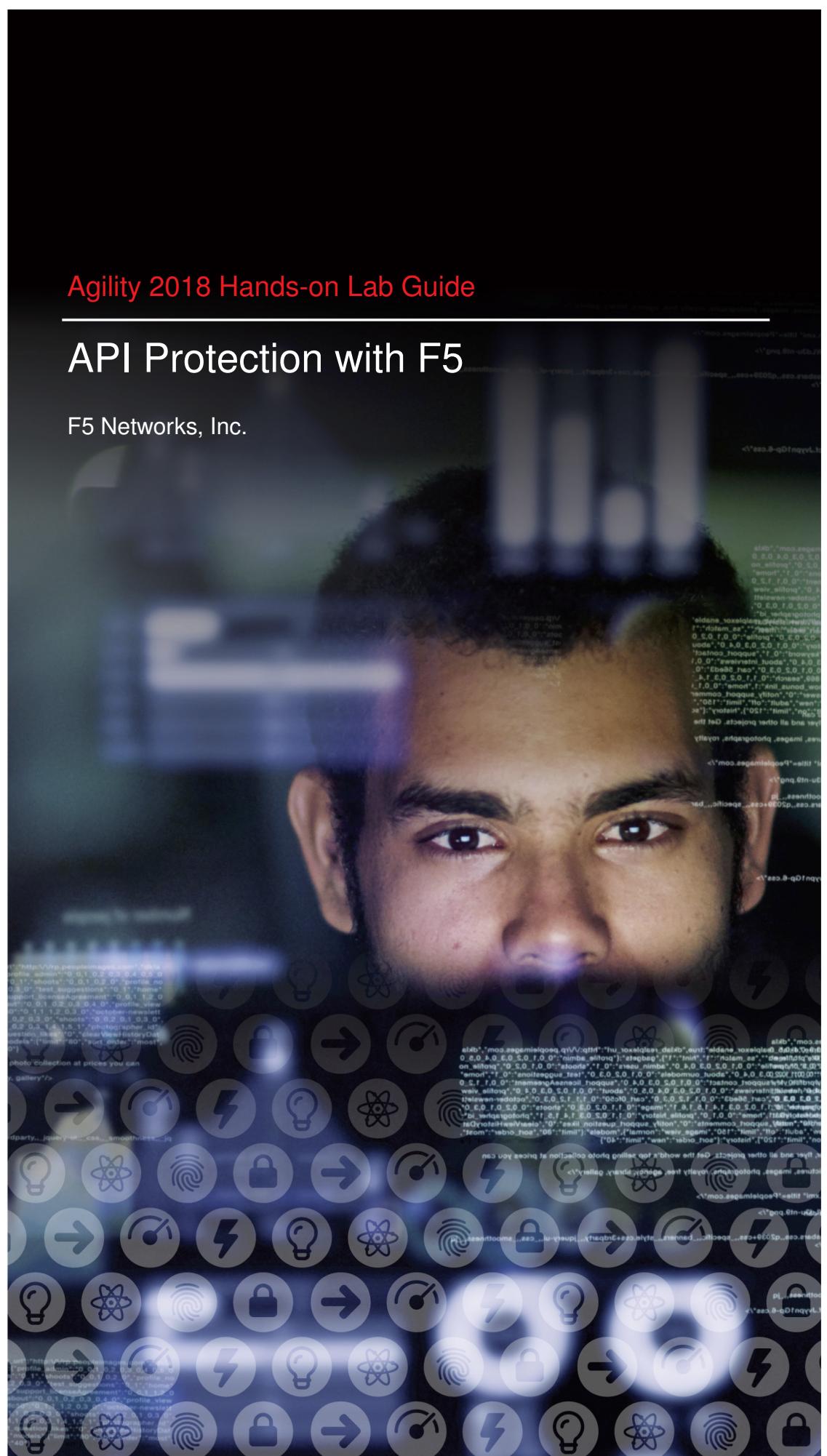




Agility 2018 Hands-on Lab Guide

API Protection with F5

F5 Networks, Inc.



Contents:

1 Getting Started	5
1.1 Lab Credentials	5
1.2 Lab Topology	6
2 API Protection with F5	7
2.1 Making API requests with POSTMAN	7
2.2 Implement Coarse-Grain Authorization	12
2.3 Adding Fine-Grain Authorization	16
2.4 API Protection By ASM	20
3 F5 BIG-IP configuration deep dive	29
3.1 BIG-IP Authorization access control configuration	29
3.2 BIG-IP API requests protection	36

1

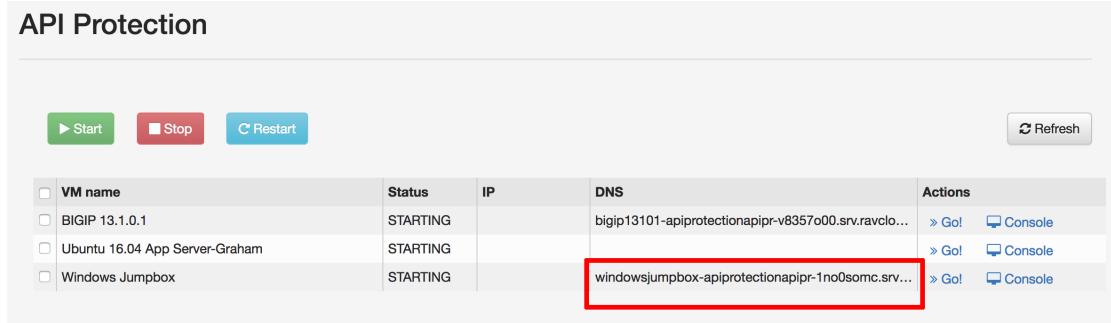
Getting Started

Instructor will provide you with access credentials to training portal as well as login page.



The screenshot shows the 'Training Portal' interface. At the top, there's a navigation bar with the Ravello Systems logo, 'Training Portal', 'Hi, student1 student1', a 'Logout' button, and 'Powered by Ravello Systems'. Below the navigation bar, the title 'API Protection' is displayed. Underneath the title, a message says 'Here are the available environments for this class:'. A table lists environments: 'Name' (API Protection - final), '# of VMs' (3), '# of running VMs' (0), and 'Actions' (a 'View' button highlighted with a red box). The 'View' button is located at the bottom right of the 'Actions' column.

Once logged in you should tap on “View”. You should be able to see Virtual Machines as shown below. Copy DNS name of Windows jumpbox to clipboard and use your Remote Desktop client to connect.



The screenshot shows the 'API Protection' environment details. At the top, there are buttons for 'Start' (green), 'Stop' (red), 'Restart' (blue), and 'Refresh' (grey). Below these are three buttons: 'VM name' (checkbox), 'Status' (dropdown), 'IP' (dropdown), and 'DNS' (dropdown). The table lists four VMs: 'BIGIP 13.1.0.1' (Status: STARTING, IP: [redacted], DNS: bigip13101-apiprotectionapipr-v8357o00.srv.ravclo..., Actions: 'Go!' and 'Console'), 'Ubuntu 16.04 App Server-Graham' (Status: STARTING, IP: [redacted], DNS: [redacted], Actions: 'Go!' and 'Console'), and 'Windows Jumpbox' (Status: STARTING, IP: [redacted], DNS: windowsjumpbox-apiprotectionapipr-1no0somc.srv..., Actions: 'Go!' and 'Console'). The 'Windows Jumpbox' row is highlighted with a red box.

Note: All work for this lab will be performed exclusively from the Windows Jumpbox. No installation or interaction with your local system is required.

1.1 Lab Credentials

The following table lists access credential for all required components:

Component	Credentials
Windows Jumpbox	admin/admin
F5 BIG-IP VE	admin/admin

The BIG-IP VE is accessible from the Windows Jumpbox at <https://192.168.1.5>

1.2 Lab Topology

The following components have been included in your lab environment:

- 1 x F5 BIG-IP VE (v13.1)
- 1 x Linux Webserver (xubuntu 14.04)
- 1 x Windows Jumphost

On the picture below you can see network topology. Basically, you will be sending various API calls to API server proxied through BIG-IP VE.



Traffic from Windows Jumpbox will be proxied through the BIG-IP to API Server.

2

API Protection with F5

In this section you will find guidelines for completion API protection lab exercises.

2.1 Making API requests with POSTMAN

2.1.1 Using Postman to make API requests

In this module you will learn how to make API requests with the Postman client to simulate calls that might be made as part of an application, for instance, a mobile app, native client app, client side webapp, or server to server API request.

