Advanced WAF Lab

Participant Guide

Last Updated: 11/05/2020

Table of Contents

Lab Environment and Setup

- **Please read all steps for each exercise before starting each one. Some exercises require new actions to be performed while others are running**
- **Use Chrome to manage the BIGP-IP**
- **Use Firefox for all application interactions and Chrome and Internet Explorer when required.**

Login Credentials

BIG-IP (version 15.1.0): https://10.1.1.8 Username: admin, password: admin

Windows Machine: Username: admin, password: admin

Lab 1 – Protecting Credentials with F5 DataSafe

The purpose of this lab is to show the new DataSafe perpetual license in 13.1 and above (also part of Advanced WAF license). You will review the login page with and without DataSafe protections. You will enable and test encryption, obfuscation, and decoy fields.

Exercise 1 – Review and Attack the Login Page

Task 1 – Review Form Fields with the Developer Tools

- Connect to the **Windows Client** (win-client) via RDP (Select an appropriate screen resolution for your screen) ensuirng that you login with username/password as **admin/admin** (change user from default Administrator if required on the logon prompt screen).
- Once connected to the Windows client, open Firefox and access http://hackazon.f5demo.com/user/login
- Right-click inside the **Username or Email** field and select **Inspect Element**. The developer tools window will open.

Question:

What is the **name** value for this field?____username_ _____

- Right-click inside the **Password** field and select **Inspect Element**.

Question:

What is the **name** value for this field?_____password _____

FOOD FOR THOUGHT: How difficult would it be for malware to know which fields to grab to steal credentials from this page? How difficult would it be for an attacker to stuff credentials into these fields? They could simply put the stolen username into the "username" field and the stolen password in the "password" field.

Task 2 – Review Methods for Stealing Credentials

- From the Windows client, in **Firefox** click the **FPS Demo Tools** Bookmark, without opening a new tab

This includes tools that behave like real malware

- On the login page of the Hackazon website enter your first name and **P@ssw0rd!** as password but do not click **Sign In**
- From the **Demo Tools** click **Steal Password** and then click on the password field

The "malware" is using JavaScript to grab the value of the password field out of the DOM (Document Object Model) even before the user submits it to the application.

- Click **OK** then clear the password you entered

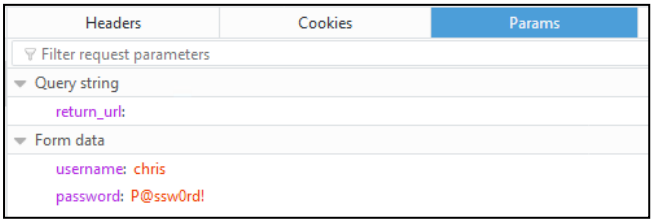
- From the **Demo Tools** click **Start Keylogger** and then enter the same password as earlier
- Watch the top of the Demo Tools

The "malware" is using JavaScript to log the password as it is typed. It could also send this capture data to some malicious site.

- In the developer tools window that opened previously, select the **Network** tab (F12), then click the trash can icon to delete the requests.
- On the login page enter your first name as username and **P@ssw0rd!** as password and click **Sign In**.

NOTE: Your login will fail, but your credentials were still sent to the web server.

- In the **Network** tab select the **/login?return_url=** entry, and then examine the **Params** tab.



The user's credentials are visible in clear text.

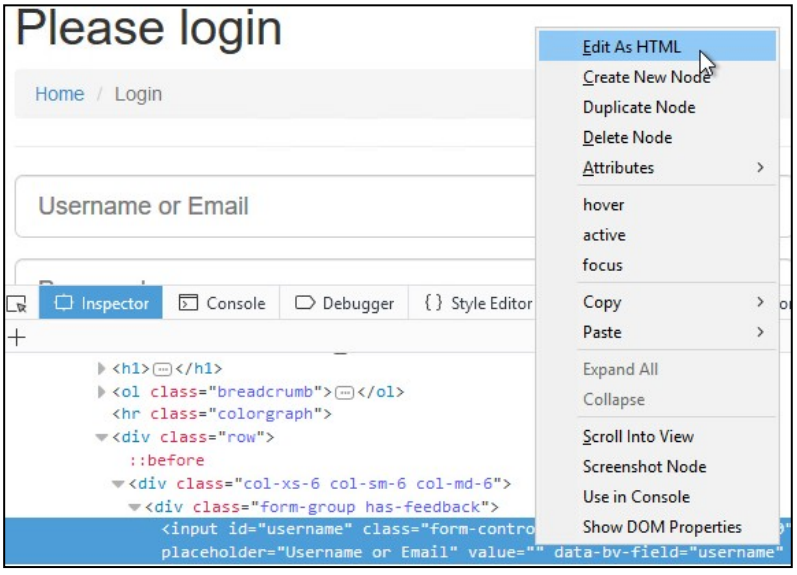
This is another way that malware can steal credentials. By "grabbing" the POST request and any data sent with it, including username and password.

Task 3 – Perform a Form Field "Web Inject"

- Return to the http://hackazon.f5demo.com/user/login page.

NOTE: It should NOT have **?return_url=** at the end of the URL in the address bar.

- Right-click inside the **Username or Email** field and select **Inspect Element** again.
- Right-click on the blue highlighted text in the developer tools window that opens and select **Edit as HTML**.



- Select all the text in the window and type **Ctrl+C** to copy the text.

- Click after the end of **data-bv-field="username">** and type **
, and then press the **Enter key twice.
- Type **Ctrl+V** to paste the copied text.

```
<input type="text" maxlength="100" required="" name="username" class="form-control input-lg" id="username" placeholder="Username or Email" value="" data-bv-
field="username"><br>

<input type="text" maxlength="100" required="" name="username" class="form-control input-lg" id="username" placeholder="Username or Email" value="" data-bv-field="username">
```

- For the new pasted entry, change the **name**, **id**, and **data-by-field** values to **mobile**, and change the **placeholder** value to **Mobile Phone Number**.

```
<input type="text" maxlength="100" required="" name="mobile" class="form-control input-lg" id="mobile" placeholder="Mobile Phone Number" value="" data-bv-field="mobile">
```

- Click outside of the edit box and examine the Hackazon login page.

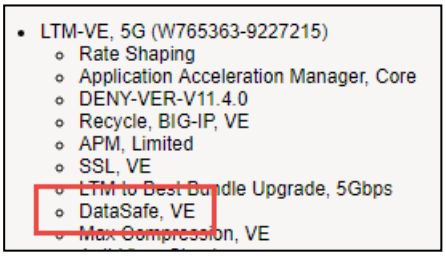
This is an example of the type of “web injects” that malware can perform to collect additional information. This same technique could be used to remove text or form fields. Note that this was done on the client side, in the browser, without any requests being sent to the server. The web application and any security infrastructure protecting it would have no idea this is happening in the browser.

- Close Firefox.

Exercise 2 – Review and Configure DataSafe Components

Task 1 – DataSafe Licensing and Provisioning

- In the Configuration Utility of the BIG-IP (connect via the TMUI access method), open the **System > License** page.

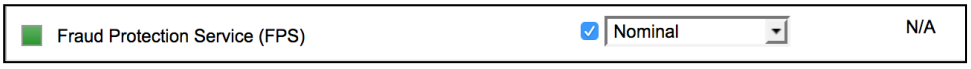


DataSafe includes only the Application Layer Encryption (ALE) module of WebSafe. Unlike WebSafe, DataSafe is licensed perpetually per device, just like ASM, APM, or any other licensed module. DataSafe is NOT included in the Best Bundle.

- Open the System > Resource Provisioning page.

When DataSafe is licensed, **Fraud Protection Service (FPS)** will display as **Licensed**.

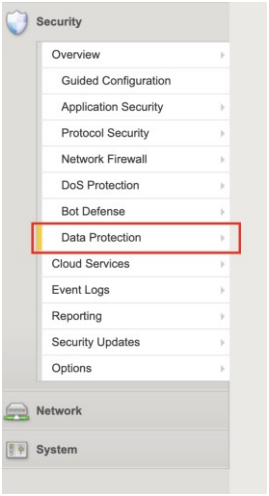
This is different than WebSafe, where Fraud Protection Services will show up as N/A.



- Expand the **Security** menu.

There is a **Data Protection** option. This is different than WebSafe where this menu option is **Fraud Protection Service**.

DataSafe



Task 2 – DataSafe Configuration

- Open the Security > Data Protection > DataSafe Profiles page on the BIG-IP and click Create.
- For Profile Name enter **Hackazon-DS**.

  Note If the '**Hackazon-DS'** profile already exists, please delete and follow instructions here.
- For **Local Syslog Publisher**, select **local-datasafe** (select the checkbox on the right side to enable this field's configuration)

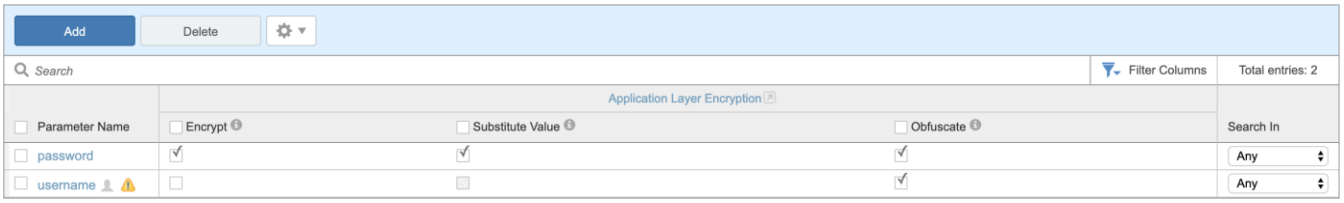
Optional: The local-datasafe Publisher can be viewed at System -> Logs -> Configuration -> Log Publishers

- Click in **Advanced** and review all other options

Data Safe will serve different Javascript files under those configured HTTP paths

- On the left menu click **URL List**, and then click **Add URL**.

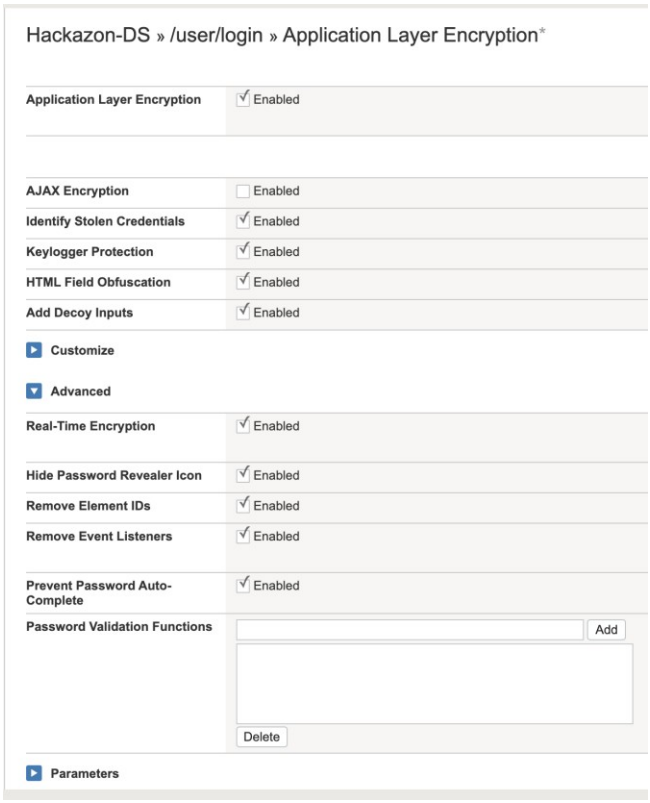
- For **URL Path** leave **Explicit** selected, and type **/user/login**.
- Click in **Advanced** and review all other options
  - Various configurations refer to where Data Safe will inject its Javascript
- From the left panel open the **Parameters** page.
  - Remember from earlier you found that the username and password parameter names are **username** and **password**.
- Click **Add**, enter a new parameter named **username**, select **Identify as Username** and then click **Repeat**.
- Create a second parameter named **password**, and then click **Create.**
- For the **username** parameter select the **Obfuscation** checkbox.
- For the **password** parameter select the **Encrypt**, **Substitute Value**, and **Obfuscate** checkboxes.



- From the left menu open the **Application Layer Encryption** page.

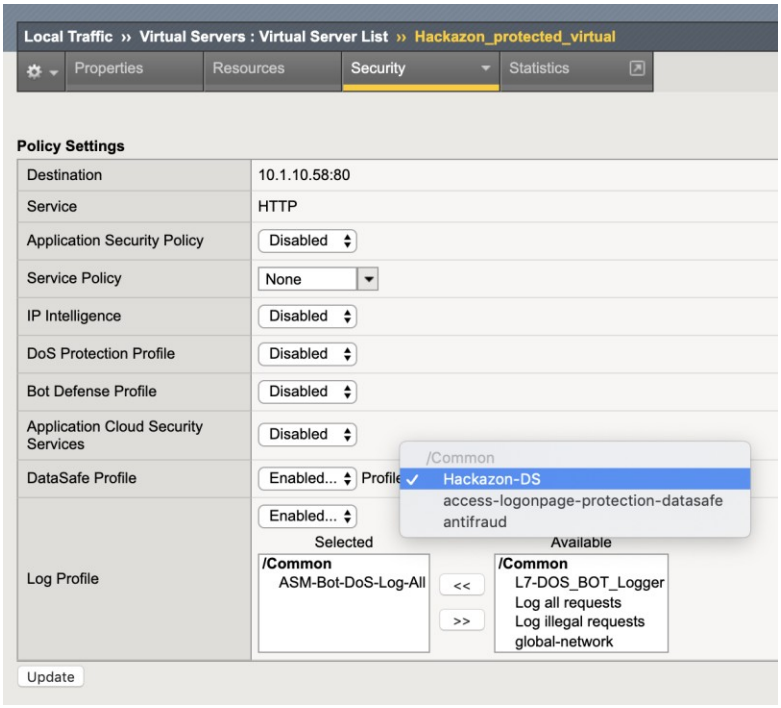
Notice that most features are enabled by default.

- Review the explanations for the different features.
- Select the **Add Decoy Inputs** checkbox
- Expand the **Advanced** section and select **Remove Element IDs** checkbox, and then click **Save**.



- Click **Save** to save the new profile
- Navigate to **Security ›› Event Logs : Logging Profiles** and select the 'ASM-Bot-DoS-Log-All' log profile.
- Ensure **Data Protection** is enabled.
- Once enabled, click on the **Data Protection** tab and ensure the '**local-datasafe'** is selected from the dropdown of the **Publisher** section.
- Enable **Login Attempt** and select the **default** template.
- Navigate to **Local Traffic ›› Virtual Servers ›› Virtual Server List** page and click **Hackazon_protected_virtual**, and then open the virtual server **Security > Policies** page.
- From the **DataSafe** Profile list select Enabled.
- From the adjacent **Profile** list box that appears, select **Hackazon-DS**, and then click **Update**.

  Note The 'ASM-Bot-DoS-Log-All' log profile will be applied already.



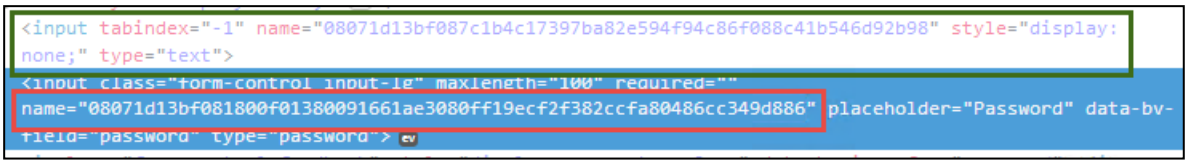
Exercise 3 – Testing DataSafe Protection

Task 1 – Review the Protected Hackazon Login Page

- From your Windows client, open a **private** Firefox window and access http://hackazon.f5demo.com/user/login.
- Right-click inside the **Password** field and select **Inspect Element**.

