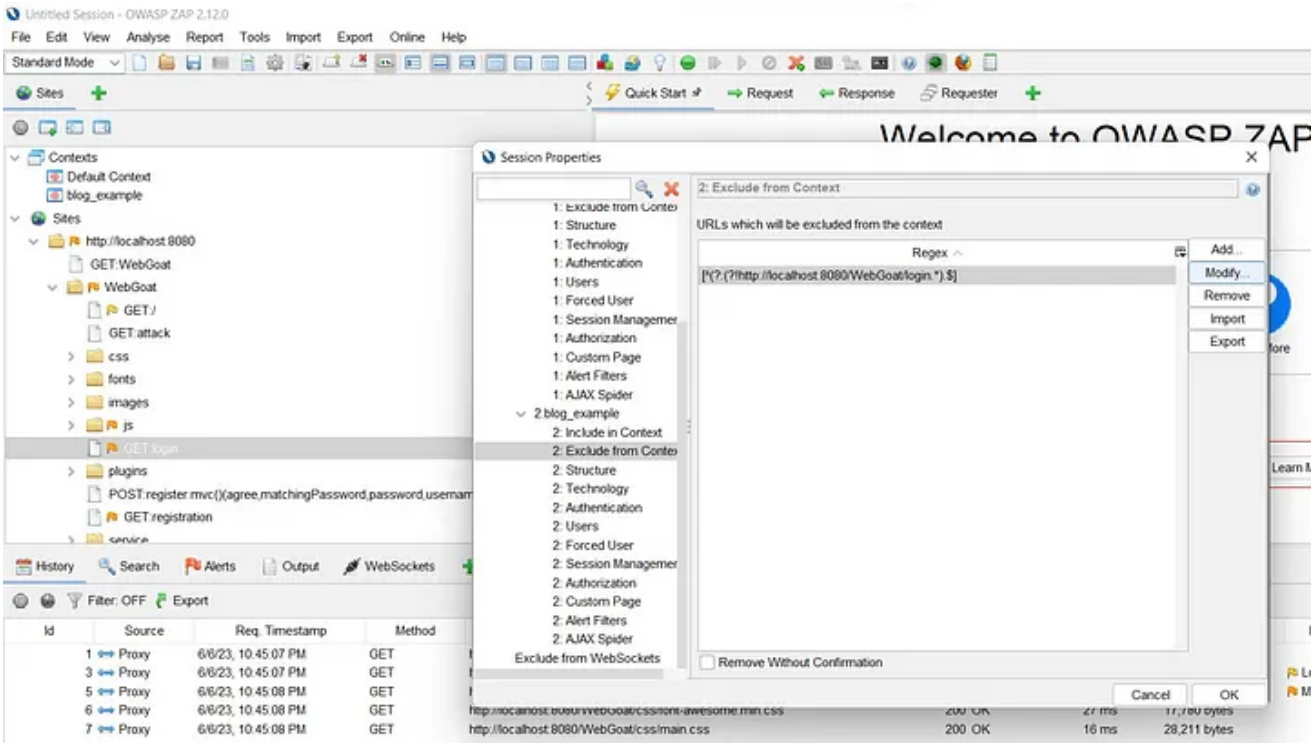
1. Locate the URL in the ZAP History tab.
2. Right-click on it
3. Select ‘Include Site in Context’ -> Select the context that you created.

The Session Property window will open, and the focus will be on the ‘Include in Context’ section and will look like this:



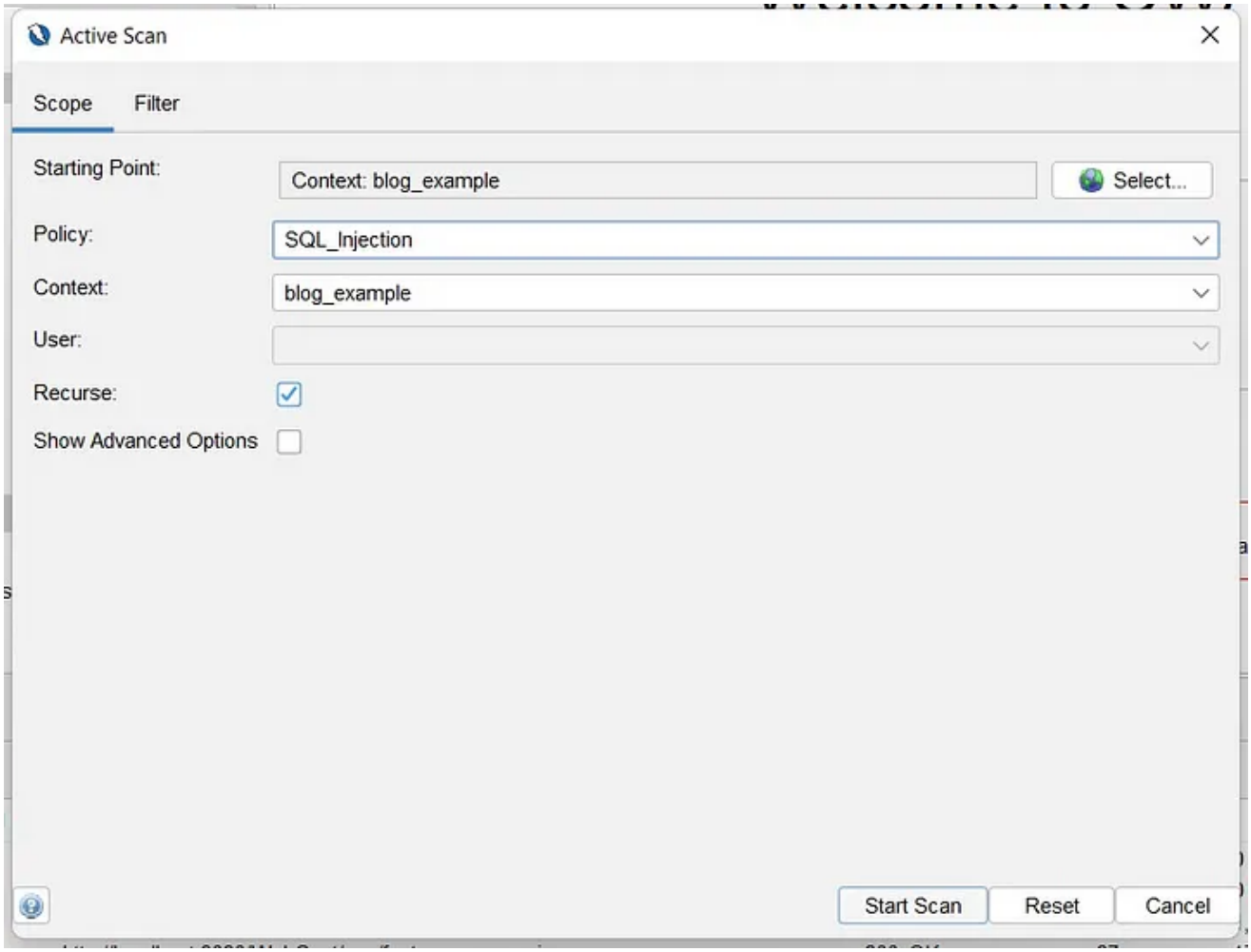
Here’s a tip that took me some time to find: you can focus on scanning only this URL. Move to the ‘Exclude from Context’ section under your context and set the following: [^(?:(?!Replace_With_Your_URL.*).$]

Example:



This will ensure that only the URL you want will be scanned. This example allows you to scan a page that doesn't require authentication.

