F5 Advanced Web Application Firewall Documentation

Patrick Zoller

Dec 15, 2021

CONTENTS:

This series of lab exercises is intended to explain and demonstrate key features of F5 Advanced Web Application Firewall. The Blueprint which we use as base for all upcoming Modules is called **Advanced WAF Demo v16 + LCC, ML and Device ID+**.

Our intend is to proviode insights on how to provide demos on the following content:

Getting to Know the Environment

- Module 1: Lab Topology
- Module 2: How to Deploy a Solution

Class 1 - Getting started with WAF, Bot Detection and Threat Campaigns

- Module 1: IP Intelligence
- Module 2: Bot Detection Lab
- Module 3: Threat Campaigns
- Module 4: Transparent WAF Policy

Class 2 - Elevated WAF Protection

- Module 1: Bot Defense
- Module 2: Behavioral DOS Protection
- Module 3: DAST Integrations
- Module 4: Login Page Protection

Class 3 - Advanced Protection and Positive Security

- Module 1: Leaked Credential Check - Credential Stuffing
- Module 2: Check how Application Traffic Insights works
- Module 3: Offline Machine Learning
- Module 4: Protecting Credentials with DataSafe

To deploy a solution you must be logged into UDF (https://udf.f5.com).

F5ers can access the blueprint directly from UDF without launching a course.

Note: If a given topic is not highlighted currently on this page or something is incorrectly documented, please send a Teams Chat to Patrick Zoller. We will do our best to prioritize the development of the content based on demand.

CHAPTER

ONE

GETTING TO KNOW THE ENVIRONMENT

The F5 Advanced Web Application Firewall Solutions lab is the cornerstone of the Security SME team's continuing effort to educate F5ers, partners, and customers on ways to efficiently use F5 AWF. This Blueprint is comprised of multiple components including Windows Jumphost, Kali Linux, Docker Enviroment. . . just to name a few. This blueprint is under content revision in hopes to add additional capabilities for others to either consume existing solutions or to build new solutions that can be shared with the community.

1.1 Module 1: Lab Topology

In this module, we will talk about the Lab Topology of the "Advanced WAF Demo v16 + LCC, ML and Device ID+" UDF Blueprint.

Note: You´ll find different BIG-IPs inside the Blueprint. To maintain the Blueprint and introduce features which are EA, its easier to approach different BIG-IPs rather then configuring the solutions on one BIG-IP.

Lab Topology:

BIG-IP Component

- `BIG-IP 15.1 - Machine Learning Demo`

Note: This BIG-IP mainly used to Demo *Offline Machine Learning*. The feature *Offline Machine Learning* is currently in Beta.

- `BIG-IP 15.1.1 - Leaked Credential Check Demo`

Note: This BIG-IP mainly used to Demo *Leaked Credential Check Demo*. The feature *Leaked Credential Check Demo* is currently in Beta.

- `BIG-IP 16.0 - Generic Demos and Device ID+`

On this BIG-IP you can run Demos on following purposes:

- Device ID+
- IP Intelligence
- Bot Detection Lab
- Threat Campaigns

- Transparent WAF Policy
- Bot Defense
- Behavioral DoS
- DAST Integration
- Login Page protection

MORE TO COME

1.2 Module 2: N/A

1.3 Module 3: N/A

CHAPTER

TWO

CLASS 1 - GETTING STARTED WITH WAF, BOT DETECTION AND THREAT CAMPAIGNS

This class will focus on a best practice approach to getting started with F5 WAF and application security.

Class 1 will give you guidance on deploying WAF services in a successive fashion. This class focuses entirely on the negative security model aspects of WAF configuration.

Class 1 - All sections

- Module 1: IPI & Geolocation Labs
- Module 2: Bot Detection Lab
- Module 3: Threat Campaigns
- Module 4: Transparent WAF Policy

2.1 Module 1: IP Intelligence (IPI)

The purpose of this lab is to show the how IP Intelligence is working.

Note: The Lab is already pre-build. Meaning, you can show/check and demo IP Intelligence without configuring anything.

Note: IP intelligence is a subscription sevice which has been licensed for the demo.

Demo IP Intelligence - pre-configured

Steps:

1. 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 **user/user**.
2. Start Chrome and click on the Add-On called **X-Forwarded-For Header**.
3. Within here, you have already an IP configured which will trigger a IPI violation in the Event Logs of AWF.

If that IP has rotated out of the malicious DB, you can try one of these alternates:

- 80.191.169.66 - Spam Source
- 85.185.152.146 - Spam Source
- 220.169.127.172 - Scanner

- 222.74.73.202 - Scanner
- 62.149.29.36 - Spam Source
- 82.200.247.241 - Phishing
- 134.119.219.93 - Spam Source
- 218.17.228.102 - Spam Source
- 220.169.127.172 - Scanner

Exercise 1 – TASK 1 - Review IP Intelligence Log entries on Advanced WAF

Steps:

1. IP Intelligence is configured on BIG-IP named **BIG-IP 16.1 - All Demos**.
2. Login to BIG-IP via WebUI (The Password of the BIG-IP instance is listed within the **Details / Documentation** Tab).
3. Navigate to Security > Event Logs > Application > Requests and review the entries. You should see a Sev3 Alert for the attempted access to uri: /xff-test from a malicious IP.

Demo IP Intelligence - What has been configured

1. Navigate to Security > Application Security > Policy Building > Learning and Blocking Settings and expand the IP Addresses and Geolocations section.

Note: These are the settings that govern what happens when a violation occurs such as Alarm and Block. We will cover these concepts later in the lab but for now the policy is in blocking but within the IPI configuration is we enabled **alarm** only.

---

1. Navigate to Security › Application Security > Security Policies > Policies List.
2. Select the Security Policy named “Hackazon-WAF-IPI”. Within the “Policy Configuration” choose “IP Intelligence”.
3. Notice at the top right that IPI is “ON” and **alarm** is configured for each category.

1. For the demo to work, **XFF inspection** in the WAF policy is enabled via to Security > Application Security > Security Policies > Policies List > Hackazon-WAF-IPI.

1. Navigate to Local Traffic > Virtual Servers and click on VS named **vs_Hackazon_III**.

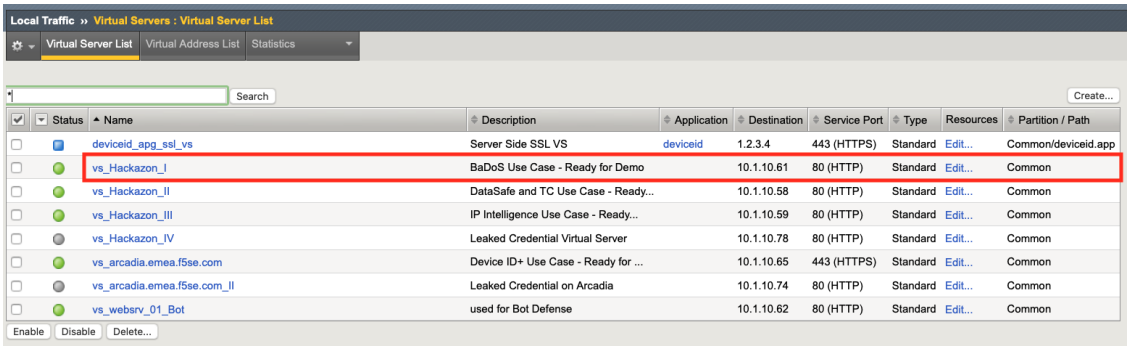
2. You´ll notice withing the Configuration that we used a HTTP Profile (Client) with XFF enabled.
3. Under the Security tab in the top middle of the GUI click on Policies and your policy settings included a Application Security Policy named **Hackazon-WAF-IPI** and a Log Profile named **ASM-BOT-DoS-Log-All**.

Note: It is best practice to enable Trust XFF in the policy when configuring IPI via WAF policy. XFF inspection is one of the advantages to consider when deploying IPI and can only be done via WAF policy. Although this setting is not needed to demonstrate this lab, it is strongly recommended to have it enabled. Attackers often use proxies to add in source IP randomness. Headers such as XFF are used to track the original source IP so the packets can be returned. In this example the HTTP request was sent from a malicious IP but through a proxy that was not known to be malicious. The request was picked up at Layer 7 due to the WAF's capabilities. This demonstrates the importance of implementing security in layers.

2.2 Module 2: Bot Detection Lab

In this Lab we want to get familar with the Bot Detection Capabilities of AWAF. The goal is to create and apply a transparent Bot Defense Profile (signatures only) and enable logging for Bot requests.

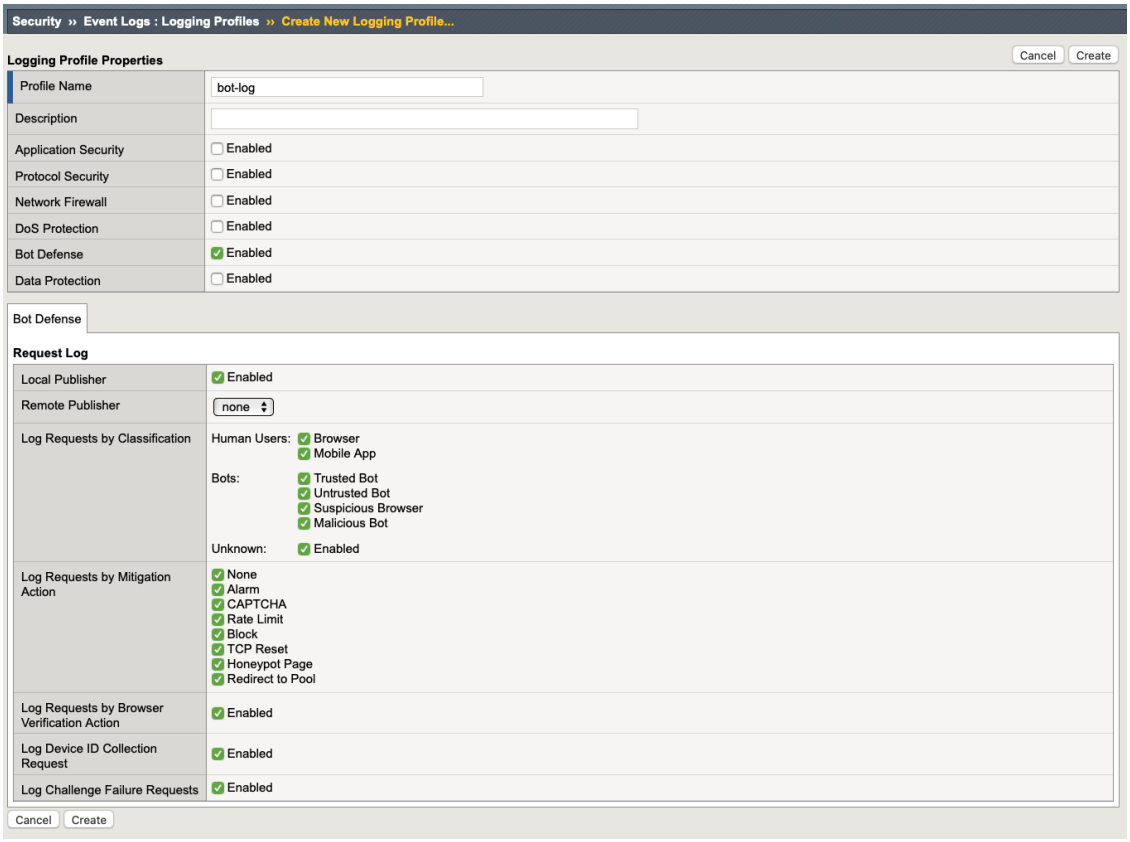
Important: To only focus on Bot Defense, we will use the "vs_Hackazon_I" virtual server for this, because there is no WAF policy attached to it. If you wanna use a different VS, please make sure that there is no WAF policy active.