Question:

What is the **name** value for this field? _____



Obfuscation - Notice that the name of the password field (outlined in red) is now a long cryptic name and is changing every second. The same is true of the username field. Perform the same for the username field.

Add Decoy Inputs – Notice that there are other random inputs being added (outlined in green). The number and order of these inputs is changing frequently.

FOOD FOR THOUGHT: Considering this obfuscation, do you think DataSafe could protect the login page from a credential stuffing or a regular brute force?

- In the developer tools window select the **Network** tab, then click the trash can icon to delete any current requests.

- On the login page enter your first name as username and **P@ssw0rd!** as password and click **Sign In**.
- In the **Network** tab select the **/login?return_url=** entry, and then examine the **Request** tab.

Questions:

What parameters were submitted?
_____Random_____

Do you see a username or password field?_____Not really_____

Do you see the username you submitted?_____Yes_____

Obfuscation – DataSafe obfuscates the names of the parameters when they are submitted in a login request.

Encryption – DataSafe encrypted the value of the password field so that it is not a readable value in the login request.

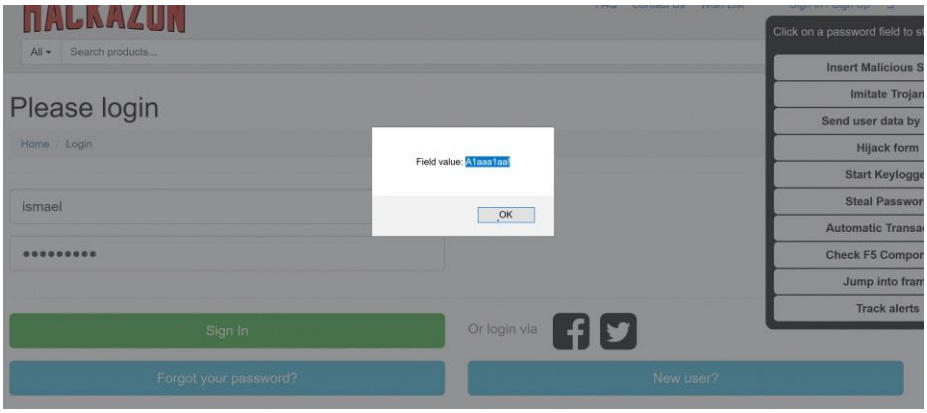
Exercise 4 – Reviewing Credential Stealing

Task 1 – Performing a Credential Stealing attack

- From the Windows client, open a **private** Firefox window (CTRL+SHIFT+P) and access http://hackazon.f5demo.com/user/login.
- In Firefox click the **FPS Demo Tools** Bookmark in the same tab where the hackazon page is loaded

This includes tools that behave like real malware

- On the login page enter your first name and **P@ssw0rd!** as password but **do not** click **Sign In**
- From the **Demo Tools** click **Steal Password** and then click on the password field
- Copy the password value from the Alert Box



- Click **OK** then click on **Sign-in**
- Open a **incognito** Chrome window and access http://hackazon.f5demo.com/user/login.

- On the login page enter your first name and paste the value you copied from the previous step. Click on **Sign In.** Repeat the login step by navigating back to http://hackazon.f5demo.com/user/login as if the attacker did not succeeded to authenticate and they are trying it again.
- On the BIG-IP, open the **System > Logs > Data Protection** and examine the entries
- What happened?

FOOD FOR THOUGHT: In the scenario above, what could the Customer do regarding to what happened?

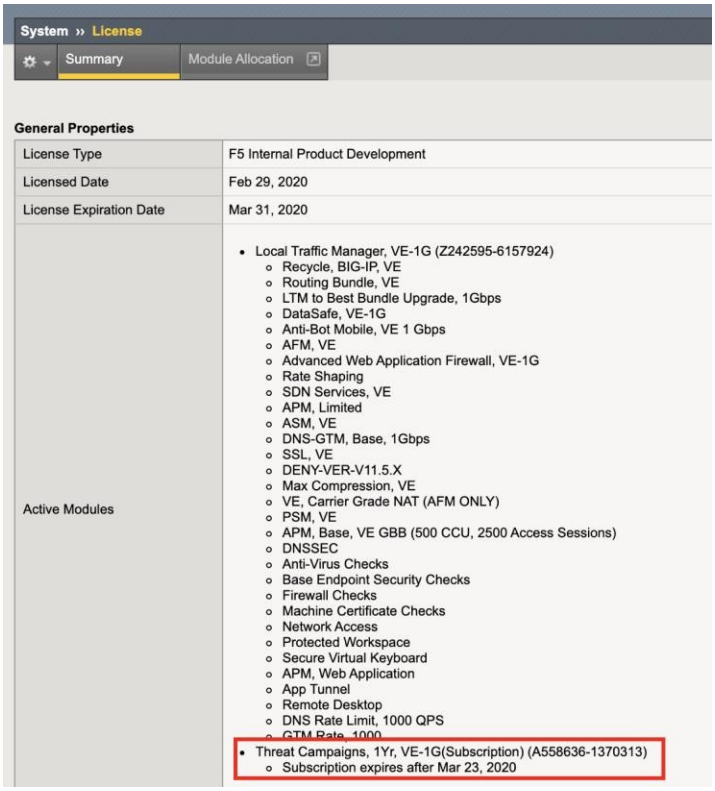
Lab 2 – Threat Campaigns

The purpose of this lab is to present Threat Campaings functionality present on Advanced WAF in 15.0 and above. You will review Threat Campaigns available, configure them in a BIG-IP ASM policy and perform a few attacks using Postman tool with malicious payload.

Exercise 1 – Review and Configure Threat Campaigns

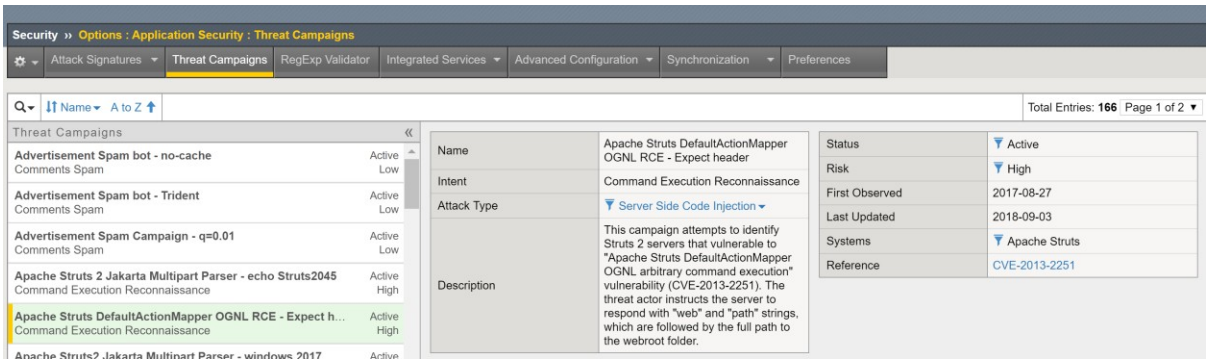
Task 1 – Threat Campaings Licensing and Components

- In the Configuration Utility of the BIG-IP, open the **System > License** page.



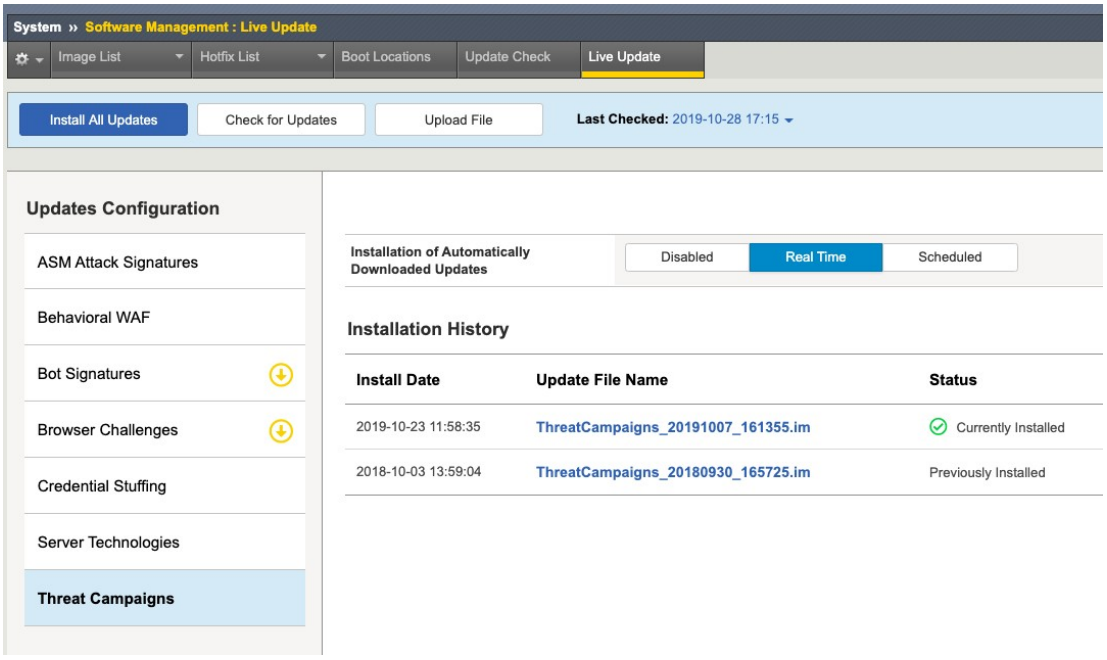
Threat Campaigns is currently available with the BIG-IP Advanced WAF License with an Add-On as Subscription.

- Open the Security > Options > Application Security > Threat Campaigns page.



This will show you all the Threat Campaigns available in the system along with their data.

- Open the System > Software Management > Live Update > Threat Campaigns page.



Threat Campaigns on BIG-IP 15.0 can now be updated automatically via Live Updates. They can also be updated manually.

Download is available from https://downloads.f5.com

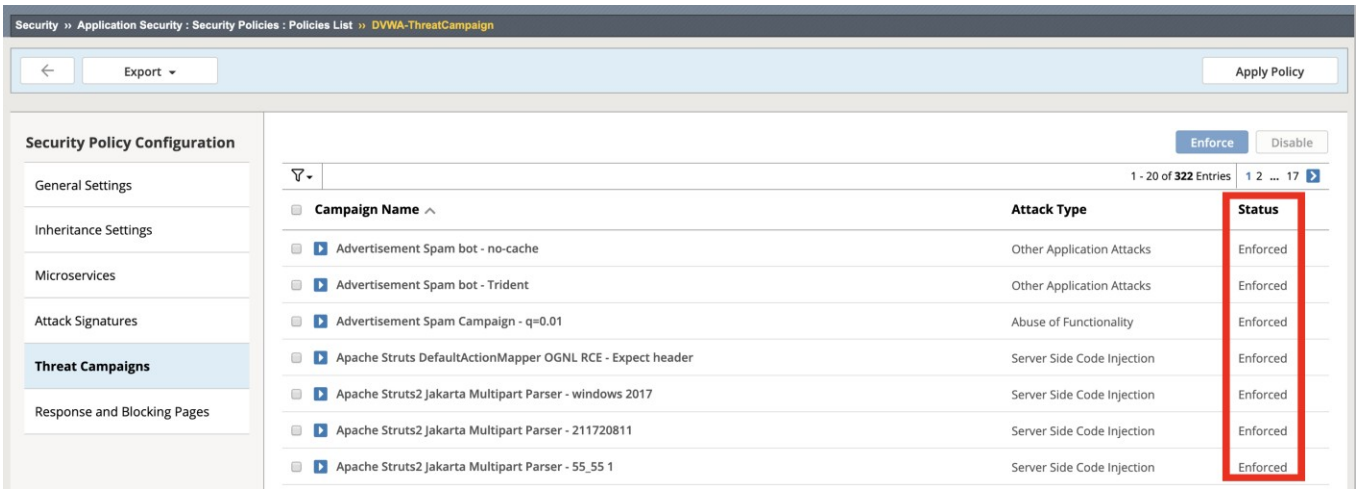
Task 2 – Threat Campaigns Configuration

- Open the Security > Aplication Security > Security Policies > Policies List page of the BIG-IP and click Create New Policy.
- For Policy Name enter **DVWA-ThreatCampaign**

- For Policy Template select Rapid Deployment. Click **OK** on the message that pops up asking whether to continue.
- For Virtual Server select **DVWA_protected_virtual**
- The **ASM-Bot-DoS-Log-All** profile should already be selected under the Logging Profiles
- Select **Blocking** for Enforcement mode.
- Leave all other settings at their defaults
- Click on the **Save** button.

- Open the Security > Aplication Security > Policy Building > Learning and Blocking Settings
- Locate **Threat Campaigns** and expand section. Observe there is no Learn flag for it and Alarm and Block is enabled. Threat Campaing Staging is disabled by default.

- Navigate to Security ›› Application Security : Security Policies : Policies List and select the DVWA-ThreatCampaign policy
- From the menu on the left, select **Threat Campaigns**
- Observe Threat Campaigns come by default Enforced

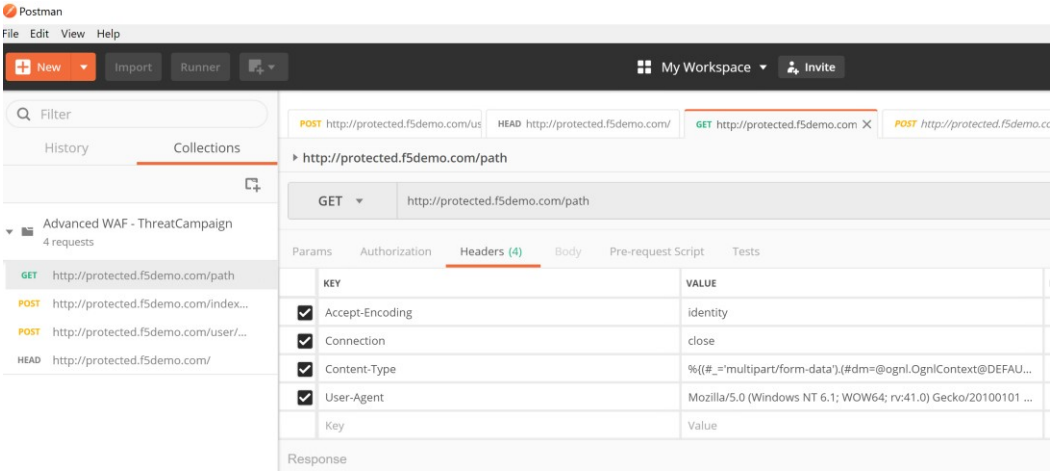


Exercise 3 – Testing Threat Campaigns

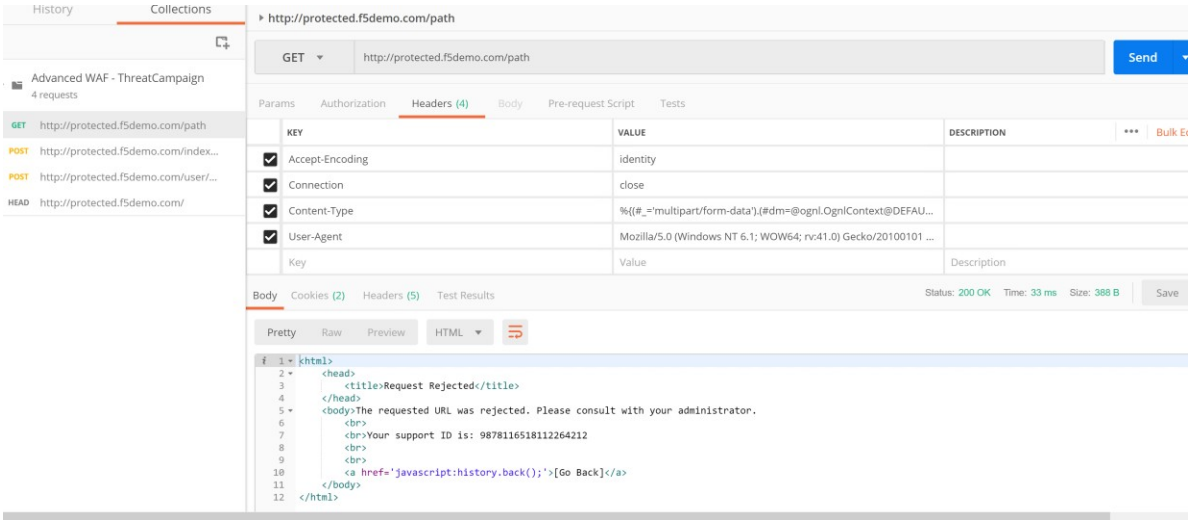
Task 1 – Launching Malicious Payload

- Locate **Postman** Application on the Windows Desktop and open it
- Click in Advanced WAF – ThreatCampaign collection

