# Lab 1 – Manually testing attacks and protections

**Step 1: Simulate Cross-Site Scripting, SQL Injection, HTTP Flood, and Bad BOT activities.**

1. When WebCarter-attacker-template CloudFormation stack has been created, go to **Outputs** tab and save the ALB endpoint for your WebCarter web site:



1. RDP to the windows server created by WebCarter-attacker-template CloudFormation stack.

You will be using this Windows instance to simulate web application attacks with the following tools:

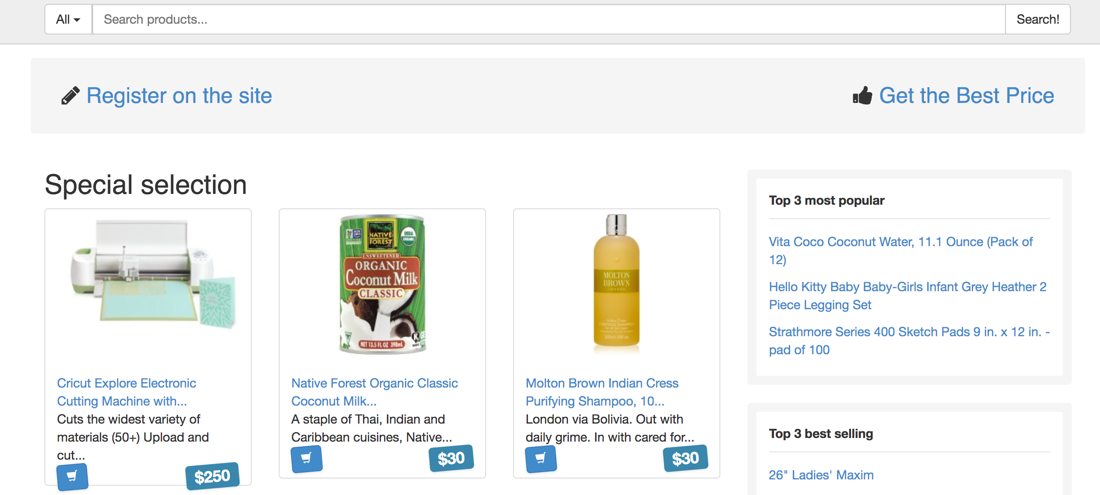
Apache JMeter: <https://jmeter.apache.org/>

Apache Benchmark (Ab): <https://httpd.apache.org/docs/2.4/programs/ab.html>

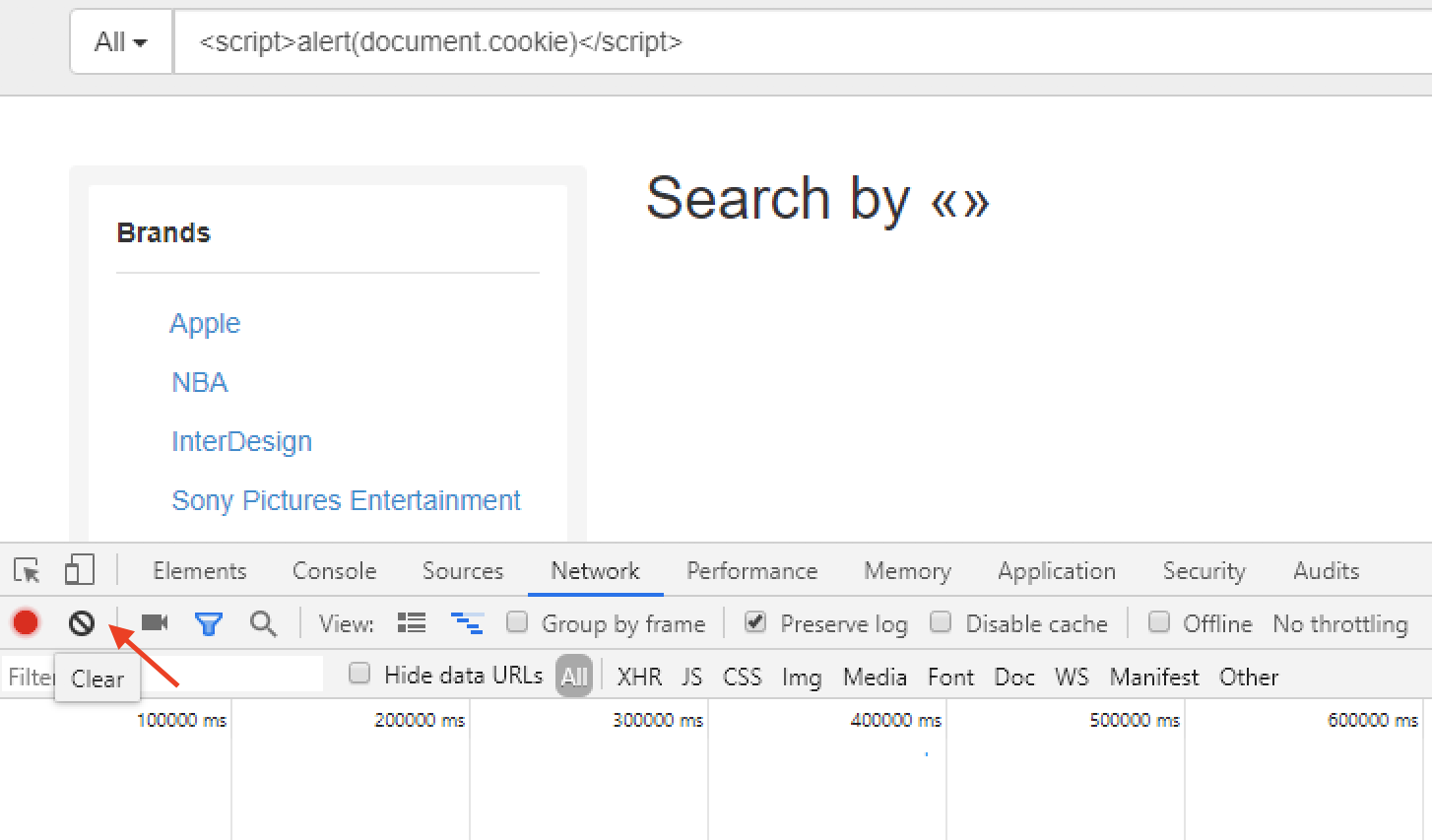
HTTRack: <https://www.httrack.com/>

These tools have been pre-installed on the Windows instance created above.