2.1.2 Connect to Client Jumphost and launch Postman

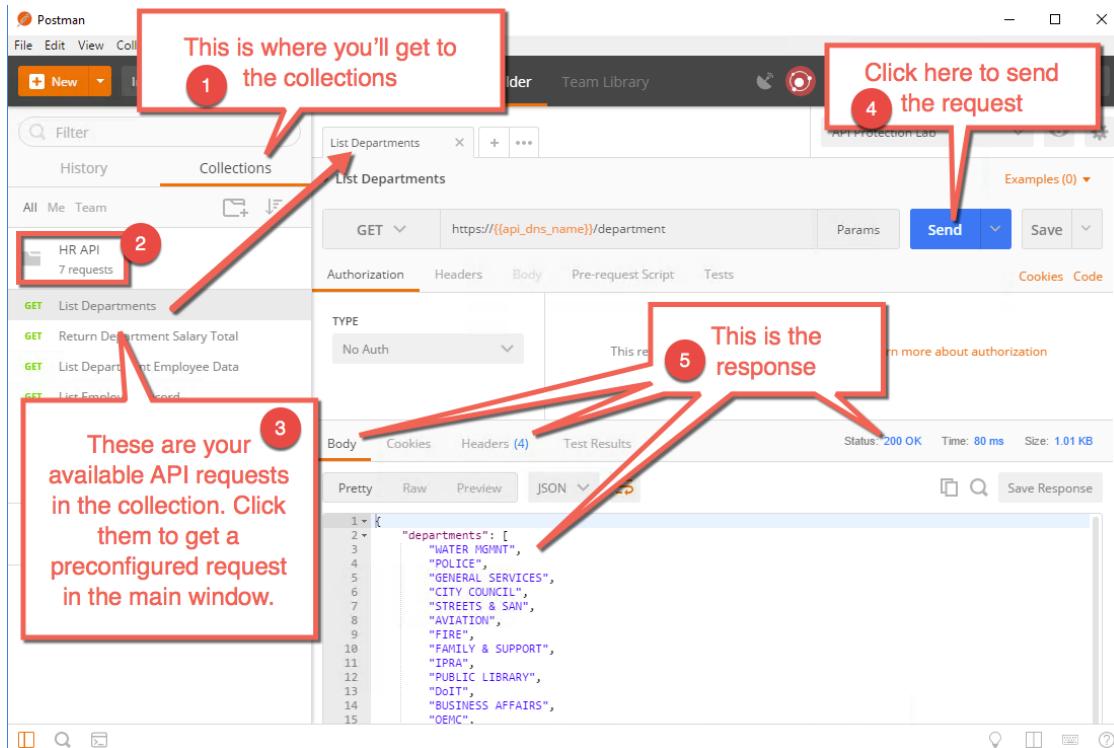
1. RDP to the client jumphost
2. Launch the Postman application. The icon looks like this:



2.1.3 Learn how to use the preconfigured API request collection

In this task you will learn how to use the preconfigured set of requests in the HR API collection.

1. Click Collections
2. Click HR API
3. Click List Departments
4. Click Send
5. Notice the returned list of departments



2.1.4 Learn how to change environment variables

In this task you will learn how to change the environment variables that are configured to alter which department you are querying data for. In this case the variables are used in the URI, but there are other variables used in some queries in the body as well.

Determine Police Department Salary Total

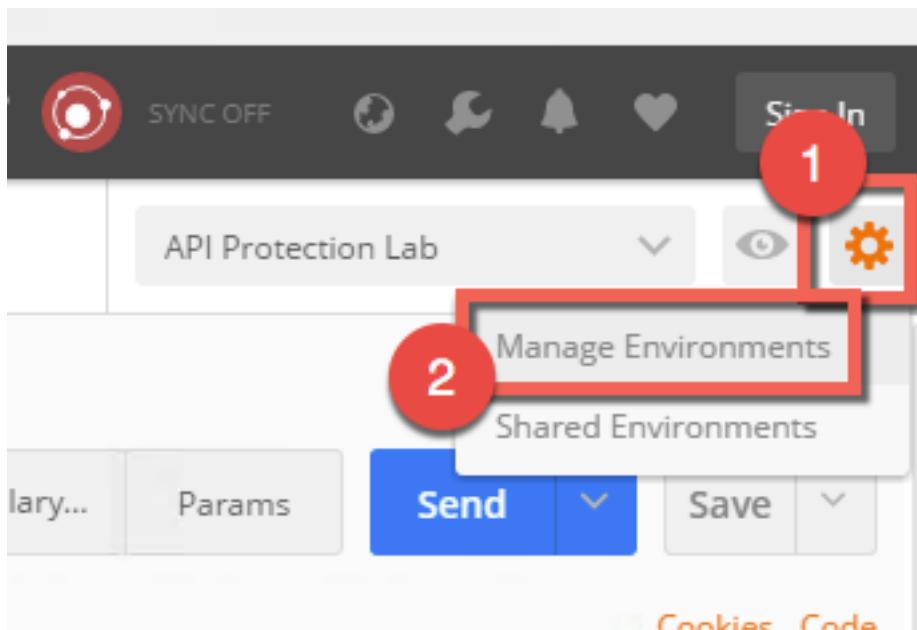
1. Click on the Return Department Salary Total request in the collection
2. Click Send
3. Notice the total returned is 1106915639.7999947

Change environment variable for department

1. Notice the GET request URI has a variable in it named {{department}}

A screenshot of the Postman interface showing a single request. The request type is 'GET' and the URL is 'https://{{api_dns_name}}/department/{{department}}/salary...'. A red box highlights the URL, specifically the part 'https://{{api_dns_name}}/department/{{department}}/salary...'. The URL is preceded by a red arrow pointing from the previous step's note.

2. Notice in the top right we have an environment set named "API Protection Lab".
3. Click the gear in the top right, then click "manage environments".



4. Click API Protection Lab

Manage Environments Environment Templates

Environments are a group of variables & values, that allow you to quickly switch between different configurations for your API requests.

[Learn more about environments](#)

API Protection Lab

5. Change the value for department from "police" to "fire" then click update

MANAGE ENVIRONMENTS

Manage Environments Environment Templates

Edit Environment

API Protection Lab

	Key	Value	Bulk Edit
<input checked="" type="checkbox"/>	api_dns_name	api.vlab.f5demo.com	
<input checked="" type="checkbox"/>	department	fire	
<input checked="" type="checkbox"/>	last_name	acevedo	
<input checked="" type="checkbox"/>	first_name	nadine	
<input checked="" type="checkbox"/>	search_field	last_name	
<input checked="" type="checkbox"/>	search_value	acevedo	

1

2

Cancel

Update

6. Click the X in the top right to close the manage environments window

Determine Fire Department Salary Total

1. Click Send
2. Notice the total returned is now 457971613.68

2.1.5 Return the Environment variables to default

1. Change the department variable back to “police”

2.2 Implement Coarse-Grain Authorization

In this module you will implement authorization requirements. You will require a valid JWT (JSON Web Token) before a client can access the API. You will then gather a valid JWT and leverage it to make an API request.

2.2.1 Add the policies to the virtual server

In this task you will add the policies you created to the virtual server.

1. Open web-browser, connect to BIG-IP <https://192.168.1.5> (login: admin, password: admin) and click Local Traffic -> **Virtual Servers**
2. Click **api.vlab.f5demo.com**
3. Change Access Profile from none to **prebuilt-api-psp**
4. Change Per Request Policy from none to **prebuilt-api-prp**

Access Policy	
Access Profile	prebuilt-api-psp
Connectivity Profile	None
Per-Request Policy	prebuilt-api-prp
VDI Profile	None
Application Tunnels (Java & Per-App VPN)	<input type="checkbox"/> Enabled
OAM Support	<input type="checkbox"/> Enabled
ADFS Proxy	<input type="checkbox"/> Enabled
PingAccess Profile	None

5. Click Update

2.2.2 Test access to the API

In this task you will test your access to the API and find it is blocked because you do not present a valid JWT.

1. Open Postman on the jumphost client
2. Select List Departments from the HR API collection and send the request
3. Review the response, note the 401 unauthorized and the header indicating you did not present a valid token

The screenshot shows the Postman interface with the 'List Departments' collection selected. A GET request is made to `https://{{api_dns_name}}/department`. The 'Authorization' tab is active, showing 'No Auth'. The response status is 401 Unauthorized, and the 'Headers' tab shows a 'WWW-Authenticate' header with the value 'Bearer error="invalid_token"'. Both the status code and the error message in the headers are highlighted with red boxes.

2.2.3 Get a JWT from the Authorization Server

1. Click the **type** drop down under the **authorization** tab.
2. Select **OAuth 2.0**
3. Click **Get New Access Token**

The screenshot shows the Postman interface with the 'Authorization' tab selected. The 'Type' dropdown is set to 'OAuth 2.0', which is highlighted with a red box. Below the dropdown, a note states: 'The authorization data will be automatically generated when you send the request. Learn more about authorization'. To the right, there is a 'Get New Access Token' button, also highlighted with a red box. The 'Access Token' and 'Available Tokens' fields are visible above the button.

Postman provides a mechanism to handle the OAuth client workflow automatically. This means it will handle getting the authorization code and then exchange it for an access token, which you will use. Without this

you would make two separate requests, one to get an authorization code and another to exchange that for an access token.

1. Fields should be prefilled, but verify they match the below:

Field	Value
Token name	employeeuser
Grant Type	Authorization Code
Callback URL	https://www.getpostman.com/oauth2/callback
Auth URL	https://as.vlab.f5demo.com/f5-oauth2/v1/authorize
Access Token URL	https://as.vlab.f5demo.com/f5-oauth2/v1/token
Client ID	9f1d39a8255e066b89a51f56b27506d39442c4f608c2f859
Client Authentication	Send as Basic Auth header

Most of this data is provided by the authorization server. The callback URL specified here is a special callback URL that the Postman client intercepts and handles rather than calling out to the [getpostman.com](https://www.getpostman.com) website.

GET NEW ACCESS TOKEN X

Token Name

Grant Type

Callback URL

Auth URL

Access Token URL

Client ID

Client Secret

Scope

State

Client Authentication

Request Token

1. Click **Request Token**
2. Select **employeeuser** in the authentication window that pops up and click **Logon**
3. Click the X to close this window

4. Make sure **employeeuser** is selected under Available Tokens drop down
5. Select **Request Headers** from the Add Authorization Data To drop down
6. Click **Preview Request**, the result should be this:

The screenshot shows the Postman interface with the following configuration:

- Authorization** tab is selected.
- Type**: OAuth 2.0
- Access Token** field contains a placeholder token: "ewogICJhbGciOiJIUzI1NilsCiAgImtpZC..."
- Available Tokens** dropdown is open, showing the selected token.
- Add authorization data to** dropdown is set to **Request Headers**.
- Preview Request** button is highlighted with a red box.