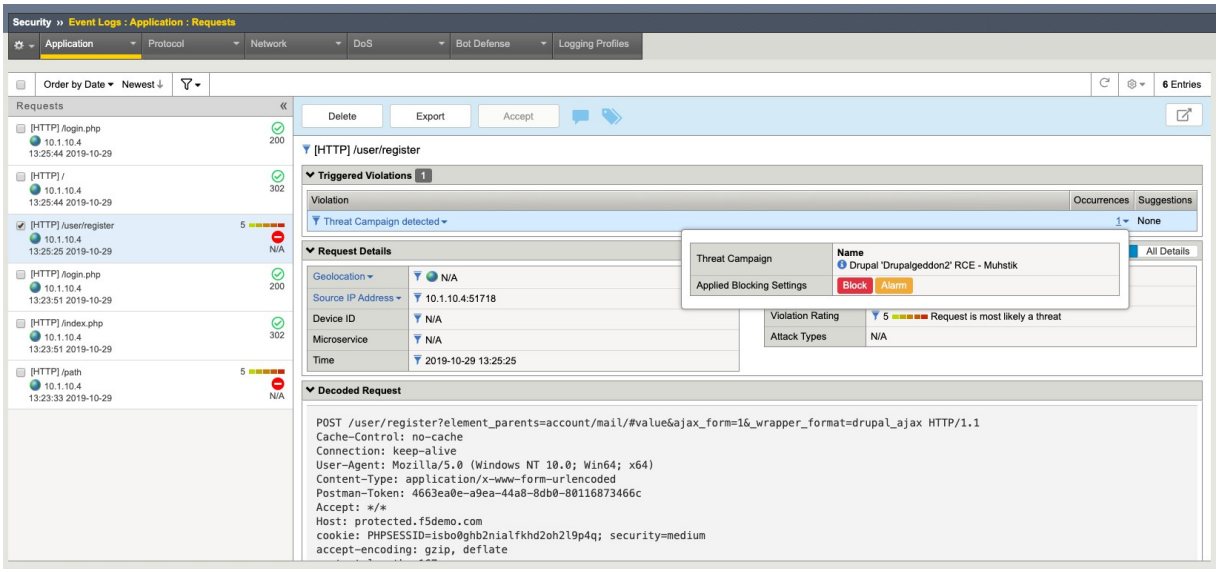
There are four requests simulating identified threats. Examine them, including Method, Headers and Body



- Now for every request click on **Send** button. Some responses shown will be a Blocking Page from the BIG-IP ASM. Take note of the Support ID in those responses.



- On the BIG-IP, open Security > Event Logs > Application >Requests
- Click on every item. Review **Support IDs** and match with the ones you took note of and verify further details of the Requests



FOOD FOR THOUGHT:

Why did only some requests generate a Threat Campaign violation?

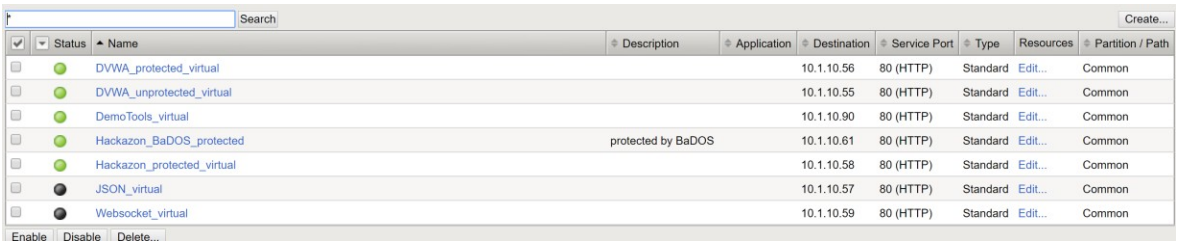
Lab 3 – Behavioral DoS (BaDoS)

The purpose of this lab is to run a simple demo of Behavioral DoS protection introduced on BIG-IP ASM 12.x in a limited way and available as unlimited on BIG-IP Advanced WAF 13.x onward. You will use some scripts running on a Kali Linux box to generate a base line traffic and also to launch specific DoS attacks.

Exercise 1 – Review BaDoS Configuration

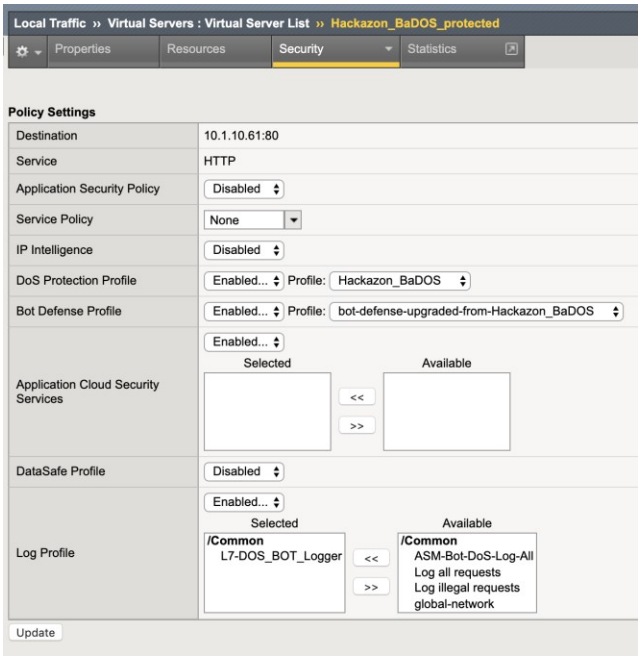
Task 1 – BaDoS Configuration

- On the BIG-IP, open Local Traffic > Virtual Servers and locate Hackazon_BaDOS_protected. Select the checkbox to the left of it and click in Enable.

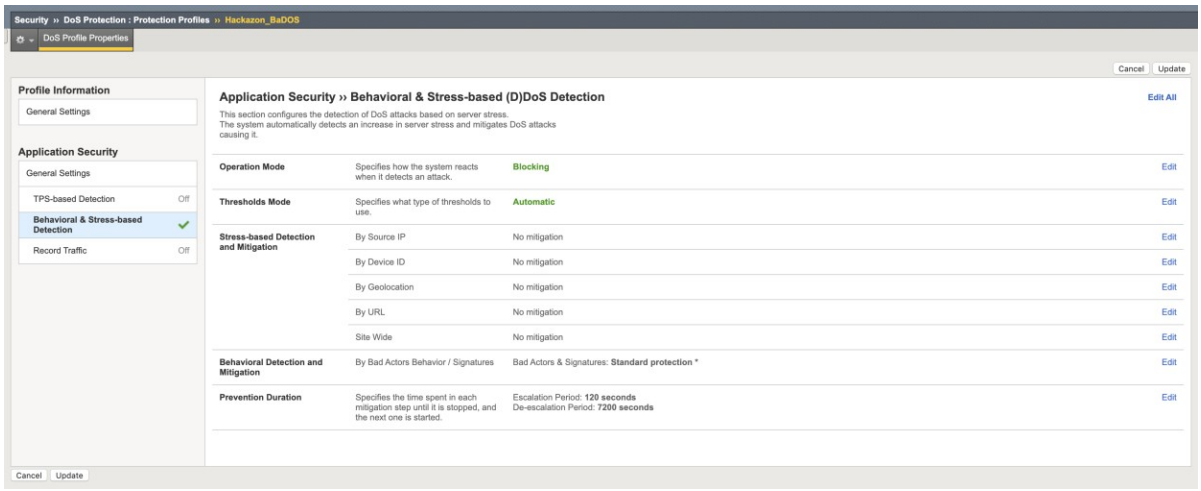


- Click in the **Hackazon_BaDOS_protected,** then open **Security > Policies**. Observe the DoS Protection and Bot Defense profiles assigned.

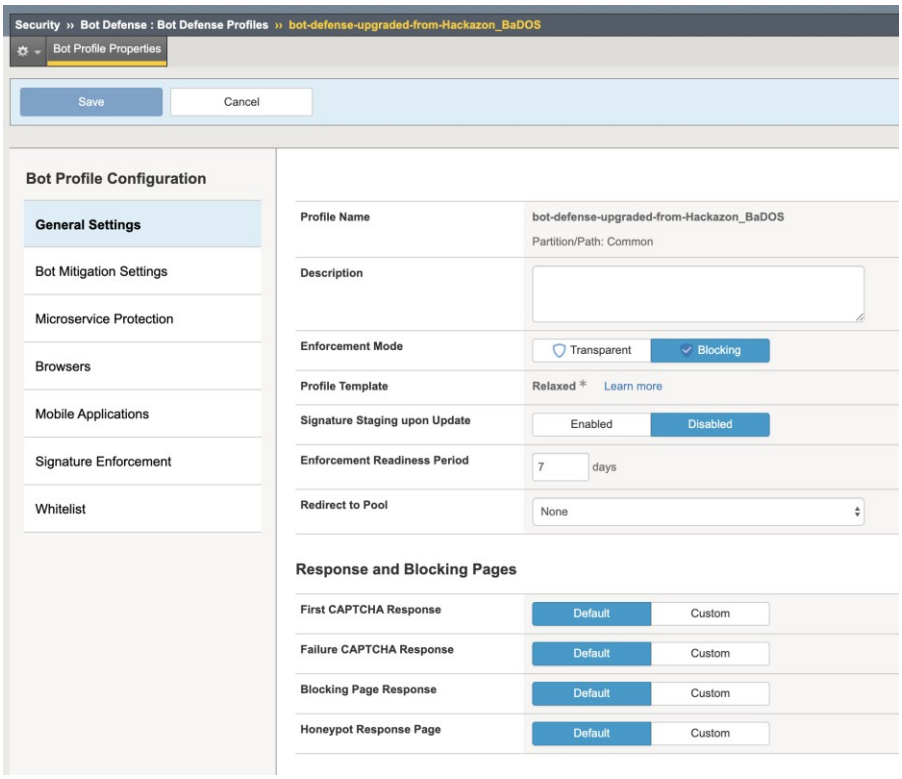
Note There is now an option to set a **Bot Defense** Profile. Proactive Bot Defense has been separated from the DoS profile from version 14.1.x. In this case, we will **disable** it (if not disabled).



- Open the Security > Dos Protection > Protection Profiles and click in Hackazon_BaDOS profile.

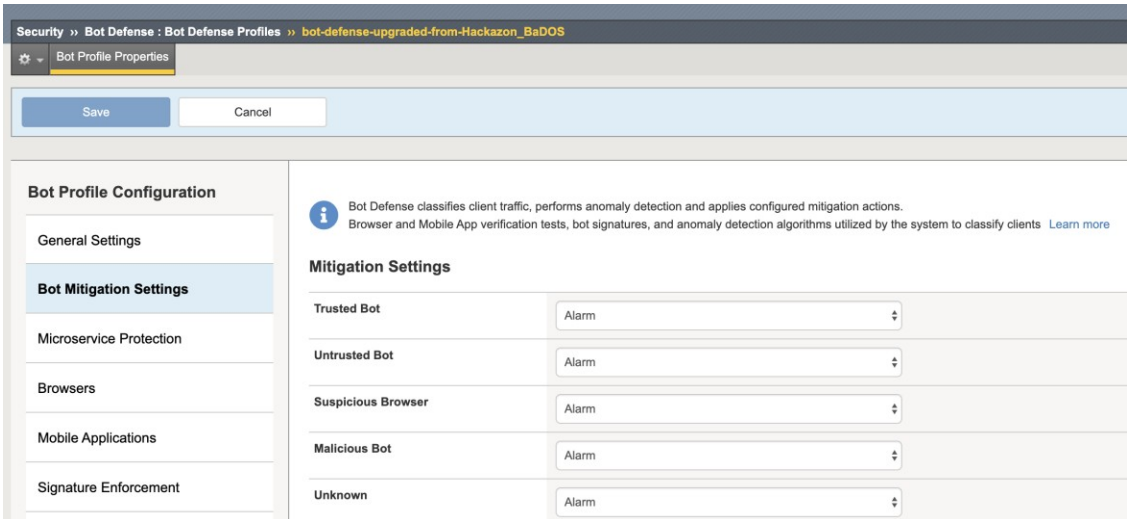


- The DOS profile has only **Behavioral & Stress-based Detection (BaDOS)** enabled. Review the settings.
- Open the Security > Bot Defense > Bot Defense Profiles and click on the **bot-defense-upgraded-from-Hackazon_BaDOS** profile.
- On the General Settings page, review the settings.



- On the **Bot Mitigation Settings** page, note that the **Mitigation Settings** are all set to Alarm.

Keep in mind, in real world environment the best practice is to block malicious categories.



- Because in this demo environment, we don´t have dozens of good and bad source IPs available for clients and attackers, we simulate them by adding an iRule to the VS, which adds a randomized X-Forwarded-For header to each request. iRule can be seen on **Local Traffic > iRules > iRule List > XFF_mixed_Attacker_Good**
- Kali Linux box has 2 IP Address assigned: 10.1.10.100 (used to generate legitimate traffic) and 10.1.10.200 (used to generate bad traffic). The iRule makes distinction on those.

- Because random IP are being generated by the iRule and inserted as an X-Forwarded-For header, there was a need to *Accept XFF* on the **Local Traffic > Profiles > XFF-http.** This will make DoS Protection Profile look for this header instead of the source IP of the connection.

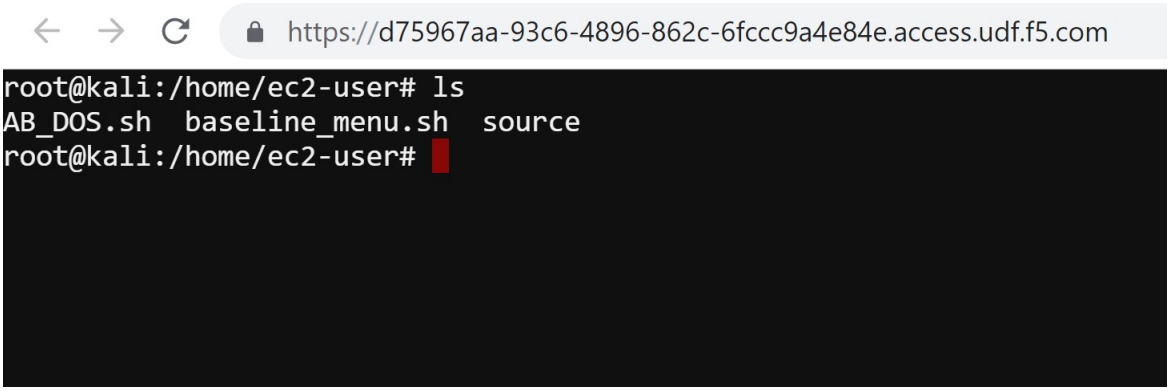
Exercise 2 – Generating Legitimate Traffic

Task 1 – Generating Legitimate Traffic

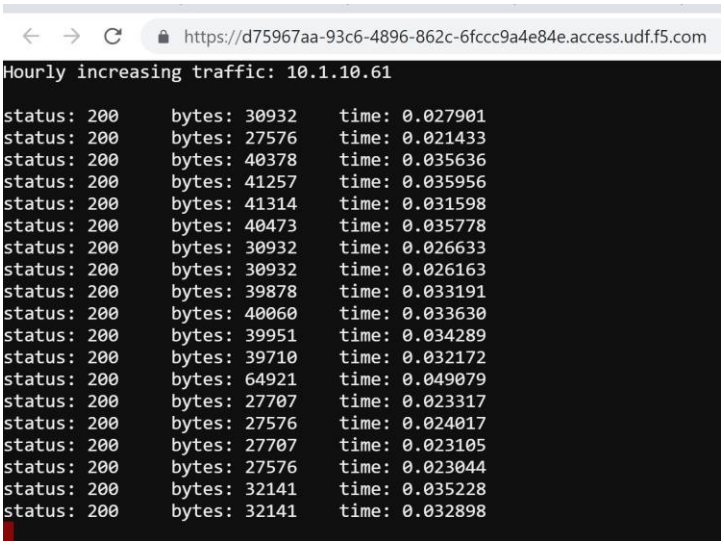
- Open two SSH console terminals for your Kali Linux client (or 2 tabs in case of using the Web Shell access method).

