SDD311 – Protecting from bots and scrapers with AWS WAF

re:Inforce 2019

June, 2019



Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

© 2019 Amazon Web Services, Inc. or its affiliates. All rights reserved.

Contents

[Pre-requisites 4](#_Toc11741934)

[Lab Prep – Launch Cloudformation Stacks Setup 4](#_Toc11741935)

[WAF Automation And Dashboard Setup 4](#_Toc11741936)

[Web Carter & Attacker workstation setup 8](#_Toc11741937)

[Lab 1 – Manually testing attacks and protections 12](#_Toc11741938)

[Lab 2 – Automating Protections and Tracking Attacks 26](#_Toc11741939)

# Pre-requisites

* A computer with Chrome
* Optional - git

# Lab Prep – Launch Cloudformation Stacks Setup

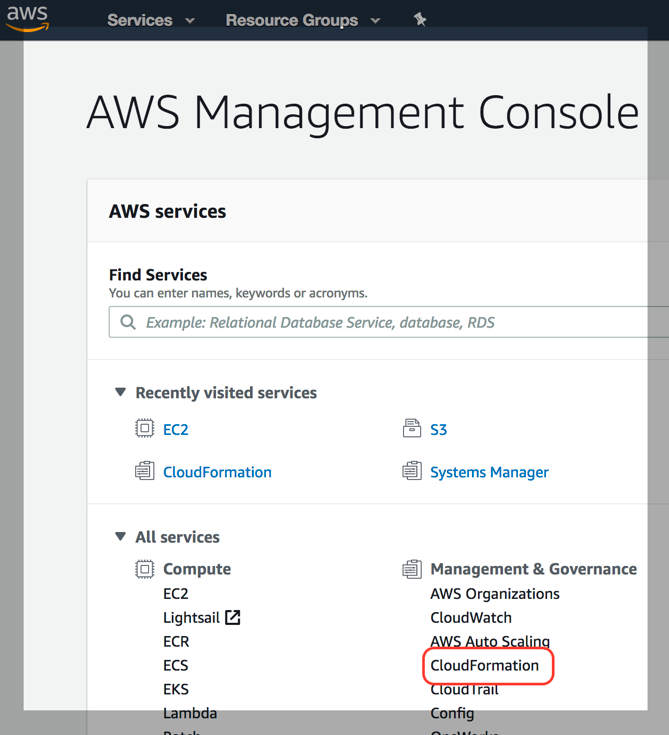
(Estimated completion time – 35 minutes)

Lab preparation will take some time to complete, so we recommend that you initiate this and let it complete in the background.

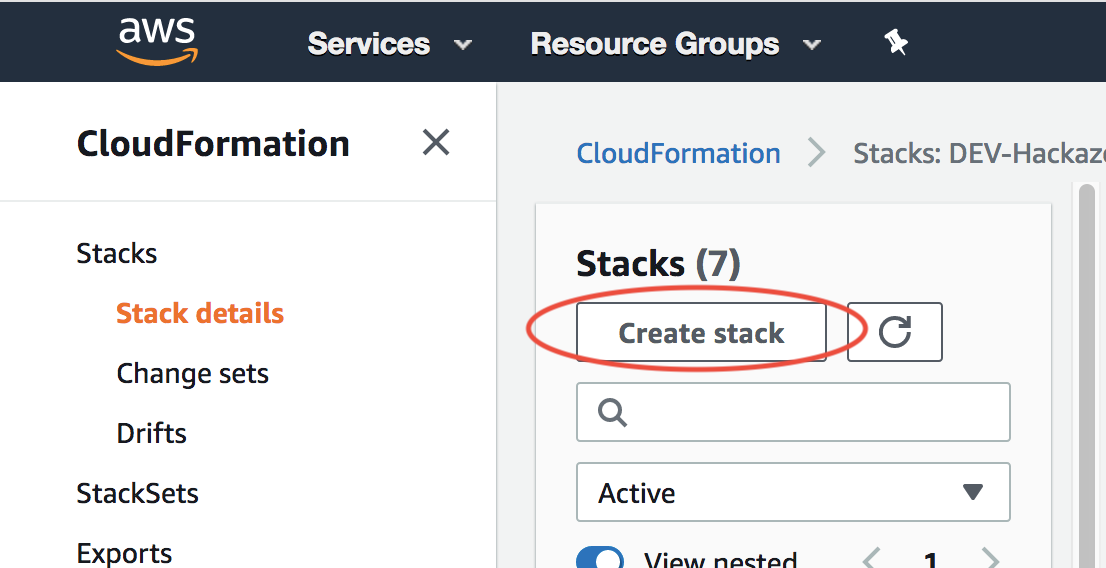
## WAF Automation And Dashboard Setup

This Cloudformation stack will stand up the automation for managing WAF and the dashboards.

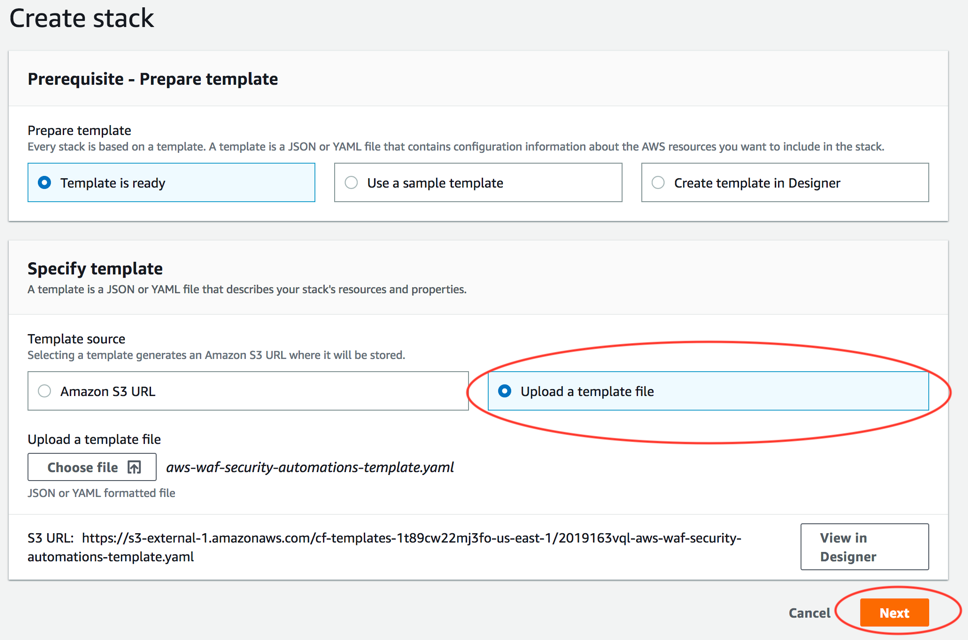
1. Open the CloudFormation console



1. Create a new stack



1. Upload the template template/aws-waf-security-automations-template.yaml



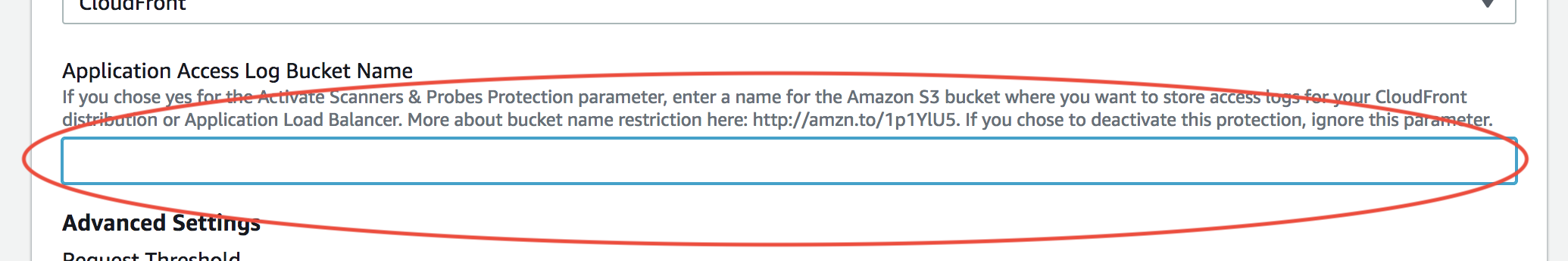
1. Enter a new stack name in **all lowercase** – e.g. waf-lab



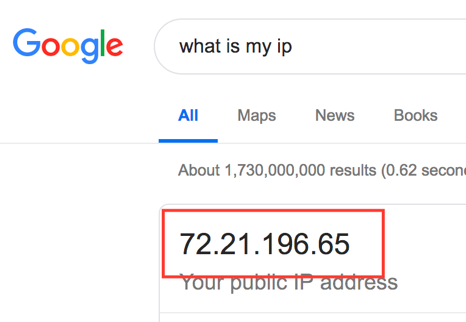
1. Update Scanner & Probe Protection to “yes – Amazon Athena log parser”:



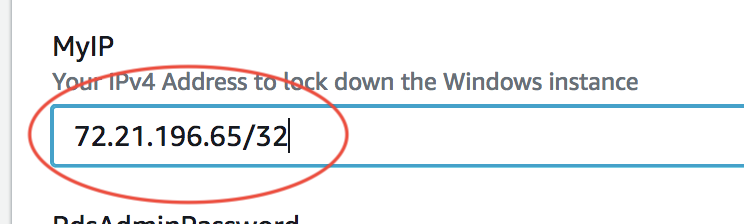
1. Enter a new S3 bucket to collect access logs for WAF



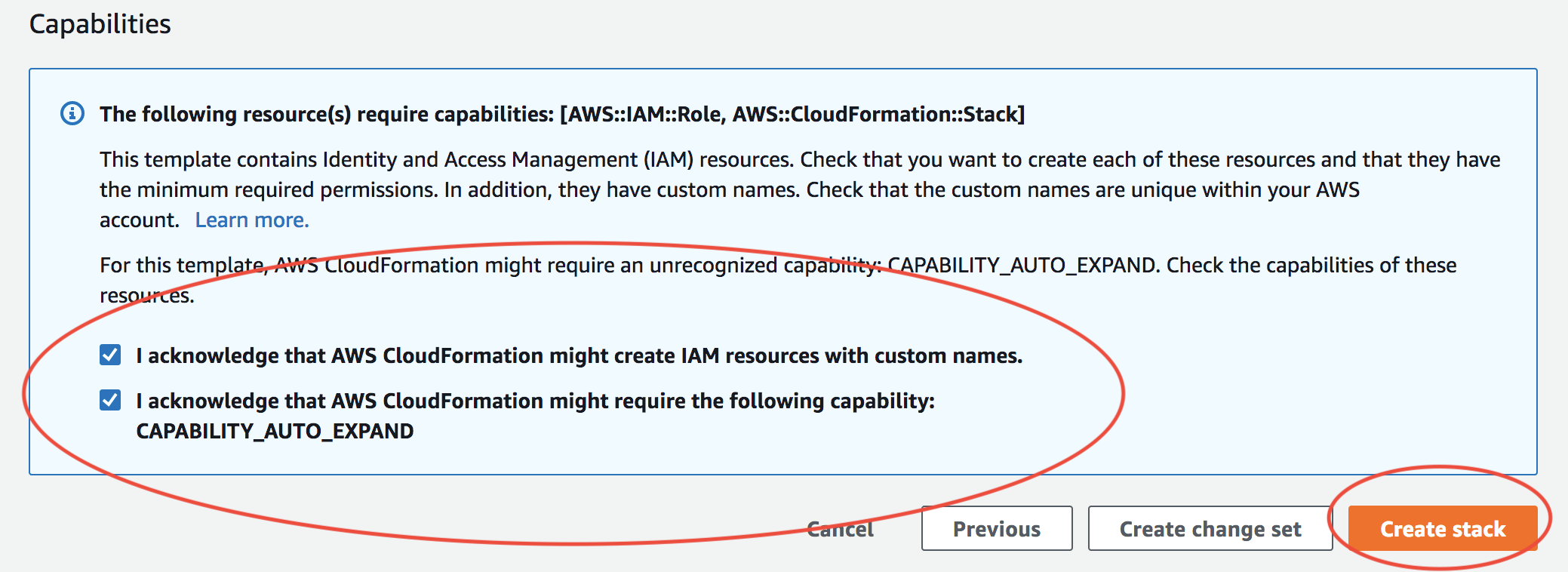
1. Get your IP from Google with “what is my ip”.



1. Enter your IP + “/32” for the attacker’s allowed ingress IP address in the Cloudformation form. Please replace 72.21.196.65 in these instructions with your IP address:



1. Scroll to the bottom of the page and hit **Next**.
2. Hit **Next** on “Configure stack options”
3. Scroll to the bottom of the page. Check **I acknowledge that AWS CloudFormation might create IAM resources with custom names.** and **I acknowledge that AWS Cloudformation might require the following capability: CAPABILITY\_AUTO\_EXPAND**, then click “Create”



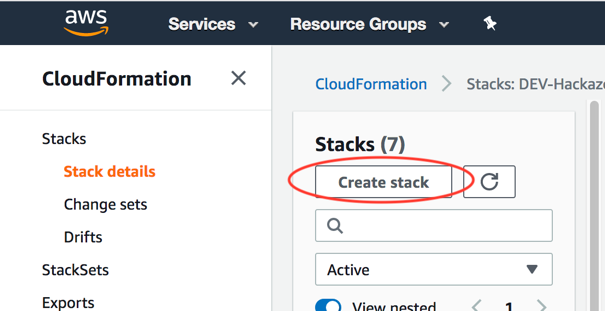
1. While you’re waiting for the stack turns to CREATE\_COMPLETE, let’s start on the next stack.