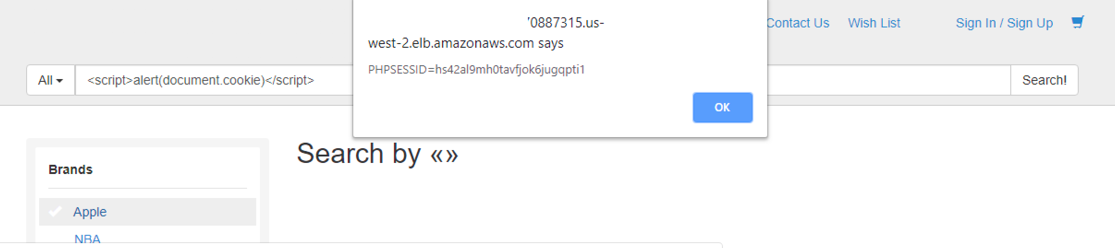
1. Open Chrome browser and navigate to the ALB endpoint you saved above:



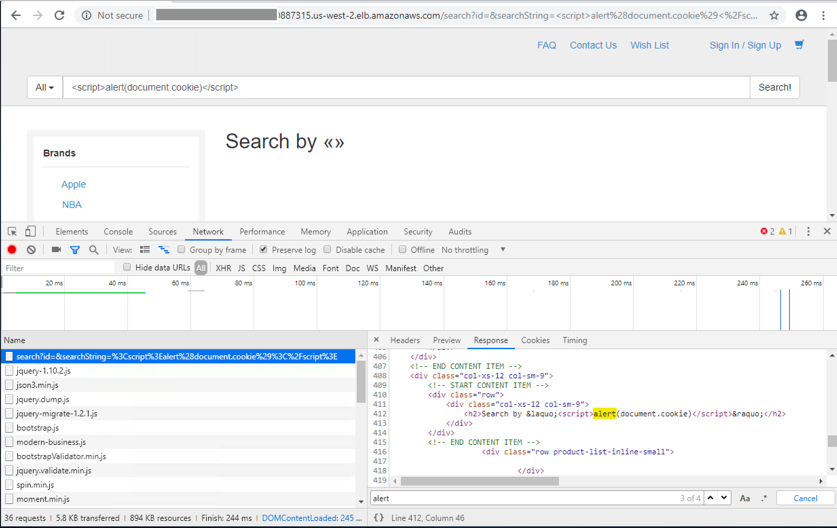
1. Right click on the page and choose “Inspect”. Chrome Developer Tools will open.
2. Go to Networking tab.
3. Type <script>alert(document.cookie)</script> in the “Search” text field in the WebCarter site and click “Search!”
4. Select the “search?id…” request in the Network tab and navigate to Response sub tab on the right. Use CTRL-F combination to search for “alert” in the response. As far as you can see, server returned your input above without validating it:



1. Click on any of the Brands filters (Apple, NBA). The script will be executed and the session cookie will be printed in the alert. Your Cross-site scripting attack was successful:

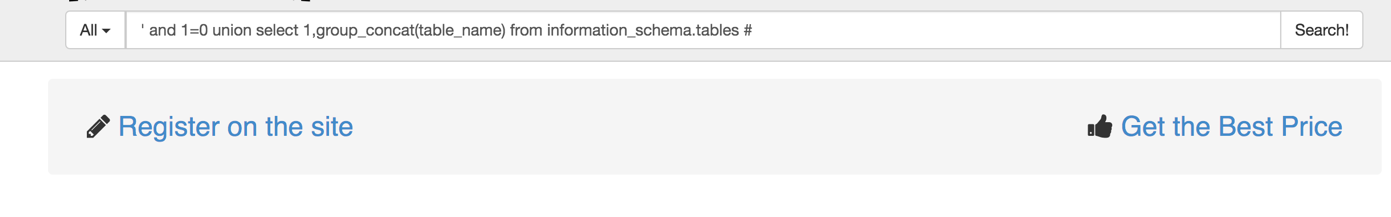


1. Click “Clear” button to clear the Developer Tools output:

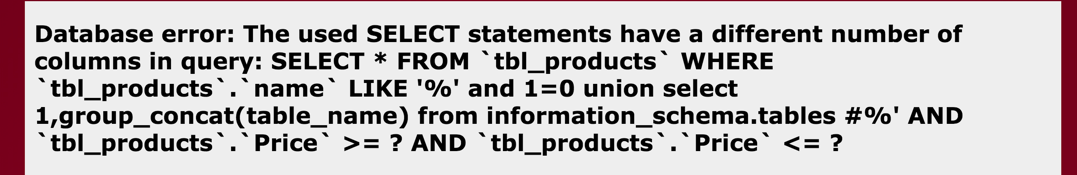


1. In the “Search” field enter the following:

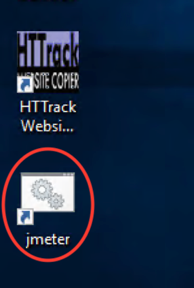
*' and 1=0 union select 1,group\_concat(table\_name) from information\_schema.tables #*



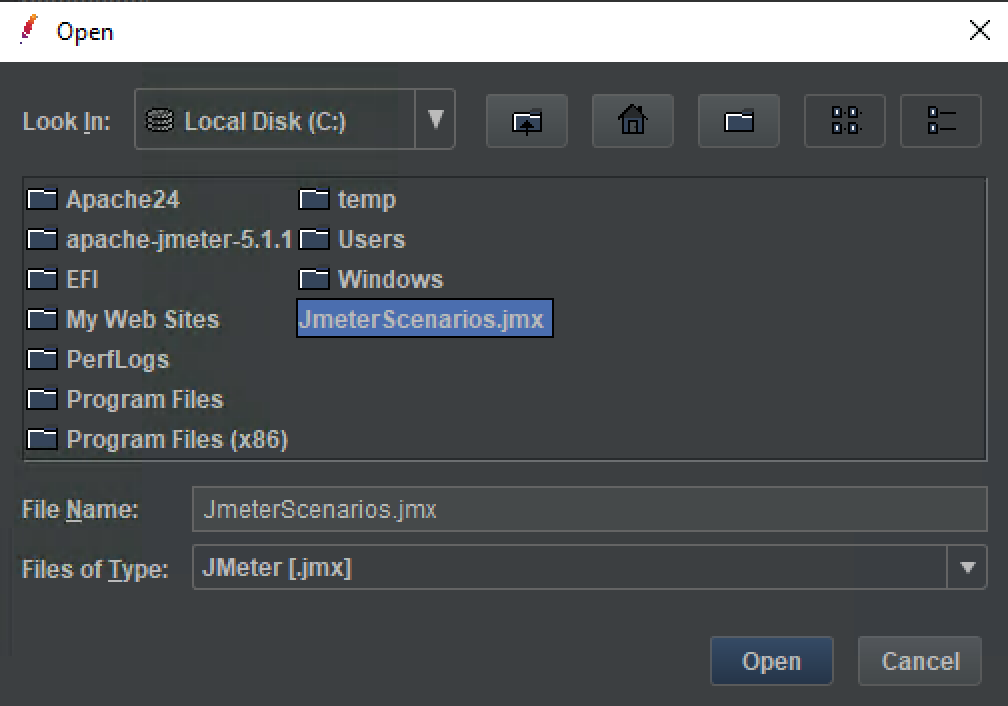
1. Server returns “Database Error” page displaying the SQL statement been executed. You started SQL Injection attack and got a desired output from the server. You can continue to explore SQL Injection vulnerabilities of WebCarter application later on.