  Note: If you have an SSH key to access UDF you could use them to log into the Kali client, if not, please, use the Web Shell SSH: On UDF, go to **Deployments > Advanced WAF Mitigation Techniques > Details > Components > Kali Linux > Details > Access Methods > Web Shell** link:

- Enter the **/home/ec2-user** folder with the command **cd /home/ec2-user**.



- The *baseline_menu.sh* script includes two options: *increasing traffic* and *alternate traffic*. Execute it on the two console (or 2 tabs) and run *increasing traffic* from first console and *alternate traffic* from the second one. Use **./baseline_menu.sh** to launch it:



The response from the script tells you the target: 10.1.10.61 (the protected Virtual Server)

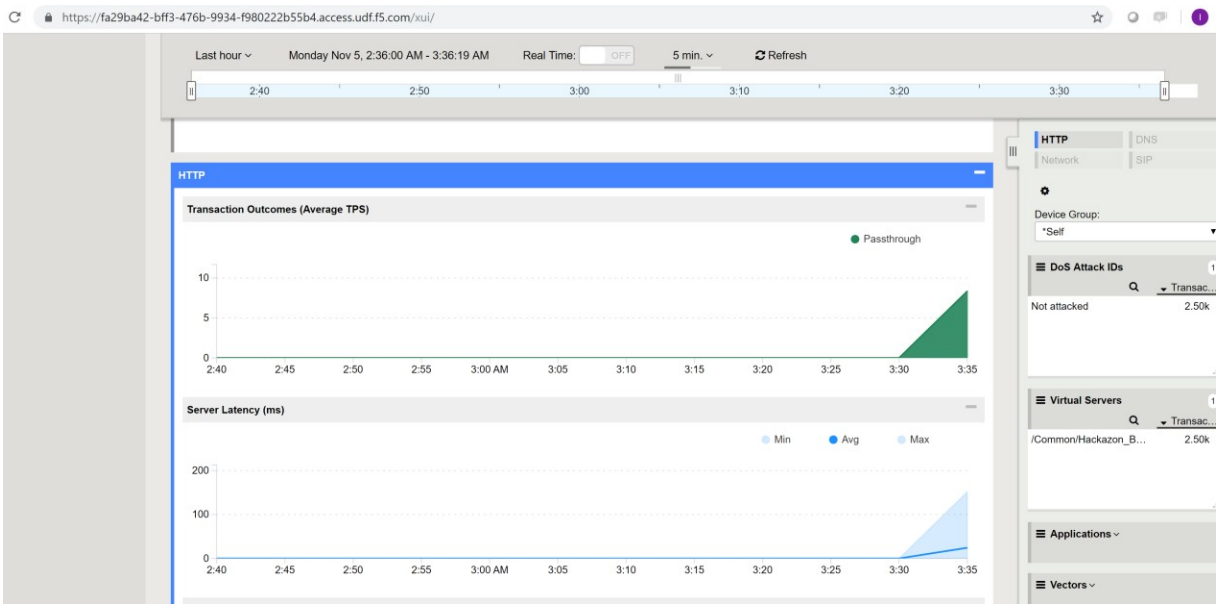
The status you receive from the server (200 OK), the transferred bytes and the time it took to get the response.

This is your valid traffic and the number of requests do change based on time.

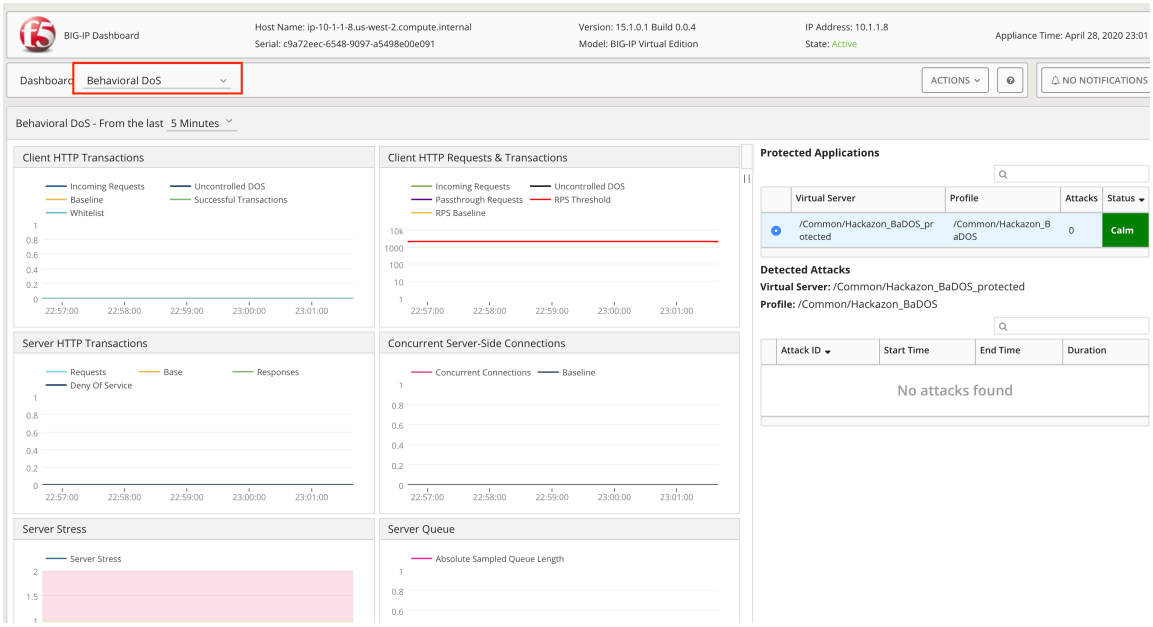
The request itself changes as well. The User-Agent changes with each request and the requested URI also. Both are randomly taken from the files in the “source” directory.

BaDOS can now learn how “good” traffic looks like.

- After a few minutes open the BIG-IP GUI on **Security > Reporting > DoS > Analysis**. You will see various metrics collected such as Average Throughtput, Number of Concurrent IPs, HTTP Transactions Per Second, Server Latency and Transaction Outcomes.



- Also open the BaDoS dashboard.



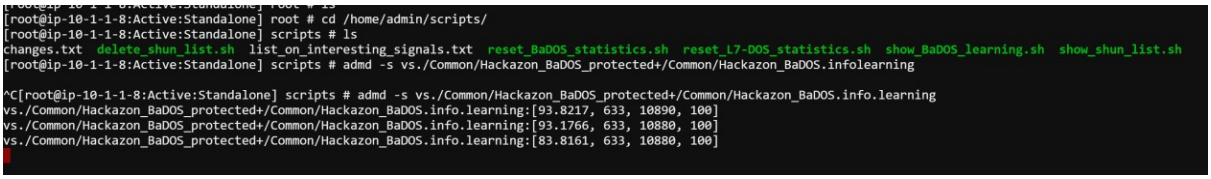
Task 2 – Observing BaDOS internal info

- From the Windows client SSH to your BIG-IP using the Command Prompt. (Note there is an issue with the Web Shell on BIG-IP version 15.x in UDF).
- Click on the search icon (magnifying glass) on the windows dock
- Type CMD in the search box and hit enter
- On the Command promt type ssh **admin@10.1.10.7**. Login with the admin user.
- After authenticating, you will be at the TMSH prompt, type '**bash'** and hit enter**.**
- There are a few scripts in the **/home/admin/scripts** directory. Feel free to examine all of them by navigating to the directory using the following command:

  cd /home/admin/scripts

- Also there is a BaDOS daemon called admd. On the command line type:

admd -s vs./Common/Hackazon_BaDOS_protected+/Common/Hackazon_BaDOS.info.learning



Output should be something like this:

vs./Common/Hackazon_BaDOS_protected+/Common/Hackazon_BaDOS.info.learning:[62.0614, 6, 7061, 100]

62.0614 average percentage approximation to the learned baselines.

For this demo, wait until you have reached at least 90%. This should happen after a couple of minutes.

The longer it runs, the better it is, because the system is self-adjusting permanently.

Exercise 3 – Launching a DoS Attack

Task 1 – Launching a DoS Attack using Kali

- Open another console terminal for you Kali Linux client (or another tab in the case when using Web Shell).
- Go the /home/ec2-user folder by typing: **cd /home/ec2-user**
- Launch AB_DOS script by typing: **./AB_DOS.sh**
- Use option 1 to start an attack:

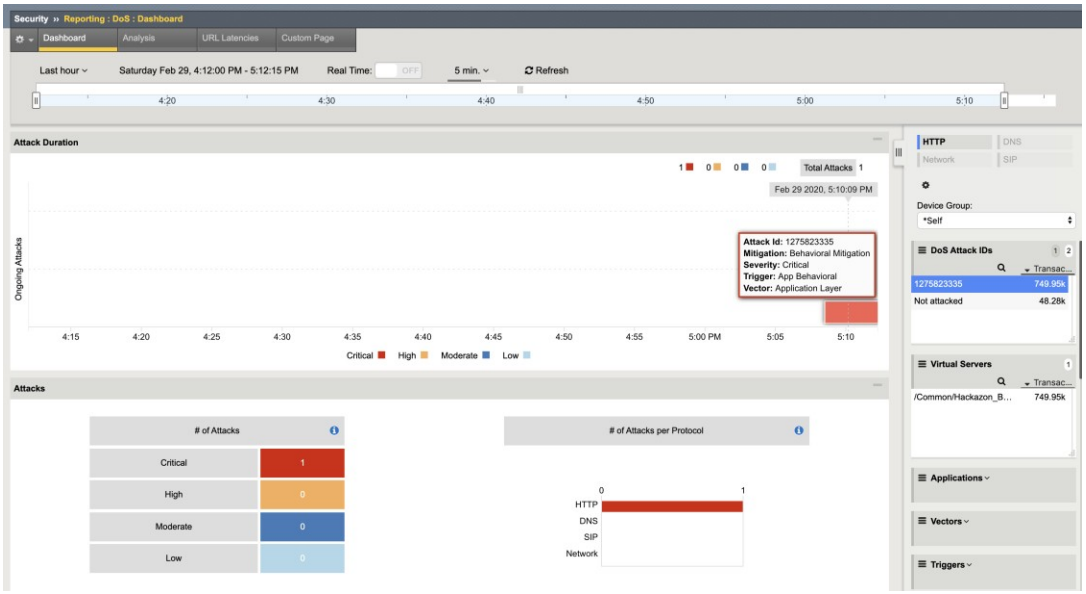
```
root@kali:/# cd /home/ec2-user
root@kali:/home/ec2-user# ./AB_DOS.sh
1) Attack start - similarity  3) Attack end
2) Attack start - score       4) Quit
#? 1
Start attack
This is ApacheBench, Version 2.3 <$Revision: 1757674 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 10.1.10.61 (be patient)
This is ApacheBench, Version 2.3 <$Revision: 1757674 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 10.1.10.61 (be patient)
This is ApacheBench, Version 2.3 <$Revision: 1757674 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 10.1.10.61 (be patient)
Completed 100000 requests
Completed 100000 requests
Completed 100000 requests
```

Task 2 – Reviewing DoS Event Log and Shun list

- On the BIG-IP GUI, go to Security > Event Logs > DoS > Application Events

You will see a DoS event is registered

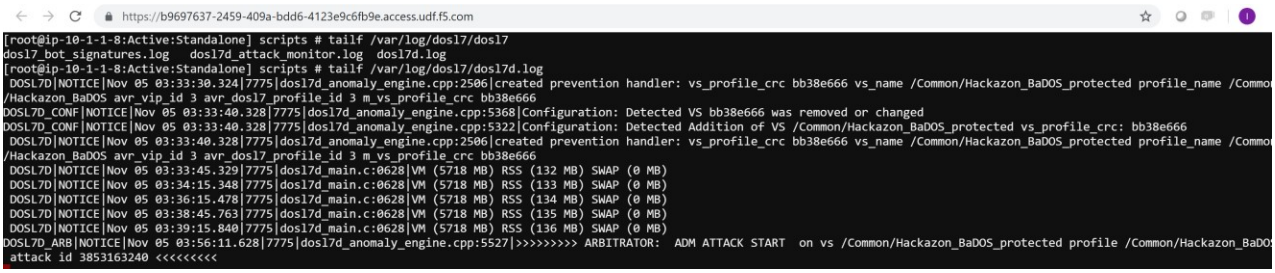
- Click on the Attack ID to view various information about the Ongoing attack



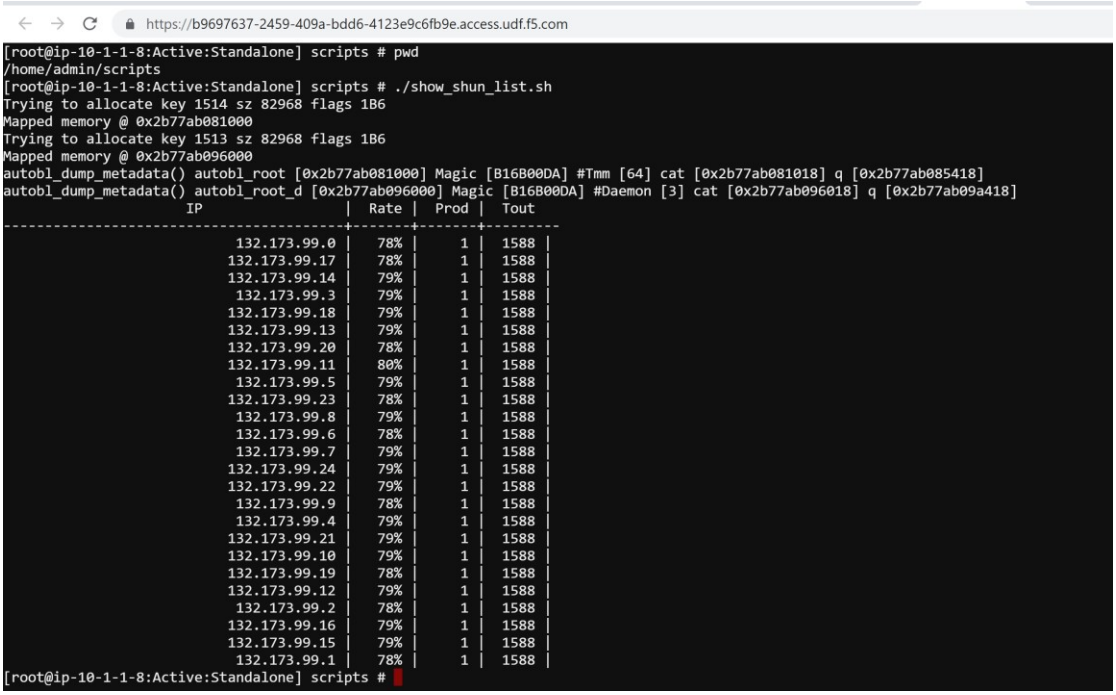
- Switch to **Analysis** menu and look at the **HTTP information**

- On the BIG-IP Shell (remember to connect via SSH from the Windows client) you will see an entry on the log when the attack was dected:

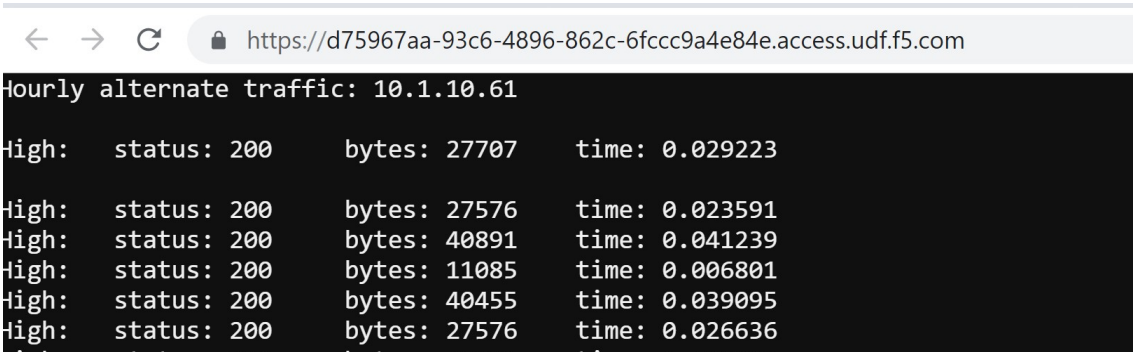
tailf /var/log/dosl7/dosl7d.log



- On the BIG-IP, navigate to **cd /home/admin/scripts** and
- Verify IP addresses are in the shun list. Type ***./show_shun_list.sh*** *on the CLI* to run the shun_list script.