7. Go to the **Headers** tab and review the inserted **Bearer** token header:

The screenshot shows the Postman interface with the following configuration:

- GET** method, URL: `https://{{api_dns_name}}/department`
- Headers** tab is selected.
- A single header is present:

Key	Value
Authorization	Bearer ewogICJhbGciOiJIUzI1NilsCiAgImtpZC...

2.2.4 Send the request with JWT and review response

1. Click **Send** and review the response.
2. Note that now it is a 200 OK instead of 401 Unauthorized and that you have response data in the body.

The screenshot shows the Postman interface with a successful API call to 'List Departments'. The response body is a JSON object containing an array of department names:

```

{
  "departments": [
    "WATER MGMT",
    "POLICE",
    "GENERAL SERVICES",
    "CITY COUNCIL",
    "STREETS & SAN",
    "AVIATION",
    "ETRS",
    "FAMILY & SUPPORT",
    "TRAN"
  ]
}

```

You have now implemented coarse grained authorization and are requiring clients to request a JWT from a trusted authorization server before allowing access to the API.

2.3 Adding Fine-Grain Authorization

2.3.1 Adding Fine-Grain Authorization

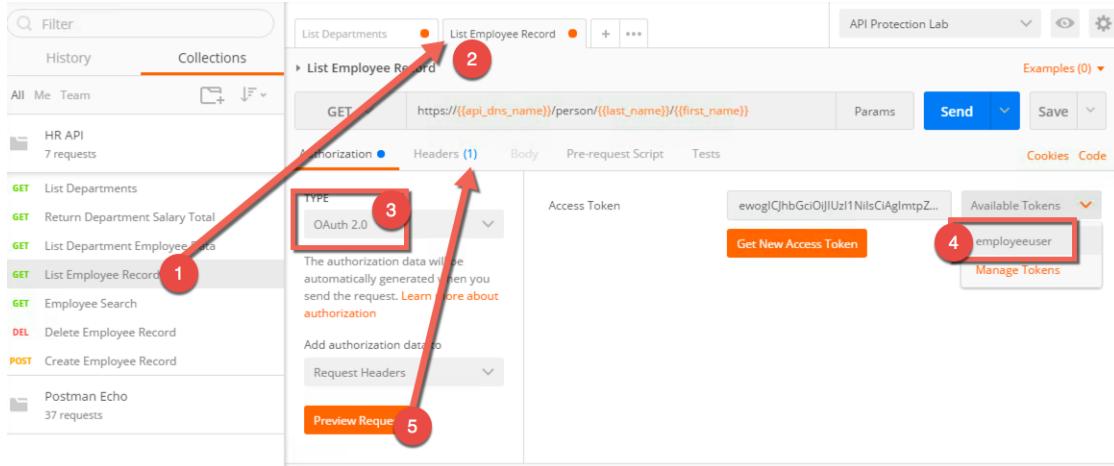
In this module you will add fine-grain controls to your policy to restrict access to parts of the API based on parameters in the JWT. The example will relate to user group membership, but it could be many parameters (e.g. company, user, group, as source, etc).

The goal is to restrict access to any person API requests to only members of the HR department.

2.3.2 Validate your policy blocks access to person requests without HR group membership

In this task you will test the settings you just put in the per request policy. You are expecting to be denied access to the /person URL because employeeuser is not in the HR group that you have marked as a required value in the JWT.

1. On the left side, select **List Employee Record**. It will now appear in another tab in the middle section and you should select it if it is not already.
2. Under Authorization type select **OAuth 2.0** for the type
3. From the **Available Tokens** drop down, select **employeeuser**
4. Make sure Add Authorization Data is set to **Request Headers**
5. Click **Preview Request** and note the header has been inserted



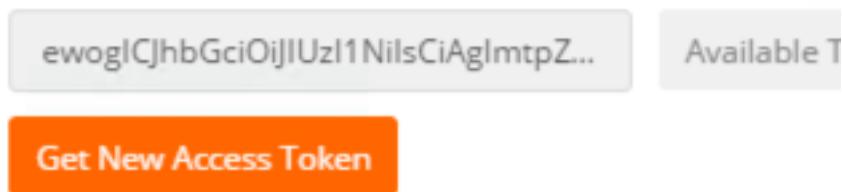
6. Click **Send**
7. The result should be a **401 unauthorized** with no data in the body. The header will report an invalid token.

You were denied access because the JWT retrieved by this user is not allowed to access that data. We can resolve this by using credentials that will generate a JWT valid for this request.

2.3.3 Acquire a JWT for hruser and validate it can access /person

In this task you will get another JWT and use that to gain access to the /person portion of the API.

1. Click **New Access Token**



2. Change the token name to **hruser**, the rest of the settings should be already correct.

GET NEW ACCESS TOKEN

Token Name hruser

Grant Type

Callback URL

Auth URL

Access Token URL

Client ID

Client Secret

Scope

State

Client Authentication

Request Token

3. Click **Request Token**
4. Select **hruser** at the logon page and press logon.

Username
 employeeus
 hruser

Password

Logon

5. A JWT should be returned and your JWT management token window will look like this:

MANAGE ACCESS TOKENS		
ALL TOKENS	Token Name	hruser
employeeuser	Access Token	ewogICJhbGciOiJIUzI1NilsCiAgImtpZCI6ImxhYiIKfQ.ewogICJ0b2tIbl90eXBIIjoiQmVhcmVlyiwKICaiaXNzlioiaHR0cHM6Ly9hcy52bGFILmY1ZGVtby5jb20iLAogICJhdWQiOlsKICAgICJodHRwczovL2FwaS52bGFILmY1ZGVtby5jb20iAgXSwKICAiZ3JvdXBzIjoiZW1wbG95ZWUsaHiiLAogICJ1c2VlylioiaHJ1c2VlyiwKICaie3ViljiaoHJ1c2VlyiwKICaianRpIjoiZmNiN2NmOTQ3MDQ5MmY3ZjI0YzU4MDRhMDdmODI0ODZhNDE3N2lyZjVhOTFmZjgzMTc1MjFjYjc0MGNIY2Y5YilsCiAgImIhdCl6MTUwODYzNTU2NCwKICAiZxhwlijoxNTA4NjQ5OTY0LAogICJuYmYiOje1MDg2MzUyNjQKfQ.HIBKN6ulCV9oPF4R4MGOMqs73v2Fp7Ez_ksfGrjct0g
	expires_in	14400

6. Notice you now have two tokens, and click the X to close the window

7. Select hruser from the Available Tokens drop down

8. Click Preview Request

List Departments List Employee Record + ... API Protection Lab

GET https://{{api_dns_name}}/person/{{last_name}}/{{first_name}} Params Send Save

TYPE: OAuth 2.0

The authorization data will be automatically generated when you send the request. [Learn more about authorization](#)

Add authorization data to Request Headers

Access Token: hruser

Available Tokens: employeeuser, hruser

Preview Request 2

Body Cookies Headers (5) Test Results Status: 401 Unauthorized Time: 17 ms Size: 170 B

9. Click Send, you should get a 200 OK response and data in the response body like this:

```

1
2
3
4
5
6
7
8
9
10
11
12
{
  "data": [
    {
      "department": "POLICE",
      "first_name": "NADINE",
      "last_name": "ACEVEDO",
      "middle_initial": "M",
      "salary": 83616,
      "title": "POLICE OFFICER"
    }
  ]
}

```

10. You can now change the token used on any request by using this process:

- Select the request
- Select the Authorization tab
- Select OAuth 2.0 from the type drop down menu
- Select the correct token from the Available Tokens drop down menu
- Make sure Authorization Data is set to Request Headers
- Click Preview Request to add the token to the headers
- Click Send on the request

In this module we've used group membership to restrict access to particular URIs, but in production you may encounter many different variations. For example, an iRule can set an APM session variable equal to the request method (e.g. GET, POST, etc) and then in the Per Request Policy you can branch on method, only allowing POST from certain users, groups, IPs, etc

JWTs are typically short lived and may or may not use refresh tokens. In this lab the JWTs have been set as valid for several hours so you will not need to get new JWTs during the lab.

More information about BIG-IP configuration settings available here [F5 BIG-IP configuration deep dive](#)

2.4 API Protection By ASM

In this module, you will demonstrate how ASM can protect JSON API against JSON parser attack, SQL Injection, Directory Traversal and DoS attacks.

2.4.1 Apply API Protections Profiles

In this task, you will apply the DoS protection profile for the API interface that will protect against known bots and DoS attacks.

1. Navigate to Local Traffic->Virtual Servers->Virtual Server List-> api.vlab.f5demo.com
2. Open Security->Policies tab.
3. Change the state of **Application Security Policy** to Enable and make sure prebuilt-**API_Security_Policy** is selected.

- Change the state of **DoS Protection Profile** to Enable and select **prebuilt-API_DoS** from the drop-down menu.
- Change the state of **Log Profile** to Enable. Highlight **Prebuilt-API_Lab_Logging** from the list of available profiles and tap << to move it into Selected.