## Web Carter & Attacker workstation setup

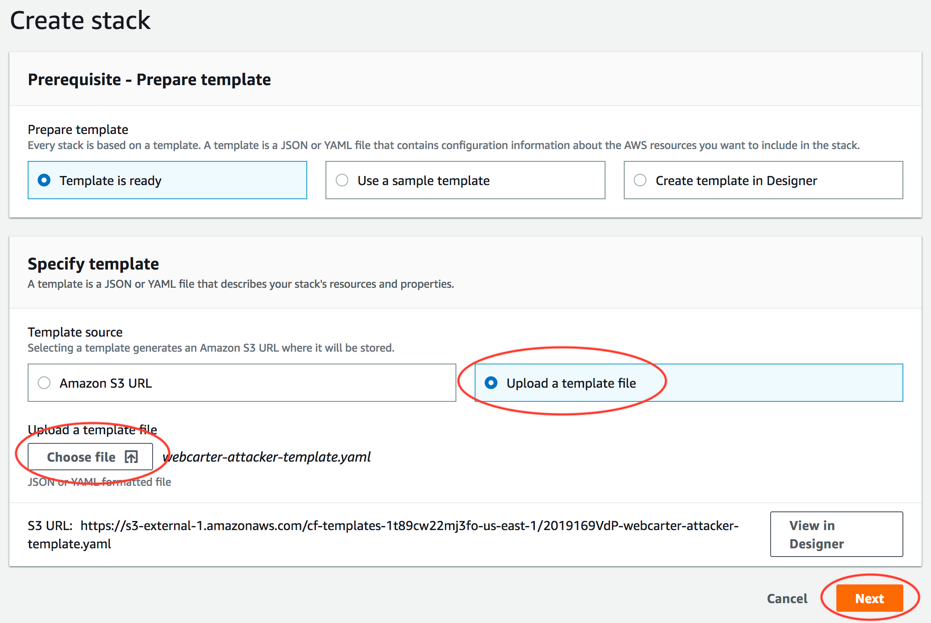
1. Open the Cloudformation console



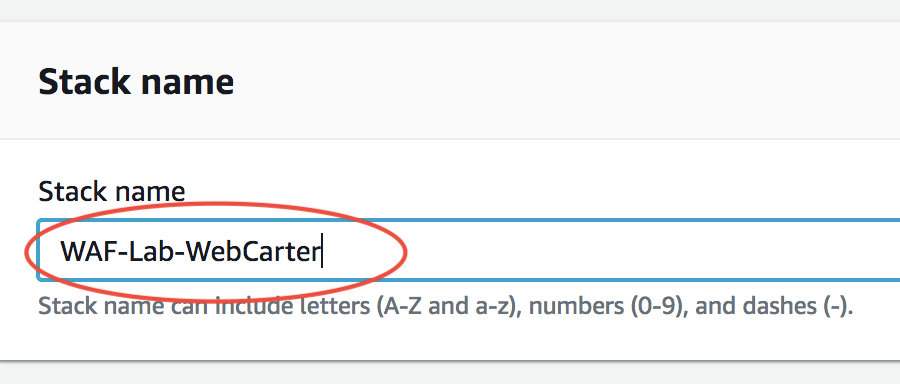
1. Create a new stack



1. Upload stack template **template/webcarter-attacker-template.yaml**



1. Enter a stack name – e.g. “WAF-Lab-WebCarter”

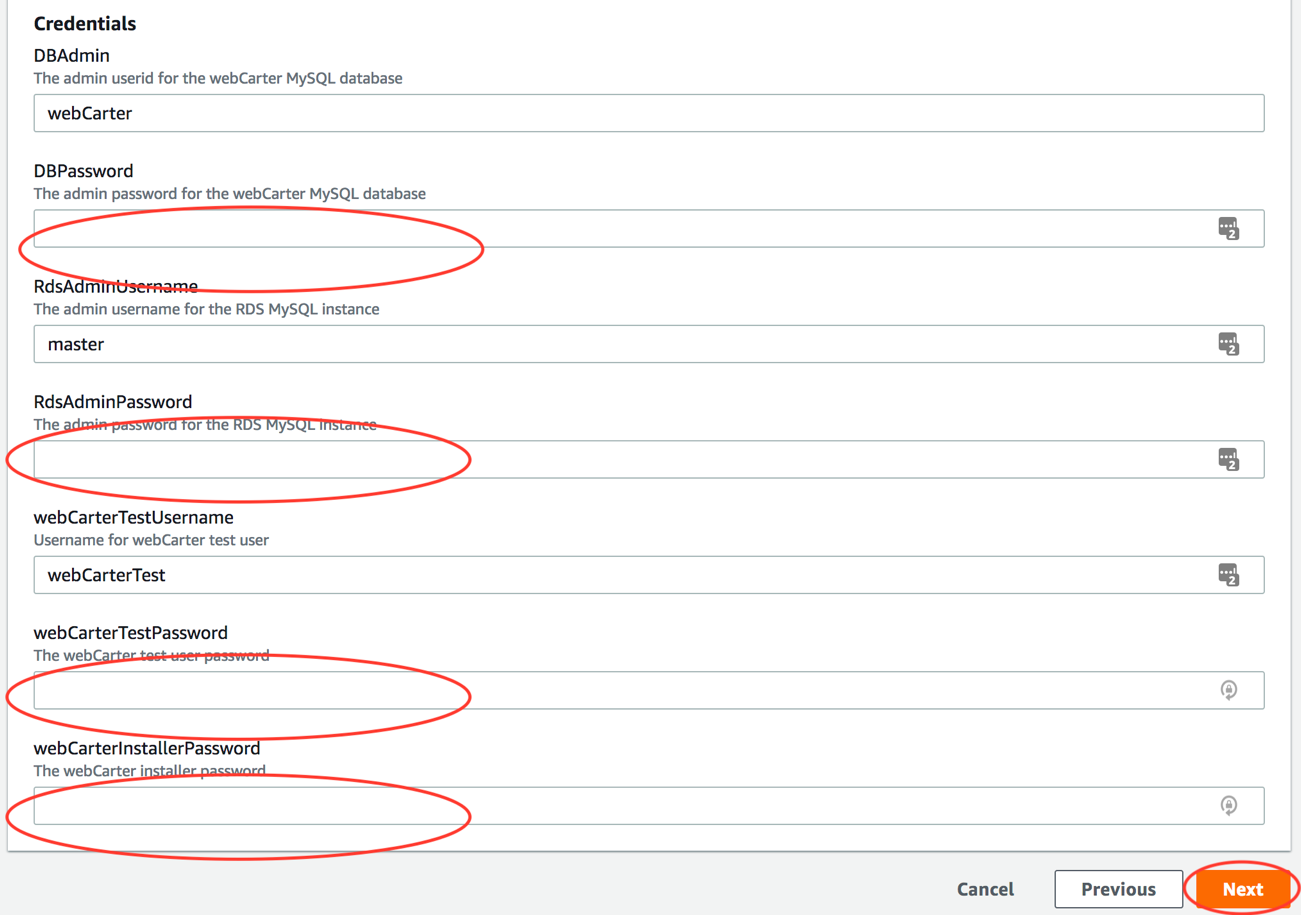


1. Enter the environmental configurations for the stack
   1. Your IP address from above
   2. An SSH in your inventory from EC2
   3. An environment tag (e.g. DEV or TEST or PROD)

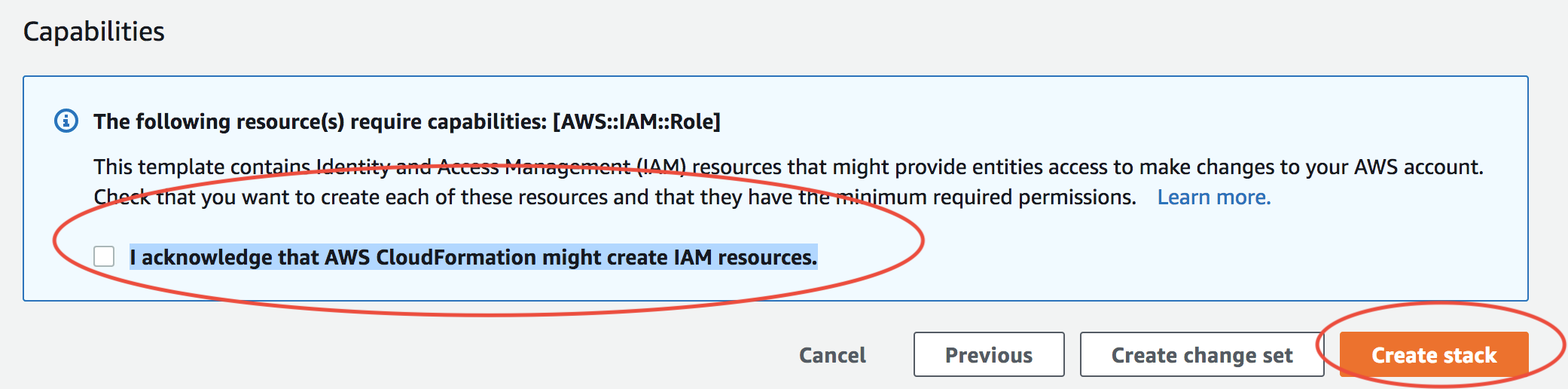


1. Enter passwords for the following credentials:
   1. RDS instance master password
   2. RDS database password
   3. Admin password
   4. Test user password

Scroll to the bottom of the page and hit **Next**



1. Scroll to the bottom of the page and hit **Next** on *Configure stack options* page
2. Scroll to the bottom of the page and check **I acknowledge that AWS CloudFormation might create IAM resources.**, the click **Create**



1. The stack will eventually turn to CREATE\_COMPLETE.

# 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:

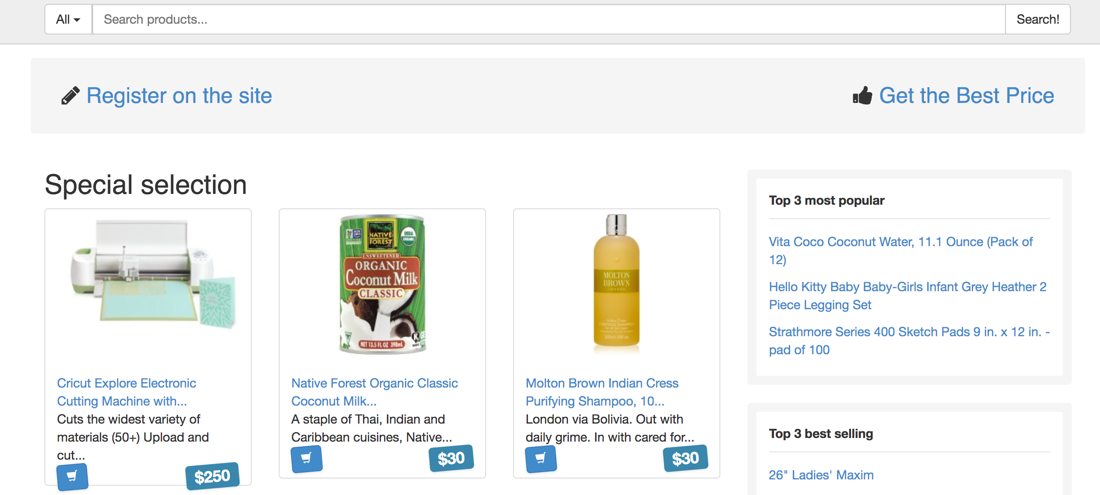
Apache JMeter: <https://jmeter.apache.org/>