- Verify your Kali Linux client is generating legitimate traffic:

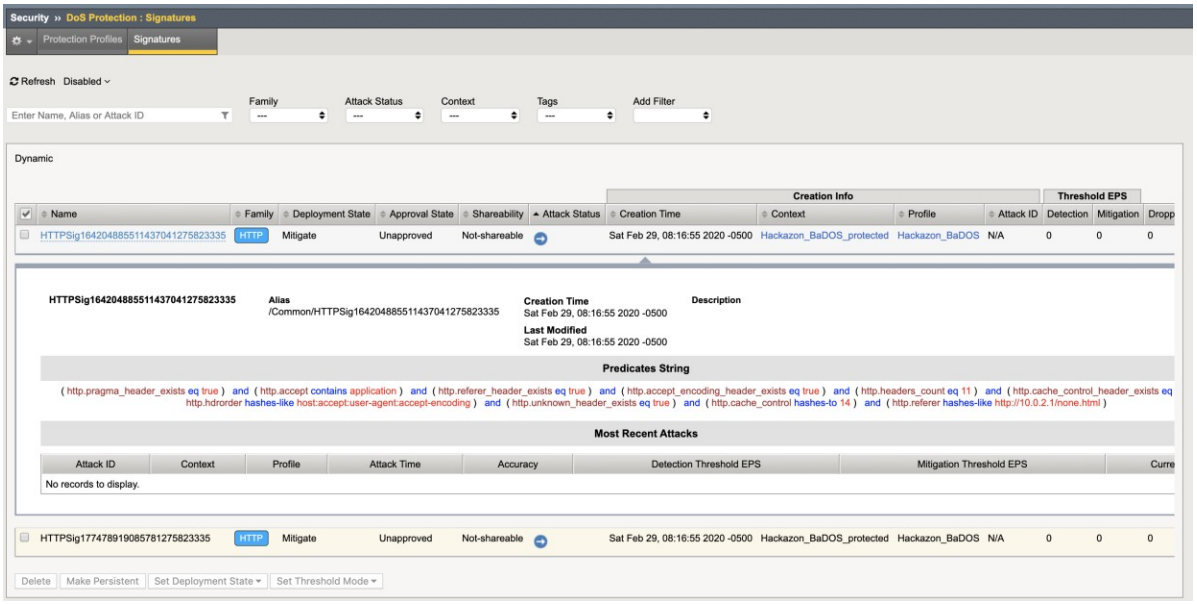


Observe legitimate traffic is still running throught with success.

Task 3 – Reviewing Dynamic Signatures

- On the BIG-IP GUI, go to Security > Dos Protection > Signatures

You will see a Dynamic Signature was created:



Observe the details about the signature. It is in Wireshark Format (Predicates String).

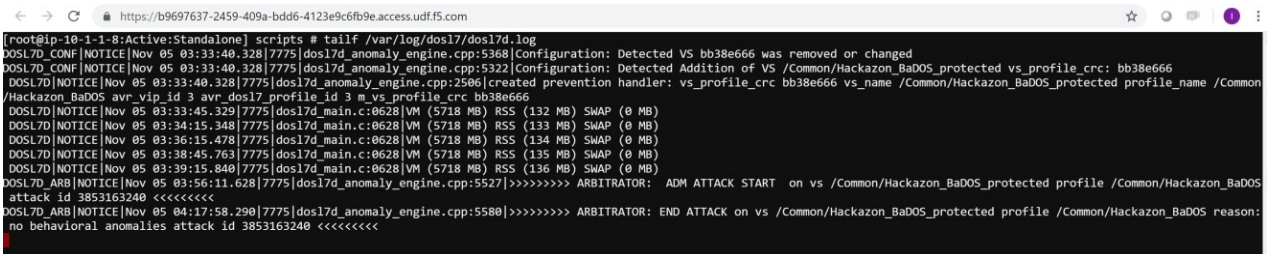
Task 4 – Stopping the Attack

- On the Kali Linux running *AB_DOS.sh* script, stop it pressing CTRL+C
- On the BIG-IP Shell when you run the ***./show_shun_list*** **script.sh** you will notice TTL of the quarantine IPs are being reduced (Tout column):

```
[root@ip-10-1-1-8:Active:Standalone] scripts # pwd
/home/admin/scripts
[root@ip-10-1-1-8:Active:Standalone] scripts # ./show_shun_list.sh
Trying to allocate key 1514 sz 82968 flags 1B6
Mapped memory @ 0x2b68772fe000
Trying to allocate key 1513 sz 82968 flags 1B6
Mapped memory @ 0x2b6877313000
autobl_dump_metadata() autobl_root [0x2b68772fe000] Magic [B16B00DA] #Tmm [64] cat [0x2b68772fe018] q [0x2b6877302418]
autobl_dump_metadata() autobl_root_d [0x2b6877313000] Magic [B16B00DA] #Daemon [3] cat [0x2b6877313018] q [0x2b6877317418]
                    IP              |  Rate |  Prod |  Tout
----------------------------------------+-------+-------+--------
                         132.173.99.0 |   78% |     1 |   801 |
                        132.173.99.17 |   78% |     1 |   801 |
                        132.173.99.14 |   79% |     1 |   801 |
                         132.173.99.3 |   79% |     1 |   801 |
```

- You would also notice the attack is ended on the BIG-IP DoS Events and in the log:

tailf /var/log/dosl7/dosl7d.log



- On the BIG-IP GUI, go to Security > Reporting > Dos > Analysis
- Examine the various statistics

Lab 4 – Public API Protection

The purpose of this lab is to run a simple demo of the Public API protection introduced on BIG-IP ASM 15.0. We are going to use a specific swagger file to import API definitions. For more information about swagger file, refer to the following link: https://swagger.io/docs/specification/about/

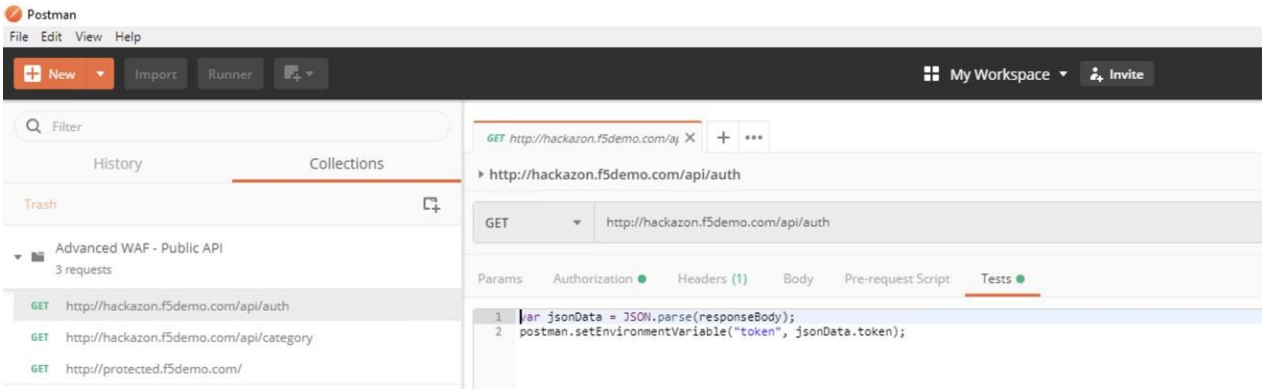
Exercise 1 – Create an API Security Policy

Task 1 – Create an API Security Policy & Import Swagger File

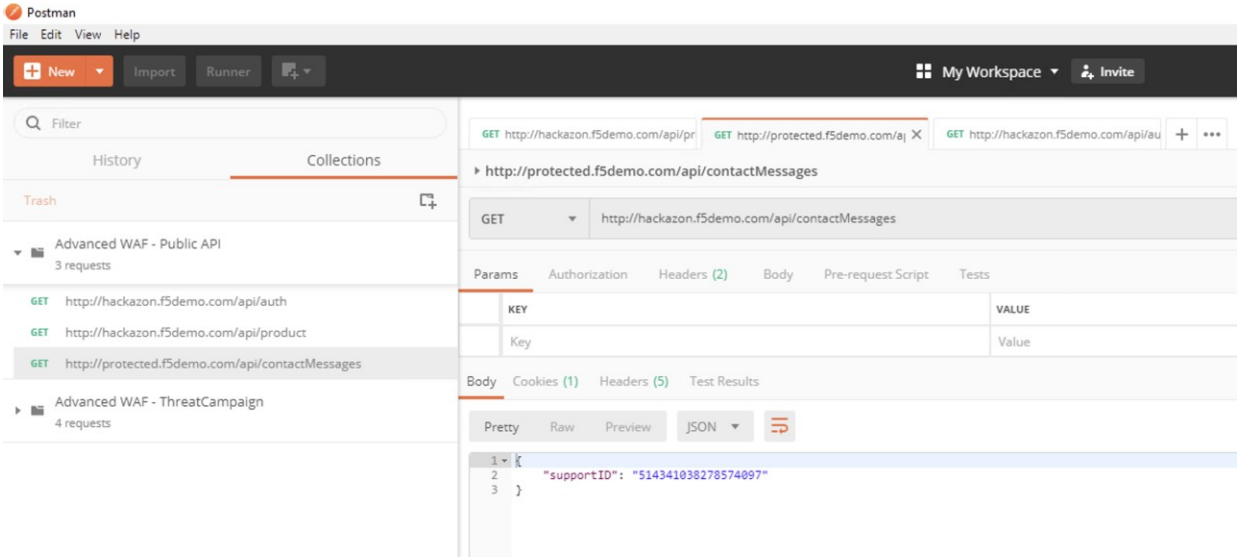
- Open Security > Application Security > Security Policies > Policies List.
- Create a new policy with the name **Hackazon-API.**
- Select **API Security** from the Policy Template drop down menu.
- Upload the **Hackazon-API-Swagger.json** swagger file from the OpenAPI (Swagger) File section.
    - The swagger file is located in the "lab guide" directory on the Win jumphost desktop.
- Click on the **Save** button.
- Navigate to Security ›› Application Security : URLs : Allowed URLs : Allowed HTTP URLs and review the URLs for the created policy.
- Navigate to Security ›› Application Security : Parameters : Parameters List and review the parameters for the created policy.
- Nvigate to Local Traffic > Virtual Servers > Virtual Server List and select the **Hackazon_protected_virtual** virtual server.
- Under the Security > Policies tab, select **Enabled** from the **Application Security Policy** drop down menu.
- Select the **Hackazon-API** policy.
- Ensure that the **ASM-Bot-Dos-Log-All** log profile is selected.
- Click the **update** button.
- Locate **Postman** Application on the Windows Desktop and open it.
- Click in Advanced WAF – Public API collection.

There are three requests.

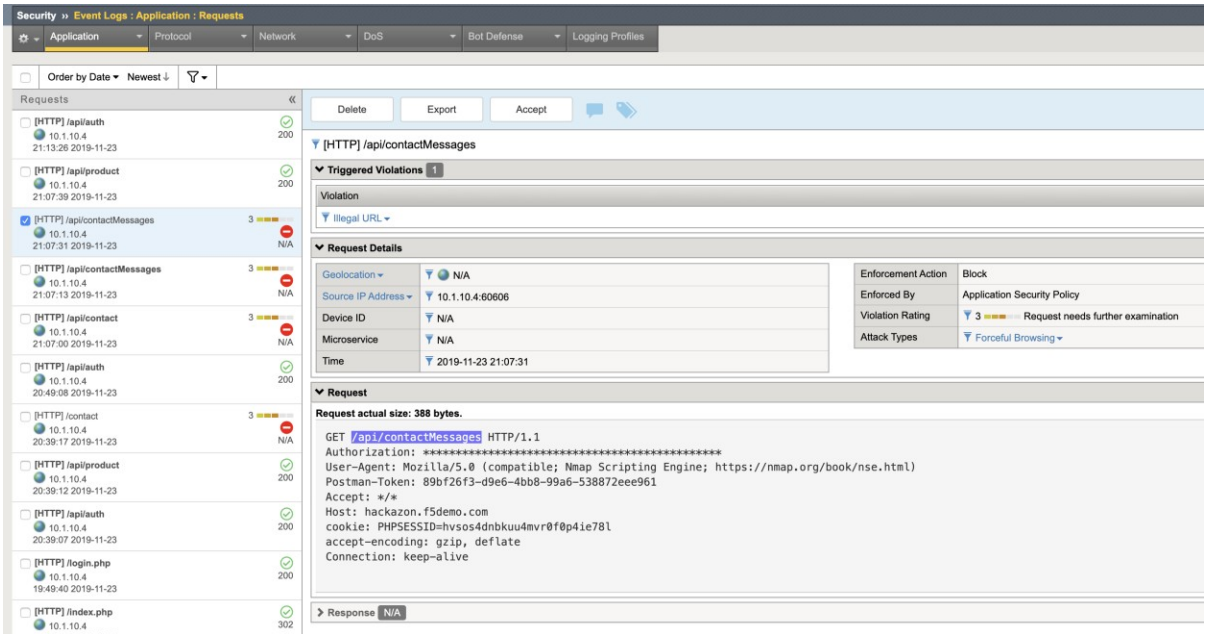
- The first request generates a token that is saved to an environment variable. Review the "Tests" tab. The code here extracts the token from the response body and saves it to the environment variable. Review the response body.
- The second request is to the product page of the API and is successful. The request includes the authorization token in the header.
- Exam the third request.



- Now for every request click on **Send** button. For the third request the response shown will be a Blocking Page from the BIG-IP ASM. Take note of the **Support ID**.



- On the BIG-IP, open Security > Event Logs > Application > Requests
- Review the event logs and find the blocked request that matches the **Support ID.** Determine why it was blocked.