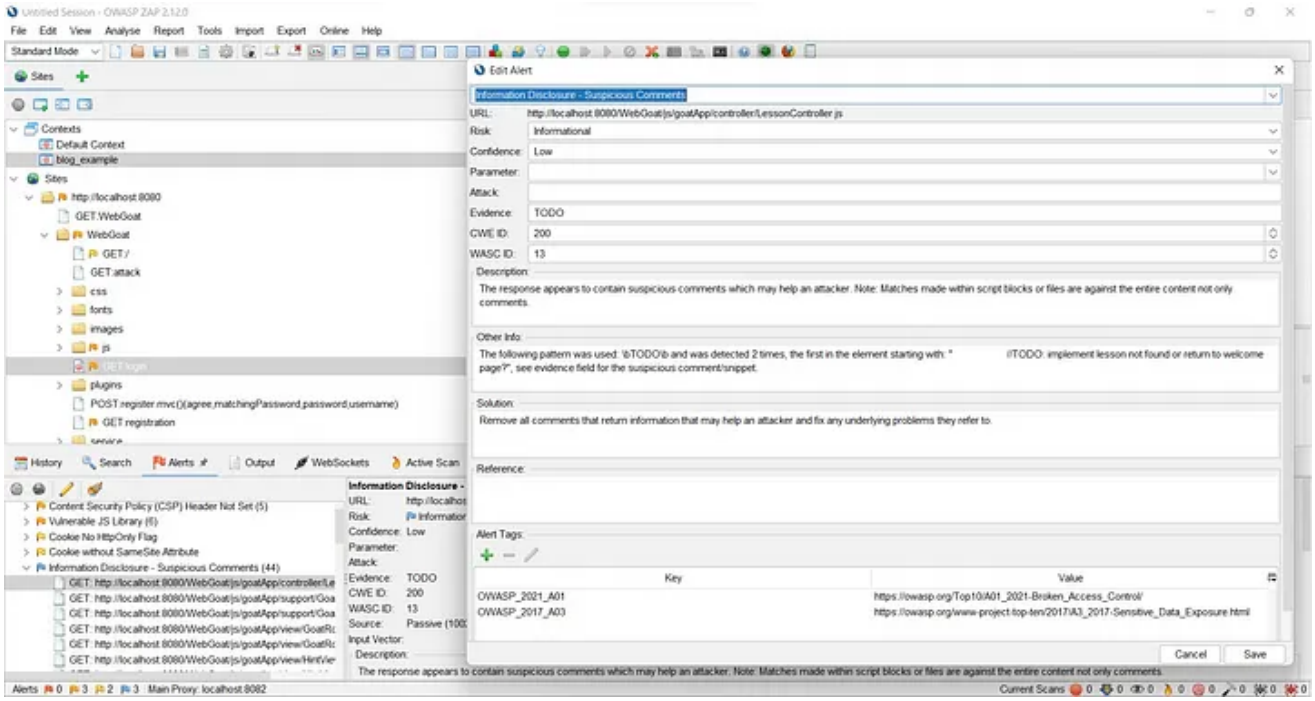
Now, right-click on the context and select Active Scan. You will see this screen:



ZAP will start an active scan on your URL after clicking on Start Scan.

Scan results

Now we have reached the most important part: reviewing the scan results and determining if our website is vulnerable to SQLi or not. Check the bottom left part to see if there are any alerts.



Here, we see an example of a scan that was performed on the WebGoat website running locally using a docker container. WebGoat is a project of a vulnerable website created by OWASP Foundation. Clicking on one of the alerts will open a page with a wealth of information about the potential vulnerability that was detected. You can find information such as the vulnerability risk level, description, solution, and more.

I hope you enjoyed reading the blog and found it informative.

If you want to do more thorough testing on your website, you are welcome to email me at info@sahartechsolutions.com or visit my website www.sahartechsolutions.com , and I will be able to provide you with consulting on how to do that or test your website for you.

Thank you for your time!

Written by Asaf Sahar

72 Followers

Senior QA Security Engineer | www.linkedin.com/in/asaf-sahar-94531121

More from Asaf Sahar

Asaf Sahar

Testing SSRF using OWASP ZAP OAST add-on

Intro

Jan 8 2

Asaf Sahar in AppsFlyer Engineering

Scaling Security Testing by OWASP ZAP API

My journey of learning about additional security vulnerabilities continues with Reflected Cross-Site Scripting (XSS). To see how this...

Dec 12, 2021 5 1

Asaf Sahar in AppsFlyer Engineering

My Journey to a QA Security Mindset SSRF vulnerability

In my previous blog post, I discussed the first vulnerability that I learned about in my journey to a QA security mindset—IDOR. If you...