Local Traffic » Virtual Servers : Virtual Server List » api.vlab.f5demo.com

Properties	Resources	Security	Statistics
Destination 10.1.10.98:443	Service HTTPS	Application Security Policy Enabled... Policy: prebuilt-API_Security_Policy	Service Policy None
IP Intelligence Disabled	DoS Protection Profile Enabled... Profile: prebuilt-API_DoS	Log Profile Enabled... Selected: /Common prebuilt-API_Lab_Logging Available: /Common Log all requests, Log illegal requests, global-network, local-dos	
<input type="button" value="Update"/>			

- Click Update

2.4.2 Running DoS Attack

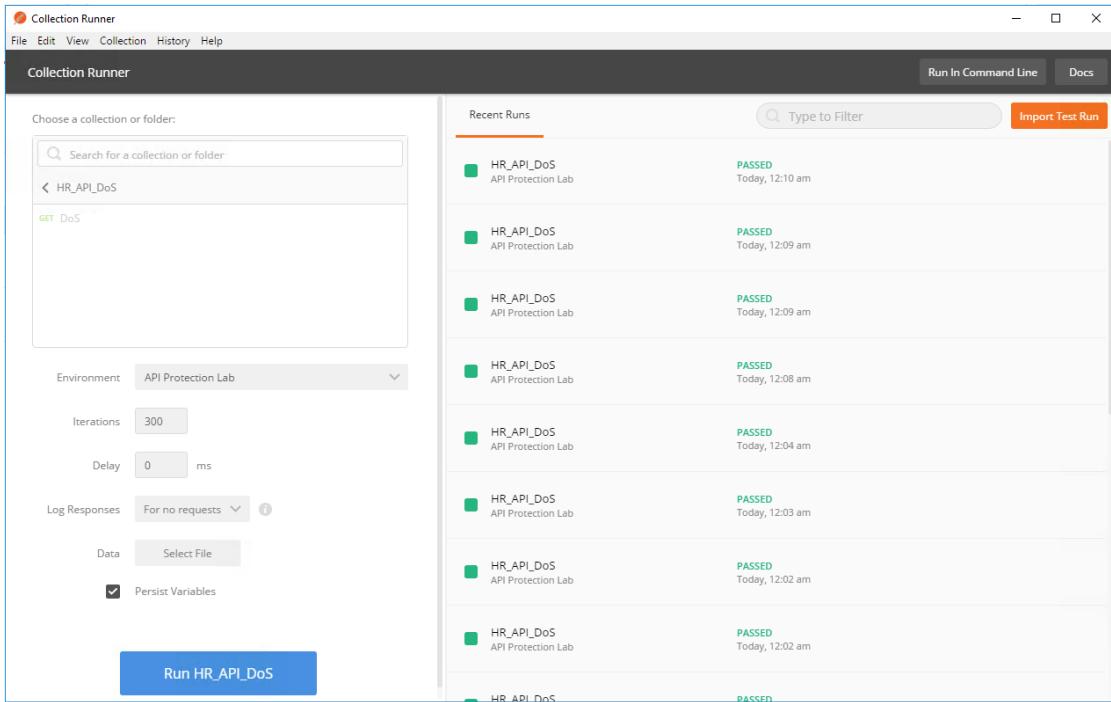
In this task, you will simulate DoS attack with Postman Runner and observe how it's been mitigated by Application DoS profile.

- On the Windows client choose HR_API_DoS collection then DoS request.
- Select token "employeeuser" from the list of available tokens and send the API call. Make sure you are getting response http code 200 and list of departments.
- Click **Save** to save the request with new access token.
- Click Runner on the top of the window:

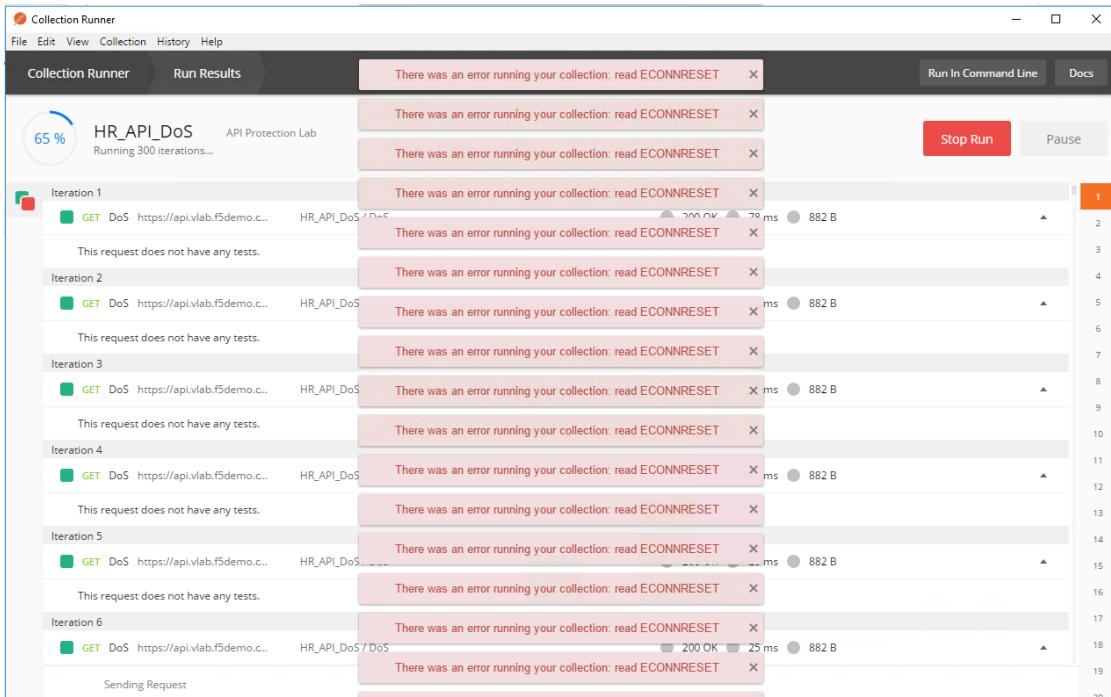
The screenshot shows a REST API testing interface with the following elements:

- Top Bar:** Contains buttons for "New", "Import", "Runner" (which is highlighted with a red box), and a "4" button.
- Filter:** A search bar labeled "Filter".
- Navigation:** Buttons for "History" and "Collections" (which is highlighted with an orange bar).
- User Selection:** Buttons for "All", "Me", and "Team".
- Collection Management:** Buttons for creating a new collection ("C+") and a dropdown menu.
- Collections List:** A list of collections:
 - HR API**: Contains 7 requests.
 - GET List Departments**
 - GET Return Department Salary Total**
 - GET List Department Employee Data**
 - GET List Employee Record**
 - GET Employee Search**
 - DEL Delete Employee Record**
 - POST Create Employee Record**
 - HR_API_DoS**: Contains 1 request.
 - GET DoS**
 - HR_API_Illegal**: Contains 5 requests.

5. Choose HR_API_DoS collection and use the screenshot below to configure parameters. Ensure right environment has been chosen, iterations updated and Log responses as "For no requests". Once the parameters have been selected click Run HR_API_DoS.



- After a short period of time Postman Runner may report failing transactions (it may not and gracefully handle the rate limiting, proceed to check logs in next step anyway):



- On the BIG-IP navigate to Security->Event Logs-> DoS -> Application Events
Look at the details of the attack detection and mitigation applied.

2.4.3 Accessing Disallowed URL

In this task, you will try accessing URL not allowed by ASM policy.

1. On the client machine Postman app proceed to “HR_API_Illegal” collection and select Disallowed URL request.
2. Use previously retrieved access token by setting Authorization Type to Oauth 2.0 and Available Tokens to hruser (similar to Module 3 Task 5 without getting a new token).
3. Click Send. ASM should return a blocking page as below:

```

1 <html>
2   <head>
3     <title>Request Rejected</title>
4   </head>
5   <body>The requested URL was rejected. Please consult with your administrator.
6   <br>
7   <br>Your support ID is: 11407609948504739885
8   <br>
9   <br>
10  <a href='javascript:history.back();'>[Go Back]</a>
11 
12 </body>
13 </html>

```

4. On the BIG-IP navigate to Security->Event Logs->Application ->Requests. Click on the last (top) request and look at the violation details:

Time	Violation Rating
2017-10-30 11:20:46	Request looks like a false positive but requires examination

Request

```

GET /testURL HTTP/1.1
cache-control: no-cache
Postman-Token: e7d69943-bf3f-4ada-beed-e5b4d3856713
Content-Type: application/json
User-Agent: PostmanRuntime/6.4.1
Accept: */*
Host: api.vlab.f5demo.com
cookie: TS01fb4a90-01afbf8dd28e753746a6d7afb5fb57ba3e11a5d23ab4704fe18f0ba7b8df6a8afa6e0e1419b75ef6d6539e23eec2b8941c7232e799; TS0142dd9e01afbf8dd287eb2a38c7fae90433c256e1f25f1b1827f56f9a55c789fff8b26fb052ecf547a61df7aeec56932c095de71316d02db1ec
accept-encoding: gzip, deflate
Connection: keep-alive

```

2.4.4 Illegal JSON parameter value - 1

In this task, you will simulate attack by running request with illegal JSON parameter.