Security » Event Logs : Application : Requests
Application
Protocol
Network
DoS
Bot Defense
Logging Profiles
Order by Date
Newest
Requests
[HTTP] /api/auth
10.1.10.4
21:13:26 2019-11-23
200
[HTTP] /api/product
10.1.10.4
21:07:39 2019-11-23
200
[HTTP] /api/contactMessages
10.1.10.4
21:07:31 2019-11-23
3
N/A
[HTTP] /api/contactMessages
10.1.10.4
21:07:13 2019-11-23
3
N/A
[HTTP] /api/contact
10.1.10.4
21:07:00 2019-11-23
3
N/A
[HTTP] /api/auth
10.1.10.4
20:49:08 2019-11-23
200
[HTTP] /contact
10.1.10.4
20:39:17 2019-11-23
3
N/A
[HTTP] /api/product
10.1.10.4
20:39:12 2019-11-23
200
[HTTP] /api/auth
10.1.10.4
20:39:07 2019-11-23
200
[HTTP] /login.php
10.1.10.4
19:49:40 2019-11-23
200
[HTTP] /index.php
10.1.10.4
302
Delete
Export
Accept
[HTTP] /api/contactMessages
Triggered Violations 1
Violation
Illegal URL
Request Details
Geolocation
N/A
Source IP Address
10.1.10.4:60606
Device ID
N/A
Microservice
N/A
Time
2019-11-23 21:07:31
Enforcement Action
Block
Enforced By
Application Security Policy
Violation Rating
3
Request needs further examination
Attack Types
Forceful Browsing
Request
Request actual size: 388 bytes.
GET /api/contactMessages HTTP/1.1
Authorization: **********
User-Agent: Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)
Postman-Token: 89bf26f3-d9e6-4bb8-99a6-538872eee961
Accept: */*
Host: hackazon.f5demo.com
cookie: PHPSESSID=hvsos4dnbkuu4mvr0f0p4ie78l
accept-encoding: gzip, deflate
Connection: keep-alive
Response N/A

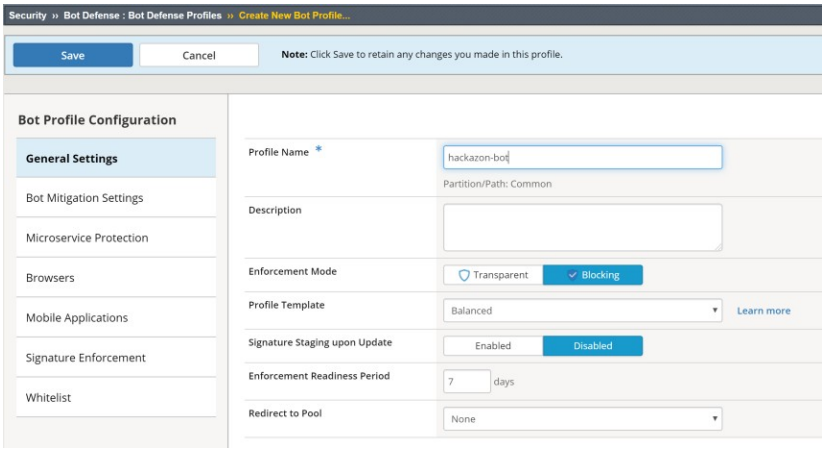
Lab 5 – Bot Defense

The purpose of this lab is to help you understand the bot detection and mitigation features in Bot Defense Profile. You will detect and block bots with increasing sophistication.

Exercise 1 – Configuring and Examining Bot Defense Profile

Task 1 – Configure General Settings

- Open **Security > Bot Defense > Bot Defense Profiles**
- Click in **Create** and perform the following modifications:
- Profile Name: **hackazon-bot**
- Enforcement Mode: **Blocking**



- Click in **Learn More** at the Profile Template
- The **Profile Template** is selected at profile creation and sets initial **Mitigation and Verification Settings** for the profile. Read about it.

Bot Defense Profile Templates

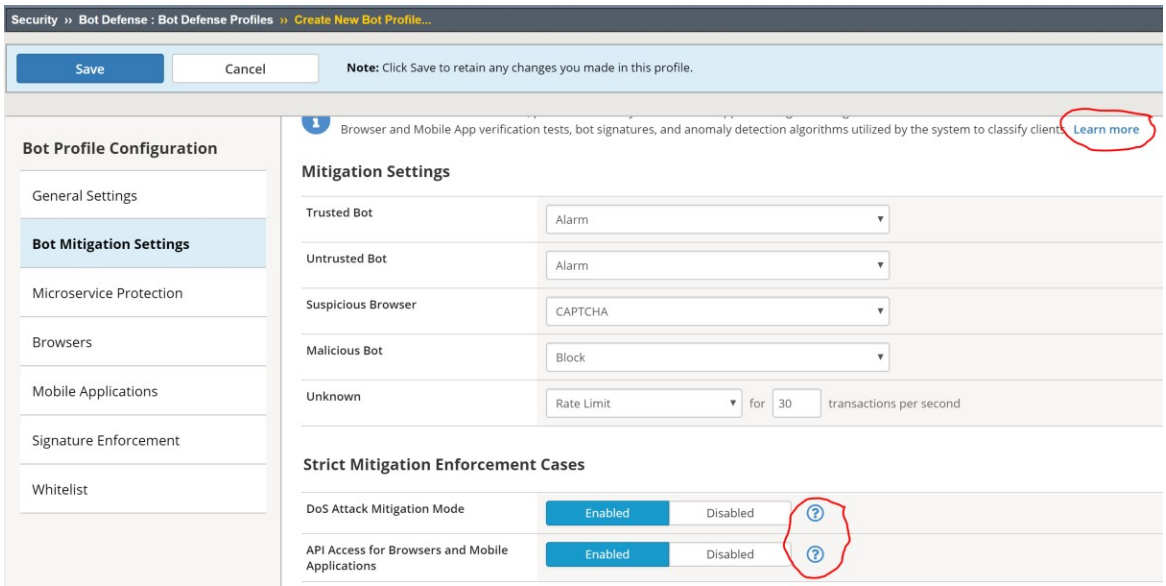
Profile templates specify **Mitigation and Verification Settings** default values. When selecting a profile template security and business requirements should be considered. The following templates are defined:

		Relaxed	Balanced	Strict
Verification	**Browser**	Challenge-Free Verification	Verify After Access (Blocking)	Verify Before Access
	Device ID Mode	None	Generate After Access	Generate Before Access
Mitigation Settings	**Trusted Bot**	Alarm	Alarm	Alarm
	Untrusted Bot	Alarm	Alarm	Block
	Suspicious Browser	Alarm	CAPTCHA	Block
	Malicious Bot	Block	Block	Block
	Unknown	None	Rate Limit	Block
	DoS Attack Mitigation Mode	Disabled	Enabled	Enabled
	API Access for Browsers and Mobile Applications	Disabled	Enabled	Enabled

- **Signature Staging** is disabled by default. If enabled, Signature Staging will not block requests that match new or updated Bot Signatures for the duration of the Enforcement Readiness Period, which is 7 days by default.
- **Redirect to Pool** could be used to send desired traffic (e.g. from malicious bot) to a specific configured pool
- The Final four options of the General Settings screen allow you to customize the Blocking Response, CAPTCHA Response and Honeypot Response pages

Task 2 – Configure Bot Mitigation Settings

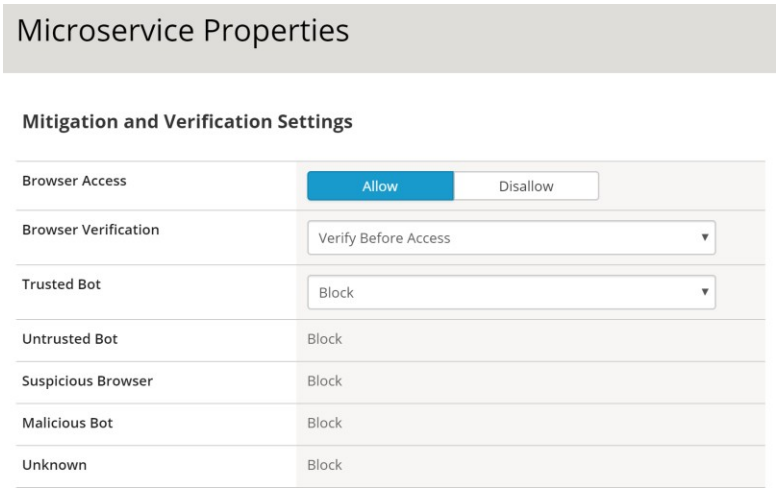
- Click in Bot Mitigations Settings.
- Examine each configuration for each Class of Bot. Click in **Learn more** to read bout the Mitigation Settings:



Task 3 – Configure Microservice Protection

For Bot Defense, a microservice is defined by a host name and/or a URI. Once the path to the microservice is defined unique Verification and Mitigation Settings can be applied to this path to protect against specific automated threats. Different mitigations can be configured according to the threats (as well customized ones).

- Click in **Microsoft Service Protection** and **Learn** at the top of the page get more details about the purpose of this feature in protecting against automated threats
- Click in **Create** and use the following settings:
- Microservice Name: login-page
- **Service Type**: Login Protection
- Protected URLs: /user/login
- Examine the remaining options and verify the Mitigation and Verification Settings are much mor stricted than the **Balanced Profile** template.



- Click in **Add** to conclude this section

Task 4 – Browser Verification

This section contains the settings for browser verification once the Bot Defense classifies as request as a Browser.

- Click in **Browsers**
- Click in the Interrogation icon to learn about **Browser Access**

Observe Browser Verification can be:

None: JavaScript and header-based browser verification is not performed. Note that a Session Opening anomaly may be detected if it is not disabled in the mitigation settings.

Challenge-Free Verification: Browser verification tests are performed on request headers. No JavaScript inject is performed.

Verify before Access: (Default) A white page with JavaScript is returned to the client before passing the request to the server. The JavaScript performs browser verification tests. If the tests fail, browser verification anomalies are reported and the mitigation is performed according to the selected mitigation settings. If the tests pass, the request is passed to the server.

Verify after Access (Detection Only): JavaScript is injected in the server response. The JavaScript performs browser verification tests. If the tests fail, browser verification anomalies are reported but not enforced. If the tests pass, the request is passed to the server.

Verify after Access (Blocking): JavaScript is injected in the response. The JavaScript performs browser verification tests. If the tests fail, browser verification anomalies are reported and the mitigation is performed according to the selected mitigation settings. If the tests pass, the request is passed to the server

- Leave the settings as default

Task 5 – Mobile Applications

- Click in **Mobile Applications** and examine the various options.

Most Mobile applications are not capable of processing javascript. Because security controls cannot validate the mobile application with javascript it would be easy for an attackers to impersonate the mobile application's user-agent header and gain unverified automated access to the backend applications. The **F5 Anti-bot Mobile SDK** allows for easy integration of F5's mobile app protections and verifications through a partner service, **Appdome**. The **F5 Anti-bot Mobile SDK** facilitates the validation of the mobile application, integration with the Unified Bot Defense protections while also blocking fraudulent requests that are impersonating the mobile application. This section is out of the scope for this lab.

Alternatively a custom signature could be created to identify the Mobile App, however, this is easy to be spoofed by attackers as the signature would be based on User-Agent headers and URL.

Task 6 – Signature Enforcement

- Click in Signature Enforcement

Signature Enforcement allows for the staging or enforcement of bot signatures IF staging is enabled for the Bot Defense profile.

Task 7 – Whitelist

- Click in **Whitelist**

Whitelist allows for bypassing Bot Defense mechanisms based on IP address and/or URL. Notice that some URLs are already listed by default (they will show up after saving the configuration). These are included because some browsers behave differently when accessing these URLs and do not some some or all cookies, which can cause false positives for Bot Defense verifications. Some common whitelist use cases: customer's internal/trusted IP subnets, customer's vulnerability scanners

- Click in **Save** to conclude the configuration

Task 8 – Bot Defense Logging Profile and Virtual Server configuration

- Click in **Application Security > Event Logs > Logging Profile**
- Click in **ASM-Bot-DoS-Log-All** and examine the **Bot Defense** tab. Observe log can be very granular logging every single request. *Food for though: Which logs you would recommend to log in a production environment?*
- Click in **Local Traffic > Virtual Servers** and click in **Hackazon_protected_virtual**
- Click in the **Security**

- Select **Disabled** (if not selected) for **Application Security Policy**
- Select **Disabled** (if not selected) for **DoS Protection Profile**
- Select **Disabled** (if not selected) for **DataSafe Profile**
- Select **hackazon-bot** for the **Bot Defense Profile**

Exercise 2 – Basic Bot Detection

In this exercise, we will use 'curl' to represent a basic a very simple bot.

Task 1 – Curl requests

- From the Windows Desktop, open a command prompt (or Git Bash – Desktop Icon) and run the following cURL requests (pay attention at the responses from every request):
    - Note should you have problems with copy/paste the Curl commands are located in the "Lab Guide" directory on the desktop of the Win jumphost.
- `curl http://hackazon.f5demo.com/`
- `curl http://hackazon.f5demo.com/user/login`
- `curl http://hackazon.f5demo.com/ -H "User-Agent: Mozilla/5.00 (Nikto/2.1.6) (Evasions:None) (Test:Port Check)"`

How different was the response for each request?

Task 2 – Review Bot Defense Logs

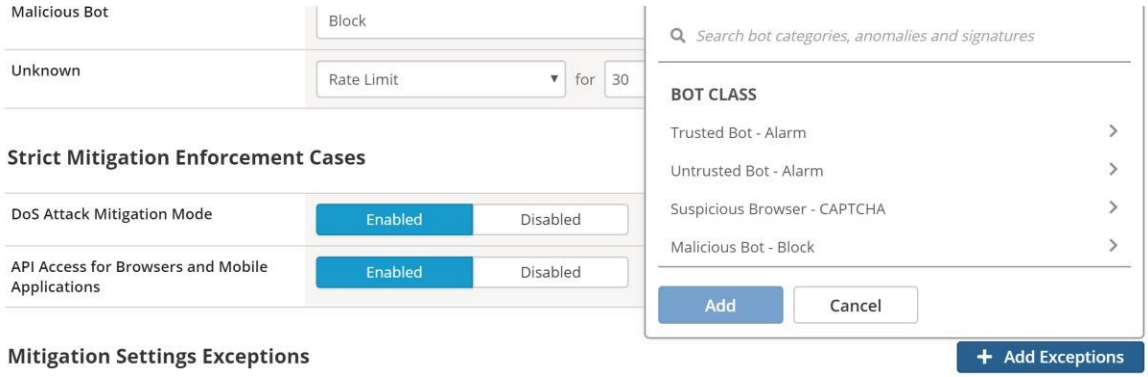
- Navigte to **Security > Event Logs > Bot Defense > Bot Requests**
- Select the third request down (as this should be the first request sent in Task 1)
- Click in **All Details** and answer the following questions:
- What is the Request Status and the Mitigation Action?
- What is the Browser Verification Status?
- What is the Bot Class, Bot Category and Bot Signature triggered?
- What is the **User Agent** of the request?
- Select the second request down (as this should be the second request sent in Task 1)
- Click in **All Details** and answer the following questions:
- The **Request Status** is **Denied** alghouth the request was sent using Curl. Why is this different from the first request?
- Select the first request down (as this should be the lastrequest sent in Task 1)
- Click in **All Details** and answer the following questions:
- What is the Request Status and the Mitigation Action?
- What is the Browser Verification Status?

- What is the Bot Class, Bot Category and Bot Signature triggered?
- What is the **User Agent** of the request?

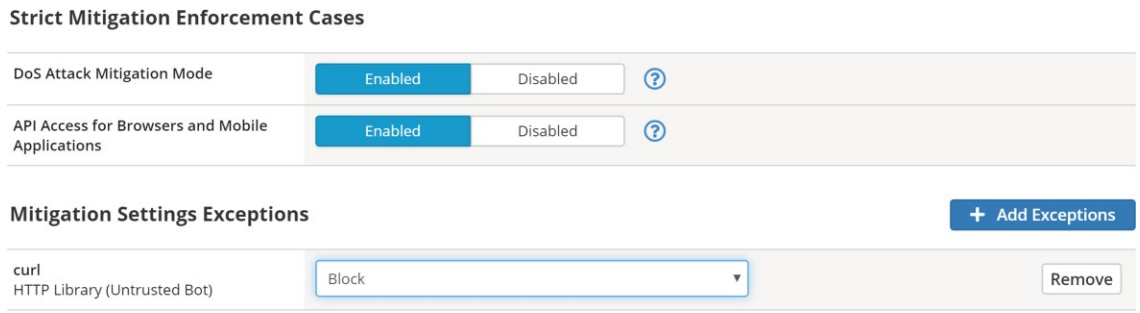
Task 3 – Create an exception

In the previous exercise, based on the Bot Defense configuration cURL is classified as an Untrusted Bot. In a case for this specific signature if one would like to create an exception this is possible. For instance, one can create an exception and configure cURL signature (part of the HTTP Library category) to **Block** instead of **Alarm**.