2.3 Create Logging Profile

Note: The **"vs_Hackazon_I"** virtual server already has a Logging Profile attached to it, which can be used for this demo. In case there is no Logging Profile attached or you want to create your own profile for this demo, use the steps described below.

1. Navigate to **Security > Event Logs > Logging Profiles** and create a new Logging Profile with the settings shown in the screenshot below (local publisher with all options enabled).
2. Give it a name and click **create**.

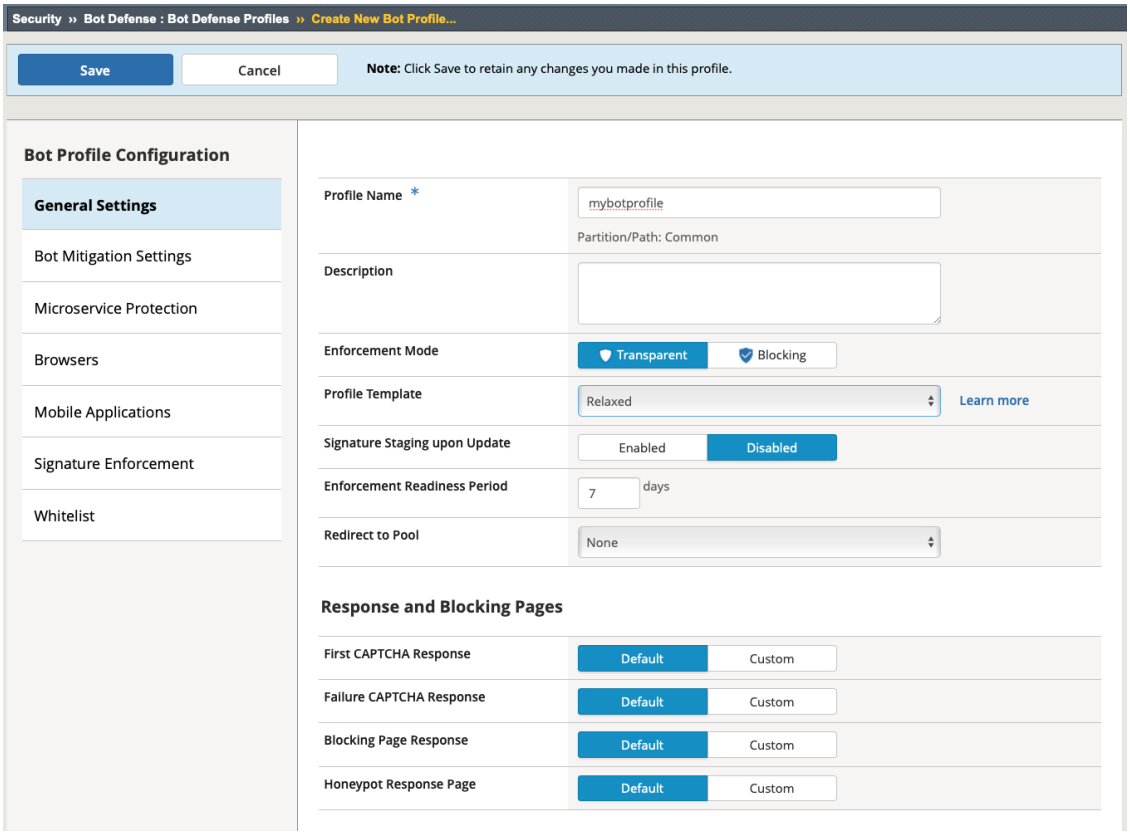


2.4 Create Bot Defense Profile

Note: The **"vs_Hackazon_I"** virtual server already has a Bot Defense Profile attached to it, which can be used for this demo. In case there is no profile attached or you want to create your own for this demo, use the steps described below.

1. Navigate to **Security > Bot Defense > Bot Defense Profiles** and click **Create**.

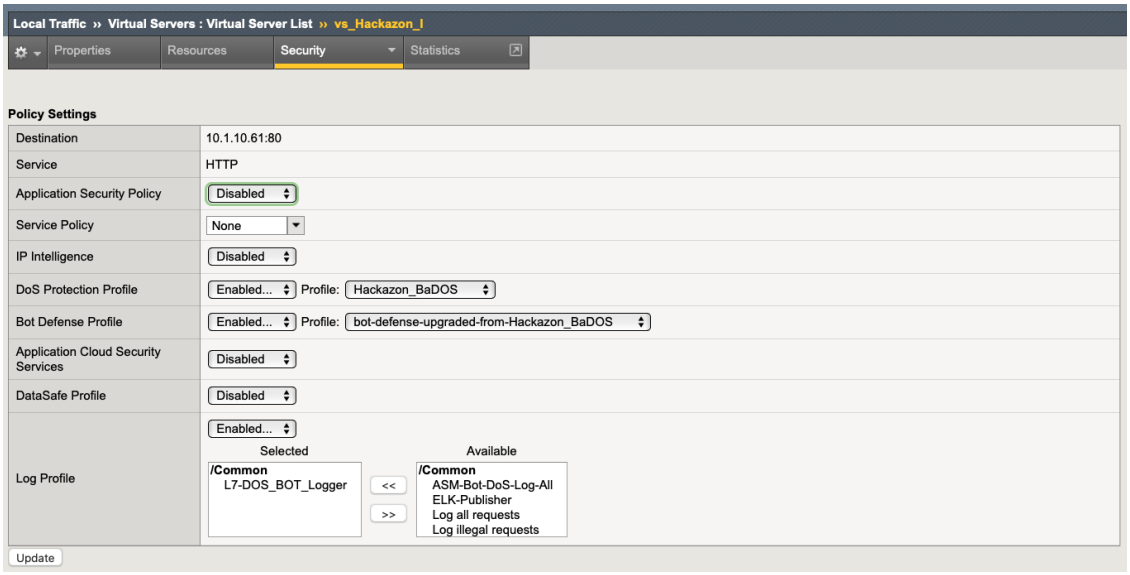
2. Choose a name (e.g. mybotprofile) and set the Enforcement mode to **transparent**. Review the **Bot Mitigation Settings** and **Signature Enforcement**, but leave all settings on default for now (We will cover more options in **Class 2 / Module 1**).
3. Click **Save**



2.5 Enable Bot Defense and Logging

1. Navigate to **Local Traffic > Virtual Servers > Virtual Server List > vs_Hackazon_I**
2. Click on the **Security** Tab and click **Policies**.
3. Enable Bot Defense and Logging with the profiles created before. (as mentioned before, you can use the pre-configured settings for this demo)
4. Click **Update**

Note: Make sure there is either the existing Logging Profile: **L7-DOS_BOT_Logger** or the **new created** Logging Profile attached to this VS.



2.6 Start generating Traffic

1. Open a ssh session to the Kali system.

   Note: To open a ssh session to UDF you need to provide your public key. For more information, please refer to the UDF documentation.

   ---

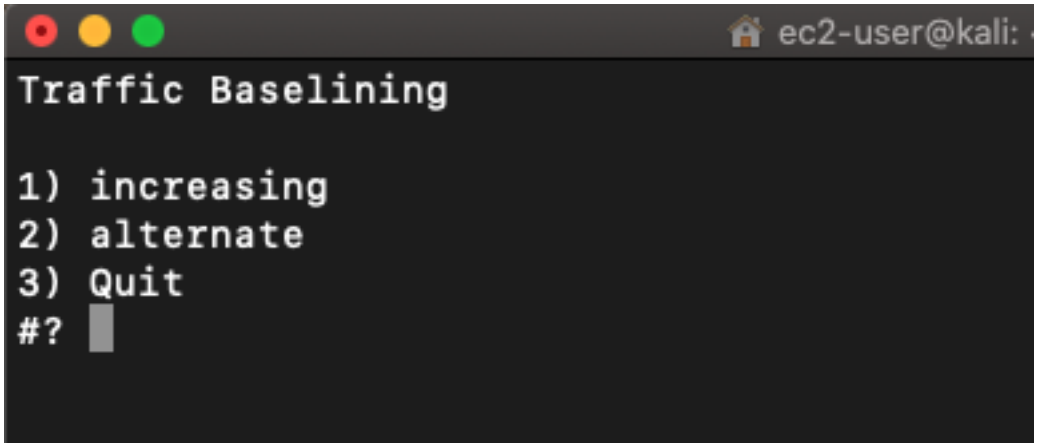
2. make sure you are in the directory:

   ```
   /home/ec2-user
   ```

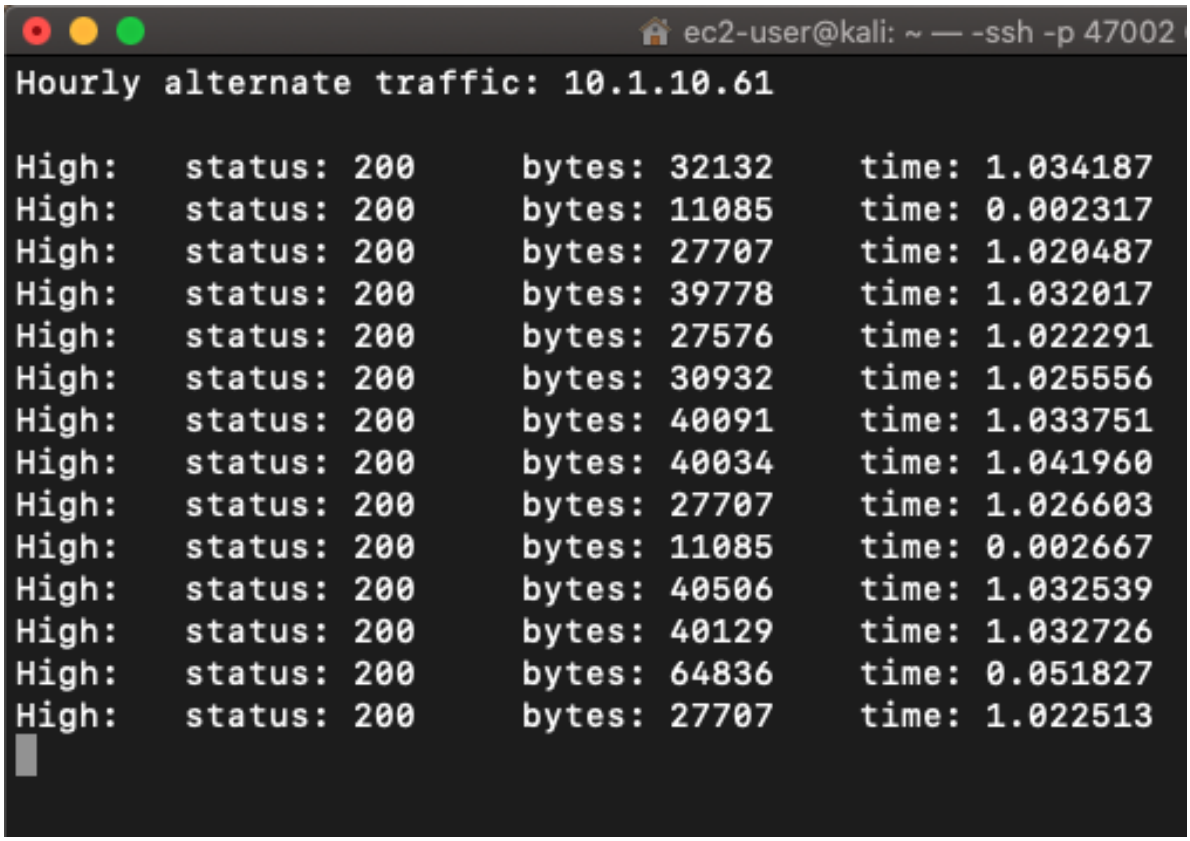
3. start generating traffic by using the script **"baseline_menu.sh"**:

   ```
   sudo su
   screen + press ENTER
   ./baseline_menu.sh
   choose 1
   de-attach by clicking Ctrl+a+d
   screen
   ./baseline_menu.sh
   choose 2
   de-attach by clicking Ctrl+a+d
   ```

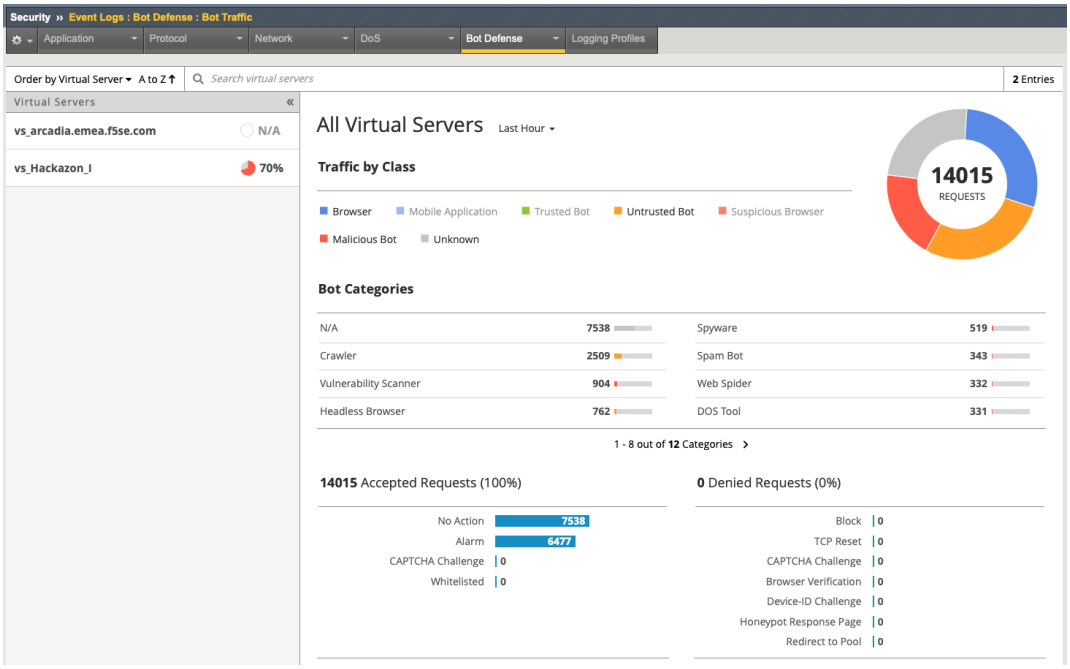
4. Activate both options:



it should look like this:

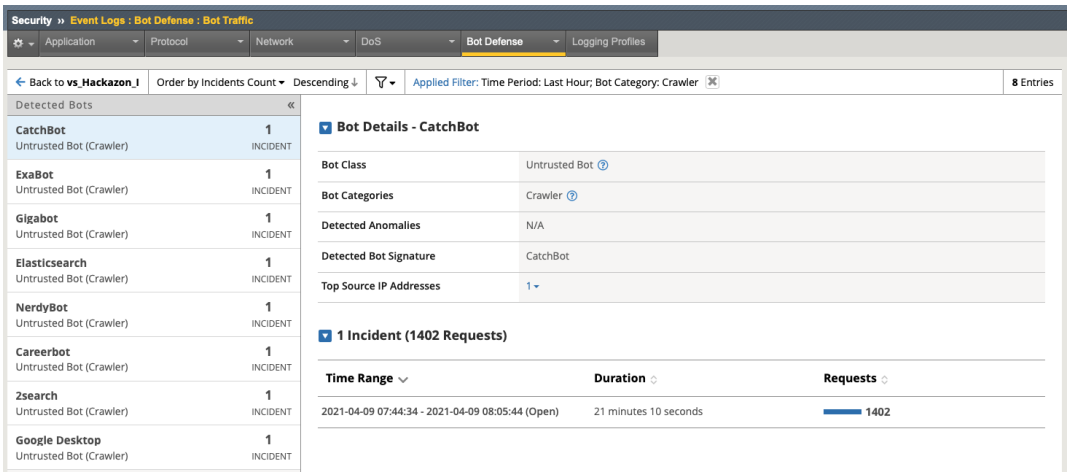


5. Navigate to **Security > Event Logs > Bet Defense > Bot Traffic** and review the Dashboard. Click on the "vs_Hackazon_I" VS to see more details for this specific Application.

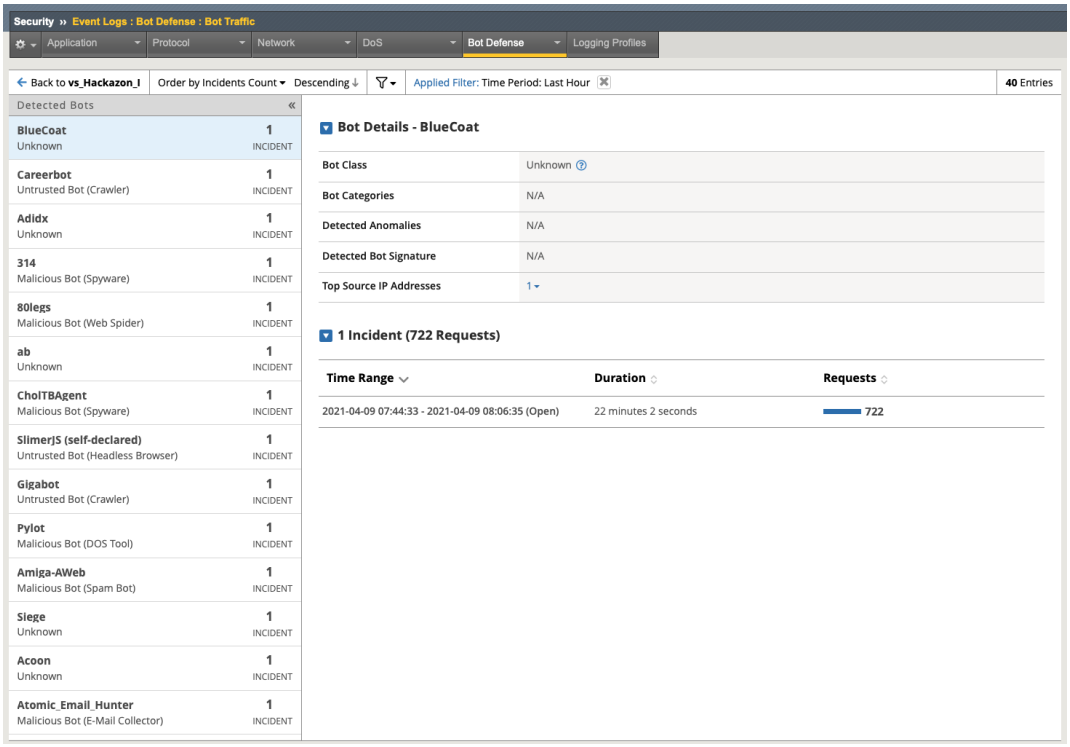


Note: It may take some time before you can see some results.

6. Click on any Bot Categories to see detected Bots (per category)



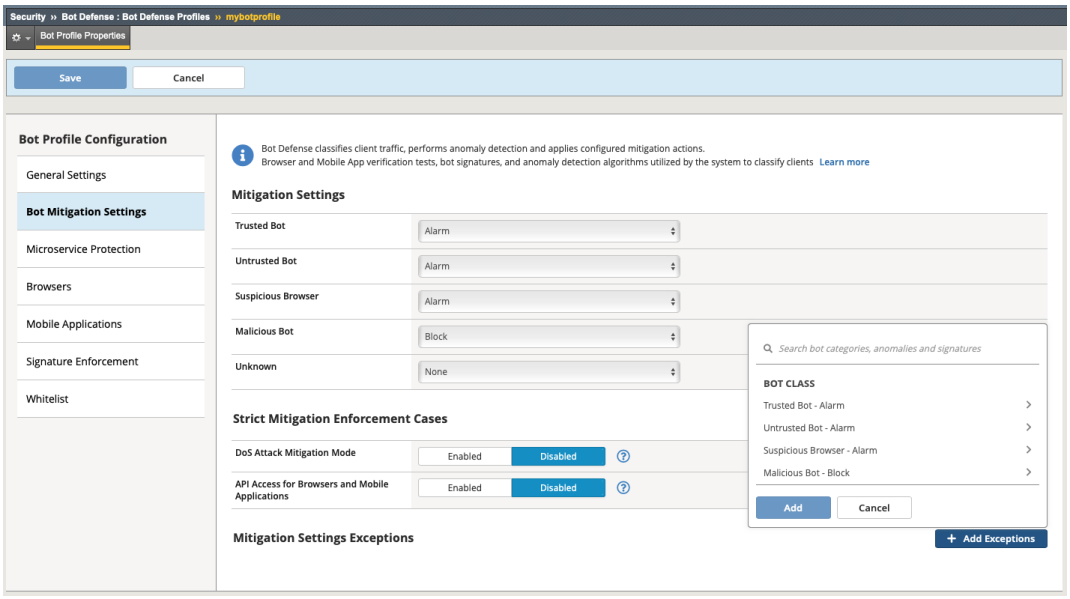
7. Go back to the Start Dashboard ans click on “detected Bots” to see all.



2.7 Override settings and create execptions for specific bots

Note: It may occur, that some Bots are detected as false positives and/or the false mitigation action will be applied. In this case, you can create exceptions to override the default settings per bot.

1. Navigate to **Security > Bot Defense > Bot Defense Profiles** and click **on the profile** (either your **own** or the preconfigured **bot-defense-upgraded-from-Hackazon_BaDOS** profile).
2. Click on **Bot Mitigation Setings**
3. On the Bottom, click on **Add Exception**



Note: The system automatically stores all seen bots (and based on signatures) sorted by classes and categories.

4. In the search field type in: **curl** to filter for this specific type, select curl (category: untrusted bot) and click add.
5. You now can define a specific action for curl, which overrides the global action for this category (untrusted bot). Exceptions are are on a per profile basis. Change the action to “block” and click “Save”.
6. Open a Terminal Server Session to the “Windows Client System” and run the “01-Curl-Bot” batch-file, located on the Desktop.
7. Back in TMUI navigate to **Event Logs > Bot Defense > Bot Requests** verify the requests seen.

Note: As the baseline script is still running, it may be needed to search for a specific log entry. Click the filter icon and select “denied”, to display only blocked requests.

Congratulations! You have just completed class 1 - module 2.

See class 2 - module 1 for more advanced configuration.