1. On the client machine Postman app chose JSON Parsing Array request.
2. Use previously retrieved access token by setting Authorization Type to Oauth 2.0 and Available Tokens to hruser (similar to Module 3 Task 5 without getting a new token).
3. Click Send. ASM should return a blocking page as below:

The screenshot shows the Postman interface. In the left sidebar, under 'HR API' requests, there is a section for 'HR_API_Illegal' with 4 requests. One of these is selected, showing a POST request to 'https://[api_dns_name]/department/[department]/'. The 'Body' tab is selected, showing the following JSON payload:

```
1: {
  "last_name": "[a'../../../../etc/passwd', 2,3]",
  "first_name": "test",
  "middle_initial": "({mod_middle_initial})",
  "title": "({mod_title})",
  "department": "({mod_department})",
  "salary": "({mod_salary})"
}
```

The response body shows a blocked page with the following HTML content:

```
1< html>
2< head>
3< title>Request Rejected</title>
4</head>
5< body>
6< p>Your requested URL was rejected. Please consult with your administrator.
7< p>Your support ID is: 11407689448504739900
8< br>
9< br>
10< a href='javascript:history.back();'[Go Back]</a>
11</body>
12</html>
```

4. On the BIG-IP navigate to Security->Event Logs->Application ->Requests. Click on the last (top) request and look at the violation details:

The screenshot shows the BIG-IP Event Log interface under 'Event Logs : Application : Requests'. A search bar at the top is set to 'Illegal Requests: Illegal Requests'. The results table shows several entries, with the last one being the most recent (2017-11-02 20:24:01). This entry is highlighted and its details are shown in a modal dialog:

Detected Keyword		last_name=[a'../../../../etc/passwd', 2,3]
Attack Signature		Signature ID: 200000190 Signature Name: Directory Traversal attempt (parameter)
Context		Parameter (detected in POST Data)
Parameter Level		Global
Parameter Name		last_name
Parameter Value		[a'../../../../etc/passwd', 2,3]
Applied Blocking Settings		Staged

Details pane on the right:
Timestamp: 2017-11-02 20:26:57
Rating: N/A
Type: Information Leakage
Response: N/A

2.4.5 Illegal JSON parameter value - 2

In this task, you will simulate another attack by running request with illegal JSON parameter.

1. On the client machine Postman app chose Parameter Length&Signatures request.

2. Use previously retrieved access token by setting Authorization Type to Oauth 2.0 and Available Tokens to hruser (similar to Module 3 Task 5 without getting a new token).
3. Click Send. ASM should return a blocking page as below:

The screenshot shows the Postman interface. On the left, there's a sidebar with various API collections and requests. The main area shows a POST request to `https://(api.dev_name)/department/((department))`. The Body tab contains a JSON payload:

```

1 {
2   "last_name": "/etc/passwd",
3   "first_name": "test3",
4   "middle_initial": "mod_middle_initial",
5   "title": "(mod_title)",
6   "department": "((department))",
7   "salary": {mod_salary}
8 }

```

The response tab shows a 200 OK status with a response body containing HTML code that blocks the browser:

```

1 <html>
2   <head>
3     <title>Request Rejected</title>
4   </head>
5   <body>The requested URL was rejected. Please consult with your administrator.
6   <br>
7   <br>Your support ID is: 1140769948084739908
8   <br>
9   <br>
10  <a href="javascript:history.back();">Go Back</a>
11 </body>
12 </html>

```

4. On the BIG-IP navigate to Security->Event Logs->Application ->Requests. Click on the last (top) request and look at the violation details:

The screenshot shows the F5 BIG-IP Security Event Log interface under the Application tab. It lists several application requests. The last request in the list is selected, showing detailed information:

- Request:**

```

POST /department/police/ HTTP/1.1
cache-control: no-cache
Postman-Token: f4fe1193-0b53-47aa-aa4f-a38d6bdadca7
Content-Type: application/json
User-Agent: PostmanRuntime/6.4.1
Accept: /*
Host: api.lab.f5dem0.com
Cookie: TS01fb4d9a=01afbd8d28e753746a6d7afb57ba1ea5d23ab4704fe18f0ba7b8df6a8afa6e0e1419b75ef6d6539e23eec2b8941c7232e799; TS0142dd9-01afbd8d28e753746a6d7afb57ba1ea5d23ab4704fe18f0ba7b8df6a8afa6e0e1419b75ef6d6539e23eec2b8941c7232e799
Accept-Encoding: gzip, deflate
Content-Length: 184
Connection: keep-alive
{
  "last_name": "/etc/passwd",
  "first_name": "test3",
}

```
- Response:** N/A
- Time:** 2017-10-30 11:22:23
- Violation Rating:** 3 (Request needs further examination)
- Attack Types:** Abuse of Functionality

2.4.6 Non-JSON request

In this task, you will simulate attack by running non JSON request.

1. On the client machine Postman app chose Non-JSON request.
2. Use previously retrieved access token by setting Authorization Type to Oauth 2.0 and Available Tokens to hruser (similar to Module 3 Task 5 without getting a new token).
3. Click Send. ASM should return a blocking page as below:

The screenshot shows the Postman interface with a non-JSON request. The request URL is `https://{{api_dns_name}}/department/{{department}}/`. The Headers tab shows `Content-Type: text/xml`. The Body tab contains the following XML payload:

```

1 < html>
2   < head>
3     < title>Request Rejected</title>
4   </head>
5   < body>
6     The requested URL was rejected. Please consult with your administrator.
7     Your support ID is: 8033866773914714142
8
9     <br>
10    <a href="javascript:history.back();">Go Back</a>
11
12 </body>
13 </html>

```

- On the BIG-IP navigate to Security->Event Logs->Application ->Requests. Click on the last (top) request and look at the violation details:

The screenshot shows the F5 BIG-IP Security Event Log under the Application Requests section. It lists three requests:

- [HTTPS] /department/police/ 10.1.10.6 20:24:01 2017-11-02
- [HTTPS] /department 10.1.10.6 20:07:27 2017-11-02
- [HTTPS] /restURL 10.1.10.6 20:02:11 2017-11-02

The third request, /restURL, is selected. The detailed view shows:

- Description:** Malformed document illegal character encountered - json syntax error
- Actual URL:** /department/police/
- Wildcard URL:** /department*
- JSON Profile:** API_LAB_JSON
- Applied Blocking Settings:** Block, Alarm, Learn
- Response:** N/A

The response body shows the raw JSON payload:

```

POST /restURL HTTP/1.1
cache-control: no-cache
Postman-Token: 146aae2b-f75d-4081-abee-6dac3bb68490
Content-Type: application/json
Authorization: Bearer ewogICJhbGciOiJIUzI1NiIsCjAgImtpZCI6ImxhYiIKfQ.ewogICJ0b2tlbl90eXBLIjoioQmVhcmVyiIwKICaiXNzIjoiaHR0cHM6Ly9yc52bgFimY1ZGVtbysjz20iAogICJhdQ10iSkICAgICJodHRwczovL2Fwa552bgFimY1ZGVtbysjzb20iC1AgXSwKICaiZ3JvdXBzijoizW1wb695ZWUsaHilAoigCJ1c2VyiolahJ1c2VyiIwKICaiC3ViIjoiL0NvbW1vb19cmVidWlsdChcy1wc3AuahJ1c2VyiIwKICaiRpijoimJEyNWUwNjz4NDQ0yZjMwYwExNjM0YzQxGfjMzIxYzQ00WZn0DYYODdi0DjmNzIyNDY4MGZnNjky0WmNNE5NyIsCjAgImhdC16MTUwOTY3NzU1MSwKICaiZXhwIjoxNTA5NjxxOTUx.AogICJ0Yy10jE1MDk2NzcyNTEKfQ.9T-jrMum8gCvPzS07rtckd2TRHcaV4uVcJfoaoWQ0V
User-Agent: PostmanRuntime/6.4.1
Accept: */
Host: api.vlab.f5demo.com
Cookie: TS01fb40a9-01af8dd28e753746a6d7afb5fb57ba3e11a5d23ab4704fe18f0ba7b8df6a8afa6e0e1419b75ef6d6539e23eec2b8941c72

```

2.4.7 Shellshock request

In this task, you will simulate Shellshock attack by running request with specific header.

- On the client machine Postman app chose Shellshock request.
- Use previously retrieved access token by setting Authorization Type to Oauth 2.0 and Available Tokens to hruser (similar to Module 3 Task 5 without getting a new token).
- Click Send. ASM should return a blocking page as below:

The screenshot shows the Postman interface with a request to 'https://{{api_dns_name}}/department'. The 'Headers' tab has 'User-Agent' set to '0 ():/bin/bash-<`ls''. The response body contains HTML code for a rejected request, including a title 'Request Rejected', a message 'The requested URL was rejected. Please consult with your administrator.', and a link to go back.

- On the BIG-IP navigate to Security->Event Logs->Application ->Requests. Click on the last (top) request and look at the violation details:

The screenshot shows the F5 BIG-IP Security Event Log for Application Requests. It lists several requests, with the last one highlighted as an 'Illegal Requests: Illegal Requests' violation. The violation details show a detected keyword 'cache-control:0x20;no-cache0x0;Postman-Token:0x20;0b2ae36f-6165-4fe6-b226-89a7d0b64340xd0xaUser-Agent:[0x20][0x20]-;0x20/bin/bash[0x20]<0x20>ls[0x20]Accept:[0x20]' and an attack signature 'Signature ID: 200003166, Signature Name: Bash Shellshock execution attempt (Header)'. The context is 'Header' and the actual header name is 'User-Agent'.