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

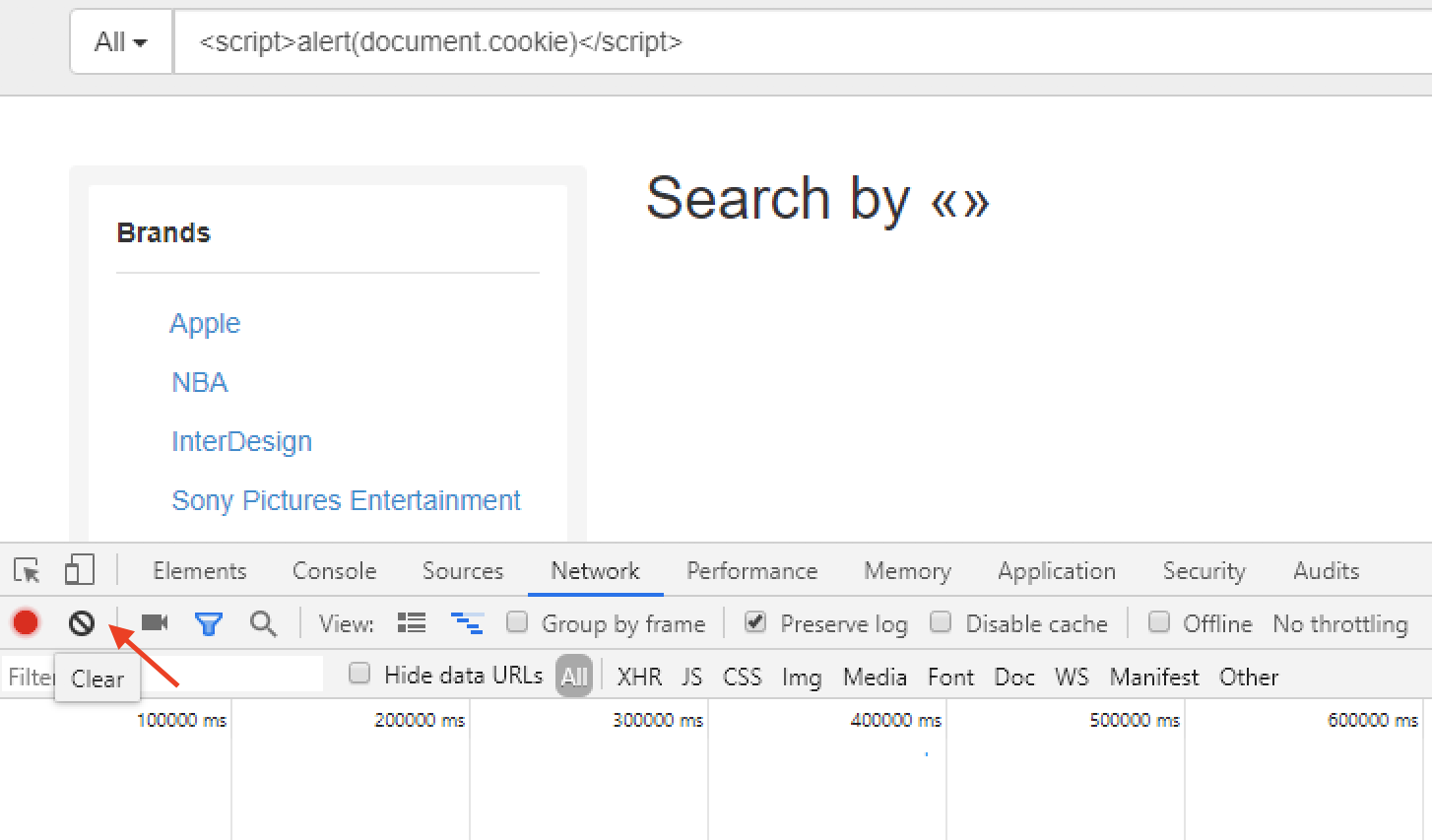
HTTRack: <https://www.httrack.com/>

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

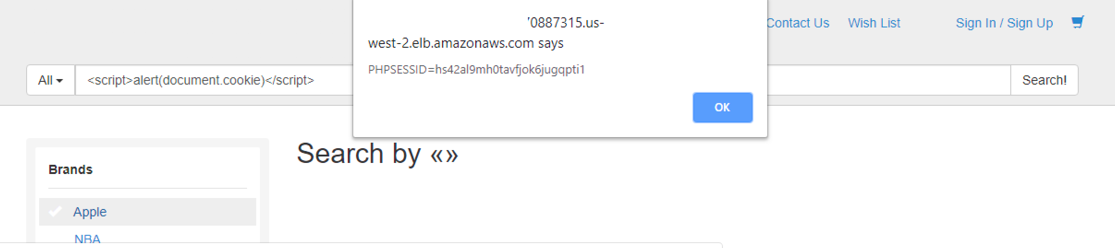
1. Open Chrome browser and navigate to the ALB endpoint you saved above:



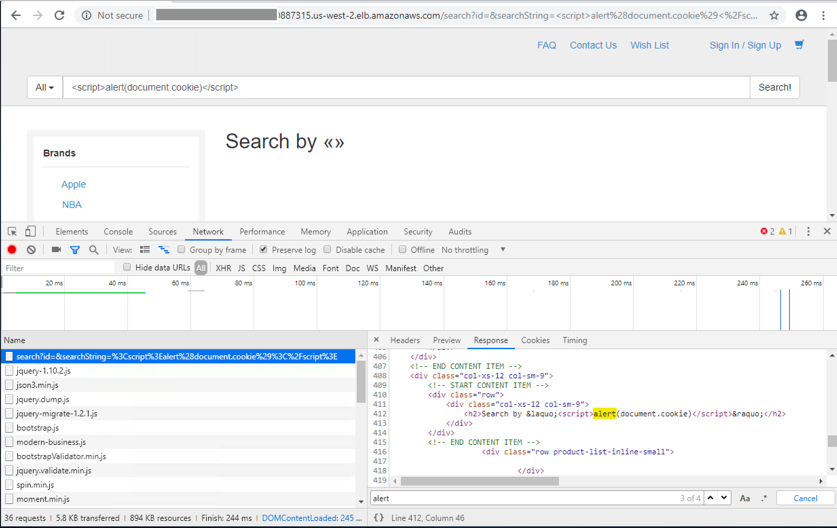
1. Right click on the page and choose “Inspect”. Chrome Developer Tools will open.
2. Go to Networking tab.
3. Type <script>alert(document.cookie)</script> in the “Search” text field in the WebCarter site and click “Search!”
4. Select the “search?id…” request in the Network tab and navigate to Response 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:

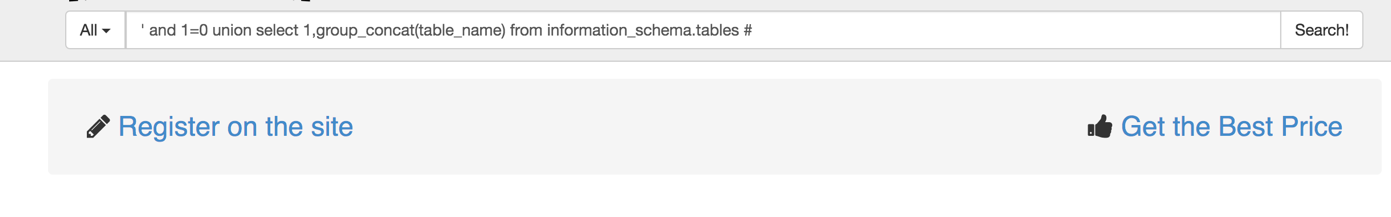


1. Click “Clear” button to clear the Developer Tools output:

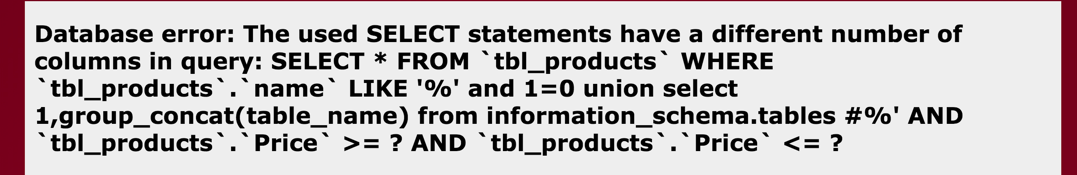


1. In the “Search” field enter the following:

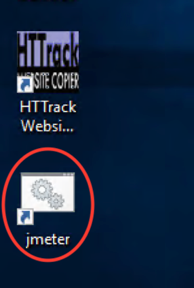
*' and 1=0 union select 1,group\_concat(table\_name) from information\_schema.tables #*



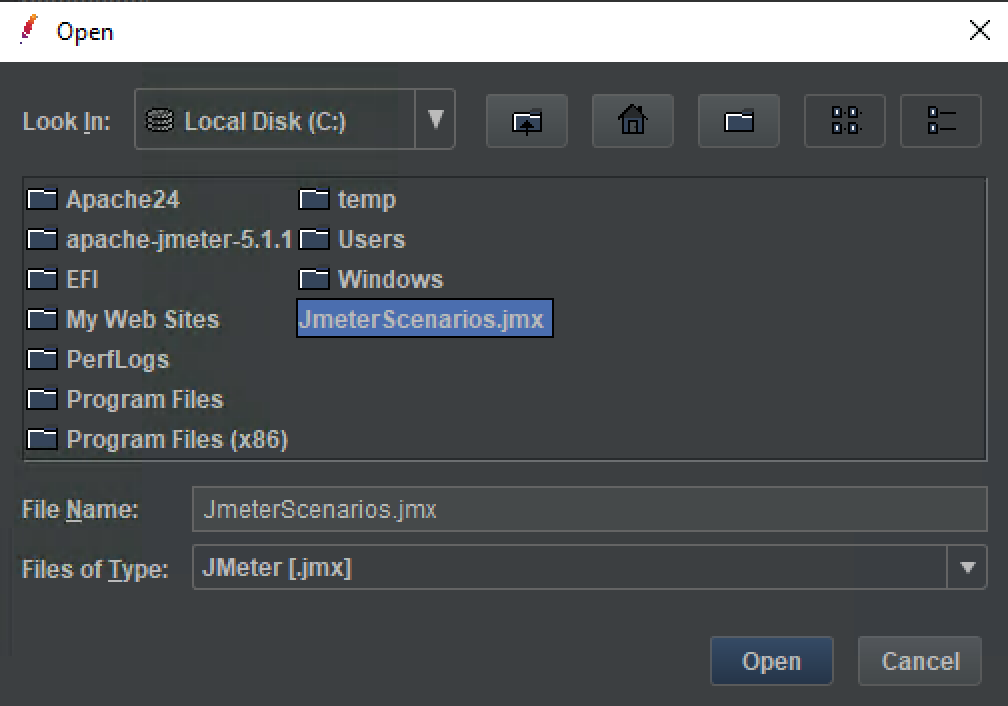
1. Server returns “Database Error” page displaying the SQL statement been executed. You started SQL Injection attack and got a desired output from the server. You can continue to explore SQL Injection vulnerabilities of WebCarter application later on.



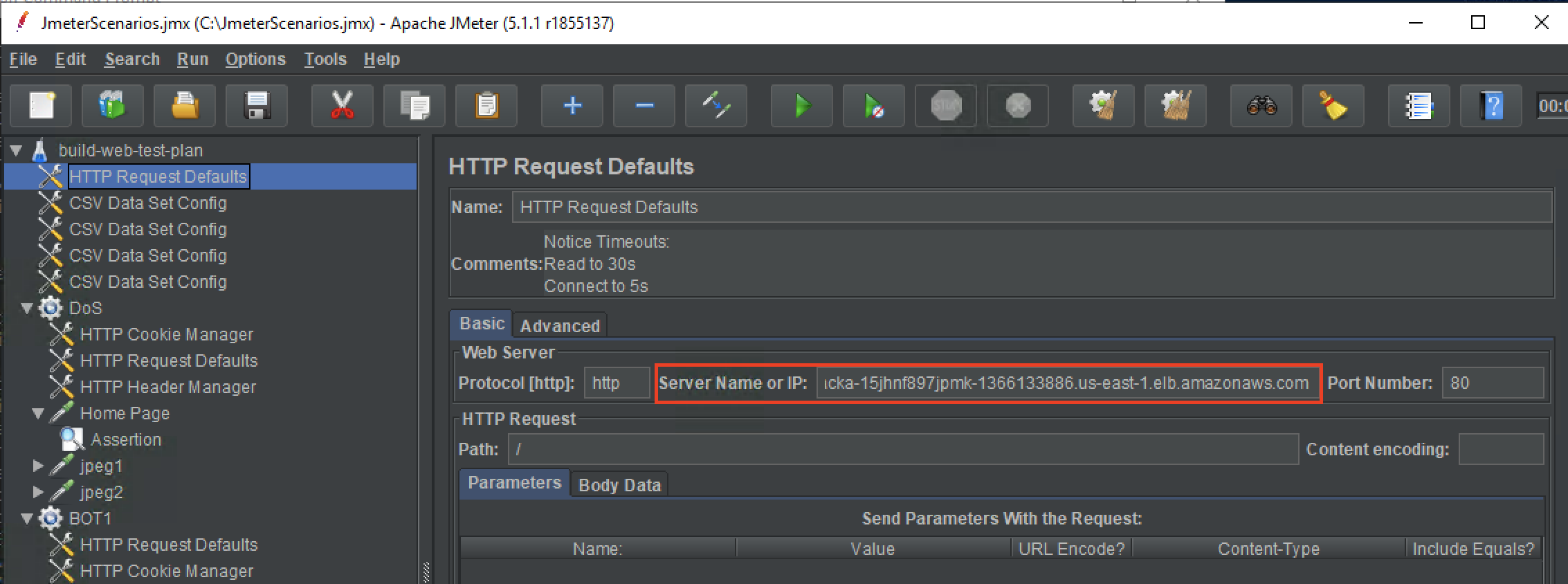
1. Run Apache JMeter by clicking on the JMeter shortcut on the desktop



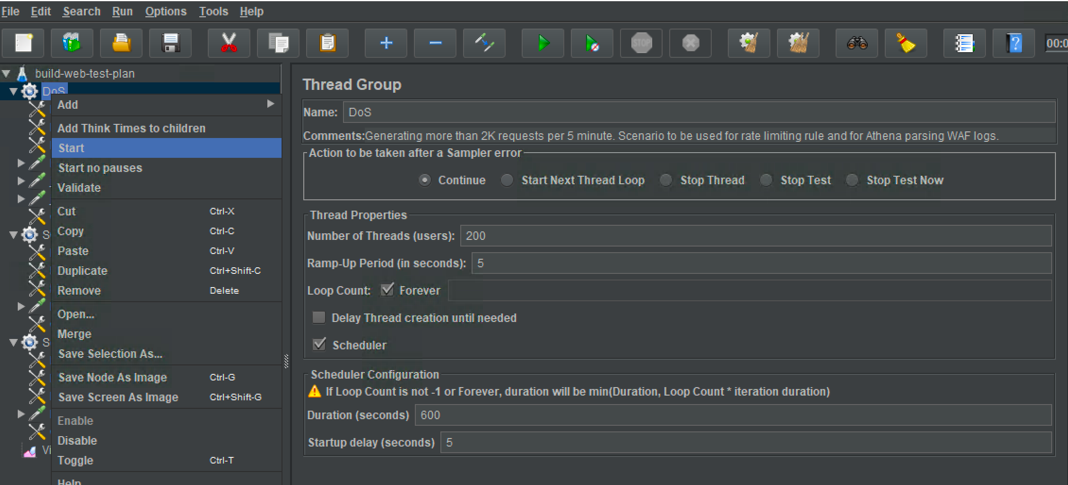
1. Go to File->Open menu and navigate to c:\JmeterScenarios.jmx file and click Open:



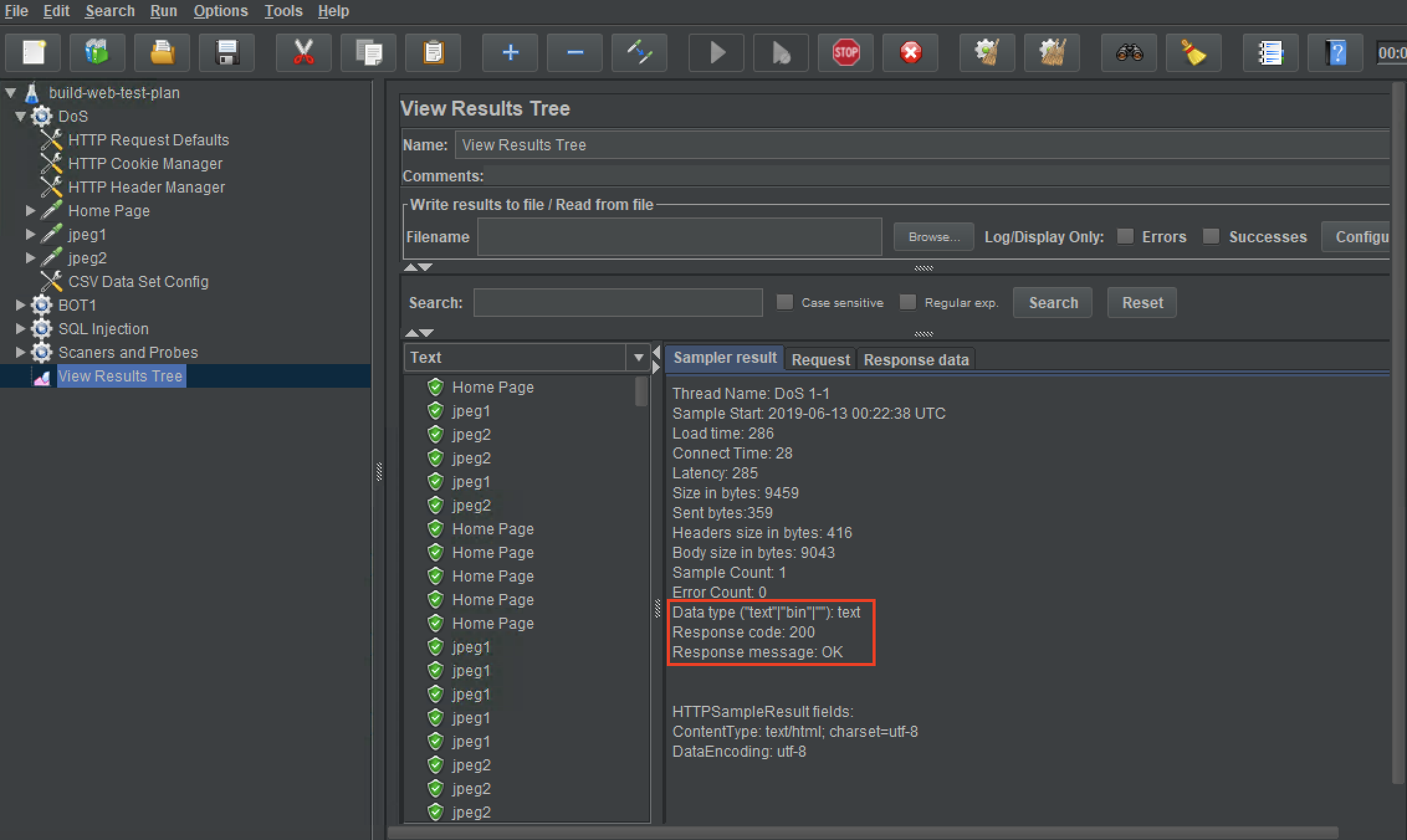
1. Click the most top “HTTP Requests Defaults” entry in the test plan menu. Update the “Server Name or IP” with the ALB endpoint saved in the p. 1 above. Enter the server name only without http:// and without the “/” suffux:



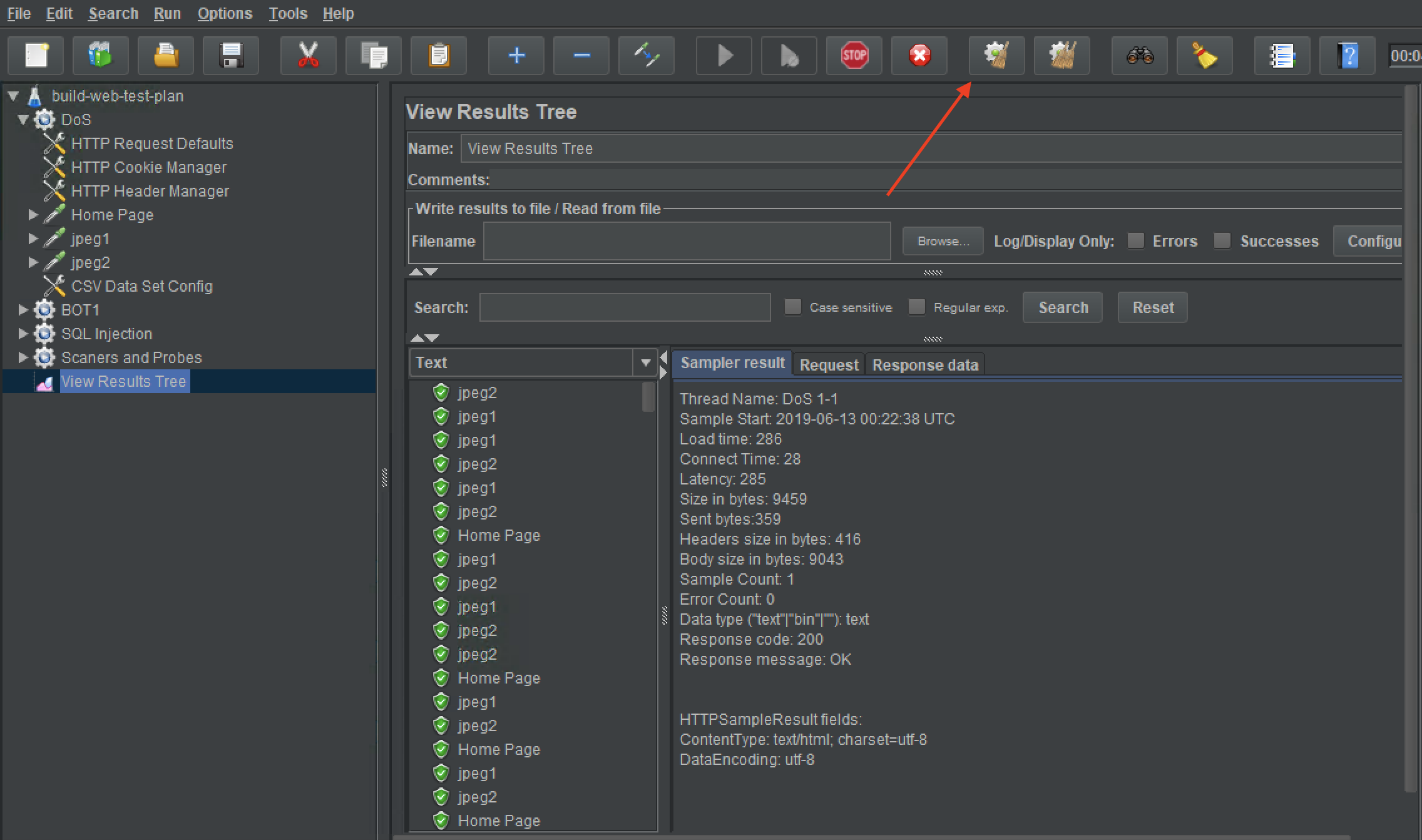
1. Save changes by navigating to File->Save menu.
2. Right click on the “DoS” scenario and then click “Start”:



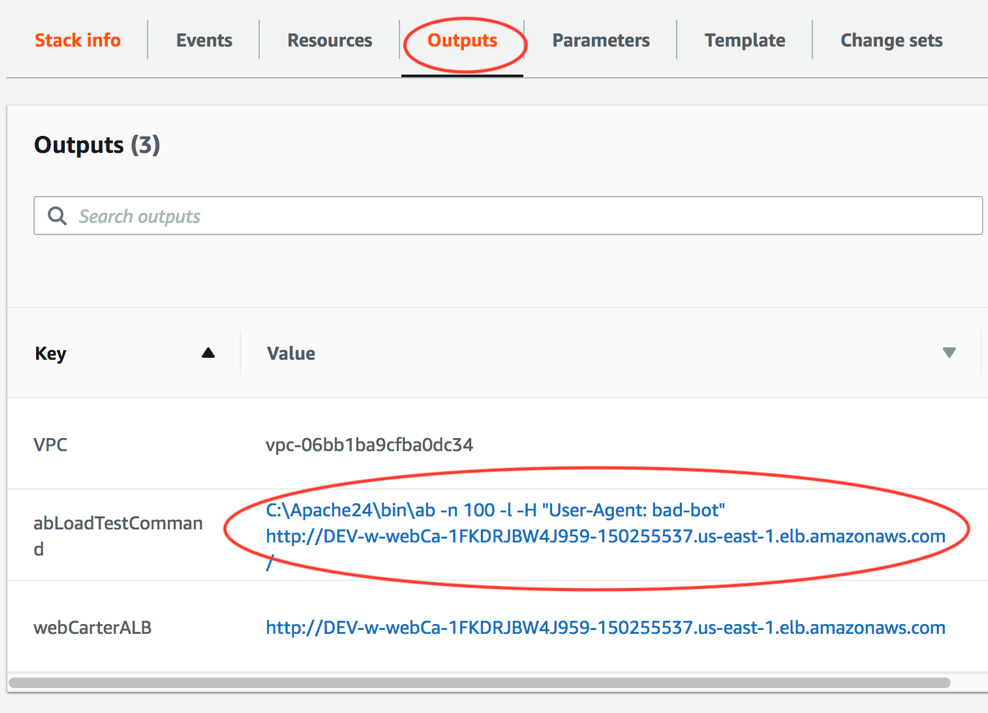
1. Navigate to “View Results Tree” and assure you’re getting 200 OK responses. Your attack was successful:



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.*

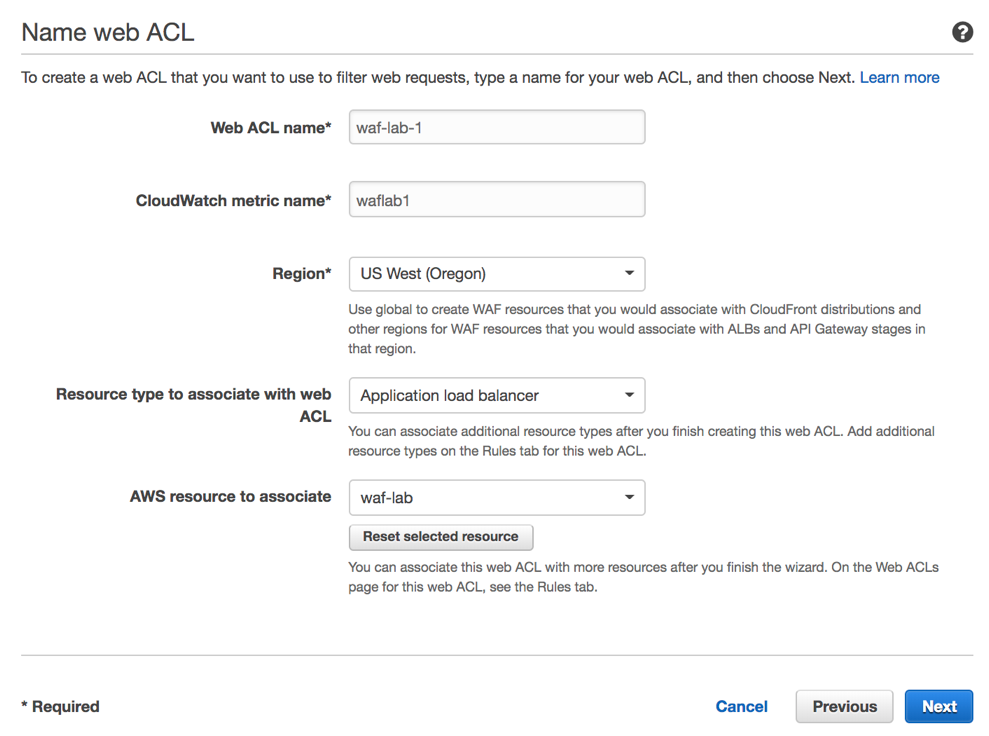
**Step 2: Configure protection against the above attacks using WAF native rules.**

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:

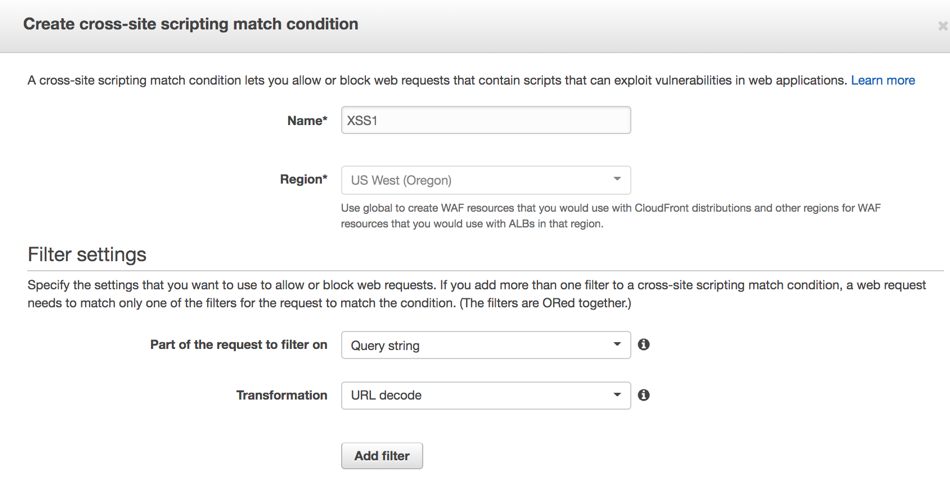


**Note**: WAF configuration is separate for each region and separate for “Global” deployment used for CloudFront deployments. You must configure your Web ACL in the same region where you have your ALB deployed.