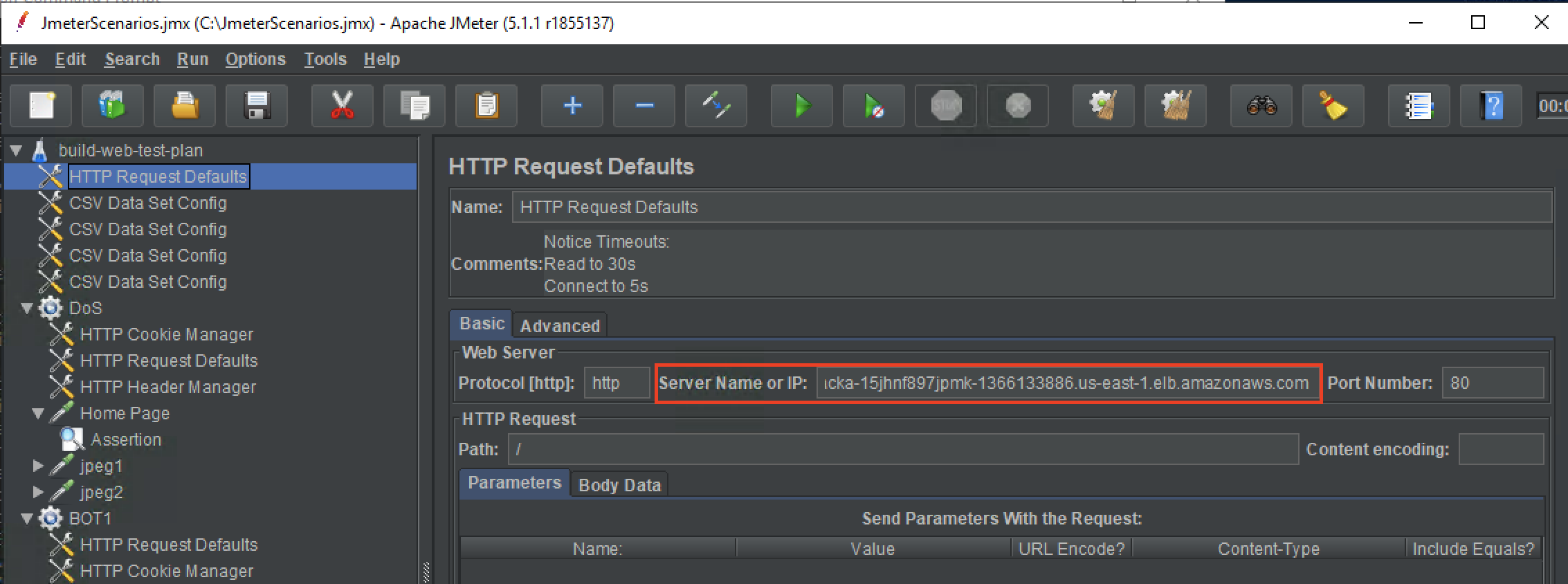
1. Run Apache JMeter by clicking on the JMeter shortcut on the desktop



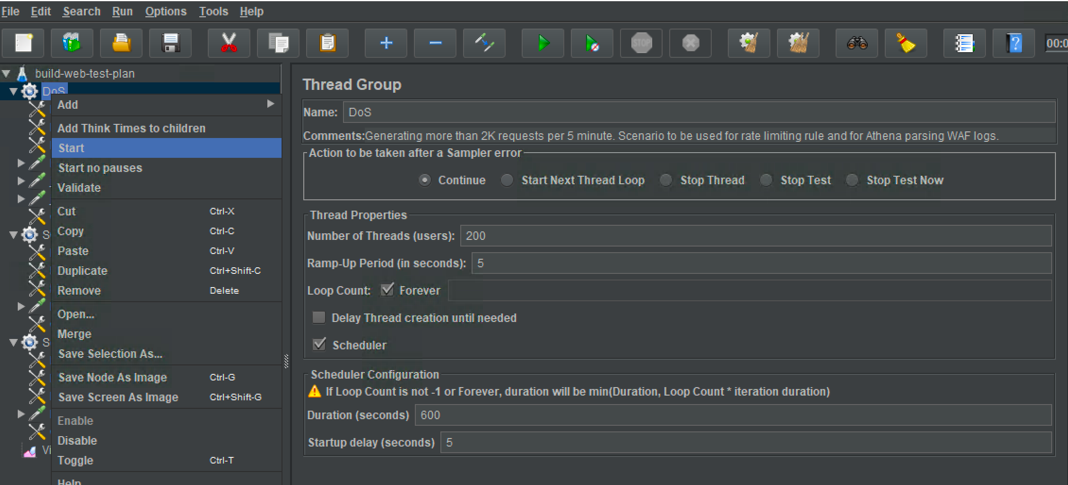
1. Go to File->Open menu and navigate to c:\JmeterScenarios.jmx file and click Open:



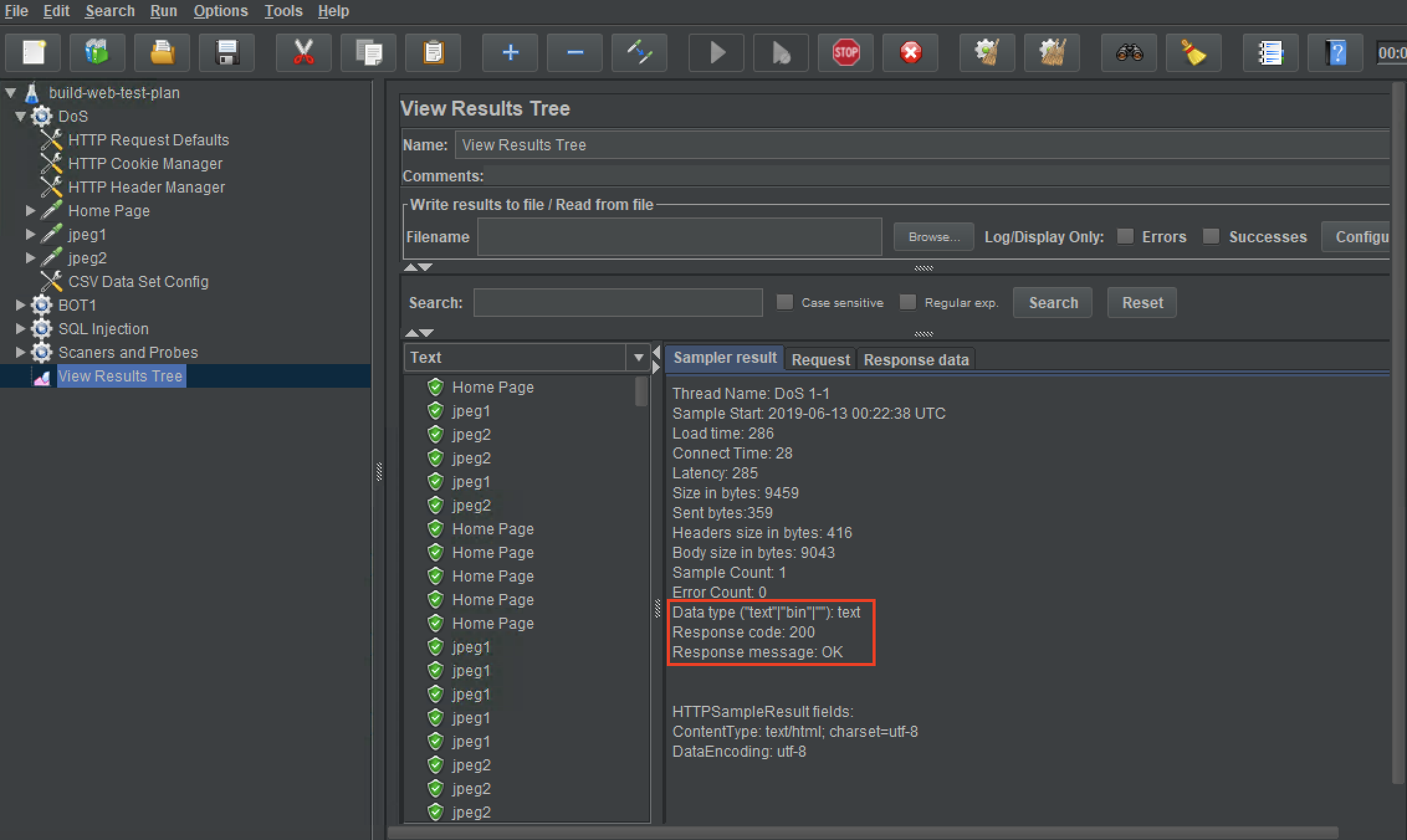
1. Click the most top “HTTP Requests Defaults” entry in the test plan menu. Update the “Server Name or IP” with the ALB endpoint saved in the p. 1 above. Enter the server name only without http:// and without the “/” suffux:



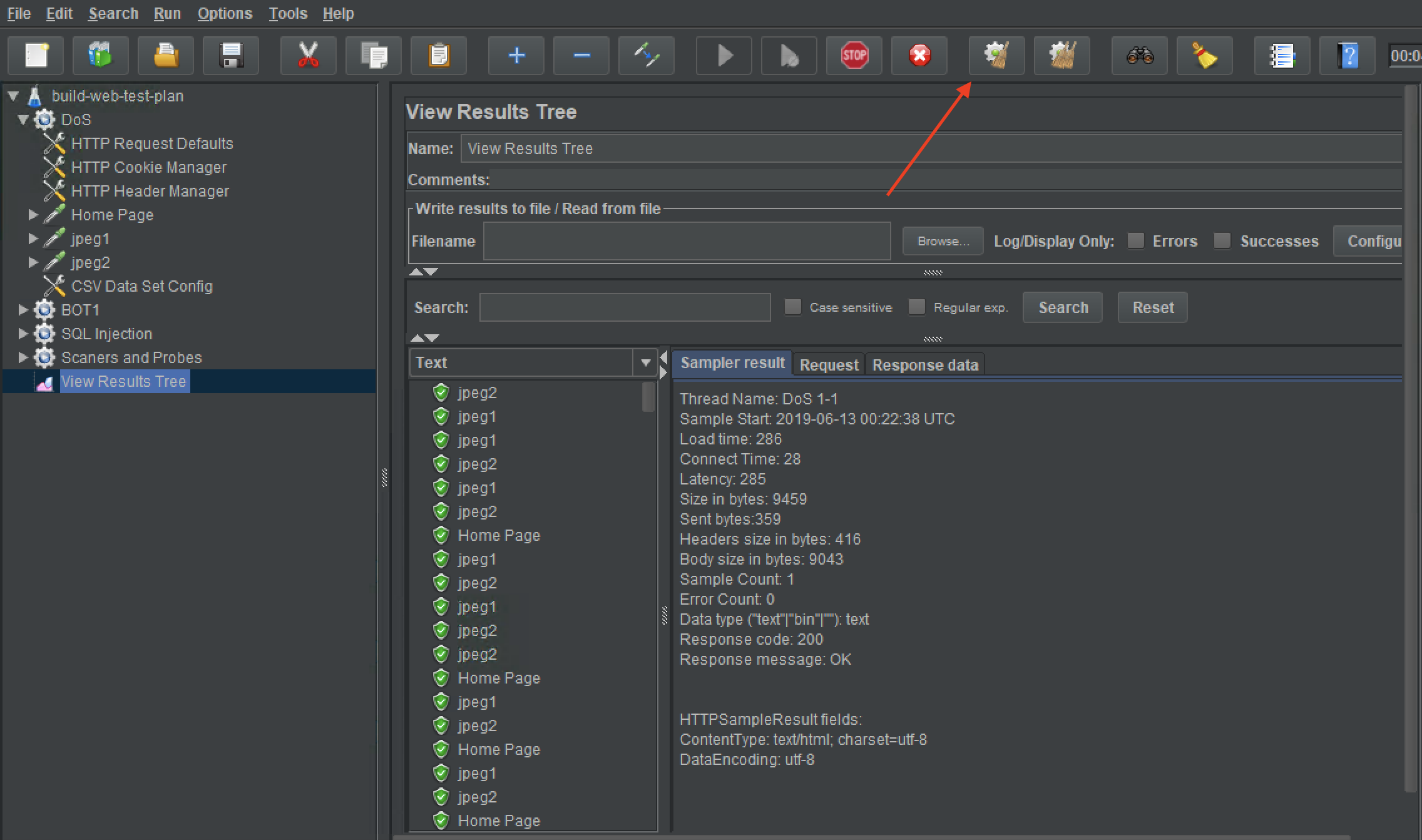
1. Save changes by navigating to File->Save menu.
2. Right click on the “DoS” scenario and then click “Start”:



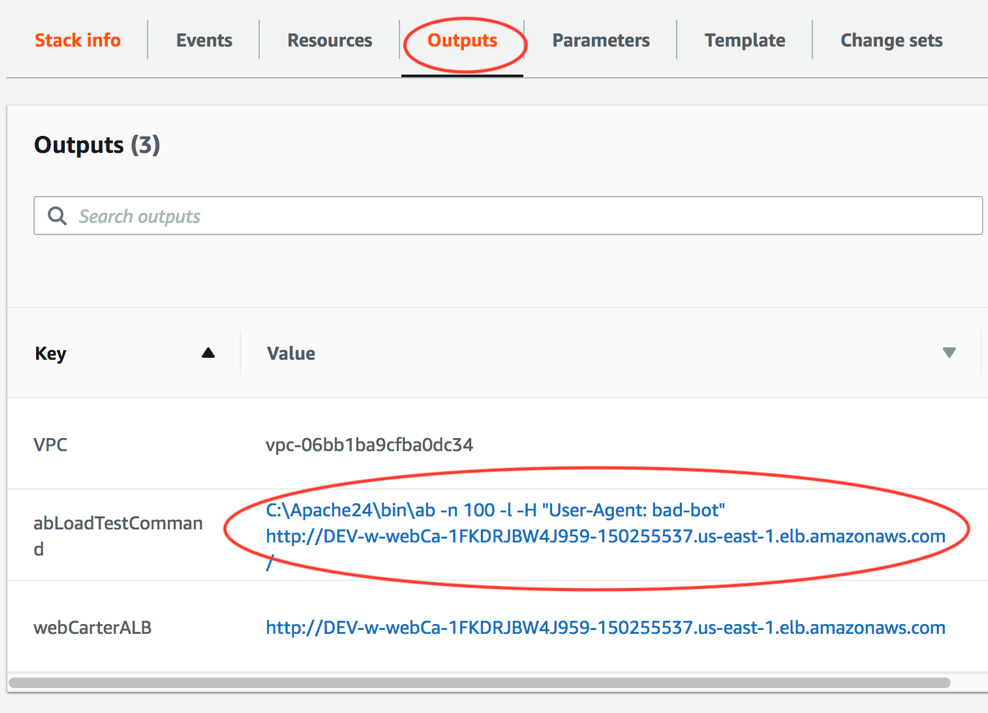
1. Navigate to “View Results Tree” and assure you’re getting 200 OK responses. Your attack was successful:



1. Clear the results tree for the next tests:



1. Open a cmd window and change directory to “C:\Apache24\bin”.
2. Run Apache Benchmark:
   * Copy the abLoadTestCommand from the **Outputs** tab in your Cloudformation stack



* Run the command in a Command Prompt window
  + Assure that ab run above was successful:

*Concurrency Level: 1*

*Time taken for tests: 6.813 seconds*

*Complete requests: 100*

*Failed requests: 0*

In this test we simulated a simple bot that causes excessive load on the server and can be identified by its *User-Agent.*

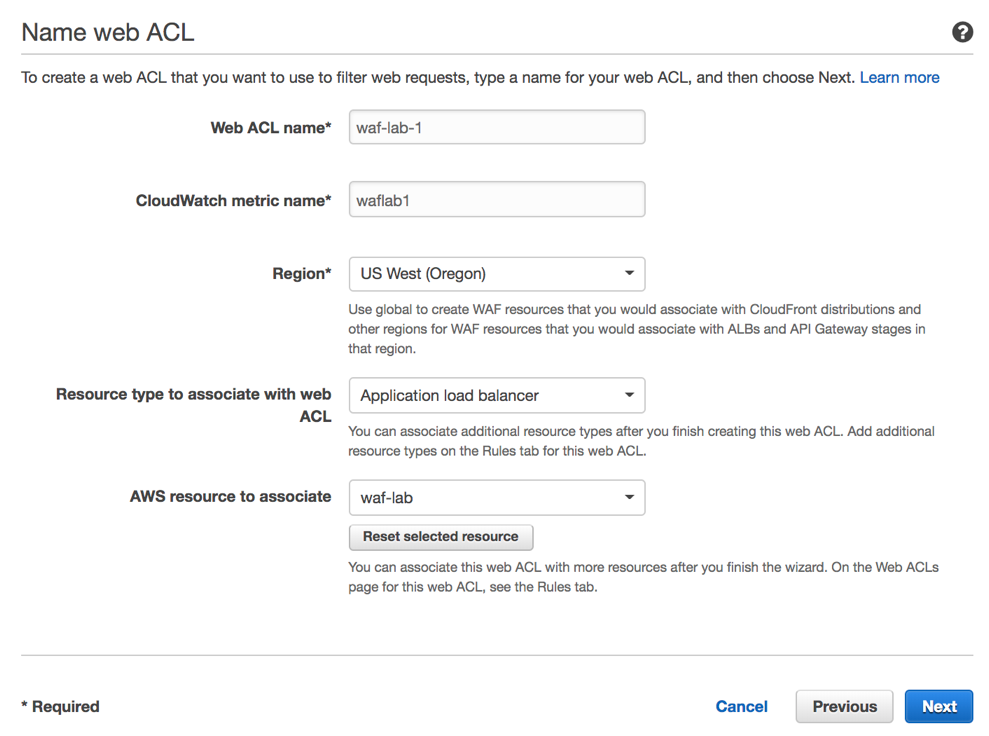
**Step 2: Configure protection against the above attacks using WAF native rules.**