Jul 12, 2021 29

Asaf Sahar in AppsFlyer Engineering

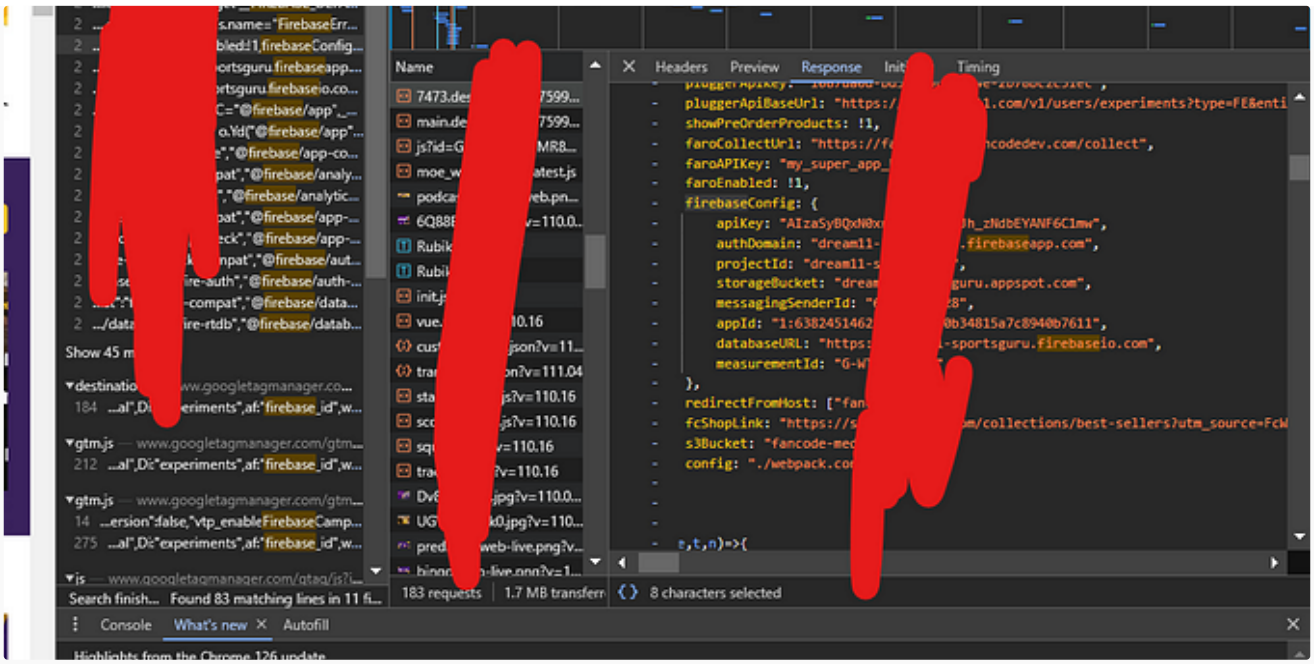
My Journey into a QA Security Mindset: Information disclosure vulnerability

My journey into a QA Security Mindset continues with learning about Information Disclosure vulnerabilities. If you would like to see how...

Aug 30, 2021 52

See all from Asaf Sahar

Recommended from Medium

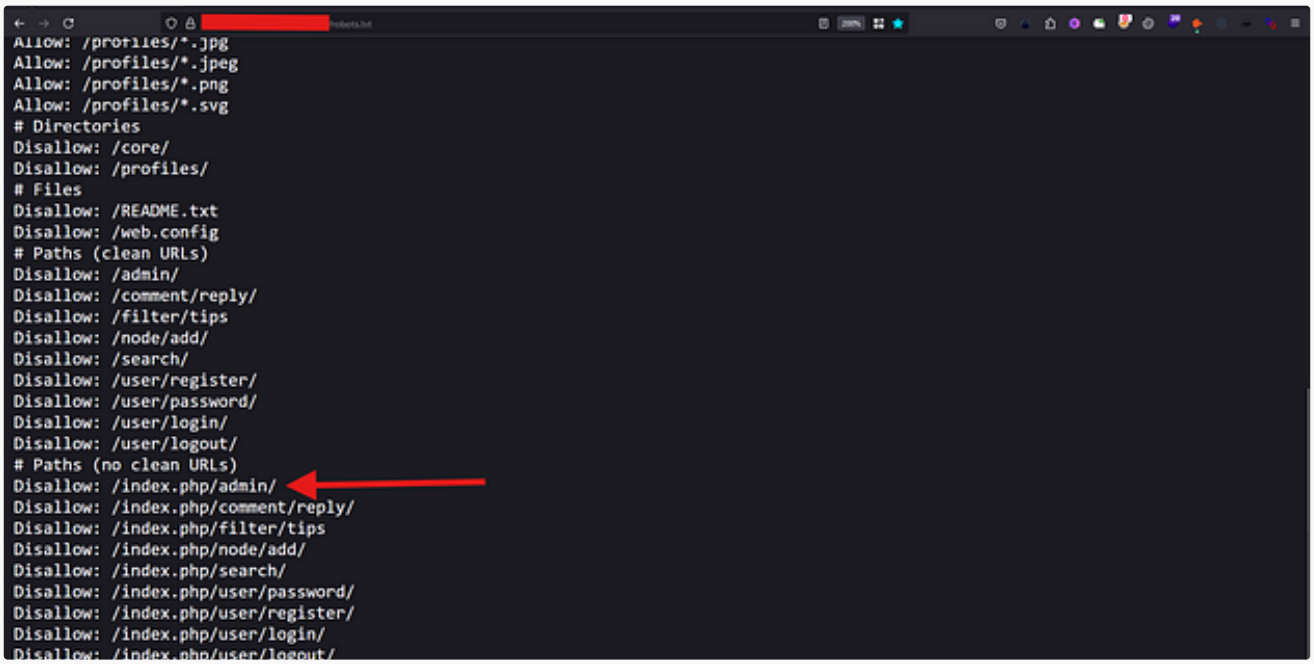


Shaikh Minhaz

How To Find Your 1st Bug For Bug Bounty Hunters (Step by Step Guide) Guarantee Result

How To Find Your 1st Bug For Bug Bounty Hunters (Step by Step Guide) Guarantee Result

Jul 31 837 12



Jayvin Gohel

Bypassed an Admin Panel Using SQL Payloads

It’s been a while since I last wrote a blog post, but I’ve got something interesting to share today. I recently managed to bypass an admin...