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

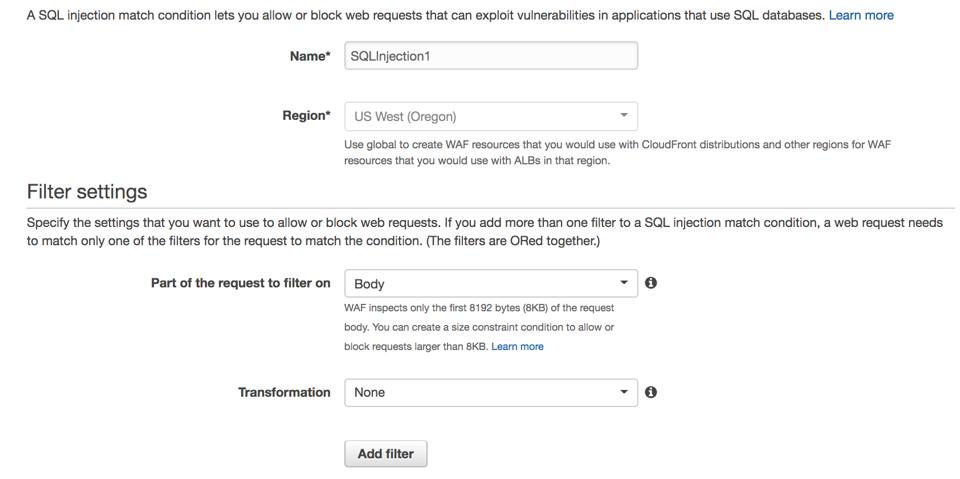


1. Create Cross-site scripting match condition. Click “Add Filter” then “Create”.

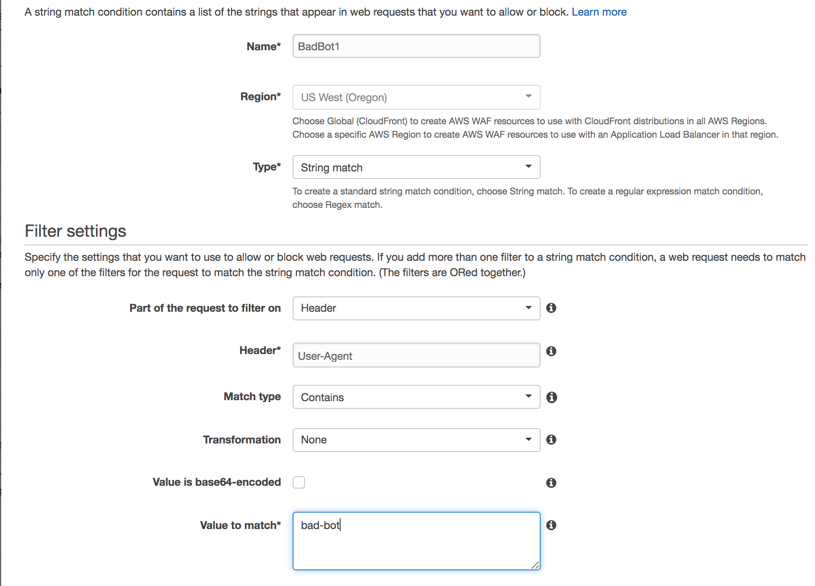


**Note**: Filters in the conditions are logically disjoined, i.e. “OR”. You can have up to 10 filters in each condition (hard limit).

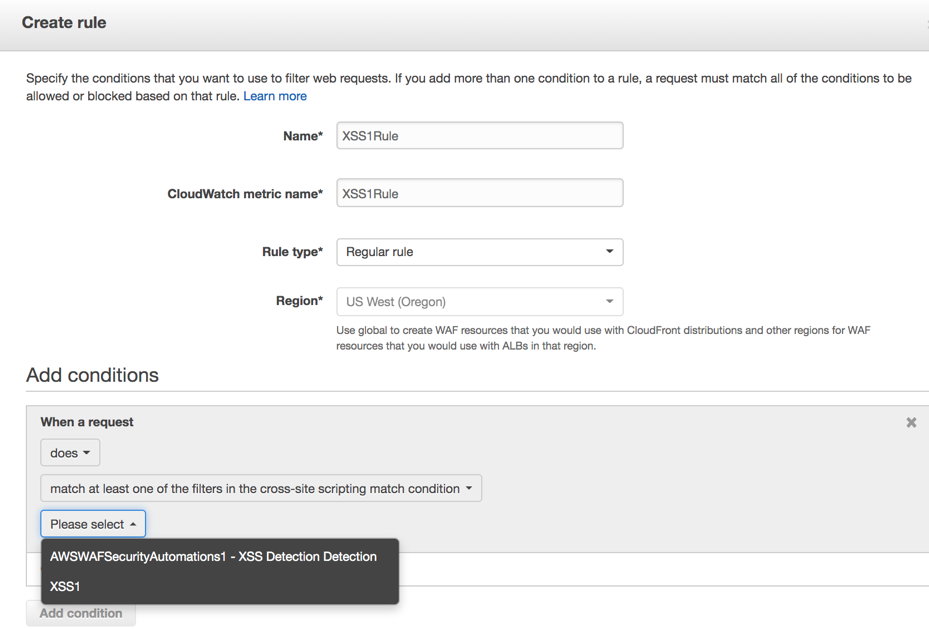
1. Create SQL injection match condition. Click “Add Filter” then “Create”.



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

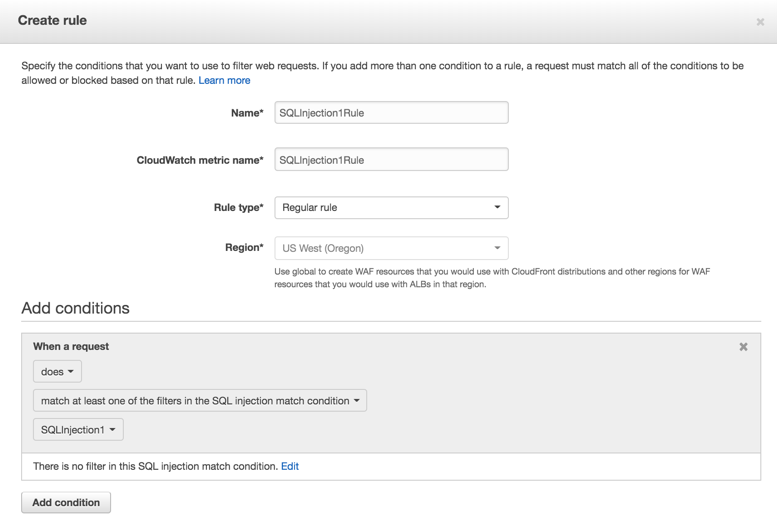


1. Click “Next” on the *Create Conditions* page.
2. Click “Create Rule” button.
3. Create a simple Cross-site scripting rule that in our example will contain a single XSS condition created above. Select the condition as below and click “Add Condition”:

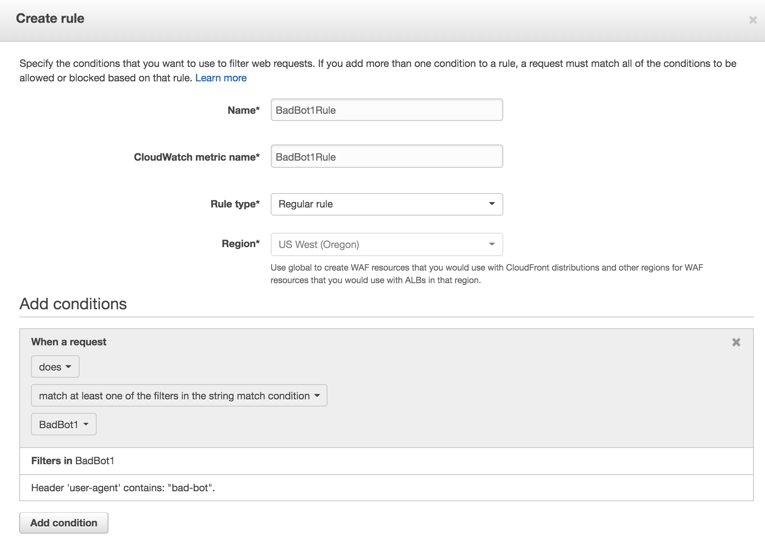


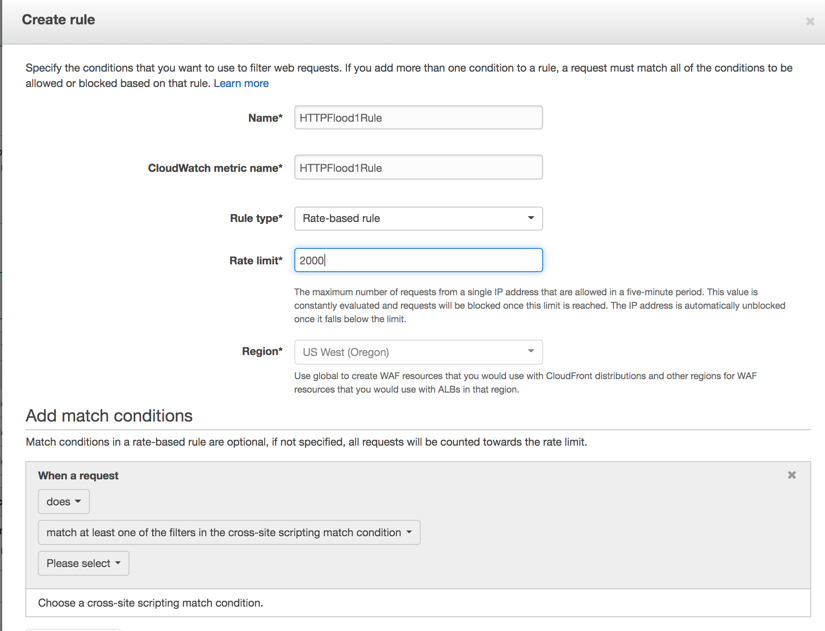
**Note**: Conditions in the rule are logically joined, i.e. “AND”. You can have up to 100 conditions of each type per account (soft limit).

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.

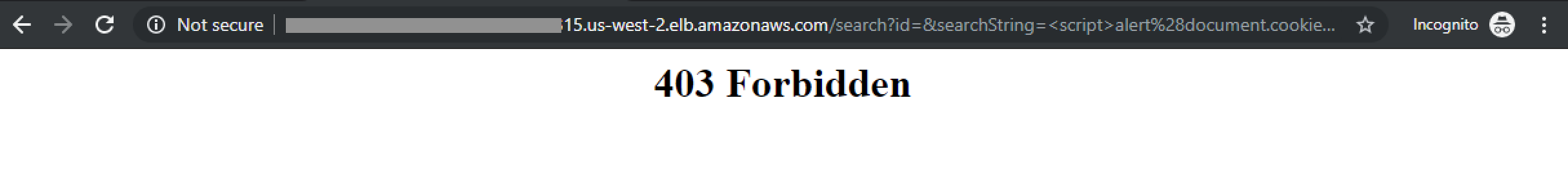
1. Set default Web ACL action as “Allow all requests that don't match any rules” and click “Review and create”.
2. Review the Web ACL and click “Confirm and create”.

**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.

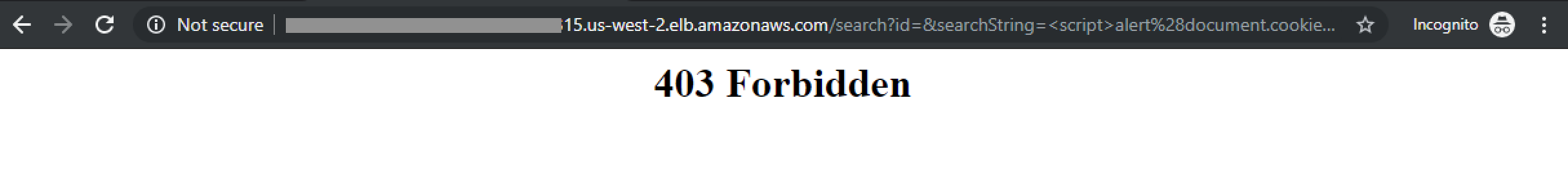
1. In a Chrome browser window navigate to your ALB endpoint and type <script>alert(document.cookie)</script> in the “Search” string. Then click “Search!”.
2. Assure you get HTTP 403 response from WAF:



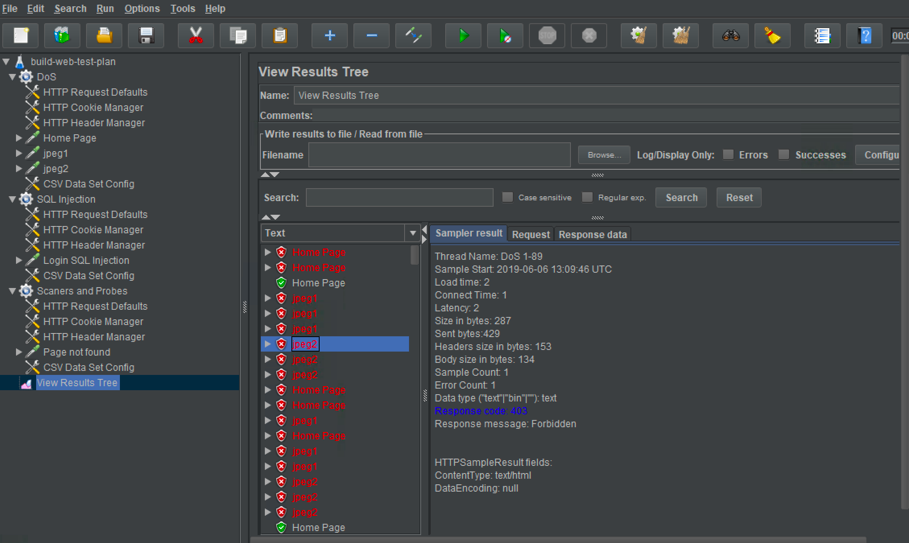
1. In the “Search” field enter the following:

*' 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:

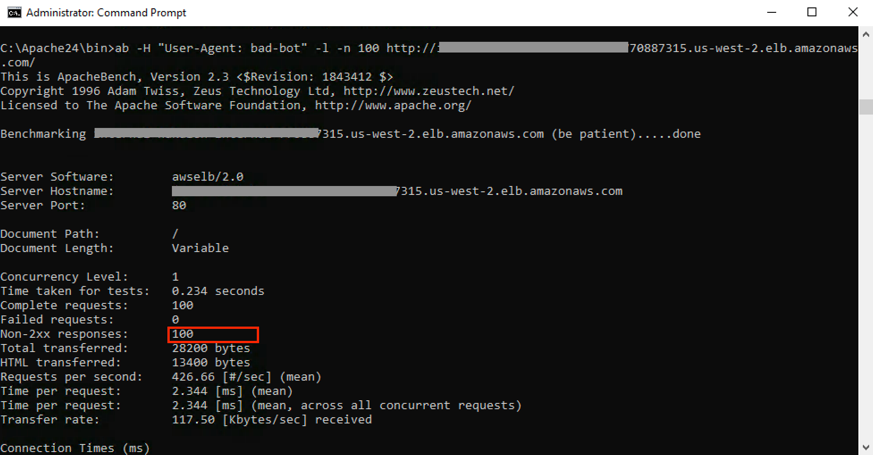


1. Go to AWS WAF Console and navigate to Rules-> HTTPFlood1Rule. Notice IP addresses that are currently blocked by this rule:



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.