1. Login to AWS console and choose EC2 service.
2. Choose “Load Balancers” in the left menu and then choose your load balancer created by CloudFormation stack.
3. Go to “Integrated Services” tab and click on “Create Web ACL” button. You’ll be directed to the WAF & Shield service console.
4. Choose the region you’re using in the “Filter” field:

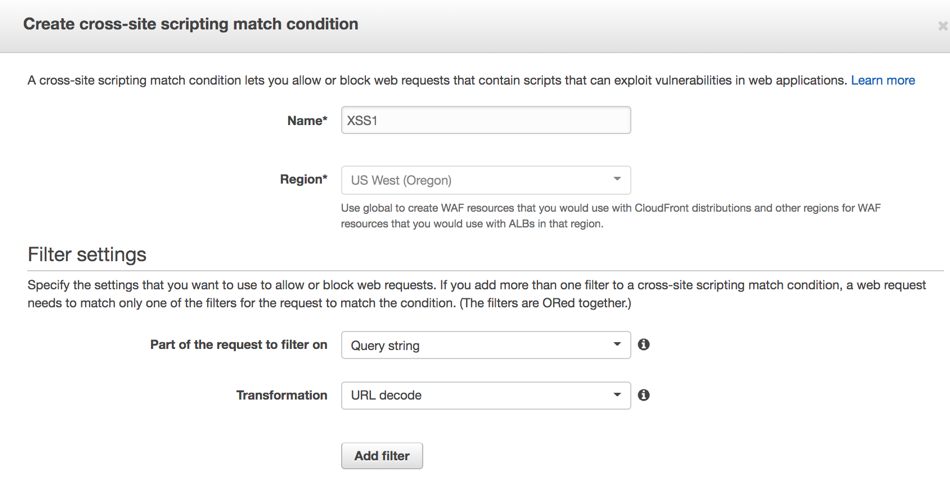


**Note**: WAF configuration is separate for each region and separate for “Global” deployment used for CloudFront deployments. You must configure your Web ACL in the same region where you have your ALB deployed.

1. Click “Create web ACL” and fill out the details as below and click “Next”:

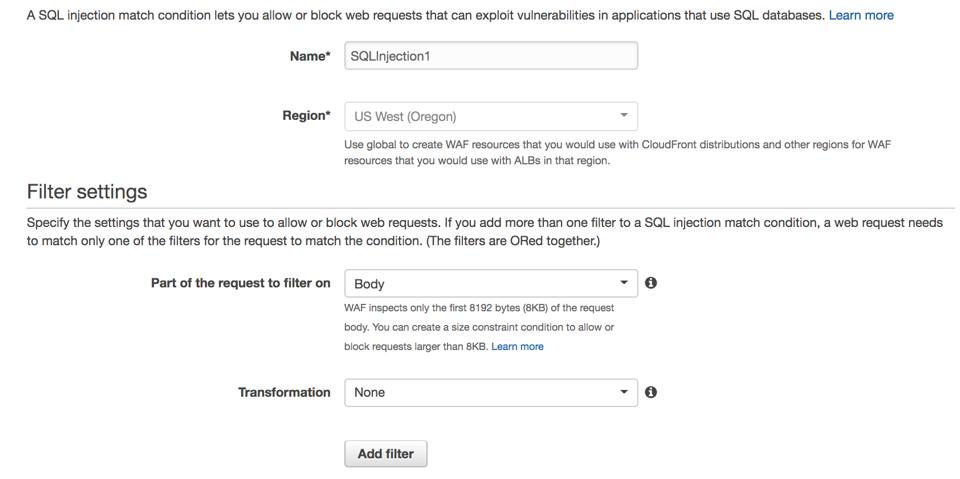


1. Create Cross-site scripting match condition. Click “Add Filter” then “Create”.

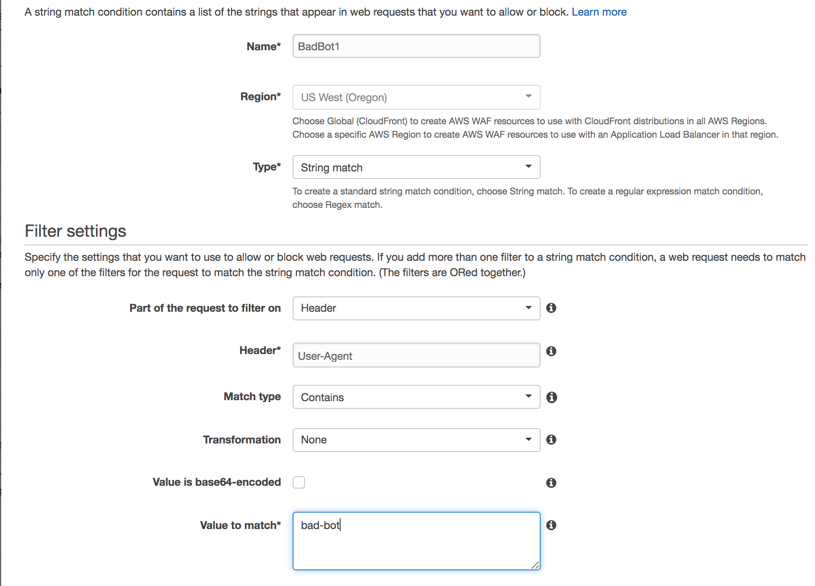


**Note**: Filters in the conditions are logically disjoined, i.e. “OR”. You can have up to 10 filters in each condition (hard limit).