2.8 Module 3: Threat Campaigns

2.9 Module 4: Transparent WAF Policy

CHAPTER

THREE

CLASS 2 - ELEVATED WAF PROTECTION

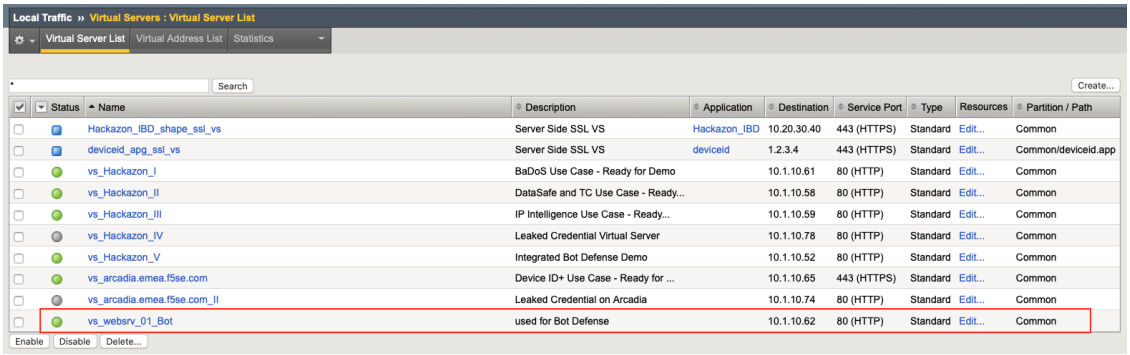
This class will focus on a best practice approach to elevating your WAF protection after becoming comfortable with the content of Class 1. This is the 2nd class in a three part lab series based on: Succeeding with Application Security which closely maps to this visualization of layered Application Security.

Here is a complete listing of all topics we cover in Class 2:

3.1 Module 1: Bot Mitigation Lab

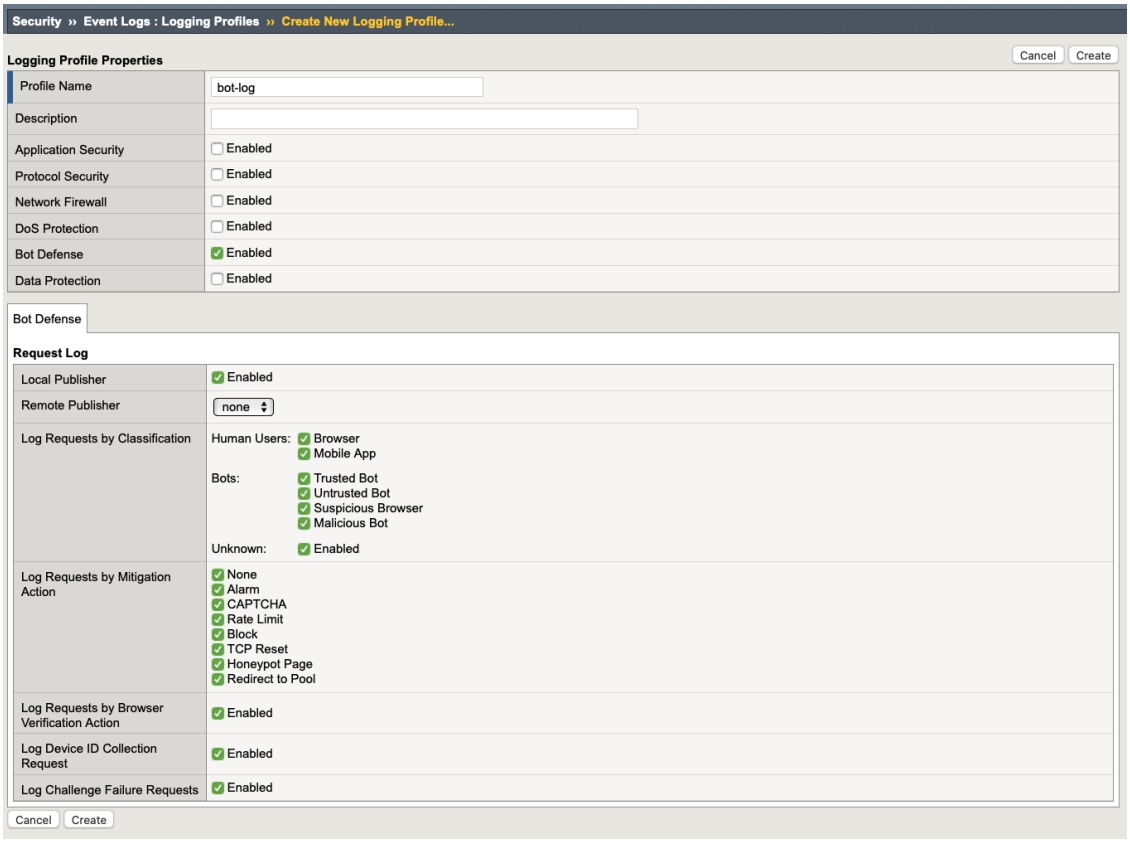
In this Lab we want to get familar with all the additional features avaialble for Bot Defense. The goal is to understand the difference between signature-based and JavaScript-based Detection capabilities and mitigation options.

Important: To only focus on Bot Defense, we will use the "vs_websrv_01_Bot" virtual server for this, because there is no WAF policy attached to it. If you wanna use a different VS, please make sure that there is no WAF policy active.



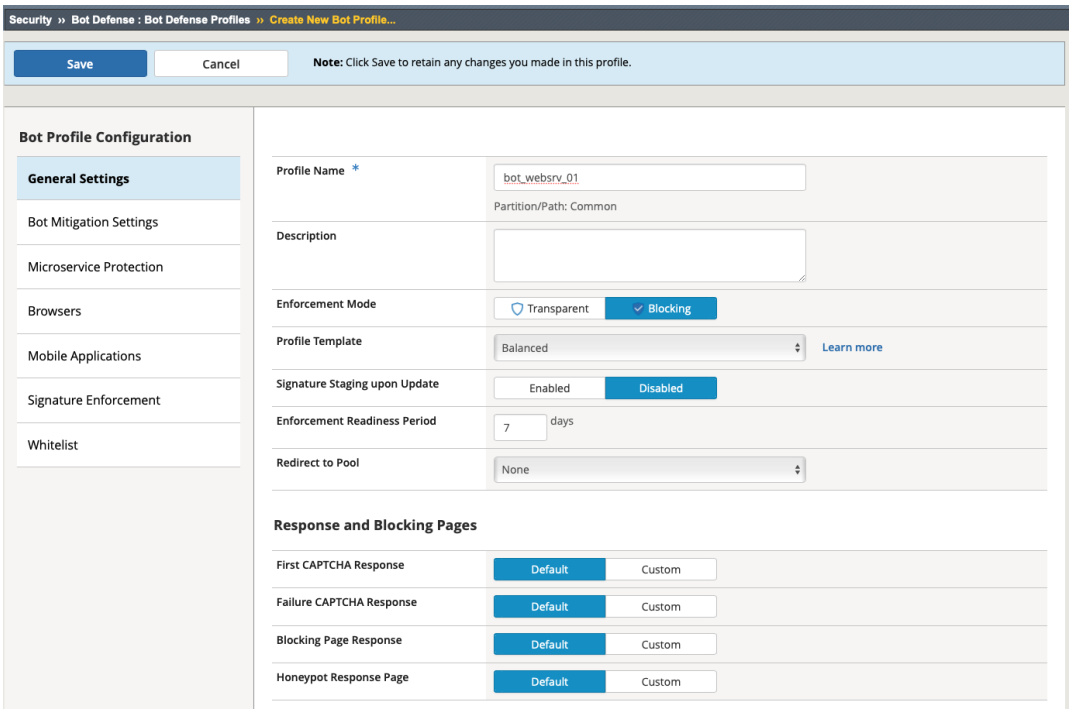
3.2 Create Logging Profile

1. Navigate to **Security > Event Logs > Logging Profiles** and create a new Logging Profile with the settings shown in the screenshot below (local publisher with all options enabled).
2. Give it a name and click **create**.

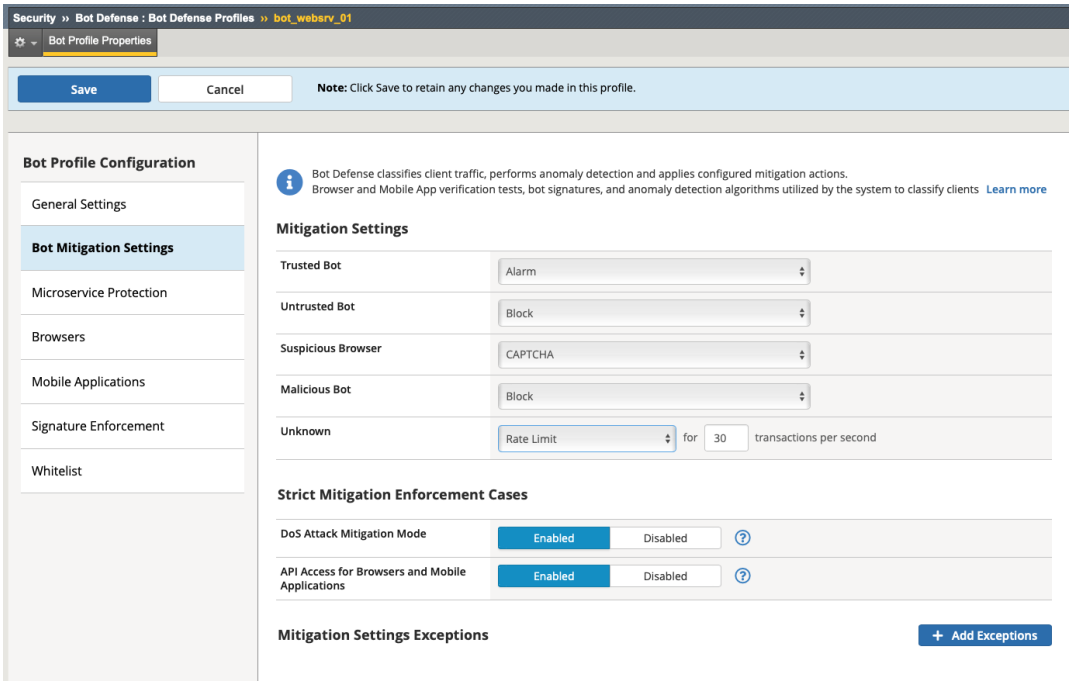


3.3 Create Bot Defense Profile

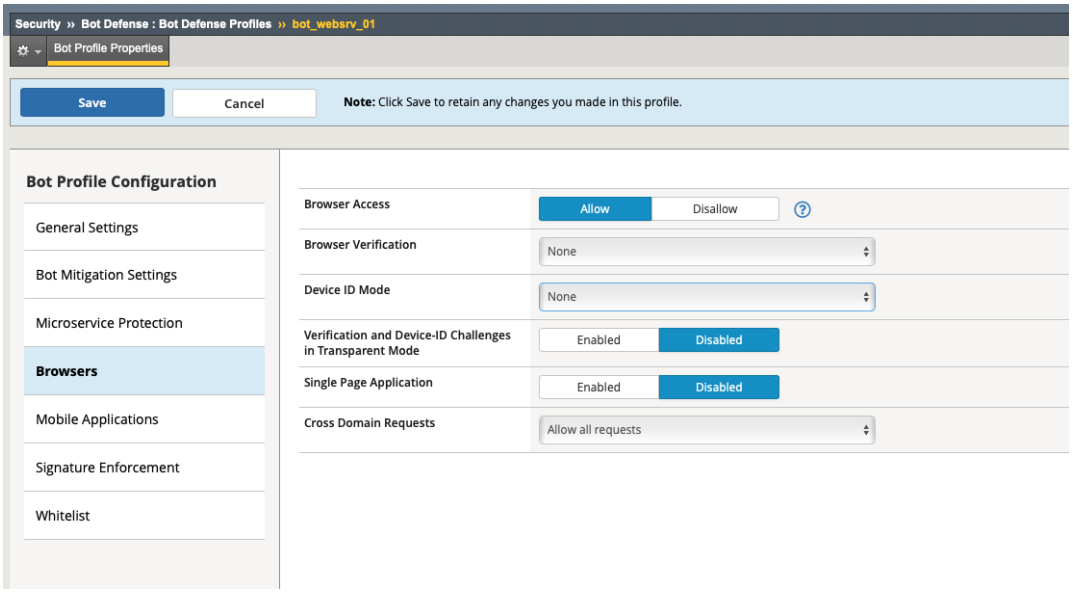
1. Navigate to **Security > Bot Defense > Bot Defense Profiles** and click **Create**.
2. Choose a name (e.g. mybotprofile) and set the Enforcement mode to **blocking**.