# Lab 2 – Automating Protections and Tracking Attacks

**Step 1: Associate WebCarter ALB with WAF-Automation Web ACL.**

1. Go to AWS WAF & Shield console, choose the region you’re working at, and open “WebCarter-lab1” Web ACL and navigate to the “Rules” tab on the right.
2. In “AWS resources using this web ACL” delete association with WebCarter ALB.
3. Choose WAF-automation Web ACL and open “Rules” tab.
4. Click “Add association”, then choose “Application load balancer” as “Resource type” and choose your WebCarter load balancer. Click “Add”.

Now your WebCarter Web application protected with Web ACL created by WAF-automation template.

**Step 2: Configure Web ACL full logging functionality.**

1. Open “Logging” tab of the WAF-automation Web ACL and click “Edit”.
2. Choose Kinesis Data Firehose stream created by waf-automation CloudFormation template.
3. Don’t add “Redacted Fields” and click “Create”.

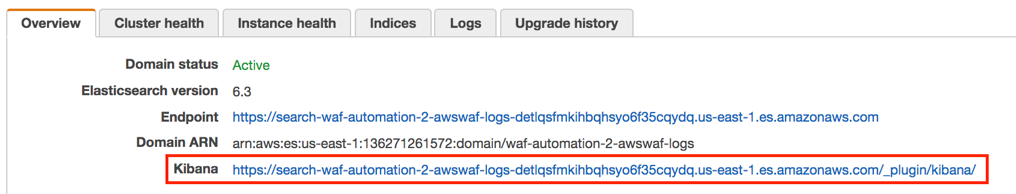
**Step 3: Configure ElasticSearch with Kibana**

WAF Automation stack created Kinesis Firehose stream pointing to ElasticSearch domain. You will configure the Web ACL logging stream JSON WAF logs to the Kinesis Data Firehose.

Kinesis Data Firehose also places a backup copy of the WAF to an S3 bucket for Amazon Athena analysis.

Below you will create indexes and visualizations in your domain.

1. Go to ElasticSearch console, choose the domain created by waf-automation template, and click on the Kibana link:



1. On the Kibana console click on “Get to work” and create an index using the following statement:

*PUT \_template/awswaf-logs*

*{*

*"index\_patterns": ["awswaf-\*"],*

*"settings": {*

*"number\_of\_shards": 1*

*},*

*"mappings": {*

*"waflog": {*

*"properties": {*

*"httpRequest": {*

*"properties": {*

*"clientIp": {*

*"type": "keyword",*

*"fields": {*

*"keyword": {*

*"type": "ip"*

*}*

*}*

*}*

*}*

*},*

*"timestamp": {*

*"type": "date",*

*"format": "epoch\_millis"*

*}*

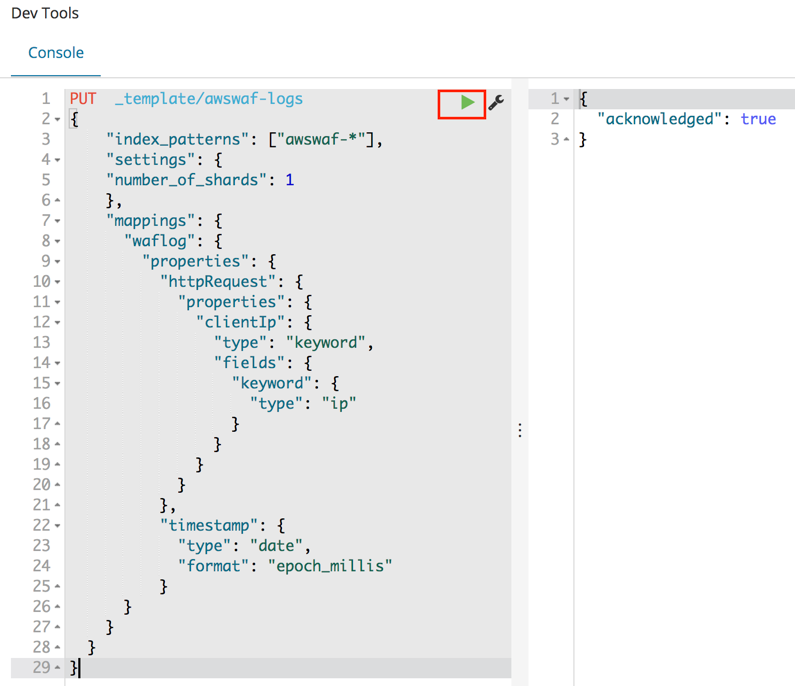
*}*

*}*

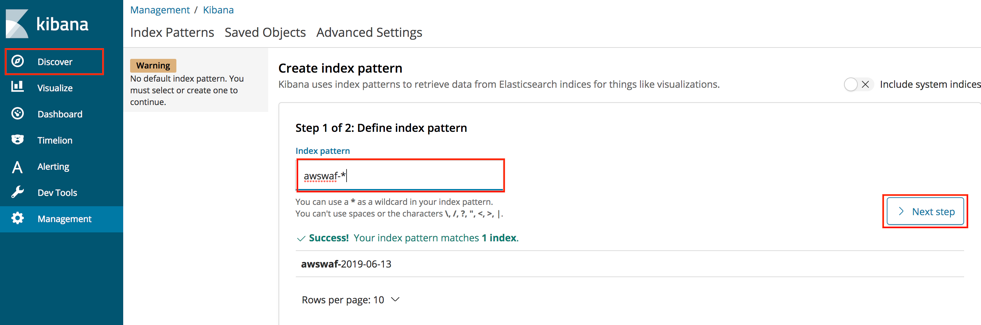
*}*

*}*

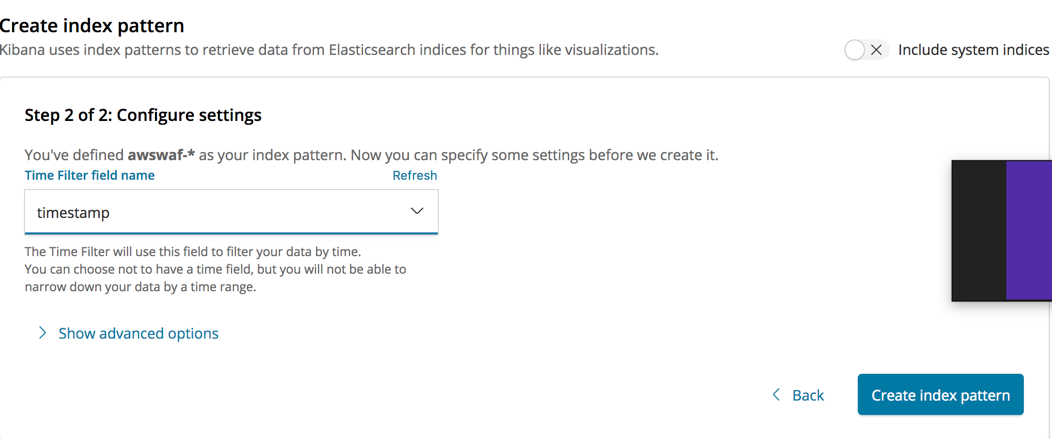
Paste this statement in Kibana console and click on the green “play” bottom to execute it:



1. Click on the “Discover” menu, define an index pattern as *awswaf-\** and click “Next step”:



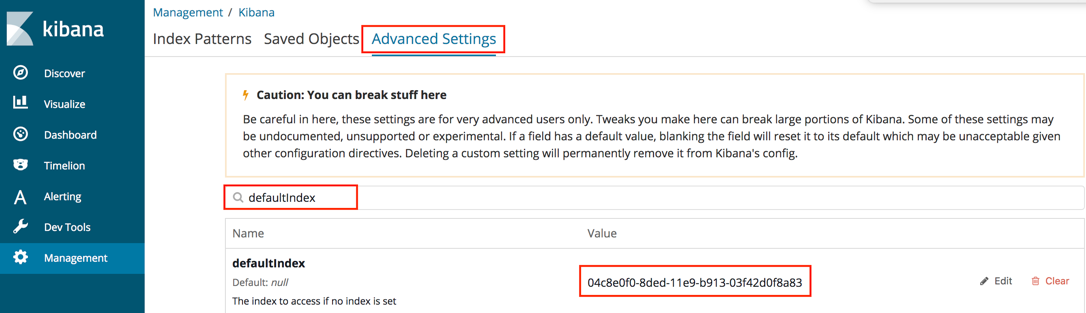
1. Choose timestamp as a Filter and click “Create index pattern”



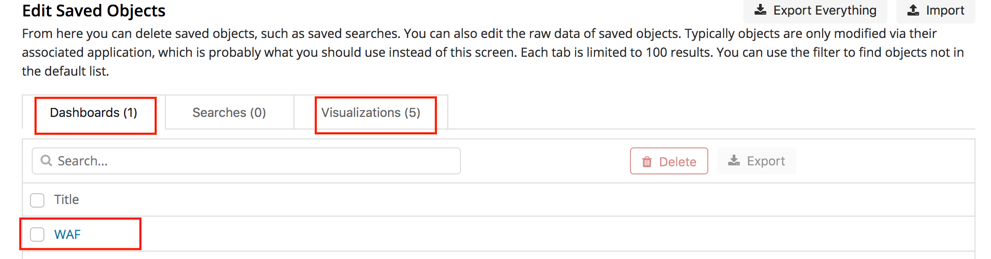
1. Now you’re able to run searches through your logs by going into the Discover tab in Kibana. For example, you can look for specific HTTP headers, query strings, or source IP addresses:

Now upload visualizations and a dashboard. You can always customize them or create your visualizations as required.

1. Go to “Management” menu and click “Advanced”. Search for the the defaultindex and copy and save the default index id value:



1. Open kibana-configruation.json file from the config directory on Github, and replace the *\"index\":\"d610d1f0-7136-11e9-a4df-61c3bd3f36bb\"* value with the one saved above. Please do it everywhere in the file.
2. Save the file.
3. Go to “Saved Objects” tab, click “Import” and upload the modified kibana-configruation.json file. Click “Yes” for the overwrite question.
4. You’ll see your dashboard and visualizations have been uploaded successfully:



1. Navigate to the “Dashboard” menu and choose “WAF” dashboard you just uploaded.

You cannot see any data yet: we’ll start generating traffic in the next steps.

**Step 4: Use ElasticSearch Service with Kibana for WAF monitoring and forensics**

1. Run JMeter BOT1 scenario on the Windows instance to generate data for this exercise.
2. Go to AWS Console and choose ElasticSearch service. Choose your waf-automation domain and click on the Kibana link.
3. Navigate to the “Dashboard” menu on the right and explore the dashboard graphs. Check the number of requests per minute per source IP address.
4. Go to AWS WAF & Shield console, choose “IP addresses” menu and check if IP addresses you see in the Kibana charts appear in any of the block lists.

**QUESTION**: Should the source IP addresses for BOT1 scenario appear in any of the block lists?

This scenario simulates a mix of legitimate traffic together with under the radar BOT traffic.

1. Go back to Kibana “Dashboard” GUI and scroll down to “Allow/block URIs”. Note that /login page is one of the top requested URIs.
2. Go to “Discover” menu and enter the following search filter:

*httpRequest.uri:/user/login*

1. Scroll through some of the requests and note that at glance, they’re coming from the same source IP address.
2. Enter the following search filter.