- Navigte to **Security > Bot Defense > Bot Defense Profiles** and click in hackazon-bot
- Click in Bot Mitigation Settings, then locate Mitigation Settings Exceptions.
- Click in Add Exceptions. Browse the various Bot Classes, Categories and Signatures



- Click Untrusted Bot > HTTP Library – Alarm > curl – Alarm and then click in Add
- Modify the newly added exception mitigation to **Block,** then click in **Save**



- (Optional) Redo Task1 and Task2 and observe results

Exercise 3 – Impersonating Bot Detection

In the previous examples, the bot was identified based on the User Agent header. It is trivial and common for the attackers to change the user-agent to a value to 'impersonate' a valid browser. Here you can find a comprehensive list of user agents: https://developers.whatismybrowser.com/useragents/explore/

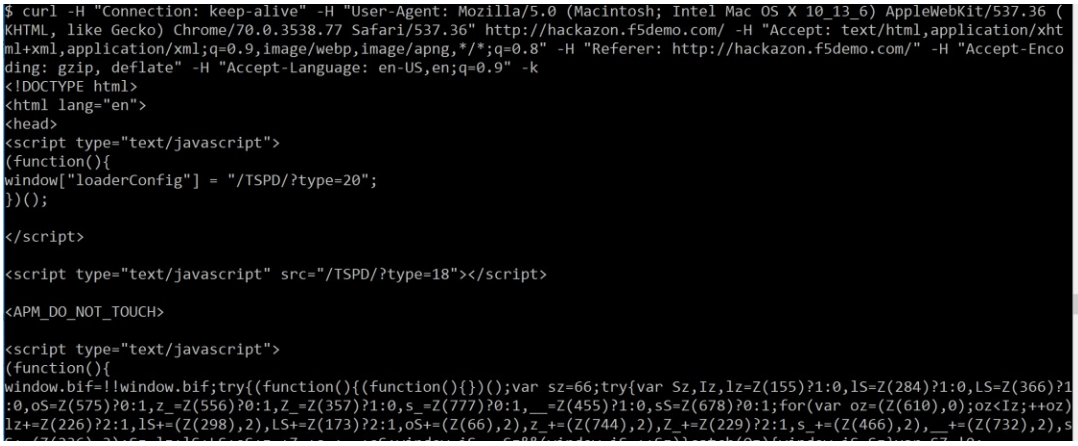
Task 1 – User-Agent spoofing

- From the Windows Desktop, open a command prompt and run the following cURL request (pay attention at the response from that request):
- `curl -H "User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36" http://hackazon.f5demo.com/`
- How the response looked line from the curl output?
- Navigate to **Security > Event Logs > Bot Defense > Bot Requests**
- Select the first request down
- Click in **All Details** and answer the following questions:
- What is the Request Status and the Mitigation Action?
- What is the Browser Verification Status?
- What is the Bot Class, Bot Category and Bot Signature triggered?
- Was any **anomaly** detected?

Task 2 – User-Agent advanced spoofing

It is still possible for automated traffic to completely impersonate a real browser by sending all of the appropriate headers and in the correct order. In this exercise we will use cURL to exactly impersonate a Chrome request with correct HTTP header orders.

- Before we use the curl, open Firefox and browse to http://hackazon.f5demo.com/
    - Refresh it around 15 times with F5
    - Before Bot Defense can inject any script into a page it needs to qualify it first (that means, determine if that page is possible to inject a response)
- From the Windows Desktop, open a command prompt and run the following cURL request (pay attention at the response from that request, if it is empty, refresh the page on Firefox again until you see the response):
- `curl -H "Connection: keep-alive" -H "User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36" http://hackazon.f5demo.com/ -H "Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8" -H "Referer: http://hackazon.f5demo.com/" -H "Accept-Encoding: gzip, deflate" -H "Accept-Language: en-US,en;q=0.9"`
- A similar response should appear (with a piece of Bot Defense javascript):

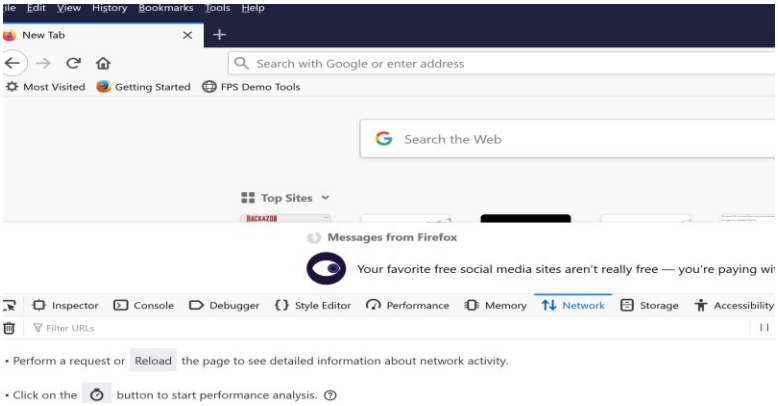


- Navigate to **Security > Event Logs > Bot Defense > Bot Requests**
- Select the first request down
- Click in **All Details** and answer the following questions:
- What is the Request Status and the Mitigation Action?
- What is the Browser Verification Status?
- What is the Bot Class, Bot Category and Bot Signature triggered?
- Was any **anomaly** detected?

Notice that the Request Status is Accepted. Because no bot signatures were triggered and all the Challenge-Free verification checks were passed this request was allowed through. Also, as indicated in the Verification Action and Challenge Status the Bot Defense profile is configured to verify AFTER access. That means Bot Defense will send the request first to the backend server and inject the bot defense javascript on the response before return it to the client.

Task 4 – Identifying Bot Defense injected code

- On the Windows Client machine launch Firefox (if not opened)
- Press F12 and click in **Network** at the Developer Tools



- Access http://hackazon.f5demo.com. Observe every request gets logged at the **Network** tab
- Examine the first request. It contains the Bot Defense javascript, which in turns trigger two additional requests (second and third). Examine that.

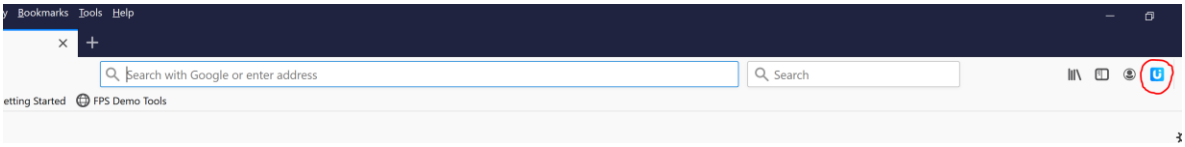
Exercise 4 – Automated Browser Detection

While command line tools and scripts are unable to process javascript challenges, there are many tools available to attackers to bypass these basic javascript challenges. There are also, tools such as Selenium that can be configured to "drive" real browsers through sophisticated web applications

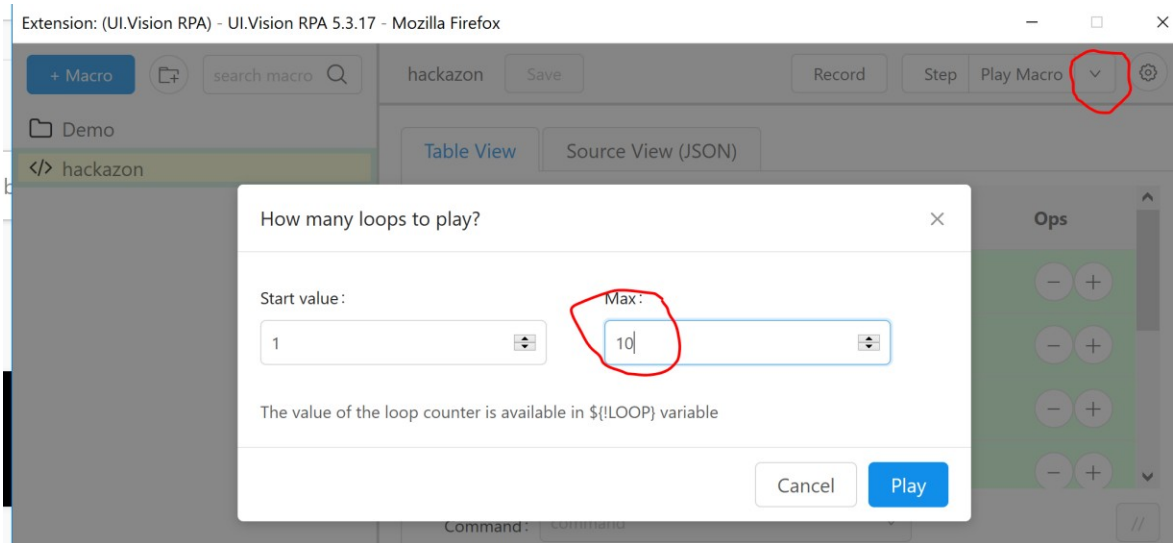
In this exercise we will use the UI.Vision (formerly Kantu) extension for Firefox to test the ability of F5 Bot Defense to detect and mitigate automated browsers. UI.Vision is a free Selenium-based extension available for most browsers (https://ui.vision/)

Task 1 – Lanching automated browsing

- Open **Firefox** (close it and re-open it) and search for the blue **UI.Vision icon**



- Click in the arrow icon and followed by **Play loop…**



- Set **Max** as 10 and click in **Play**
- Observe the hakazon page being loaded and browsed automatically. Wait until you see a blocking page from the Bot Defense.
    - The first time you run this procedure will take more requests to block the automated process as some of the URLs are getting qualified before a Javascript is inserted into the response. After the first time you see a Blocking Page, if you close the browser and repeat the process the Blocking Page will appear faster.
- Review the Bot Defense logs at Security > Event Logs > Bot Defense > Bot Requests

Exercise 5 – Impersonating Bot Detection (bonus)

Task 1 – Replaying User-Agent advanced spoofing

On the Exerciese 3, Task 2, it was possible to make a request using curl simulating Chrome with the correct headers. The request made to the backend server and the Bot Defense javascript was injected in the response prior to return it to the client. What does it happen if an attacker replay that request simulating a DoS attack?

- On the Windows Client machine launch Bash (Gitbash desktop icon)
- Execute the following curl command in an infinite loop:

- `while true;do curl -H "Connection: keep-alive" -H "User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36" http://hackazon.f5demo.com/ -H "Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8" -H "Referer: http://hackazon.f5demo.com/" -H "Accept-Encoding: gzip, deflate" -H "Accept-Language: en-US,en;q=0.9"; done;`
- Observe the curl responses with the HTML + injected javascript
- Leave it running for a few seconds until the response changes to Bot Defense blocking page (sample below). At this time, cancel the requests pressing CTRL+C a few times or close the bash window

  <html><head><title>Request Rejected</title></head><body>The requested URL was rejected. Please consult with your administrator.

Your support ID is: < 9403282495987779806>

<a href='javascript:history.back();'>[Go Back]</body></html>
- Navigate to Security > Event Logs > Bot Defense > Bot Requests
- Select the first request down
- Click in **All Details** and answer the following questions:
- What is the Request Status and the Mitigation Action?
- What is the Browser Verification Status?
- What is the Bot Class, Bot Category and Bot Signature triggered?
- Was any **anomaly** detected?
- Can you find out for how long this client will be blocked based on the Anomaly Detected?

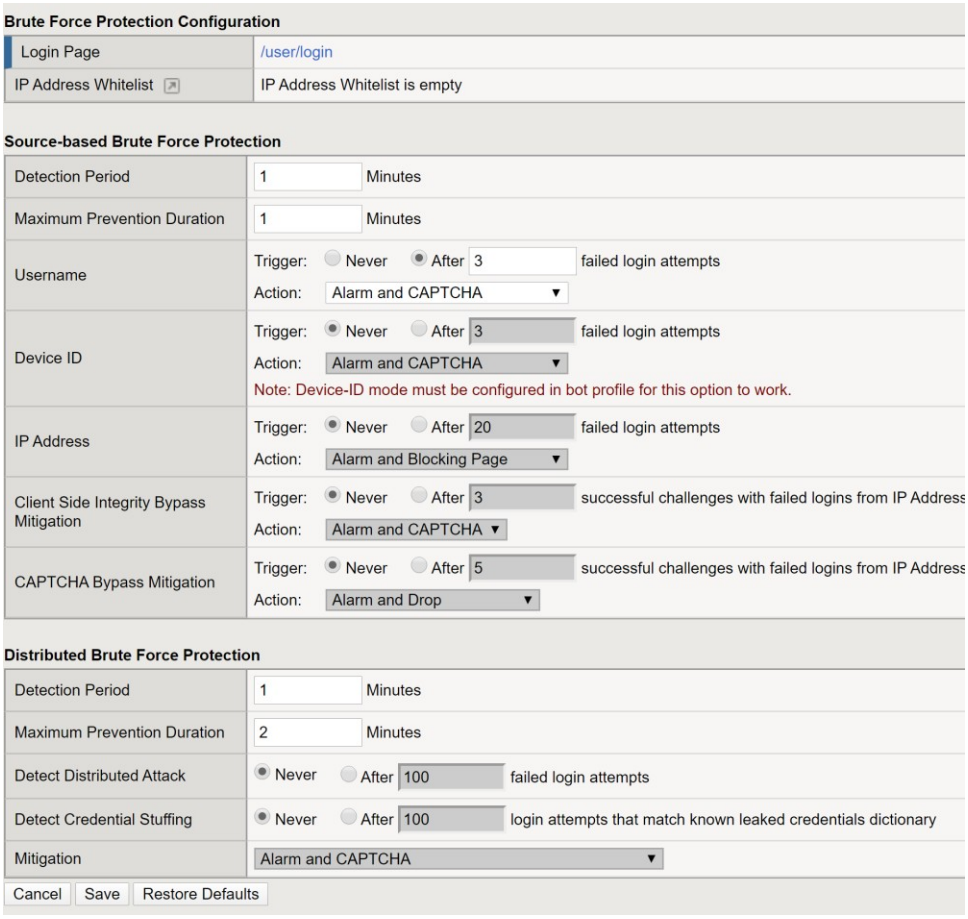
Lab 6 – Brute Force

The purpose of this lab is to help you understand the Brute Force protection capabilities in the BIG-IP ASM/Adv. WAF. A base policy for Brute Force was defined in advance as well the login page for Hackazon application.