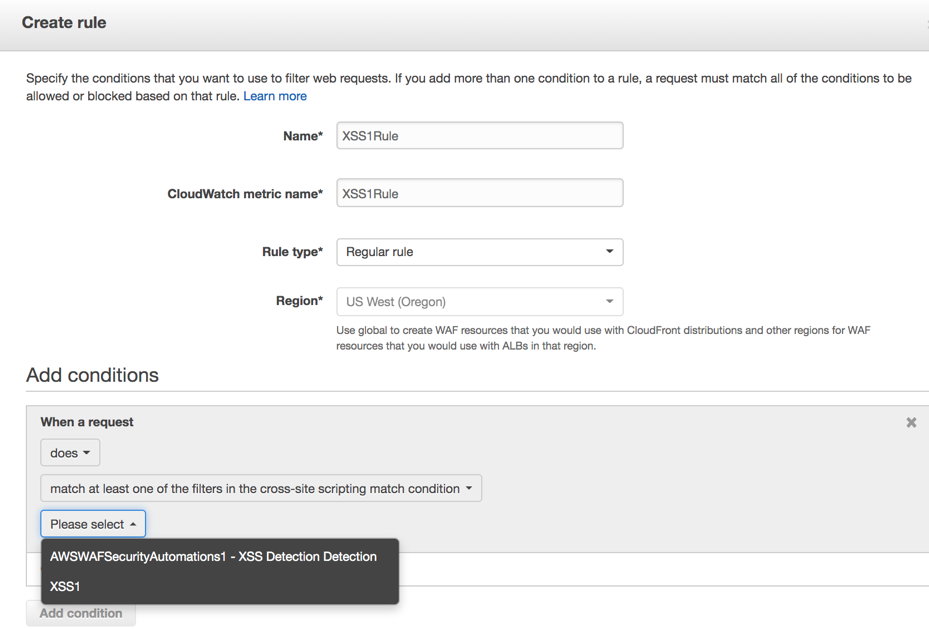
1. Create SQL injection match condition. Click “Add Filter” then “Create”.



1. Create String and regex match condition. Click “Add Filter” then “Create”.

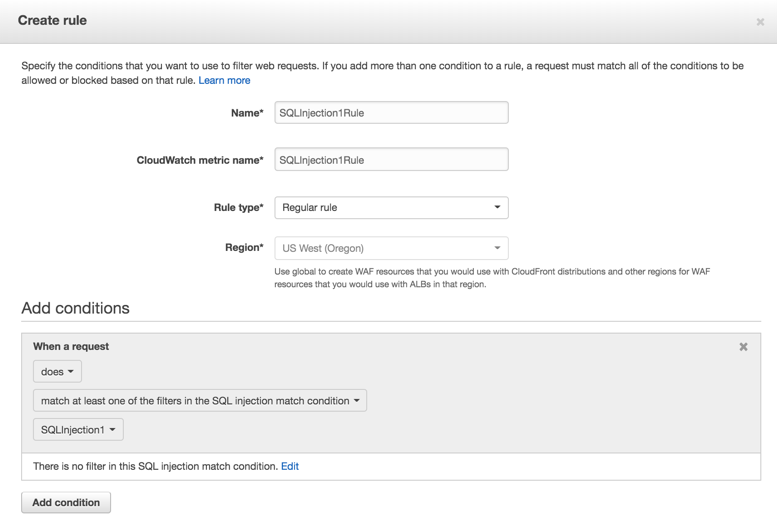


1. Click “Next” on the *Create Conditions* page.
2. Click “Create Rule” button.
3. Create a simple Cross-site scripting rule that in our example will contain a single XSS condition created above. Select the condition as below and click “Add Condition”:

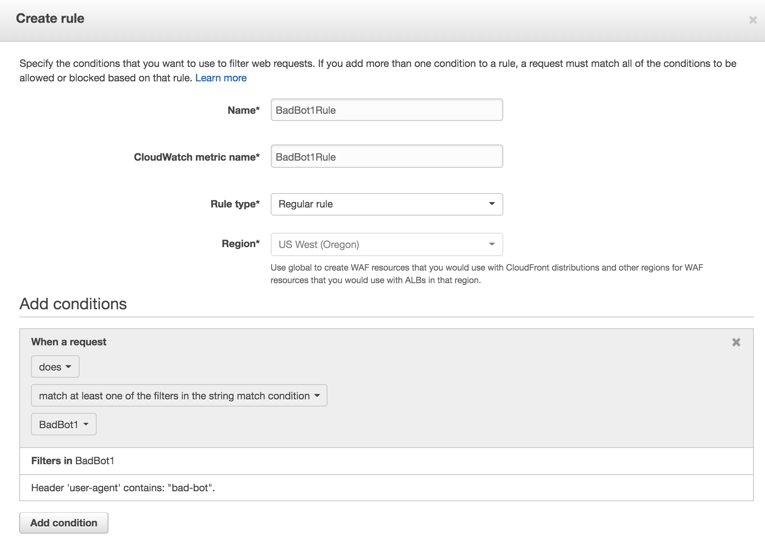


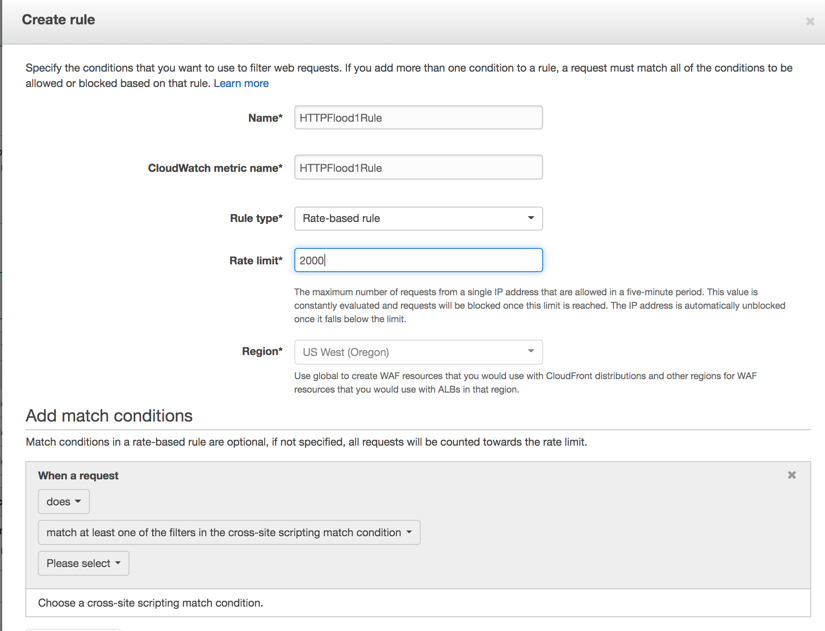
**Note**: Conditions in the rule are logically joined, i.e. “AND”. You can have up to 100 conditions of each type per account (soft limit).