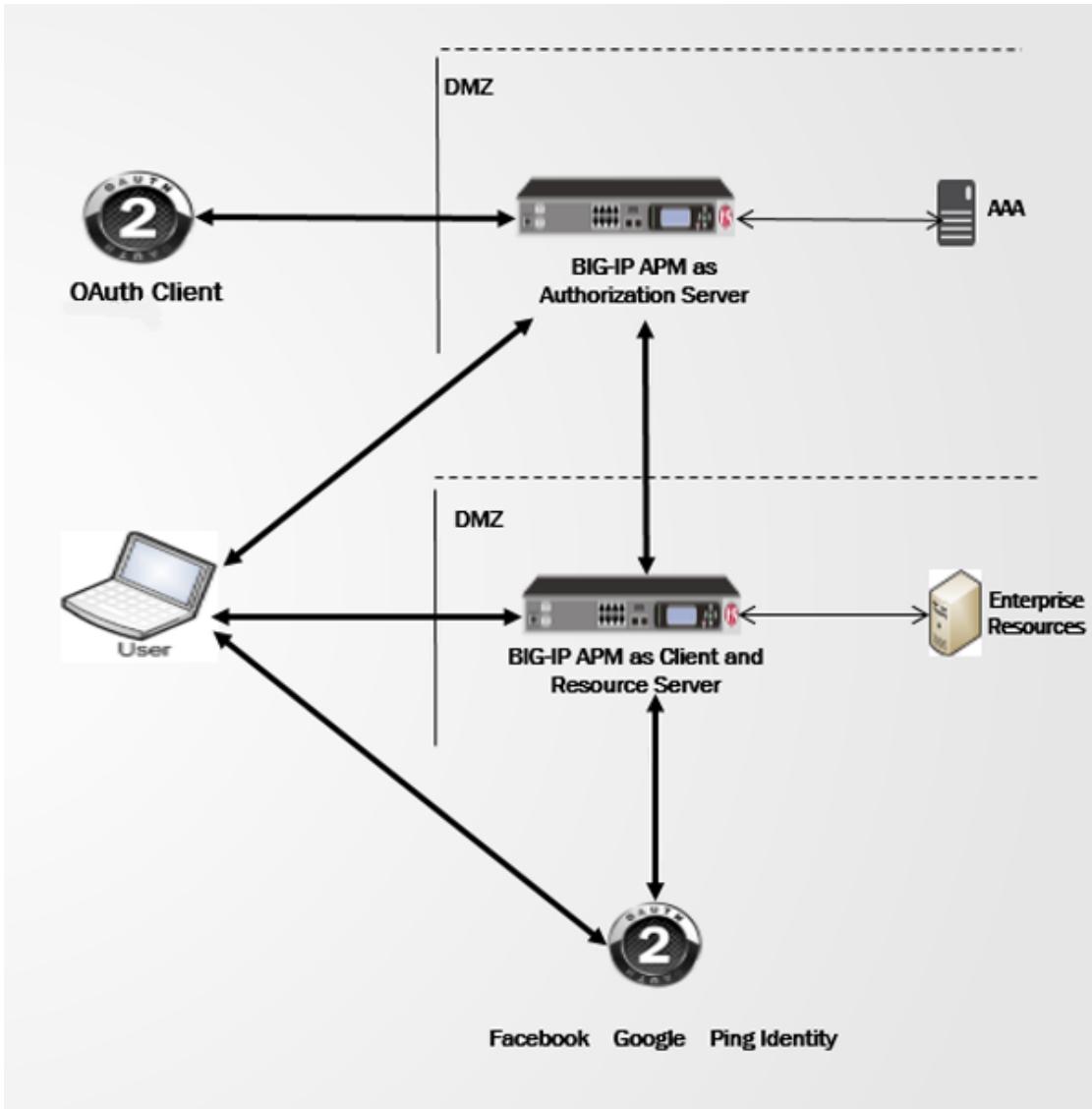
More information about BIG-IP configuration settings available here [F5 BIG-IP configuration deep dive](#)

F5 BIG-IP configuration deep dive

In this section you will find detailed information about BIG-IP configuration used in the lab exercises.

3.1 BIG-IP Authorization access control configuration

BIG-IP APM can be used as authorization server as well resource server. On the diagram below this is shown as separate hardware components.



Although hardware segregation is not a strict requirement and those components can be hosted on the same BIG-IP, only VIPs should be different. In this lab we have used **api.vlab.f5demo.com** as resource server and **as.vlab.f5demo.com** as an authorization server. Check the steps below in order to configure coarse-grain authorization.

3.1.1 Creation of JWK (JSON Web Key)

In this task you will check configuration settings for JWK which is used for validating the sent JWT. In this lab we have used Octet and a shared secret, but options includesolutions like public/private key pair as well.

Go to Access -> Federation -> JSON Web Token -> Key Configuration -> prebuilt-api-jwk

It is configured according to data below

Field	Value
Name	api-jwk
ID	lab
Type	Octet
Signing Algorithm	HS256
Shared Secret	secret

3.1.2 OAuth provider

In this task you will check configuration settings for an OAuth provider so that JWT can be validated.

Go to Access -> Federation -> OAuth Client/Resource Server -> Provider -> prebuilt-as-provider

The configuration settings is shown below.

Access » Federation : OAuth Client / Resource Server : Provider » prebuilt-as-provider

Properties					
General Properties					
Name	prebuilt-as-provider				
Description	<input type="text"/>				
Type	F5				
Ignore Expired Certificate Validation	<input type="checkbox"/>				
Trusted Certificate Authorities	ca-bundle.crt				
Allow Self-Signed JWK Config Certificate	<input checked="" type="checkbox"/>				
Use Auto-discovered JWT	<input checked="" type="checkbox"/>				
OpenID URI	<input type="text"/> Discover Last discovery time: 2017-10-31 11:11:53				
Authentication URI	<input type="text"/> https://as.vlab.f5demo.com/f5-oauth2/v1/authorize				
Token URI	<input type="text"/> https://as.vlab.f5demo.com/f5-oauth2/v1/token				
Token Validation Scope URI	<input type="text"/> https://as.vlab.f5demo.com/f5-oauth2/v1/introspect				
Userinfo Request URI	<input type="text"/>				
Issuer	<input type="text"/> https://as.vlab.f5demo.com				
Signing Algorithm	<table border="1"> <thead> <tr> <th>Allowed</th> <th>Blocked</th> </tr> </thead> <tbody> <tr> <td>ES256 ES384 HS256 HS384 HS512 RS256 RS384 RS512</td> <td><input type="button"/> <input type="button"/></td> </tr> </tbody> </table>	Allowed	Blocked	ES256 ES384 HS256 HS384 HS512 RS256 RS384 RS512	<input type="button"/> <input type="button"/>
Allowed	Blocked				
ES256 ES384 HS256 HS384 HS512 RS256 RS384 RS512	<input type="button"/> <input type="button"/>				
Key (JWK)	<table border="1"> <thead> <tr> <th>Allowed</th> <th>Blocked</th> </tr> </thead> <tbody> <tr> <td>/Common/prebuilt-api-jwk</td> <td><input type="button"/> <input type="button"/></td> </tr> </tbody> </table>	Allowed	Blocked	/Common/prebuilt-api-jwk	<input type="button"/> <input type="button"/>
Allowed	Blocked				
/Common/prebuilt-api-jwk	<input type="button"/> <input type="button"/>				

Most of these settings have been discovered automatically from Authorization Server with the use of OIDC.

3.1.3 Token Configuration

In this task you will check the configuration of some of the values retrieved automatically via OIDC discover tool. Those values had to be adjusted manually because the OIDC AS cannot provide you with the values specific to your audience.

1. Go to Access -> Federation -> JSON Web Token -> Token Configuration -> auto_jwt_prebuilt-as-provider
2. Make sure **https://api.vlab.f5demo.com** have been defined as Issuer

General Properties	
Auto Discovered	The token is auto generated by OpenId discovery.
Name	auto_jwt_prebuilt-as-provider
Issuer	<input type="text" value="https://as.vlab.f5demo.com"/>
Use Provider List Settings	<input checked="" type="checkbox"/>
Access Token Expires In	0
Audience	<input type="text" value="https://api.vlab.f5demo.com"/> <input type="button" value="Add"/>

3. Under Additional Key make sure prebuilt-api-jwk is added into Allowed

Keys (JWK)		
Auto Discovered Key	Allowed	Blocked
	No available items	<input type="button"/> <input type="button"/>
Additional Key	Available	Allowed
	Filter <input type="text"/> <input type="text" value="/Common/prebuilt-as-jwk"/>	<input type="button"/> <input type="button"/> <input type="text" value="/Common/prebuilt-api-jwk"/>

The object prebuilt-as-jwk was precreated for the authorization server function. It matches api-jwk (and prebuilt-api-jwk) exactly because a shared key is needed on both the authorization server and resource side. In this case you could have reused it instead of making a new one, but in production your authorization server may not be the Big-IP you are protecting the API server on, and you would need to create it as shown here.

3.1.4 JWT Provider

In this task you will check configuration for a JWT provider that can be selected in a per request or per session policy for JWT validation.

Go to Access -> Federation -> JSON Web Token -> Provider List -> prebuilt-as-jwt-provider

Make sure prebuilt-as-provider is selected

Access » Federation : JSON Web Token : Provider List » **prebuilt-as-jwt-provider**

	Properties	
General Properties		
Name	prebuilt-as-jwt-provider	
Access Token Expires In	0	
Provider	<input type="text" value="/Common/prebuilt-as-provider"/> /Common/AzureAD ▾ Add	
Cancel Save		

3.1.5 Per-session policy

In this task you will check per session policy which validating the JWT token and collecting the claims data from parameters inside the JWT.