*!httpRequest.uri:/user/login && httpRequest.clientIp:10.192.30.50*

**QUESTION**: *Do these requests look legit? In the real-life scenario why can we see large number of malicious and legitimate requests coming from the same IP address?*

*Can we just block all requests coming from that IP address?*

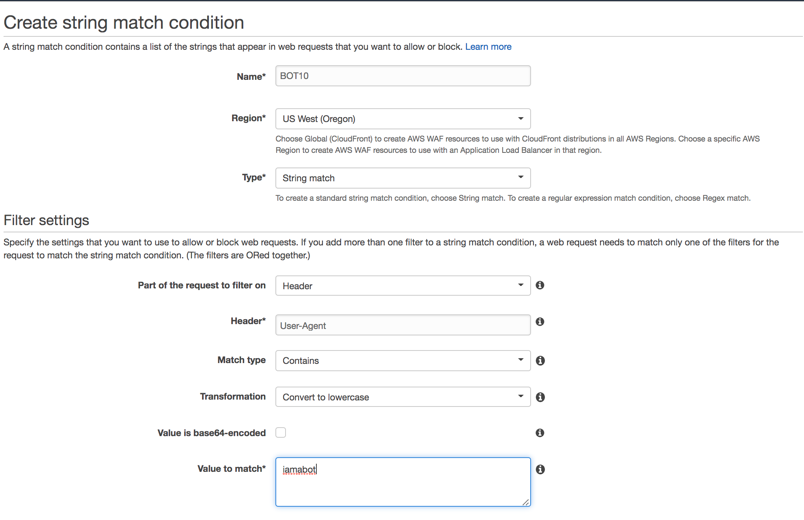
*Can we block all requests to this URI?*

1. Look at the User-Agent header value for the requests returned by the search below:

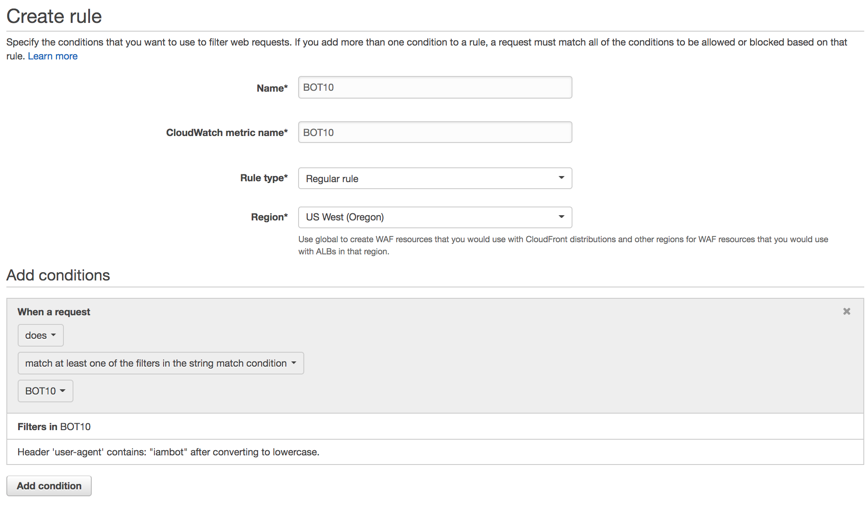
*httpRequest.uri:/user/login && httpRequest.clientIp:10.192.30.50*

Note that large number of requests has User-Agent value “IamaBOT”. We’ll create a WAF rule that blocks these requests below.

1. Go to AWS WAF & Shield console and navigate to “String and regex matching”.
2. Click “Create condition” and create BOT10 condition as following:



1. Click “Add filter” then “Create”.
2. Go to “Rules” menu and click “Create rule”.
3. Create BOT10 rule as following:



1. Click “Create”.
2. Navigate to “Web ACLs” menu and click on “WAF-Automation..” ACL.
3. Click “Rules” tab then “Edit Web ACL” button.
4. In the “Rules” drop-down list choose BOT10 rule and click “Add rule to Web ACL”.
5. Assure rule action set to “Block” and click “Update”.
6. Wait for couple for minutes and refresh Kibana dashboard. Check if you see blocked request to the login page.
7. Note the “Terminating Rule ID” in the “WAF Rules Hit” visualization and check the rule that block these requests. To do so, run the following AWS CLI command. Update the region and the rule id as required:

*aws waf-regional get-rule --region us-west-2 --rule-id d3458c9b-5bba-4361-a7ce-e28d66a86d27*

**QUESTION**: Does number of allowed requests to the login page look reasonable after we blocked a “IamABOT” BOT?

1. Enter the following filter and try to see if there is anything abnormal in the requests returned:

action:"ALLOW" && httpRequest.uri:"/user/login".

**QUESTION**: If we cannot find how to identify malicious requests and we cannot block this source IP entirely, what can be next steps of our investigation?

In the next steps we’ll look at ALB access logs using Amazon Athena.

1. Go to Amazon Athena AWS console.
2. Go to Query Editor and choose “awswafsecurityautomations..” as your Database.
3. Run the following SQL query:

SELECT \* from app\_access\_logs where request\_url like limit 1;

app\_access\_logs query points to the ALB access logs on the corresponding S3 bucket.

You can get an additional information about incoming requests from these logs, such as HTTP response code, SSL cypher and protocol, processing time. This information usually helps you to investigate incoming requests to your Web application.

1. Get familiar with the values returned.
2. Run the following SQL query:

*SELECT*

*client\_ip,*

*request\_url,*

*target\_status\_code,*

*COUNT(\*) as counter*

*FROM*

*app\_access\_logs*

*WHERE*

*parse\_datetime(time, 'yyyy-MM-dd''T''HH:mm:ss.SSSSSS''Z') > DATE\_ADD('minute', -5, NOW()) AND*

*target\_status\_code = ANY (VALUES '400', '401', '403', '404', '405', '500')*

*GROUP BY*

*client\_ip,*

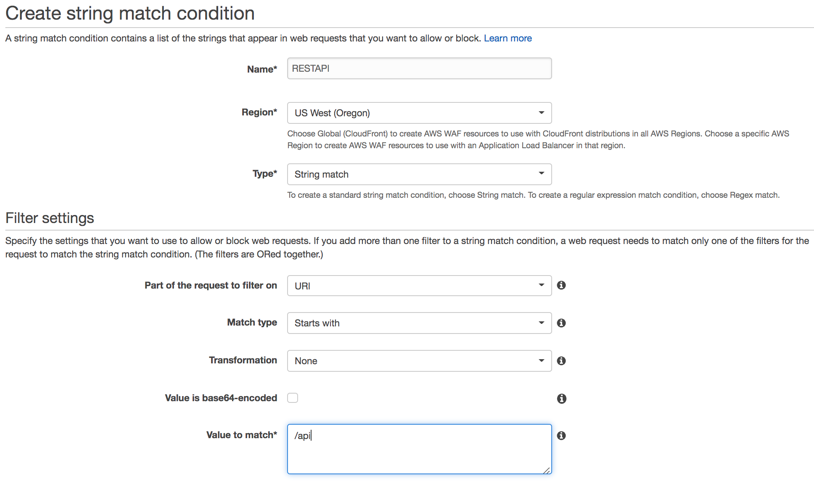
*request\_url,*

*date\_trunc('minute', parse\_datetime(time, 'yyyy-MM-dd''T''HH:mm:ss.SSSSSS''Z')),*

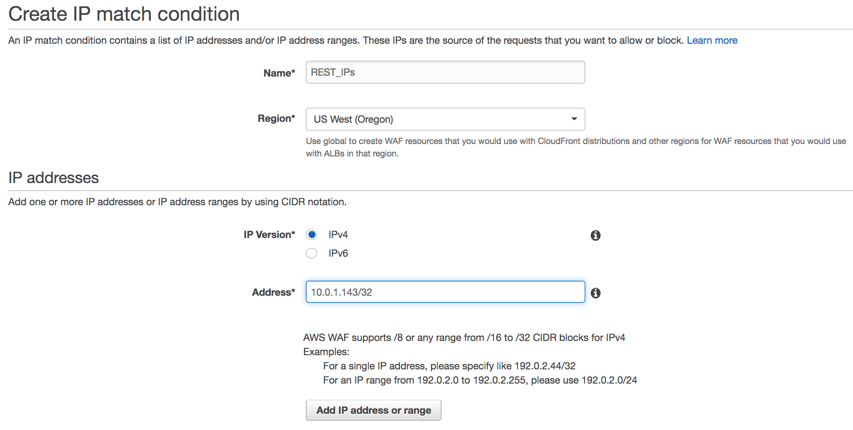
*target\_status\_code;*

As far we can see, there are malicious requests coming from the same IP address and targeting our Web appliucation REST API endpoint. We know that REST API shall be open only to our partners, and we decide quickly to block it for the IP address returned above.

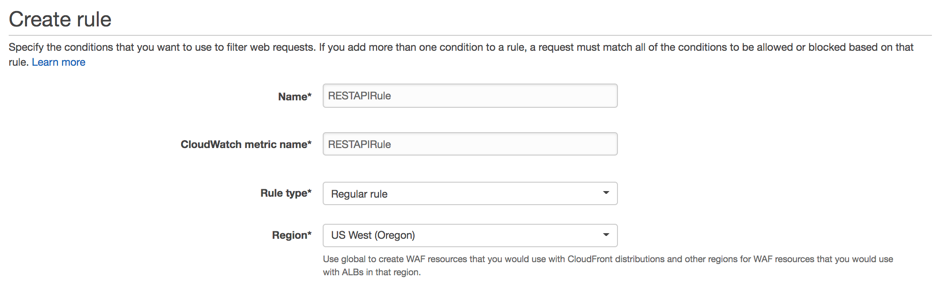
1. Go to AWS WAF & Shield console and navigate to “String and regex matching”.
2. Click “Create condition” and create RESTAPI condition as following:

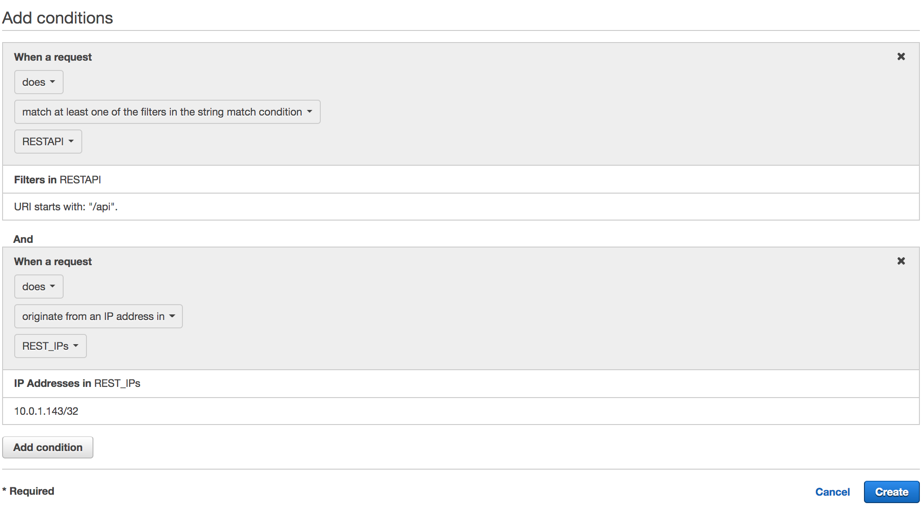


1. Click “Add filter” then “Create”.
2. Go to “IP addresses” and click “Create condition”:



1. Click “Add IP address…” then “Create”.
2. Go to the “Rules” menu and click “Create rule”:





Note we added two conditions in this rule.

1. Navigate to “Web ACLs” menu and click on “WAF-Automation..” ACL.
2. Click “Rules” tab then “Edit Web ACL” button.
3. In the “Rules” drop-down list choose RestAPIRule rule and click “Add rule to Web ACL”.
4. Assure rule action set to “Block” and click “Update”.
5. Wait for couple for minutes and refresh your Kibana dashboard. Check if you see blocked request to the api endpoint.
6. Go to Athena console and run the following query:

SELECT

client\_ip,

request\_url,

target\_status\_code,

COUNT(\*) as counter

FROM

app\_access\_logs

WHERE

parse\_datetime(time, 'yyyy-MM-dd''T''HH:mm:ss.SSSSSS''Z') > DATE\_ADD('minute', -5, NOW()) AND

elb\_status\_code = '403'

GROUP BY

client\_ip,

request\_url,

date\_trunc('minute', parse\_datetime(time, 'yyyy-MM-dd''T''HH:mm:ss.SSSSSS''Z')),

target\_status\_code

You can see that requests from that IP address to /api url been blocked by the WAF rule.

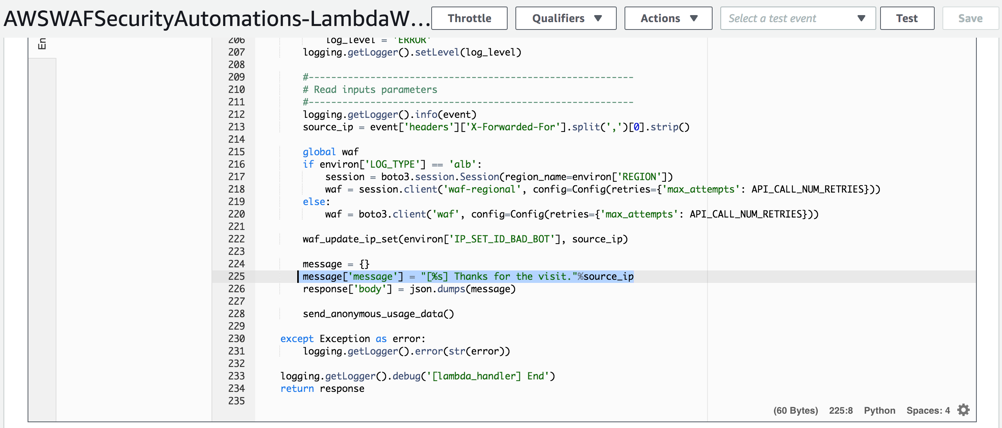
**Step 5: Honeypot for bad bots and scrapers.**

1. Open Chrome browser and navigate to the WebCarter home page.
2. Right click on the page and choose “Inspect”.
3. Verify the honeypot link is present on the page. This link has been inserted into your application by AWS Systems Manager, which was configured by WAF Automation CloudFormation stack.