3. Go to Mitigation Settings and change it as seen in the picture below. Leave all other settings as default.



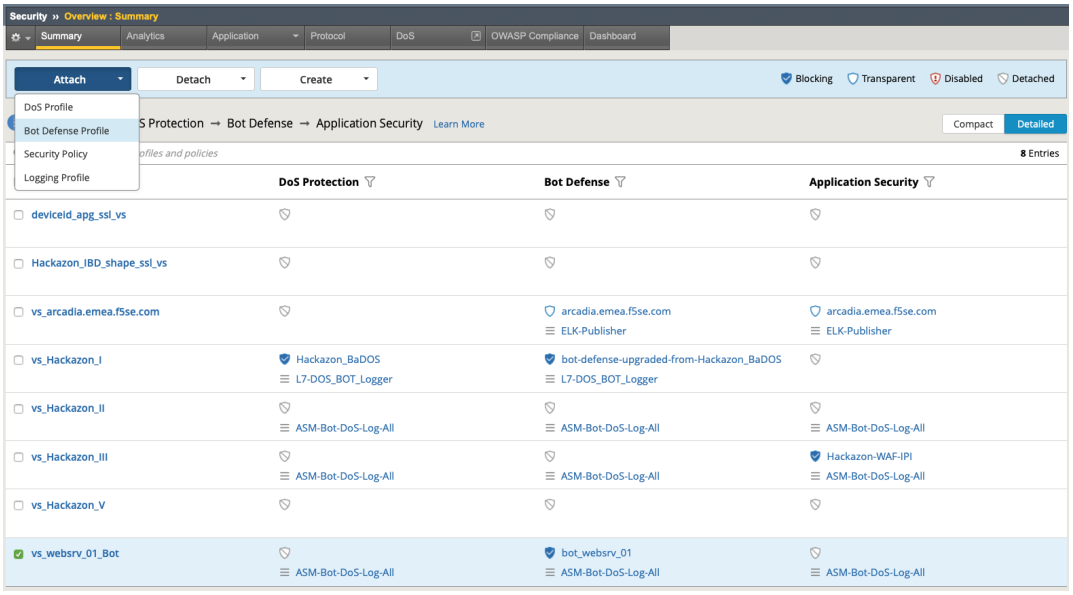
4. Go to **Browsers** and make sure that **Browser Verification** and **Device ID Mode** are disabled (none). Leave all other settings as default.



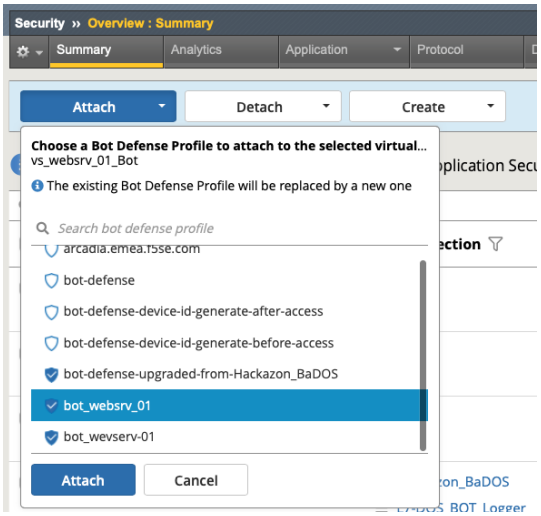
5. Click **Save**

3.4 Enable Bot Defense and Logging

1. Navigate to **Security > Overview** and select the "vs_websrv_01_Bot" Virtual Server
2. Click on **Attach** and select **Bot Defense Profile**.



3. Choose the profile you've just created and click **Attach**

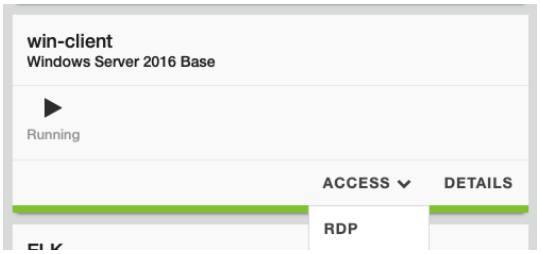


4. Do the same for the **Logging Profile** and use the profile you've just created.

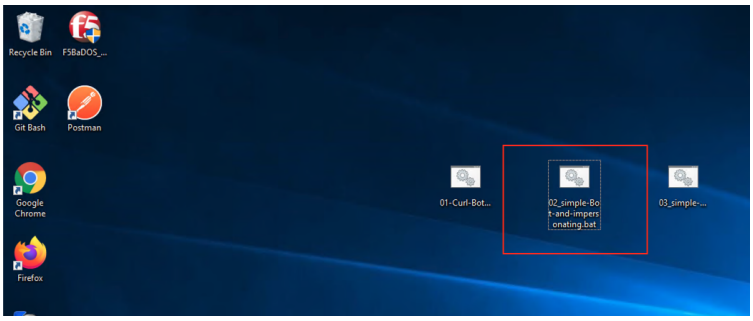
3.5 Create and review simple Bot-Requests

We will use the "win-client" virtual machine provided by this deployment to create simple Bot-Requests.

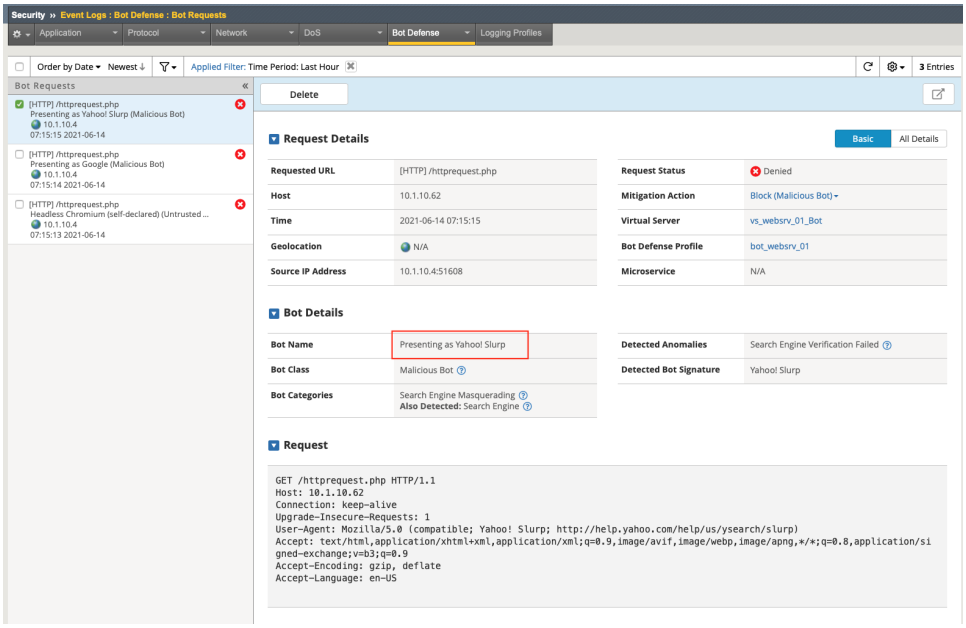
1. Open the RDP session



2. Double-click on the "02-Simple-Bot-and-impersonating.bat" batch file located on the desktop. This will generate three different requests.



3. Go back to the TMUI and click on: **Security > Event Logs > Bot Defense > Bot Requests**

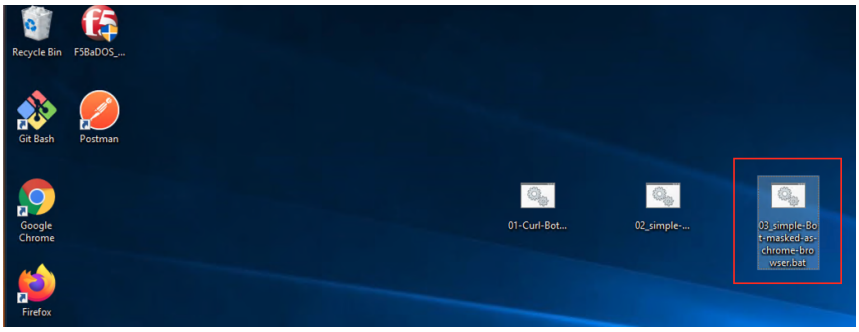


4. Review all (three) logs and see the "block" reason for each request. All requests where classified as malicious bots with the attempt to masquerade as a good bot (i.e. search bot).

Note: All requests were made with curl and customized user agents to simulate different requests/attacks.