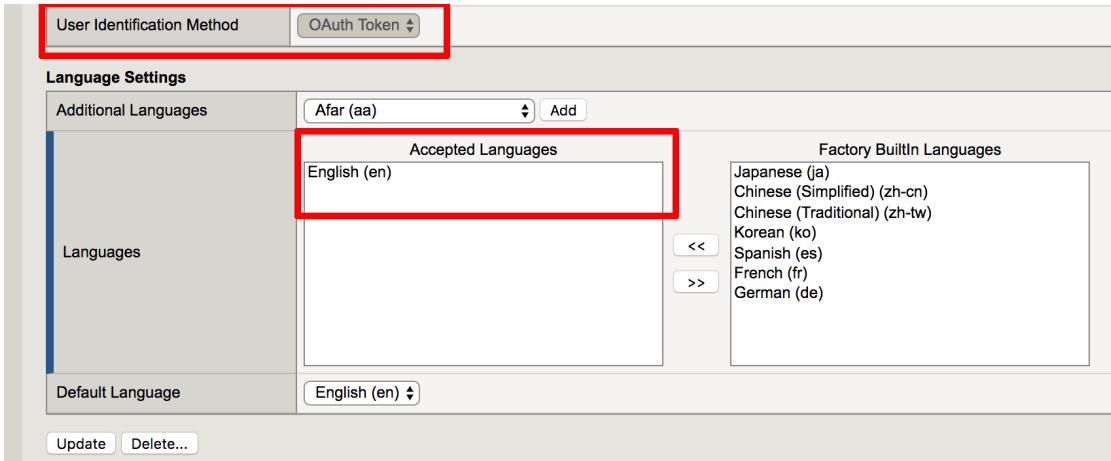
Go to Access -> Profiles/Policies -> Access Profiles (Per-Session Policies) -> prebuilt-api-psp

Make sure profile type is set to OAuth-Resource Server and Profile Scope is Profile.

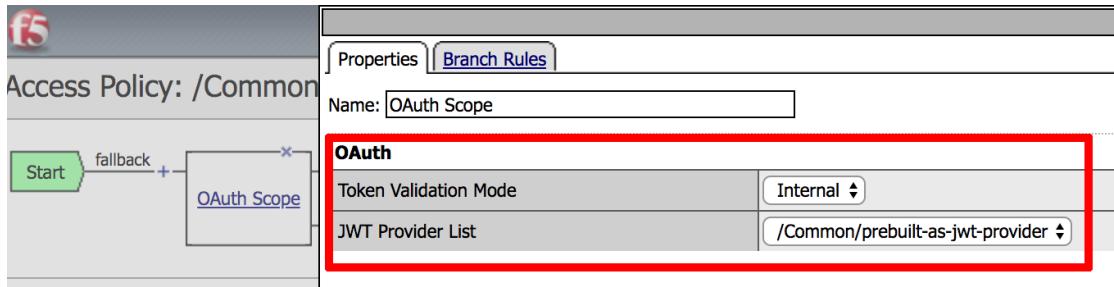
Access » Profiles / Policies : Access Profiles (Per-Session Policies) » **prebuilt-api-psp**

	Properties	SSO / Auth Domains	Access Policy	Logs
General Properties				
Name	prebuilt-api-psp			
Partition / Path	Common			
Parent Profile	access			
Profile Type	OAuth-Resource Server			
Profile Scope	<input type="button" value="Profile"/>			

Also make sure that the User Identification Method is set to OAuth token and default language is English.



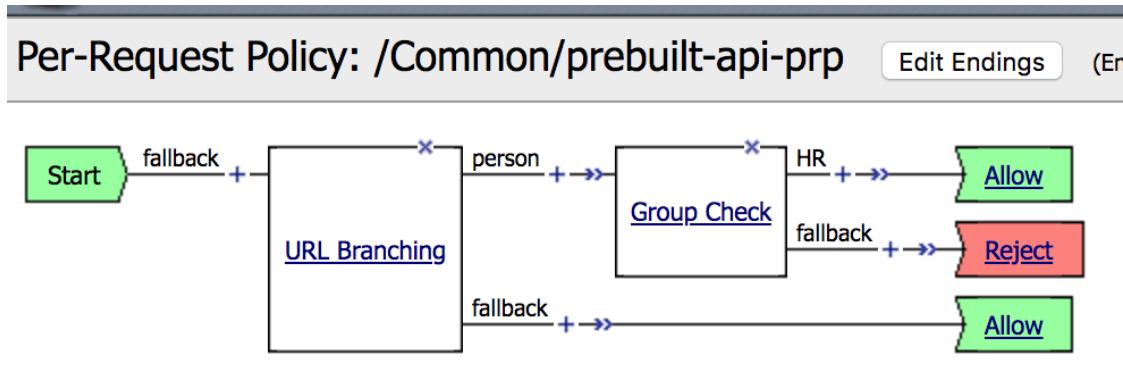
Click **Access Policy** tab and then click **Edit Access Policy for Profile “prebuilt-api-psp”**. Examine the policy and click on OAuth Scope. Make sure Token Validation Mode is set to “Internal” and JWT Provider List is set to as-jwt-provider. The validation mode is set to be internal because JWT token will be validated internally instead of making an external call.



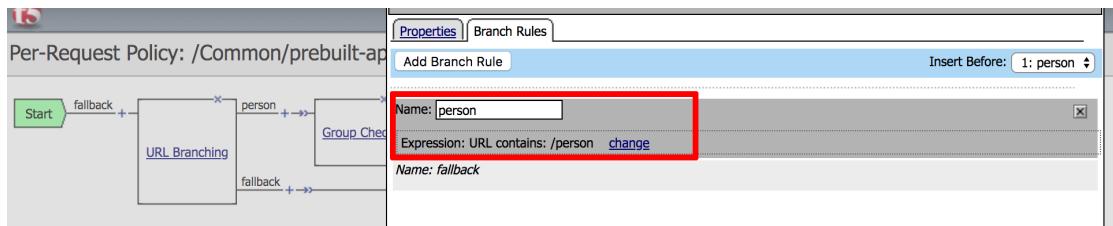
3.1.6 Per-request policy

You will check a per request policy to validate authorization on each request by checking for the presence and validity of a JWT.

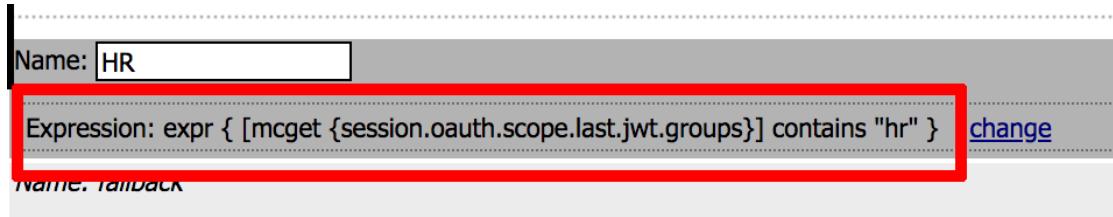
Go to Access -> Profiles/Policies -> Per-Request Policies -> prebuilt-api-prp —> Click **Edit**



Click on **URL Branching** and and then **Branch Rules**. Note the URL expression.



Click on **Group Check** and and then **Branch Rules**. Note the expression.



Close the tab in the browser

3.2 BIG-IP API requests protection

In this module you will examine security policies created for API server protection.

3.2.1 Logging profile

In order to check for logging configuration settings to Security -> Event Logs -> Logging Profiles —> prebuilt-API_Lab_Logging and check logging settings.

Security » Event Logs : Logging Profiles » Edit Logging Profile

Edit Logging Profile

Logging Profile Properties	
Profile Name	prebuilt-API_Lab_Logging
Partition / Path	Common
Description	
Application Security	<input checked="" type="checkbox"/> Enabled
Protocol Security	<input type="checkbox"/> Enabled
Network Firewall	<input type="checkbox"/> Enabled
DoS Protection	<input checked="" type="checkbox"/> Enabled
Bot Defense	<input checked="" type="checkbox"/> Enabled

Application Security DoS Protection Bot Defense

Configuration Basic ▾

Storage Destination	Local Storage ▾
---------------------	-----------------

Storage Filter Basic ▾

Request Type	All requests
--------------	--------------

Cancel **Update**

3.2.2 Application Security Policy

Go to Security -> Application Security -> Security Policies —> prebuilt-API_Security_Policy and make sure it is in blocking mode.

The screenshot shows the 'Policy Summary' page for the 'prebuilt-API_Security...' policy. The 'Virtual Server' field is set to 'api.vlab.f5demo.com'. The 'Enforcement Mode' field is set to 'Blocking'. Both of these fields are highlighted with red boxes.

Check hostname configuration for requested secured by this policy. Go to Security -> Application Security -> Headers -> Hostnames and make sure api.vlab.f5demo.com is configured. Any request with different hostname will be blocked by the policy.

The screenshot shows the 'Host Names' section of the Application Security Headers configuration. The host name 'api.vlab.f5demo.com' is listed and highlighted with a red box.

Check JSON profile configuration. Go to Security -> Application Security -> Content Profiles -> JSON Profiles --> prebuilt-API_LAB_JSON and check its settings.