Sep 8 122 2

Lists

Staff Picks

751 stories · 1399 saves

Stories to Help You Level-Up at Work

19 stories · 846 saves

Self-Improvement 101

20 stories · 2930 saves

Productivity 101

20 stories · 2480 saves

Karthikeyan Nagaraj in System Weakness

How Finding an SQL Injection Vulnerability Earned a $1000 Bug Bounty

Bug Bounty Reports Explained by Karthikeyan Nagaraj

Sep 11 1K

ZeroDay Freak

Exploiting Web Applications with SQL Injection: A Step-by-Step Guide

SQL Injection remains one of the most critical and widespread vulnerabilities in web applications, often leading to severe data breaches...

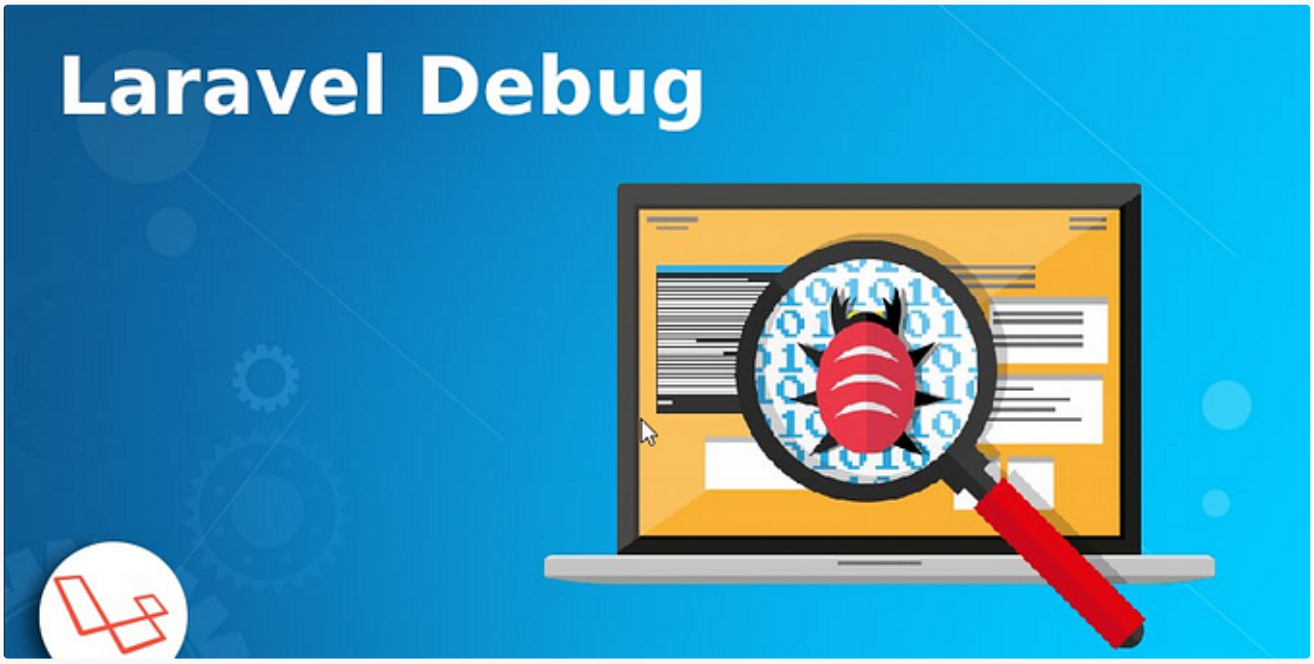
Jun 1 1

codingbolt

Blind SQL injection with time delays and information retrieval

Blind SQL Injection with Time Delays: The Art of Information Extraction

Oct 3 5



Professor.0xx01

Found Bug: Cross-Site Scripting (XSS) in Laravel Debug Mode ‼

Hello Hunters ‼ Hope you all are doing well.

May 18 52

See more recommendations