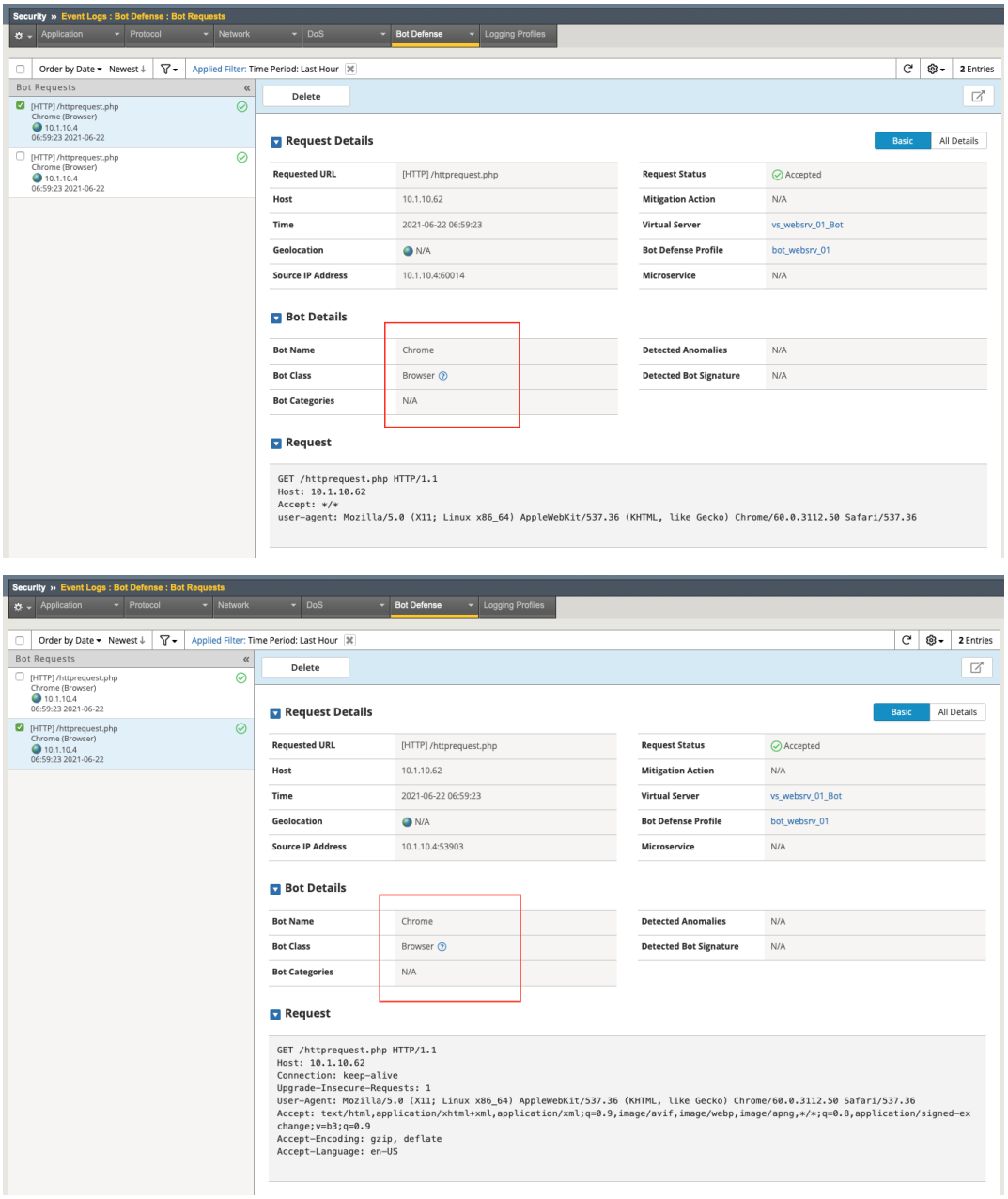
---

5. Go back to the Windows client and double-click on the "03-Simple-Bot-masked-as-Chrome-Browser.bat" batch file.

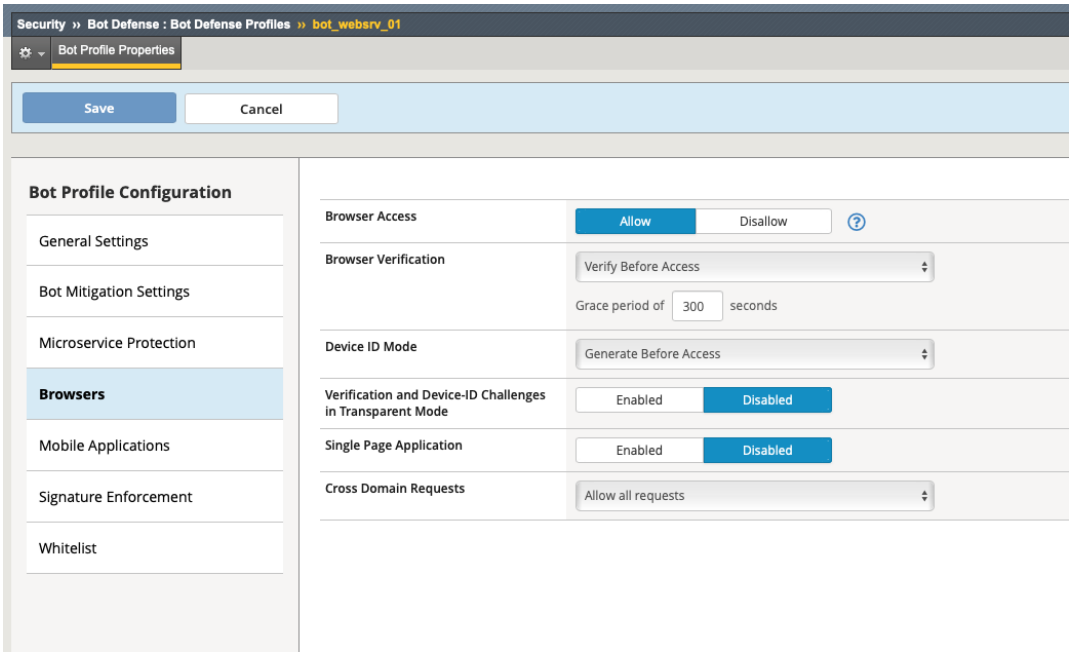


6. Go back to the Eventlog and review the result for this request. As you can see both requests were classified as a valid Browser and were allowed. Lets see how we can get more accurate results.

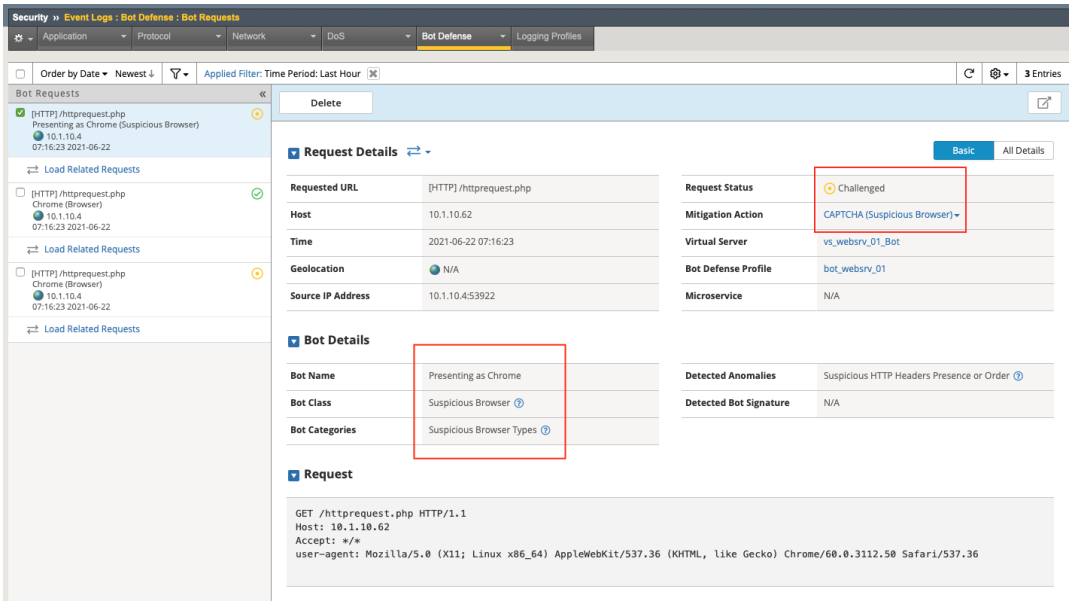
Note: One request was made with curl and a customized user agent, but the other one was made with a headless chrome and a customized user agent to simulate different bots but masked as valid browsers.



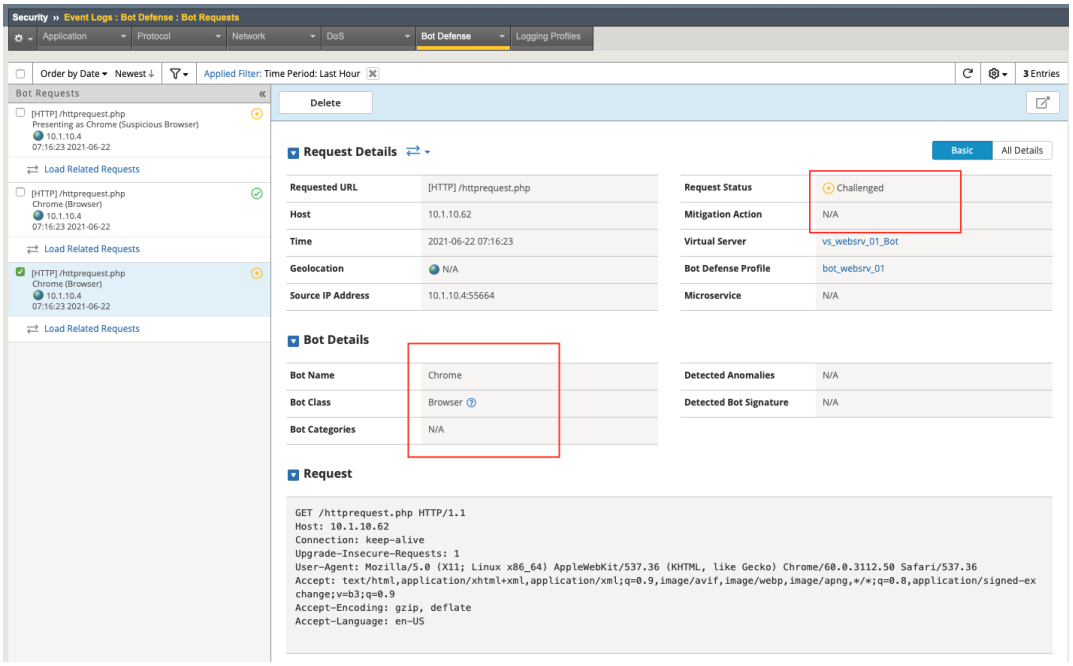
7. Go to **Security > Bot Defense > Bot Defense Profiles** and select our Bot Defense Profile (bot_websrv_01)
8. Within the profile go to **Browsers** and set "Browser Verification" to **Verify Before Access** and "Device ID Mode" to **Generate Before Access**.

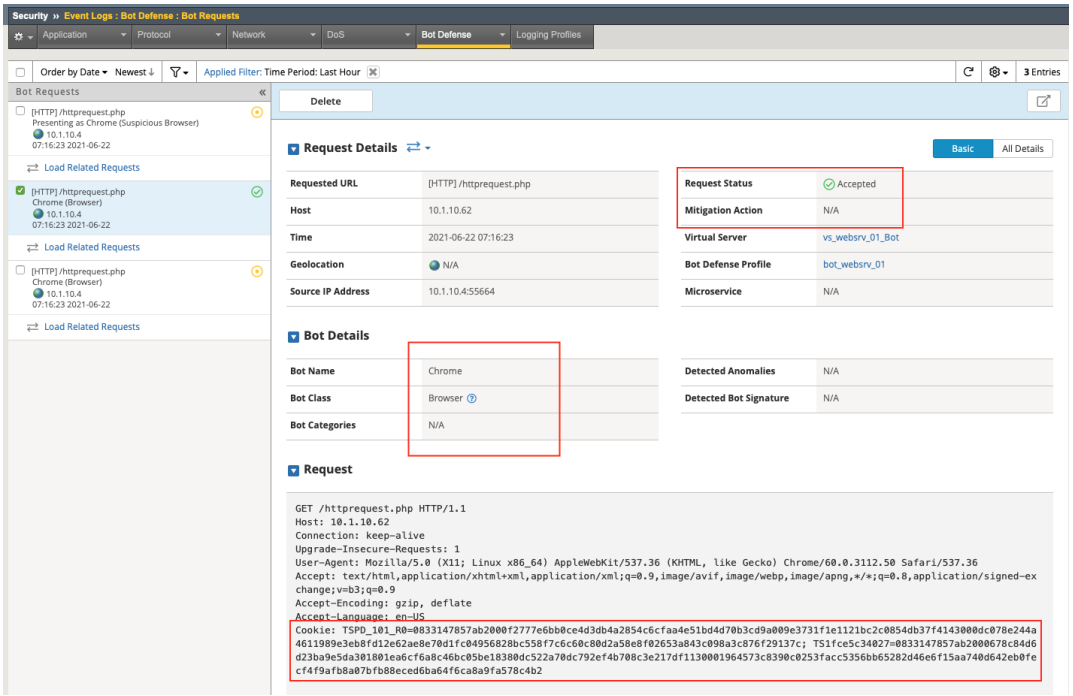


9. Click **save** and go back to the Windows Client RDP Session.
10. Double-click again on the "03-Simple-Bot-masked-as-Chrome-Browser.bat" batch file and review the log entries in the TMUI.
11. As you can see, one request (made with curl) was classified as "suspicious Browser" and the status is "challenged".



12. The second one (made with headless chrome and a customized user agent) was classified as "Browser" and also challenged. But this time the automated browser was able to solve the JS challenge and the request was allowed.