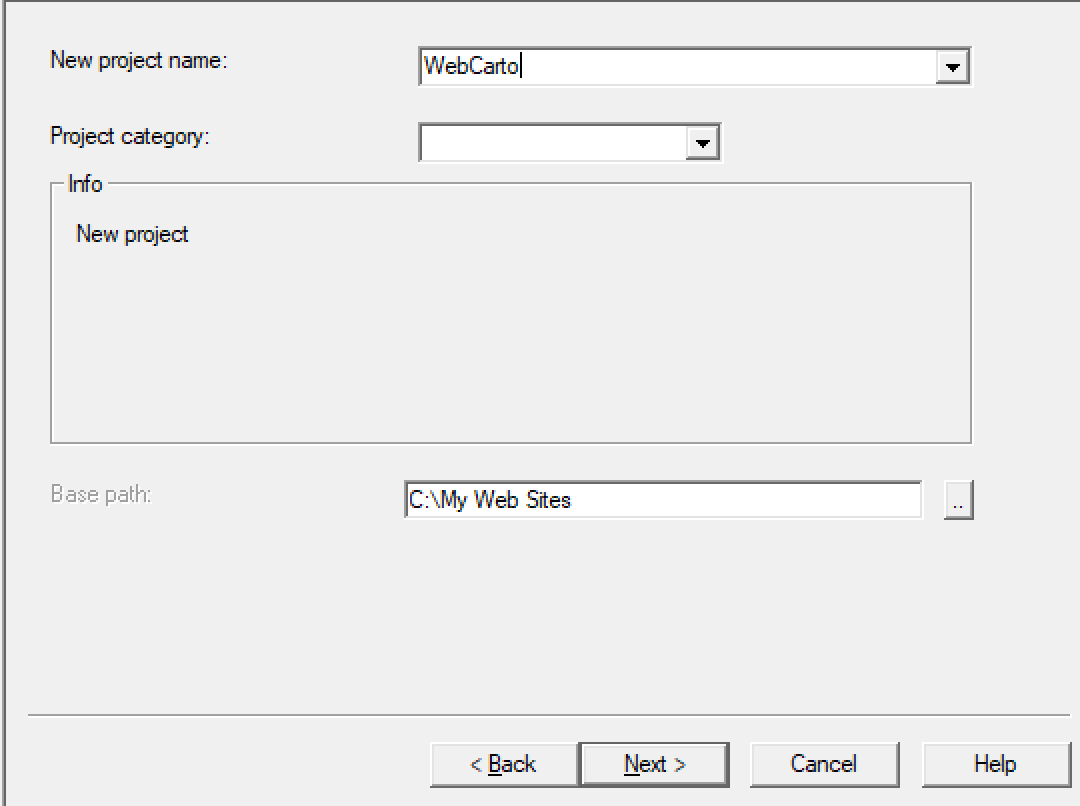


WAF Automation CloudFormation stack configured an API Gateway instance with Lambda function that parses the source IP address of the request and adds the source IP address to the black list.

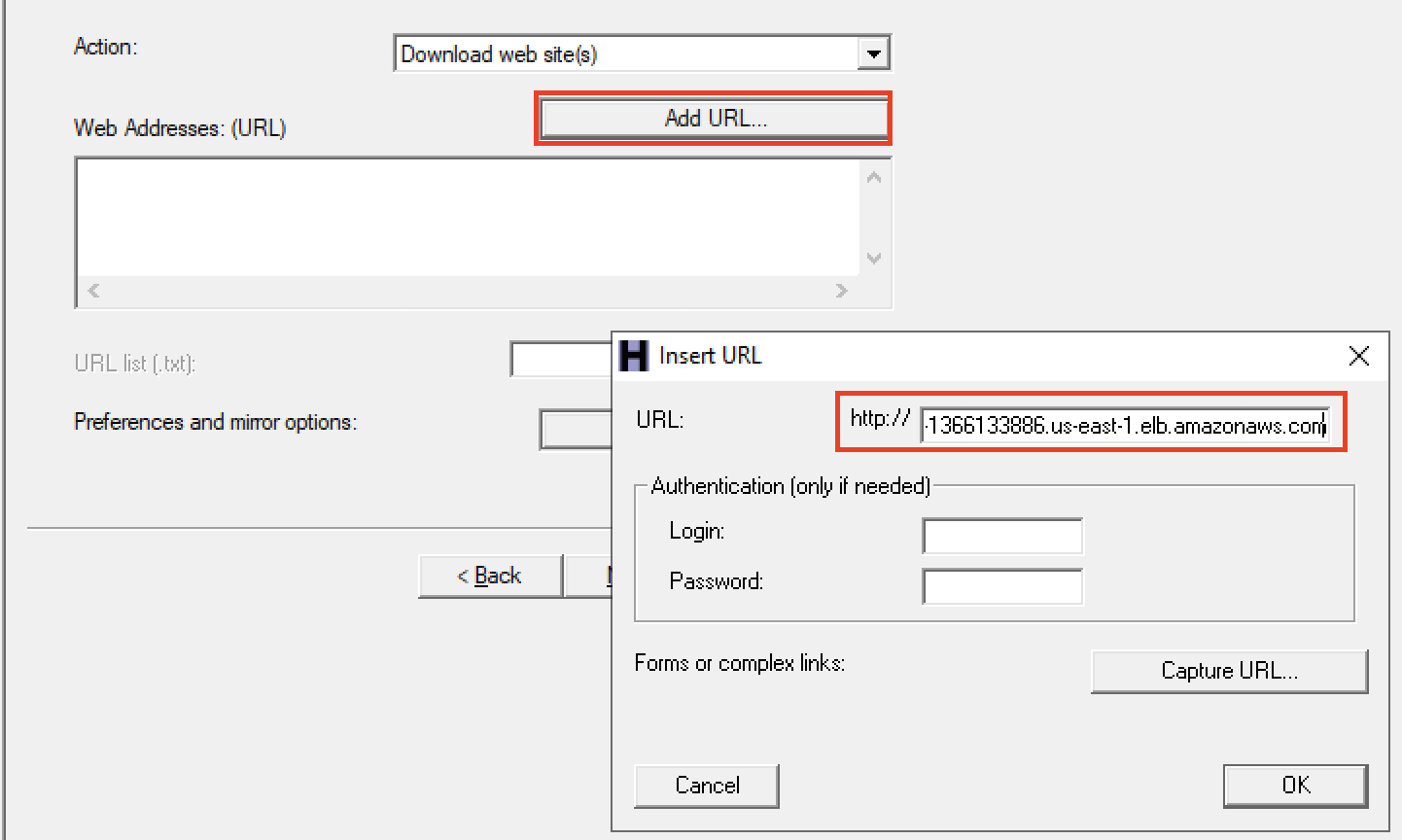
1. To review / customize the honeypot lambda function, go to AWS Console and choose API Gateway service.
2. Choose “Security Automation – WAF..” API and click “ANY”.
3. Click on the Lambda function on the right if you’d like to customize this function:



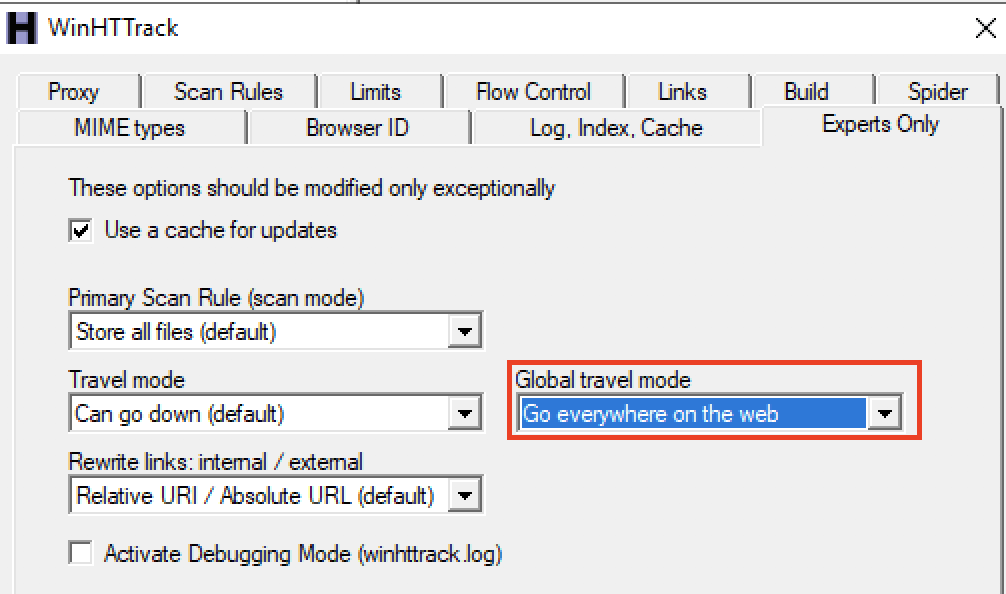
1. Run HTTTRack web copier on your Windows instance.
2. Click “Next” on the Welcome page and enter your project name (WebCarter for example):



1. Click “Add URL”, enter the ALB endpoint and click “OK”:



1. “Set options” button and go to “Experts Only” tab. Change “Global travel mode” to “Go everywhere…”. Due to out honeypot is on another domain, this will allow us to simulate a bad bot hitting the “no-follow” hidden honeypot link above:



1. Click “Next” then “Finish”.
2. You can cancel the test after you see the site been downloading. It was enough time for the copier to follow the honeypot URL.
3. Navigate to WAF& Shield AWS Console, “IP Addresses” and choose “.. Bad Bot Set”. Assure you see your external IP address in the list.

**Note**: In this simulation HTTTrack is accessing an ALB using 10.x.x.x IP address, while accessing API GW honeypot endpoint using public IP address. Due to this simulation network topology, we can see our public source IP address been added to the block list by the Lambda function, although traffic from HTTTrack is not been blocked. In the real-life scenario scraper will be blocked based on its public IP address.

**Step 3: WAF Automation using AWS Lambda and Amazon Athena.**

In the above steps we demonstrated manual creation of the WAF rules using information we’ve got from Kibana dashboard and Athena SQL queries. Below we’ll work with automated protection configured by WAF Automation CloudFormation template.

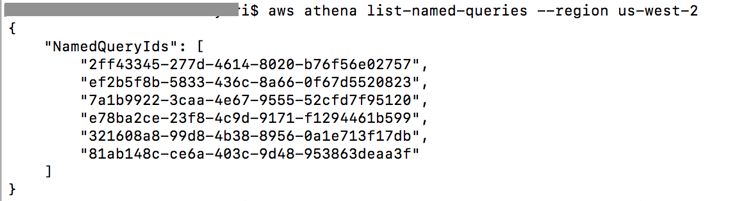
1. Go to WAF & Shield AWS Console, IP Addresses and click on “.. Scanners & Probes Set”.
2. Assure the list is currently empty.
3. Go to Amazon Athena console and navigate to “Saved Queries” tab.
4. Click on “ScannersProbesLogParser” query.
5. For our exercise, change the time window from 240 minutes to 5 minutes:

*DATE\_ADD('minute', -5, NOW())*

Now this modified query will be looking for any IP address that got >=50 of the 4xx responses from the application within last 5 minutes.

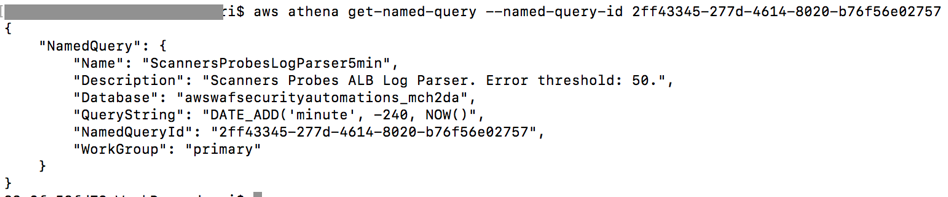
1. Click “Save as” and save the modified function with the new name “ScannersProbesLogParser5min”.
2. In AWS CLI run the following command updating the region as required (here and in all commands below):

*aws athena list-named-queries --region us-west-2*

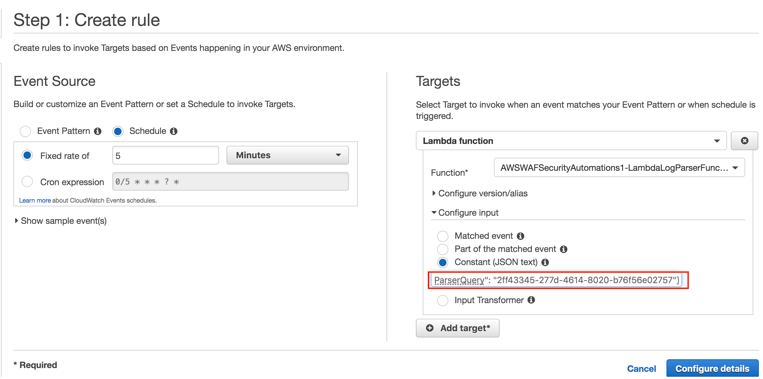


1. Copy the id of the first query in the list and run the following command with that query id:

aws athena get-named-query *--region us-west-2* --named-query-id 2ff43345-277d-4614-8020-b76f56e02757



1. Check the query name and assure this is the query we just saved on the console.
2. Go to AWS Lambda console and search for “[LambdaLogParserFunction](https://us-west-2.console.aws.amazon.com/lambda/home?region=us-west-2" \l "/functions/AWSWAFSecurityAutomations1-LambdaLogParserFunction-OF5T61R0EN0C)”.
3. Open “LambdaLogParserFunction” and click on the “CloudWatch Events”.
4. Click on “…LambdaAthenaAppLogParser…” one.
5. In the CloudWatch window click “Action->Edit” on the top right corner.
6. In the JSON input definition scroll right and replace the query id with the one you saved above:



1. Click “Configure details” and then “Update rule”.
2. Run “Scanners and Probes” scenario on JMeter.
3. Go to the results tree in JMeter and verify that application returns 404 “Not found” response.
4. After about 5 minutes go to AWS WAF & Shield console, navigate to “IP addresses” and click on “…Scanners & Probes Set”. Verify that your source IP address appeared in the blocking list.
5. Go to JMeter results tree, clear the results and check that currently JMeter is getting HTTP 403 “Forbidden” response from WAF.
6. Stop JMeter test and check that in about 5 minutes your source IP address will disappear from the “…Scanners & Probes Set” IP list.

In this scenario we used automated detection of probes and scanners, that are getting 4xx responses from the application and stay undetected by the rate-limiting rules.