1. Create SQL Injection rule choosing the SQL injection condition created above:

****

1. Create Bad Bot rule choosing the String match condition created above:

****

1. Create HTTP Flood rule changing the Rule type to “Rate-based rule”:

**Note**: You can add match conditions to the rate-based rule when crafting your custom signatures.

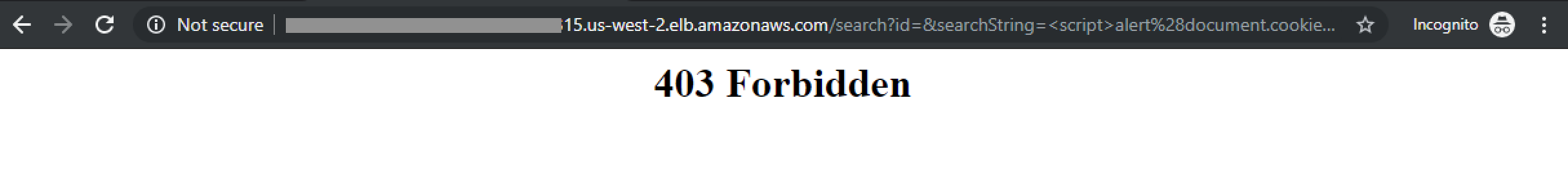
1. Set default Web ACL action as “Allow all requests that don't match any rules” and click “Review and create”.
2. Review the Web ACL and click “Confirm and create”.

**Note**: Due to we started creating a Web ACL from the ALB console, your Web ACL became associated with your load balancer. You also can just create Web ACL and then associate it with one or more load balancer instances.

**Step 3: Demonstrate protection against the above attacks using WAF native rules.**

In this step we’ll repeat the attacks in Step 1 above and assure they’re blocked by the rules we created in Step 2.

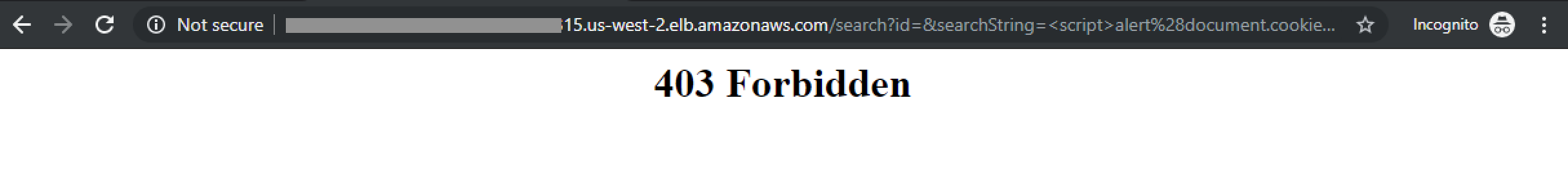
1. In a Chrome browser window navigate to your ALB endpoint and type <script>alert(document.cookie)</script> in the “Search” string. Then click “Search!”.
2. Assure you get HTTP 403 response from WAF:



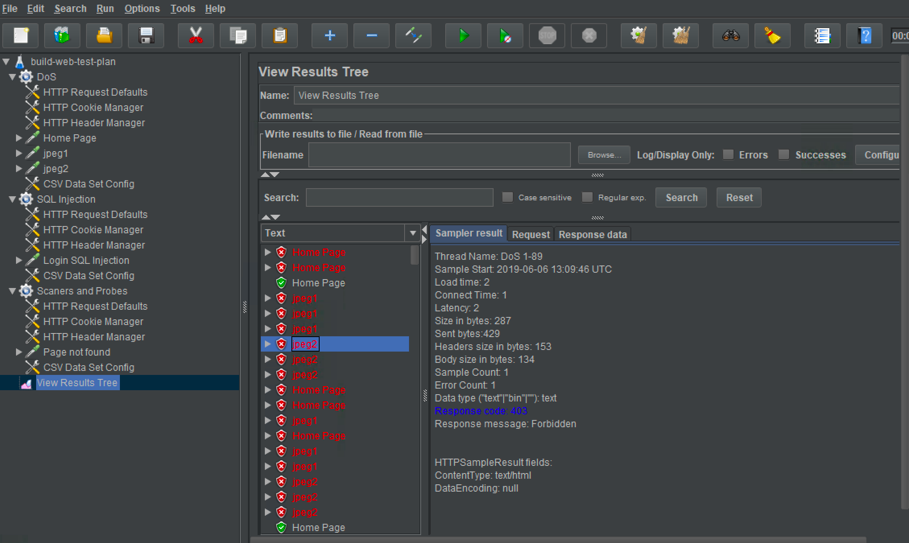
1. In the “Search” field enter the following:

*' and 1=0 union select 1,group\_concat(table\_name) from information\_schema.tables #*

1. Assure you get HTTP 403 response from WAF:



1. Start JMeter DoS scenario and navigate to the Results tree. Watch the request till they start failing after reaching 2000 requests / 5 min as configured in your rate-based rule:

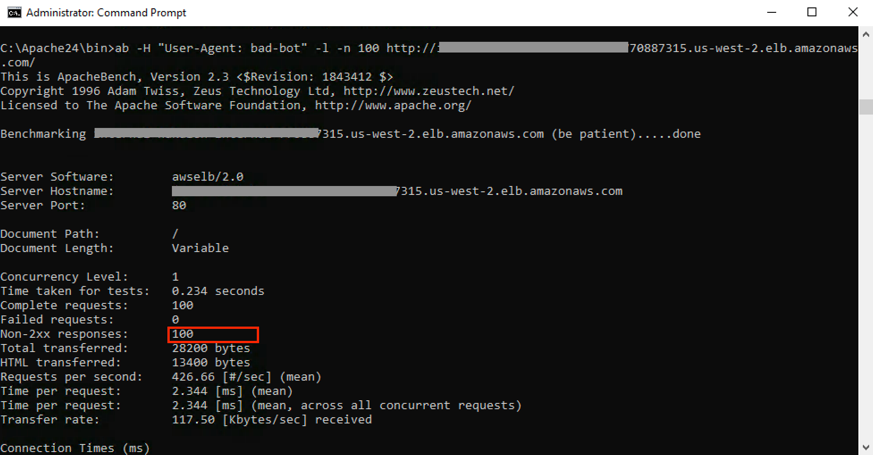


1. Go to AWS WAF Console and navigate to Rules-> HTTPFlood1Rule. Notice IP addresses that are currently blocked by this rule:



These IP addresses configured in JMeter in “CSV Data Set Config” section.

1. Repeat ab test as above and notice Non-2xx responses from WAF:



In this lab you protected your Web application against Cross-site scripting, SQL Injection, HTTP flood and simple BOTs using AWS WAF native rules.