Note: This is not part of this LAB but it can be identified with the "CSHUI" part of Bot Defense (Client Side Human Intercation and Counting Anomalies"). It is based on ongoing checks, while the user browses through the application and is looking at HTML responses, for Mouse / Keyboard / Touch anomalies, Rapid surfing or session opening and many others.

Note: Shape Solutions can provide the same and even more accurate results because of the more advanced JS and the AI based classification.

3.6 Module 2: Behavioral DOS (BADOS) Protection

F5's AWAF product include advanced functionality for defending L7DoS attacks. While there are well described self-paced labs available on Clouddocs, the goal of this article is to focus on the Behavioral DOS Dashboard improvements, rather showing the Behavioral DOS Protection configuration.

Note: If you are intrested into the Behavioral DOS Protection configuration, please visit Introduction to L7 Behavioral DoS.

Behavioral DOS Protection Overview

With TMOS Release 14.1 a new enhanced BADOS (Layer 7 Behavioral DoS) Dashboard was introduced. Use Case of the new Dashbaord: A customer would use this dashboard to get visibillity into Denial of Service attacks which are tracked and mitigated Behavioral DoS.

Note: The dashboard is reached via **Statistics > Dashboard and selecting Behavioral DoS** from the menu of dashboards.

The new dashboard provides:

- Client HTTP Transactions
- Client HTTP Requests & Transactions
- Server HTTP Transactions
- Concurrent Server-Side Connections
- Server Stress
- Server Queue
- TLS Handshake
- Connections Mitigation
- Layer 3-4 & SSL Mitigations
- HTTP Mitigation
- CPU - Utilization
- Memory - Utilization

The new Dashboard also provide the ability to:

- Zoom-in and Zoom-out
- Show legends of each chart, on placing mouse on any chart
- Custom time selection view

When an Attack is been recognized the new Dashbaord will provide details about:

- Status of the Protected Application - change from "Calm" to "Attack"
- Attack ID
- Information on the Server Stress Level

- Mitiagtion Action

3.7 Module 3: DAST Integration

3.8 Module 4: Login Page protection

CHAPTER

FOUR

CLASS 3 - ADVANCED PROTECTION AND POSITIVE SECURITY

This lab will focus on how to progress your application security to the limits of what F5 WAF can offer.

This is the 3rd Class which is mainly based on: Succeeding with Application Security which closely maps to this visualization of layered Application Security.

Here is a complete listing of all topics we cover in Class 3:

4.1 Module 1: Leaked Credential Check - Credential Stuffing

In this module, we will Demo Leaked Credentials Check (LCC) to detect ATO attacks and provide a compliance check for NIST guidance with a single login attempt with leaked credentials.

> **Warning:** Don´t use the Application ID / Access Token outside of F5 and for production. Application ID / Access Token to be used for demo purposes only!!!

Leaked Credentials Check Overview

Leaked Credential Check (LCC) provides access to a database of compromised credentials, which can be used to detect ATO attacks and provide a compliance check for NIST guidance. LCC is an add-on feature to BIG-IP Advanced WAF which been consumed as a subscription service.

Advantages of F5 Leaked Credential Check (LCC)

- Integrated with F5 Advanced WAF; Can be enabled within minutes.
- LCC policy can be managed and enforced by the application owner.
- Flexible deployment options based on number of users; Not based on number of AWF instances.
- Policy configuration and enforcement can be automated via tmsh CLI and BIG-IP REST API.

Flow trough the Demo

LCC is configured on BIG-IP named `BIG-IP 16.1 - All Demos`. Login to that BIG-IP instance to check the LCC configuration. The Password of the BIG-IP instance is listed within the `Details / Documentation` Tab.

1. Within *Security › Application Security* > *Security Policies* > *Policies List* you´ll notice a Security Policy named `LCC`.
2. The Policy is attached to Virtual Server `arcadia.emea.f5se.com_vs` and `Hackazon_protected_vs`.

3. Check the LCC configuration under *Security › Cloud Services > Cloud Security Services Applications > f5-credential-stuffing-cloud-app*.
4. You will notice a predefined API Key ID and API Key Secret configuration. Additional the Endpoint which will be used is called `f5-credential-stuffing-blackfish`.

Note: In the 15.1.1 the colour of the traffic light only reflects what happened when a credential check attempt was last made. Before that the icon will stay blue (unknown). lt is not a health monitor which can be used to indicate the current state of the service or whether the service has expired etc.

---

1. The LCC feature is configured within the Brute-Force Protection profile. Normally a login page is specified for the login credentials to be captured by Advanced WAF. The information required to manually identify the login URL can be found by reviewing the HTML source code and snooping the HTML traffic generated as a user logs into the site (e.g. keyboard F12).

Note: There is also the option to create login pages automatically Creating Login Pages for Secure Application Access.

---

1. *Leaked Credential Detection* is enabled within the Brute Force Protection configuration.

2. The following mitigation actions can be configured as an *Action*:

Action	Description
Alarm	report the Leaked Credentials Detection violation in event log
Alarm and Blocking Page	report the Leaked Credentials Detection violation in event log and send the Blocking Response Page
Alarm and Honeypot Page	report the Leaked Credentials Detection violation in event log and send the Honeypot Response Page
Alarm and Leaked Credentials Page	report the Leaked Credentials Detection violation in event log and send the Leaked Credentials Page

1. Within that demo `Learning and Blocking Settings` for Leaked Credential Detection have been set to `Alarm` and `Block`.

2. The Honeypot Page and the Leaked Credentials Page can be configured in the Response and Blocking Pages screen (see screenshot below).

3. **RDP to windows machine called *win-client*. The Password of the instance is listed within the `Details / Documentati`**

    1. Launch Chrome. Spot the Folder called `Leaked Credentials Check demo`.
    2. Choose the bookmark called `Hackazon -- Login`.
    3. Login with username `demo33@fidnet.com` and password `mountainman01`
    4. Your login is blocked by LCC as those credentials are known as leaked credentials.
    5. Alternatively you can also select the Arcadia bookmark in the `Leaked Credentials` Chrome Folder and you can also try other username/password combinations like usernam `admin` with password `12345678`.

4. Go back to to the BIG-IP instance to check in the request log for the blocked request with the Leaked credentials detection violation.

Demo Leaked Credentials Check with a Script

Note: In this demo you can do it without ASM enabled first - Hydra will find credentials and password that worked, and then do it with ASM enabled.

---

1. **Remove ASM policy named `LCC` from Virtual Server `Hackazon_protected_virtual` on BIG-IP Instance `BIG-IP`**

    1. Launch the attack:

2. SSH or use Web Shell of UDF Instance called `kali`.
3. Run `sudo su`.
4. Check you are in directory */home/ec2-user*, else move to this directory.
5. Launch the Brute Force stuffing attack (be careful, copy paste does not work every time because of the “”).
6. `hydra -C cred_list.txt -V -I 10.1.10.78 http-form-post "/user/login?return_url=:username=^USER^&password=^PASS^:S=My Account"`. This is the VS on the BIG-IP named `Leaked Credential Check Demo`.
7. Within your Putty or Web Shell Session You should see one line with `[80][http-post-form] host: 10.1.10.78 login: demo33@fidnet.com password: mountainman01`. This means attack passed with this credential.

1. Login to Hackazon (demo1/demo1 or with the previous stolen cred), to show it works and that there is no Captcha.

2. Try with a distributed attack. Here we simulate a Bot network sending a Credential Stuffing attack with thousand leaked credentials.
    1. Enable ASM policy `LCC` on VS `Hackazon_protected_virtual`.
    2. SSH or use Web Shell of UDF Instance called `kali`.
    3. Check you are in directory */home/ec2-user*, else move to this directory.
    4. Launch the Brute Force stuffing attack (be careful, copy paste does not work every time because of the “”).
    5. `hydra -C cred_list.txt -V -I 10.1.10.78 http-form-post "/user/login?return_url=:username=^USER^&password=^PASS^:S=My Account"`. This is the VS on the BIG-IP named `Leaked Credential Check Demo`.
    6. Keep attack on going and RDP to windows machine called `win-client`.
    7. Launch Chrome and click Hackazon login bookmark.
    8. Login as demo1 / demo1, you should see a Captcha. You are a legitimate user, but the website is protecting itself. Proof you are a legitimate user by answering the CAPTCHA.
    9. Go to BIGIP and check Brute Force and cred stuffing logs *Security* > *Event Logs* > *Application* > *Brute Force Attack*.

Additional information

The following cloud related commands could help to identify whether the cloud connection is working.

1. `tmsh show security cloud-services application-stats`

2. `tmctl app_cloud_security_service_stat`

4.2 Module 2: Check how Application Traffic Insights works

In this module, we will work with Application Traffic Insights and get the identifier reported back into /var/ltm/log of BIG-IP. Additional BIG-IP reports into ELK Stack and within the ELK Dashboard we correlate data i.e. Device Identifier, User, IP and Unified BOT Information for anaylses.

Application Traffic Insights Overview

Application Traffic Insight (ATI) is a Proof of Value (PoV) tool that allows customers to explore different Shape products and understand their benefits and value (free of charge for a period of 60 days). ATI is an easy-to-use, self-service deployment with multiple deployment options on a variety of platforms (F5 or otherwise).

Note: If you haven´t worked with Application Traffic Insight (ATI) before, please review the Application Traffic Insight (ATI) Article on Volterra Docs.

Application Traffic Insights is easy-to-use as it allows flexible and easy deployment using Big-IP, NGINX, SSE, AWS and JS snippet Give the customer compelling Proof of Value (PoV) charts/dashboards.

Device Traffic Dashboard

Bot Assessment Dashboard

Check how Application Traffic Insights Overview* works

1. Connect to BIG-IP named “BIG-IP 16.1 - All Demos” via TMUI.

2. Within the WebUI of the BIG-IP instances navigate to iApps › Application Services : Applications › Application_Traffic_Insight and select *Reconfigure*.

3. Within the iApp configuration you will find predefined JS Injection configuration in the *1JS* part. Furthermore the 1JS gets been injected on the Virtual Server named *arcadia.emea.f5se.com_vs*. We leave the rest of the configuration untouched.

Note: Volterra on ATI on VoltConsole cover the Application Traffic Insight onboarding in more detail.

Application Traffic Insights and iRule

Application Traffic Insights includes two identifiers - a residue-based identifier and an attribute-based identifier. The residue-based identifier is based on local storage and cookies. The attribute-based identifier is based on signals collected on the device. The two identifiers always have different values.

1JS writes both the residue-based and attribute-based identifiers in a single, first-party cookie called *_imp_apg_r_*. The *_imp_apg_r_* cookie is URL encoded with the following format:

%7B%22diA%22%3A%22AT9cyV8AAAAAd60uXCtYafPTZGLaVAku%22%2C%22diB%22%3A%22ASJ4gFmzPo%2Fa8AHJceW

This cookie can be decoded via https://www.urldecoder.org/ to get the response in clear text. The decoded cookie has the following format:

```
"diA": "AT9cyV8AAAAAd60uXCtYafPTZGLaVAku"
"diB": "ASJ4gFmzPo/a8AHJceWhykudRoXeBGlP"
```

Note: Here, diA represents the residue-based identifier and diB represents the attribute-based identifier.

How to decode Application Traffic Insights _imp_apg_r_ cookie with an iRule

1. Within BIG-IP we use an iRule named *print_deviceid* and do a URL decoding of the *_imp_apg_r_* cookie and log *diA* and *diB* into /var/log/ltm of BIG-IP.
2. The irule named *print_deviceid* has been attached to Virtual Server named *arcadia.emea.f5se.com_vs*.