Exercise 1 – Configuring and Brute Force Protection

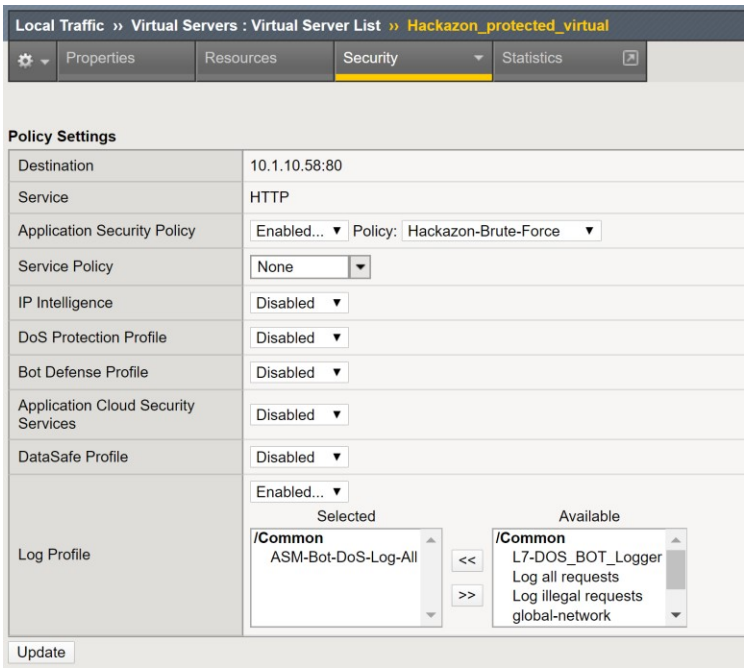
Task 1 – Configure Brute Force

- Open **Security > Application Security**
- Click in **Hackazon-Brute-Force**
- Open **Security > Application Security > Sessions and Login > Login page list**
- Click in **/user/login** and examine the Login page details
    - The login page was previously defined for you. That success/fail criteria should work in conjunction with the customer to determine what kind of responses a login attempt occur will determine if the attempt failed or succeded.
- Open **Security > Application Security > Brute Force Attack Prevention** and click in **Create**
- Examine the various options and configure them as the following:



- Click in **Create** and **Apply Policy** in the sequence

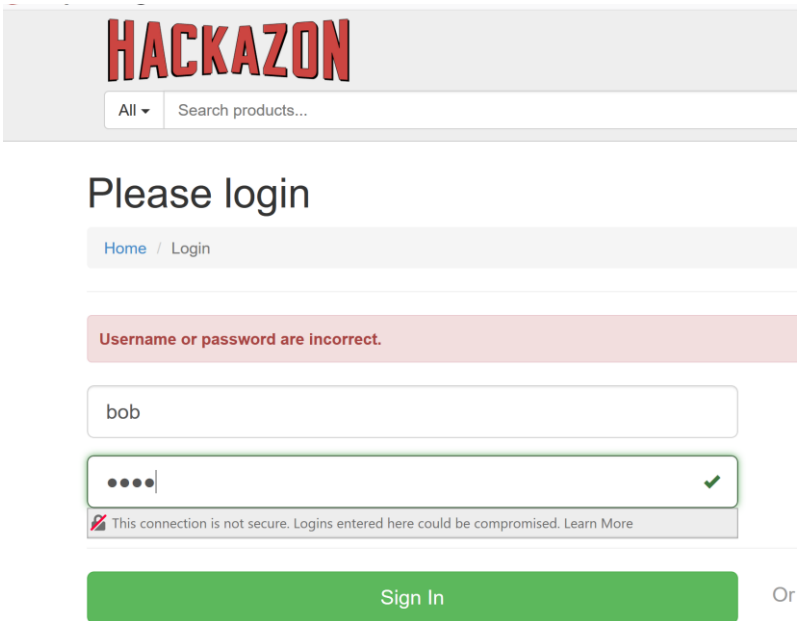
- Navigate **to Local Traffic > Virtual Servers** and click in **Hackazon_protected_virtual**
- At the **Security > Policies** tab, select **Hackazon-Brute-Force** as **Application Security Policy**
- Make sure **Bot Defense Profile** and **DoS Profile** options are disabled
- Click in **update**



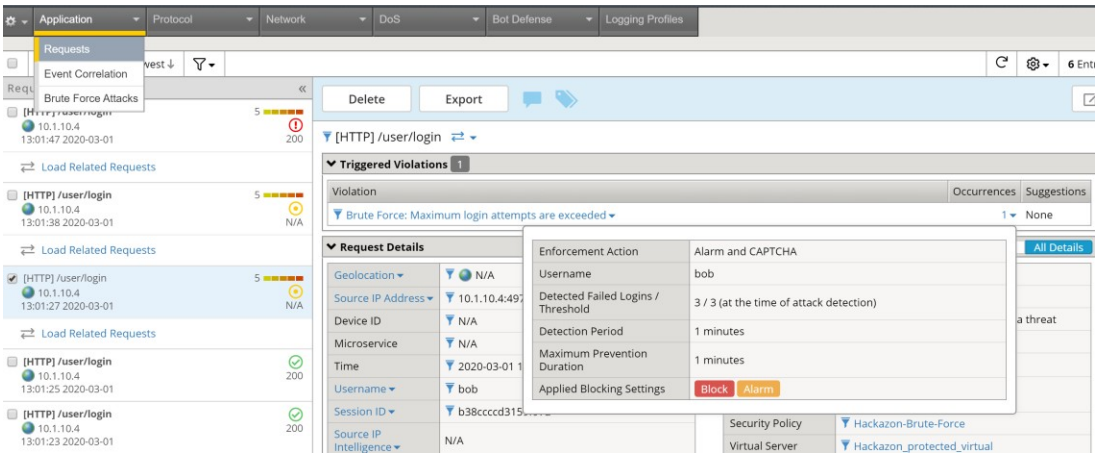
Exercise 2 – Source-based Brute Force Protection

Task 1 – Test Username source-based protection

- Open Firefox and browse to http://hackazon.f5demo.com.
- Click in **Sign-in**
- Try various passwords for the user '**bob**'



- After a few attemps Brute Force mitigation will kick in and a Captcha will return to the user
- Examine the Event Logs at **Security > Event Logs > Application > Requests** (Source-based Requests and mitigations are available at this log)



- Observe the Captcha mitigation will be sent to whatever client trying to authenticate as ‘bob’.
- Open Google Chrome for instance and try to authenticate with ‘bob’ meanwhile the Prevention Duration is in place BIG-IP ASM a Captcha will be returned as well.

Exercise 3 – Credential Stuff Brute Force Protection

Credential stuffing is a type of cyberattack where stolen account credentials typically consisting of lists of usernames and/or email addresses and the corresponding passwords (often from a data breach) are used to gain unauthorized access to user accounts through large-scale automated login requests directed against a web application.[1] Unlike traditional brute force attack, credential stuffing attacks do not attempt to brute force or guess any passwords - the attacker simply automates the logins for thousands to millions of previously discovered credential pairs using standard web automation tools like Selenium, cURL, PhantomJS or tools designed specifically for these types of attacks such as: Sentry MBA, SNIPR, STORM, Blackbullet and Openbullet. As a matter of curiosity, this website can tell if your credential

was leaked somehow: https://haveibeenpwned.com/.

Task 1 – Launching an automated brute force attack

- Open a Kali SSH terminal

  Note: If you have an SSH key to access UDF you could use them to log into the Kali client, if not, please, use the Web Shell SSH: On UDF, go to **Deployments > Advanced WAF Mitigation Techniques > Details > Components > Kali Linux > Details > Access Methods > Web Shell** link:

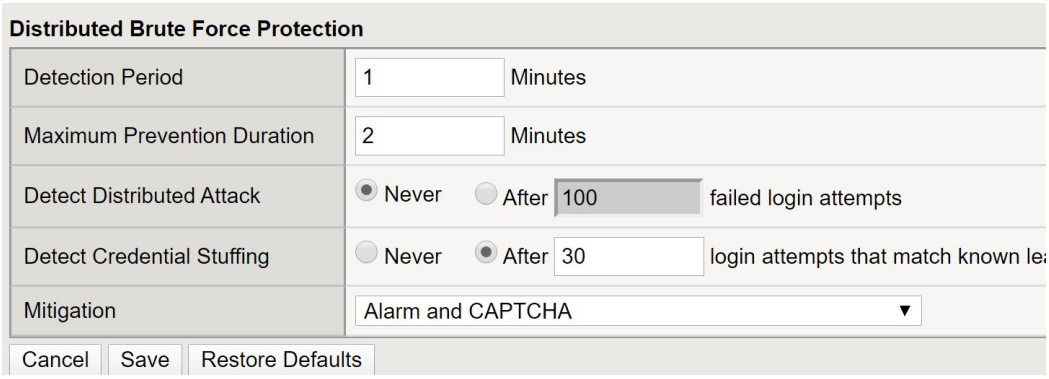
- Browse to /home/ec2_user/brute-force
  - `cd /home/ec2-user/brute-force`

    Note: it is possible to copy the command and past it onto the web shell using **SHIFT+INSERT** key combination for pasting

- Examine the file cred.txt with known leaked credentials (and valid Hackazon account on purpose)
  - `less cred.txt`
- Execute an automated brute-force attack using Hydra tool (there is an ASM signature for Hydra but it is disabled for this test)
  - `hydra -C cred.txt 10.1.10.58 http-post-form "/user/login:username=^USER^&password=^PASS^:incorrect"`
- After a couple of seconds hydra will identify valid users. Stop the tool using CTRL+C.
- Test the username + password combination found at the http://hackazon.f5demo.com
- Review **Security > Event Logs > Application > Requests** and confirm no request was blocked

Task 2 – Protecting against credential stuffing attack

- Navigate to Application **Security > Brute Force Attack Prevention**
- Click in **/user/login** and configure the **Distributed Brute Force Protection** as the following:



- Click in **Save** and **Apply Policy** in the sequence
- **Repeat Task 1** executing the automated attack using **hydra**
- Verify the outcome of the hydra. You should see something similar with lots of false positive:

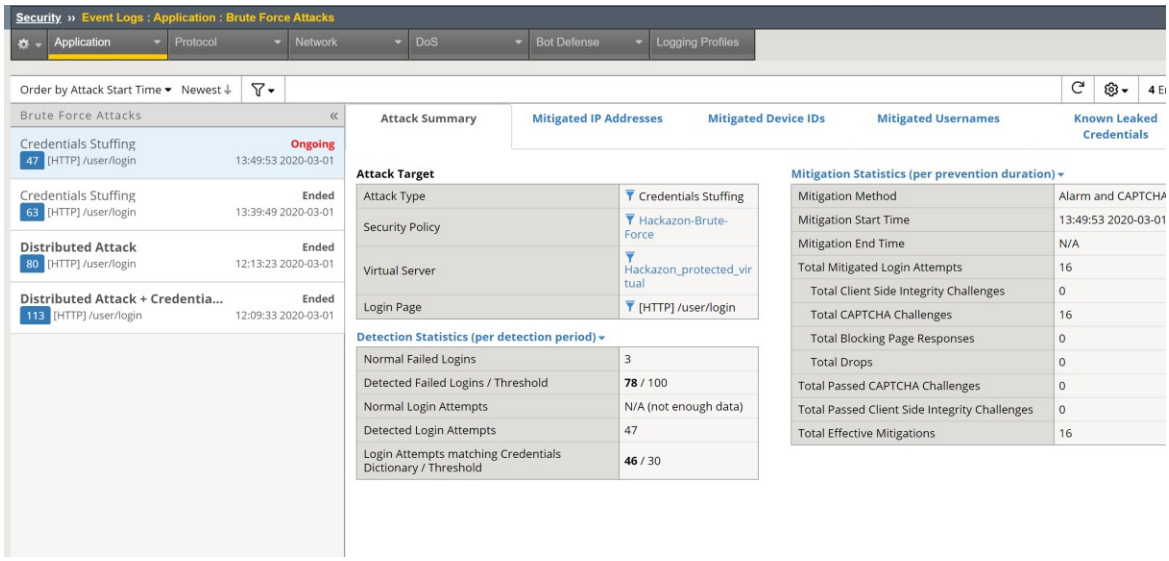
```
root@kali:/home/ec2-user/brute-force# hydra -C cred.txt 10.1.10.58 http-post-form "/user/login:username=^USER^&password=^PAS
Hydra v8.3 (c) 2016 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purpos

Hydra (http://www.thc.org/thc-hydra) starting at 2020-03-01 21:39:26
[WARNING] Restorefile (./hydra.restore) from a previous session found, to prevent overwriting, you have 10 seconds to abort.
[DATA] max 16 tasks per 1 server, overall 64 tasks, 858 login tries, ~0 tries per task
[DATA] attacking service http-post-form on port 80
[80][http-post-form] host: 10.1.10.58   login: cabtnmohamed011@gmail.com   password: p4a37gzU0MAI3299
[80][http-post-form] host: 10.1.10.58   login: election2014@mail.ua   password: october26
[80][http-post-form] host: 10.1.10.58   login: election2014@mail.ru   password: october26
[80][http-post-form] host: 10.1.10.58   login: mmm1432m@hotamil.com   password: 123456
[80][http-post-form] host: 10.1.10.58   login: ramyhakam1@gmail.com   password: PENCILSOFT71110
[80][http-post-form] host: 10.1.10.58   login: mohamednoman2001@gmail.com   password: tedata2000
[80][http-post-form] host: 10.1.10.58   login: mahsa1776.asgharian@gmail.com   password: 1776fm
[80][http-post-form] host: 10.1.10.58   login: rreza974@yahoo.com   password: 2231571R
[80][http-post-form] host: 10.1.10.58   login: texasbita@yahoo.com   password: 77!@#$^mil
[80][http-post-form] host: 10.1.10.58   login: clpostroet@drt.890m.com   password: ail/u/9/#inbox
[80][http-post-form] host: 10.1.10.58   login: ah2000693@yahoo.com   password: ah1031985
[80][http-post-form] host: 10.1.10.58   login: abdoelsonbaty1990@yahoo.com   password: bodybody10101990
[80][http-post-form] host: 10.1.10.58   login: 01229577236@yahoo.com   password: 123456
[80][http-post-form] host: 10.1.10.58   login: saeed.niaze28@yahoo.com   password: 09156541545s
[80][http-post-form] host: 10.1.10.58   login: ahmed-1983-12-21@hotmail.com   password: rahaf_2013
[80][http-post-form] host: 10.1.10.58   login: ahmed22man26@yahoo.com   password: 0557443019
[STATUS] 63.00 tries/min, 63 tries in 00:01h, 795 to do in 00:13h, 16 active
```

- Review **Security > Event logs > Application > Requests**
- Review **Security > Event logs > Application > Brute Force Attacks**

Observe that **Brute Force Attacks** report works only for the **Distributed Brute Force protection** section

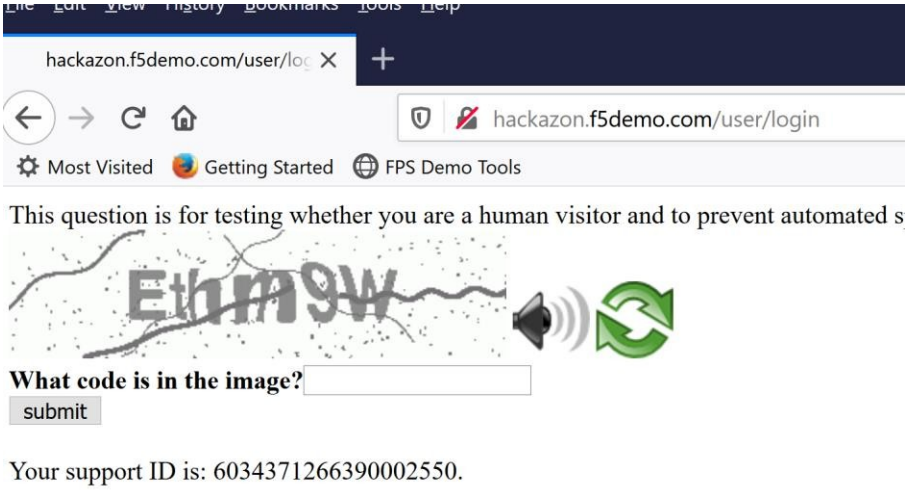
However, if there is an attack going on (either Distributed or Credential Stuffing) when a Source-based mitigation is also triggered, the statistics are counted into the ongoing Distributed/Credential stuffing attack.

- Browse the various options of the report below. Observe an amount of Captcha as Total CAPTCHA Challenges and Total Passed CAPTCHA Challenges. Hydra is not capable of passing a CAPTCHA challenge.



- While the attack is still going on, open Firefox and try to authenticate yourself in http://hackazon.f5demo.com

  You will observe a Captcha challenge will be send even if the client is a different client as well client IP address. This happens because the **Distributed Brute Force protection** mitigation applies to **ALL** users visiting your site.



- What would be a less intrusive mitigation to end users when detecting distributed/credential stuffing attack? Can you try that and observe the behavior? Comment with your colleagues/instructor.