Security » Application Security : Content Profiles : JSON Profiles » JSON Profile Properties

JSON Profile Properties

Profile Name	prebuilt-API_LAB_JSON
Description	(empty)
Maximum Total Length Of JSON Data	<input type="radio"/> Any <input checked="" type="radio"/> Length: 10000 bytes
Maximum Value Length	<input type="radio"/> Any <input checked="" type="radio"/> Length: 20 bytes
Maximum Structure Depth	<input type="radio"/> Any <input checked="" type="radio"/> Length: 10
Maximum Array Length	<input type="radio"/> Any <input checked="" type="radio"/> Length: 1000
Tolerate JSON Parsing Warnings	<input type="checkbox"/> Enabled
Parse Parameters	<input checked="" type="checkbox"/> Enabled

Cancel **Update**

Check for allowed URLs in the security policy. Go to Security -> Application Security -> Allowed URLs -> Allowed HTTP URLs. Click on **/department*** and check Header-Based Content Profiles - Make sure prebuilt-API_LAB-JSON content profile is attached to this URL.

Attack Signatures Header-Based Content Profiles HTML5 Cross-Domain Request Enforcement Meta Characters Methods Enforcement

Request Header Name	(explicit, case insensitive)			
Request Header Value	(wildcard, case sensitive)			
Request Body Handling	Form Data			
Add				
Order	Request Header Name	Request Header Value	Request Body Handling	Profile Name
1	Content-Type	*json*	JSON	prebuilt-API_LAB_JSON
default	Any		Apply value and content signatures	N/A
Up Down Delete				

Cancel **Update**

Check for allowed parameters in the security policy. Go to Security->Application Security->Parameters->Parameters List. Check length and other settings by clicking into each parameter. For example maximum length for parameter “salary” is 10.

Edit Parameter

Parameter Name	salary (<i>Explicit</i>)
Parameter Level	Global ▾
Perform Staging	<input type="checkbox"/> Enabled
Allow Empty Value	<input checked="" type="checkbox"/> Enabled
Allow Repeated Occurrences	<input checked="" type="checkbox"/> Enabled
Sensitive Parameter	<input type="checkbox"/> Enabled
Parameter Value Type	User-input value

Data Type

Data Type	Integer
Minimum Value	<input checked="" type="radio"/> Any <input type="radio"/> Value: 0
Maximum Value	<input checked="" type="radio"/> Any <input type="radio"/> Value: 0
Maximum Length	<input type="radio"/> Any <input checked="" type="radio"/> Value: 10

[Cancel](#)

[Update](#)

Check attack signatures configuration. Attack signature set is required to limit policy only to relevant signatures for the protected API. Go to Security -> Application Security -> Policy -> Policy Properties and click on Attack Signatures Configuration.

Policy General Features

HTTP protocol compliance failed (19 out of 19 subviolations are enabled) Learn Alarm Block

Attack Signatures

Learn	Alarm	Block	Signature Set Name	Signature Set Category
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	prebuilt-API_Lab_SigSet	User-defined

Enable Signature Staging

Updated Signature Enforcement Retain previous rule enforcement and place updated rule in staging Note: Newly added signatures are always placed in staging regardless of this setting.

Apply Response Signatures for these File Types

Add Delete

Click on prebuilt-API_Lab_SigSet, scroll down to “Signatures” section and examine the set of signatures.

3.2.3 API DoS Protection Profile

In this section you will check for configuration settings in DoS protection profile that will protect API interface against known bots and DoS attacks. Navigate to Security -> DoS Protection -> DoS Profiles and click on prebuilt-API_DoS profile. Check for the example of white list for known ip addresses.

Properties

Name	prebuilt-API_DoS
Partition	Common
Description	
Threshold Sensitivity	High

Whitelists

Default Whitelist	/Common/prebuilt-Wh
HTTP Whitelist	Use Default

Shared Objects

Filter Shared Objects

Address Lists (1)

- prebuilt-Whitelist_1

Properties

prebuilt-Whitelist_1

Description:

Contents:

Name/Address

1.1.1.1

Click on **Application Security** tab and make sure its enabled and Heavy URL Protection is activated.

Security » DoS Protection : DoS Profiles » prebuilt-API_DoS

Properties	Application Security																								
Application Security <table border="1"> <thead> <tr> <th colspan="2">General Settings</th> </tr> </thead> <tbody> <tr> <td>Proactive Bot Defense</td> <td>Off</td> </tr> <tr> <td>Bot Signatures</td> <td>✓</td> </tr> <tr> <td>Mobile Applications</td> <td>Off</td> </tr> <tr> <td>TPS-based Detection</td> <td>✓</td> </tr> <tr> <td>Behavioral & Stress-based Detection</td> <td>Off</td> </tr> <tr> <td>Record Traffic</td> <td>Off</td> </tr> </tbody> </table>		General Settings		Proactive Bot Defense	Off	Bot Signatures	✓	Mobile Applications	Off	TPS-based Detection	✓	Behavioral & Stress-based Detection	Off	Record Traffic	Off										
General Settings																									
Proactive Bot Defense	Off																								
Bot Signatures	✓																								
Mobile Applications	Off																								
TPS-based Detection	✓																								
Behavioral & Stress-based Detection	Off																								
Record Traffic	Off																								
Application Security » General Settings <table border="1"> <thead> <tr> <th colspan="3">Edit All</th> </tr> </thead> <tbody> <tr> <td>Application Security</td> <td>Enable this setting to protect your web application against DoS attacks.</td> <td>Enabled</td> </tr> <tr> <td>Heavy URL Protection</td> <td>Configure Heavy URL include list, automatic detection, and exclude list</td> <td>Automatic Detection: Disabled Heavy URLs: Not configured Ignored URLs: Not configured</td> </tr> <tr> <td>Geolocations</td> <td>Overrides the DoS profile's Geolocation Detection Criteria threshold settings by selecting countries from which to allow or block traffic during a DoS attack.</td> <td>Not configured</td> </tr> <tr> <td>Trigger iRule</td> <td>Enable this setting if you have an iRule that manages DoS events in a customized manner.</td> <td>Disabled</td> </tr> <tr> <td>Single Page Application</td> <td>Enable this setting if your website is a Single Page Application.</td> <td>Disabled</td> </tr> <tr> <td>URL Patterns <i>Example: /product/*php</i></td> <td>Configure URL patterns to be used. Each URL pattern defines a set of URLs which are logically the same URL with the varying part of the pattern acting as a parameter.</td> <td>Not configured</td> </tr> <tr> <td>Performance acceleration</td> <td>Configure TCP fastL4 profile to be used as fast-path for acceleration</td> <td>Disabled</td> </tr> </tbody> </table>		Edit All			Application Security	Enable this setting to protect your web application against DoS attacks.	Enabled	Heavy URL Protection	Configure Heavy URL include list, automatic detection, and exclude list	Automatic Detection: Disabled Heavy URLs: Not configured Ignored URLs: Not configured	Geolocations	Overrides the DoS profile's Geolocation Detection Criteria threshold settings by selecting countries from which to allow or block traffic during a DoS attack.	Not configured	Trigger iRule	Enable this setting if you have an iRule that manages DoS events in a customized manner.	Disabled	Single Page Application	Enable this setting if your website is a Single Page Application.	Disabled	URL Patterns <i>Example: /product/*php</i>	Configure URL patterns to be used. Each URL pattern defines a set of URLs which are logically the same URL with the varying part of the pattern acting as a parameter.	Not configured	Performance acceleration	Configure TCP fastL4 profile to be used as fast-path for acceleration	Disabled
Edit All																									
Application Security	Enable this setting to protect your web application against DoS attacks.	Enabled																							
Heavy URL Protection	Configure Heavy URL include list, automatic detection, and exclude list	Automatic Detection: Disabled Heavy URLs: Not configured Ignored URLs: Not configured																							
Geolocations	Overrides the DoS profile's Geolocation Detection Criteria threshold settings by selecting countries from which to allow or block traffic during a DoS attack.	Not configured																							
Trigger iRule	Enable this setting if you have an iRule that manages DoS events in a customized manner.	Disabled																							
Single Page Application	Enable this setting if your website is a Single Page Application.	Disabled																							
URL Patterns <i>Example: /product/*php</i>	Configure URL patterns to be used. Each URL pattern defines a set of URLs which are logically the same URL with the varying part of the pattern acting as a parameter.	Not configured																							
Performance acceleration	Configure TCP fastL4 profile to be used as fast-path for acceleration	Disabled																							

Go to **Bot Signatures** section, expand Bot Signature Categories and make sure Benign category is selected for blocking.

Go to **TPS-based Detection**, make sure it is in blocking mode and Thresholds Mode is set to Manual. Expand “How to detect attackers and which mitigation to use” section and check setting in subsection “By Source IP”.

Operation Mode	Specifies how the system reacts when it detects an attack.	Blocking	Edit
Thresholds Mode	Specifies what type of thresholds to use.	Manual	Edit
How to detect attackers and which mitigation to use	<p>By Source IP</p> <p>Consider an IP as an attacking entity if either of the following conditions occur:</p> <p>Relative Threshold: TPS increased by: 500 % and reached at least 2 transactions per second OR</p> <p>Absolute Threshold: TPS reached: 3 transactions per second</p> <p>Set default criteria</p> <p>Select mitigation methods to use on the attacking IP's:</p> <p><input type="checkbox"/> Client Side Integrity Defense <input type="checkbox"/> CAPTCHA Challenge <input checked="" type="checkbox"/> Request Blocking Rate Limit</p>	Close	

Hint: More information on DoS Prevention available in the manual:

https://support.f5.com/kb/en-us/products/big-ip_asm/manuals/product/asm-implementations-12-1-0/1.html

WE MAKE APPS



FASTER.
SMARTER.
SAFER.



F5 Networks, Inc. | f5.com