How to test Application Traffic Insights

1. To verify and view the logged values, connect to BIG-IP named "BIG-IP 16.1 - All Demos" via SSH.

2. Run *run util bash* followed by *tail -f /var/log/ltm* in the SSH Session.
3. RDP to windows machine called *win-client*.
4. Launch Chrome.
5. Open Devtools (Keyboard F12), select XHR in the Devtools and select the Browser Tab named *Device ID check*.
6. Check the request and response in Chrome.
7. Also check the cookie on the Devtools under Application.

1. You may want to do further test by running *Chrome* in Incognito Modus and compare the values of *diA* and *diB* with the outcome of the previous test.
2. Also check *tail -f /var/log/ltm* in the SSH Session as the values of *diA* and *diB* of the *_imp_apg_r_* cookie have been written to the file.

Application Traffic Insights and ELK |

Within the UDF Environment you will find an instance called **ELK**. Here we run an ELK Container which is used to visualize Device Identifier and correlate data i.e. Username to Device ID; Geo IP to Device ID. Additional **AWF Unified Bot Protection** log events into ELK. Those logs been correlated as well.

Note: This is a MVP. So please reach out if you have use cases which we should add to the Demo.

Steps:

1. RDP to windows machine called *win-client*. The Password of the instance is listed within the **Details / Documentation** Tab.
2. Launch Chrome and choose the bookmark called **Kibana - Dashboard**.
3. Klick the Button left to “Home”. Within the Kibana Section you can choose between **Discover** or **Dashboard**.

Note: Within the Dashboard you will find pre-configured Visualizations. The Dashboard has only a limited space in terms of sizing. In case you want to anaylses a specific Visualization, use the function called **Maximize Panel**.

Demo Use Cases - Single Device accessing unauthorized accounts

Within here we will Demo sudden fluctuations in Users per DeviceID.

Steps:

1. Launch Chrome and discover the browser and access the bookmark called **Device ID check**. This will launch the **Arcadia Application**.
2. Navigate to the **Login** section of the Application.
3. Try to login with different random Username.
4. Go back to **Device ID+ Kibana** and select **Dashboard**.
5. Here you will see that a single Device (single **Device ID Type A** and **Type B**) tried to access the App with differnet Username.

1. If you like to Demo it with Postman, open **Postman**, start **New Runner Tab** by navigating to the **File** Menu of Postman.
2. From **Runner** drag the collection **Device ID+ ELK** into the Field **RUN ORDER**.
3. Choose the Source Data File named **Demo_1.csv** by using the **select file** menu.
4. Via **preview** check which Data we will Post via Runner to login page of **Arcadia Application**.
5. Now Press **Run Device ID+ ELK** in Runner.

Demo Use Cases - Deliberate use of proxy networks

Within that use case you will cover a single Device accessing unauthorized accounts from different Source IPs.

You will use Postman Runner to simulate 10 Request with 10 different Username using 10 different IPs but the same Device ID.

Steps:

1. Open **Postman**, start **New Runner Tab** by navigating to the **File** Menu of Postman.
2. From **Runner** drag the collection **Device ID+ ELK** into the Field **RUN ORDER**.
3. Choose the Source Data File named **Demo_2.csv** by using the **select file** menu.
4. Via **preview** check which Data we will Post via Runner to login page of **Arcadia Application**.
5. Now Press **Run Device ID+ ELK** in Runner.

6. Go back to your Kibana Dashboard.
7. Within here you see again there is only one **Device ID Type A / Device ID Type B** identifier generated.
8. The requests coming from 10 different geo locations.
9. Ten Usernames have been used with one **Device ID Type A / Device ID Type B** to logon to the page.

Demo Use Cases - Unusual Devices accessing user accounts

Within this Demo we will use Postman Runner to simulate requests coming from different devices sitting behind a proxy network. The Source IP will be the same however, the **Device ID Type A / Device ID Type B** will change on the malicious request. You´ll also see valid request coming from username **xyzgood**.

Steps:

1. Open **Postman**, start **New Runner Tab** by navigating to the **File** Menu of Postman.
2. From **Runner** drag the collection **Device ID+ ELK** into the Field **RUN ORDER**.
3. Choose the Source Data File named **Demo_3.csv** by using the **select file** menu.
4. Via **preview** check which Data we will Post via Runner to login page of **Arcadia Application**.

5. Now Press **Run Device ID+ ELK** in Runner.
6. Go back to your Kibana Dashboard.
7. Within here you see that various **Device ID Type A / Device ID Type B** have been generated by a single IP.

8. If you invest further, you´ll see potential valid requets as these coming from a unique User by a Unique IP generating a single Device Identifier.
9. On the other hand you see differnt Device Identifier been generated by the same IP using random Usernames.

4.3 Module 3: Offline Machine Learning

To test Offline Machine Learning please follow the Documentaion within UDF until its available on Readthedocs.

4.4 Module 4: 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.

Note: The Lab is already pre-build. Meaning, you can show/check and demo encryption and obfuscation for password field with help of Datasafe without configuring anything. If you plan to demo from scratch, please go to **Exercise 1 – TASK 1 - Review and Attack the Login Page** and follow the instructions.

Demo Protecting Credentials with F5 DataSafe - pre-configured

---

Note: KNOWN BUG in 14.1 (fixed in 15.1.1): you need to enable AJAX, save, and disable AJAX in Datasafe profile for /user/login to make it works.

Steps:

1. 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 **user/user**.
2. Start Chrome and click bookmark Hackazon Login or Hackazon Home
3. Move cursor to the password field, right click and select *Inspect*.
4. You can see obfuscation (password field name changing every 5 seconds).
5. Next go to tab **Network** in Developers tools.
6. Login into Hackazon with username/password as **demo1 /demo1**
7. Check encryption and obfuscation for password field.

Exercise 1 – TASK 1 - Review and Attack the Login Page

Purpose: Review `Form Fields` with the Developer Tools of your Browser.

Steps:

1. Datasafe is configured on BIG-IP named **BIG-IP 16.1 - All Demos**.
2. Login to BIG-IP via WebUI and detach the *DataSafe Profile* from Virtaul Server `Local Traffic ›› Virtual Servers : Virtual Server List ›› vs_Hackazon_II`.
3. 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).
4. Once connected to the Windows client, open **Firefox** and access **Hackazon Login** Bookmark.
5. Right-click inside the field called **Username or Email** and select **Inspect Element**. The developer tools window will open.

Question:

What is the **name** value for this fieldc alled **Username or Email**? username

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

Question:

What is the **name** value for this field called **Password**? password

---

Note: `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.

Exercise 1 – TASK 2 - Review Methods for Stealing Credentials

Steps:

1. 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.
2. On the login page of the Hackazon website enter your first name and **P@ssw0rd!** as password but do not click **Sign In**.
3. From the **Demo Tools** click **Steal Password** and then click on the password field.

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

---

1. Click **OK** then clear the password you entered.
2. From the **Demo Tools** click **Start Keylogger** and then enter the same password as earlier.
3. Watch the top of the Demo Tools.
4. The "malware" is using JavaScript to log the password as it is typed. It could also send this capture data to some malicious site.

5. In the developer tools window that opened previously, select the Network tab (F12), then click the trash can icon to delete the requests.
6. 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.

---

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

2. The user's credentials are visible in clear text.
3. This is another way that malware can steal credentials. By "grabbing" the POST request and any data sent with it, including username and password.

Exercise 1 – TASK3 – Perform a Form Field ``Web Inject``

Steps:

1. Return to the **Hackazon — Login** page.

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

---

1. Right-click inside the **Username or Email** field and select **Inspect Element** again.
2. Right-click on the blue highlighted text in the developer tools window that opens and select **Edit as HTML**.
3. Select all the text in the window and type **Ctrl+C** to copy the text.
4. Click after the end of **data-bv-field="username">** and type **
, and then press the **Enter key twice.
5. Type **Ctrl+V** to paste the copied text.
6. 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**.
7. Click outside of the edit box and examine the Hackazon login page.

Note: 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.

---

1. Close Firefox.

Exercise 2 – TASK1 – Review and Configure DataSafe Components

Within the exercise we will cover DataSafe Licensing and Provisioning.

Steps:

1. Datasafe is configured on BIG-IP named **BIG-IP 16.1 - All Demos**.
2. In the Configuration Utility of the BIG-IP (connect via Chrome Bookmark or launch https://10.1.1.9/tmui/login.jsp).
3. The Password of the BIG-IP instance is listed within the `Details / Documentation` Tab.

Note: DataSafe is NOT included in the Best Bundle but DataSafe IS INCLUDED in Advanced WAF.

1. Open the System > Resource Provisioning page

Exercise 2 – TASK2 – DataSafe Configuration

Steps:

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

Note: If the **Hackazon-DS** profile already exists, please delete and follow instructions here.

1. For **Local Syslog Publisher**, select **local-datasafe** (select the checkbox on the right side to enable.
2. Optional: The local-datasafe Publisher can be viewed at System -> Logs -> Configuration -> Log Publishers.
3. Click in **Advanced** and review all other options Data Safe will serve different Javascript files under those configured HTTP paths.
4. On the left menu click **URL List**, and then click **Add URL**.
5. For **URL Path** leave **Explicit** selected, and type **/user/login**.
6. Click in **Advanced** and review all other options. Various configurations refer to where Data Safe will inject its Javascript.
7. From the left panel open the **Parameters** page. Remember from earlier you found that the username and password parameter names are **username** and **password**.
8. Click **Add**, enter a new parameter named **username**, select **Identify as Username** and then click Repeat.
9. Create a second parameter named **password**, and then click **Create.**
10. For the **username** parameter select the **Obfuscation** checkbox.
11. For the **password** parameter select the **Encrypt**, **Substitute Value**, and **Obfuscate** checkboxes.
12. From the left menu open the **Application Layer Encryption** page.

Note: Notice that most features are enabled by default.

1. Review the explanations for the different features.
2. Select the **Add Decoy Inputs** checkbox

3. Expand the **Advanced** section and select **Remove Element IDs** checkbox, and then click **Save**.

4. Click **Save** to save the new profile
5. Navigate to **Security ›› Event Logs : Logging Profiles** and select the 'ASM-Bot-DoS-Log-All' log profile.
6. Ensure **Data Protection** is enabled.
7. Once enabled, click on the **Data Protection** tab and ensure the **local-datasafe** is selected from the dropdown of the **Publisher** section.
8. Enable **Login Attempt** and select the **default** template. Click Update.

9. Navigate to **Local Traffic ›› Virtual Servers ›› Virtual Server List** page and click **Hackazon_protected_virtual**, and then open the virtual server **Security > Policies** page.
10. From the **DataSafe** Profile list select Enabled.
11. 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 – TASK1 – Testing DataSafe Protection

Review the Protected Hackazon Login Page

Steps:

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

Question:

1. What is the **name** value for this field?

2. **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.
3. **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.

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

---

1. In the developer tools window select the **Network** tab, then click the trash can icon to delete any current requests.
2. On the login page enter your first name as username and **P@ssw0rd!** as password and click **Sign In**.
3. In the **Network** tab select the **/login?return_url=** entry, and then examine the **Params** tab.

Question:

1. What parameters were submitted? Random
2. Do you see a username or password field? Not really
3. Do you see the username you submitted? Yes
4. **Obfuscation** – DataSafe obfuscates the names of the parameters when they are submitted in a login request.
5. **Encryption** – DataSafe encrypted the value of the password field so that it is not a readable value in the login request.