RSA NetWitness Using the REST API

Before You Begin

Requirements

This lab exercise guide is intended to be used in conjunction with the RSA University virtual lab ("eLab") environment.

If you have not already done so, please view the document "Accessing RSA University Virtual Labs", which is available on the content page of the RSA University site: https://community.rsa.com/community/training/content

Problems/Questions?

If you have any problems or questions, please email: rsauniversity@rsa.com

Additional Resources:

RSA University on YouTube: https://www.youtube.com/watch?v=EbyWW2aMz88

NetWitness Logs and Packets Documentation: https://sadocs.emc.com/

NetWitness Courses: https://community.rsa.com/community/training/overview

Objectives

At the end of this lab exercise, you should be able to:

- Use the NetWitness Web GUI to find metrics
- Perform REST queries from within a browser
- Use curl to automate REST calls to the backend
- Use Python scripts to programmatically set and/or get metrics over time
- Find metrics of interest within NetWitness Logs and Packets
- Secure the use of REST by using SSL/HTTPS and best practices
- Use includes in your scripts to query several Packet Decoders or appliances of the same type

Resources

As you complete these exercises, you can make use of the following resources. These resources appear under the Resources tab in the eLearning module:

- This lab guide
- Sample Python script that you can upgrade to HTTPS

Note

As you work through this lab exercise, there are times when you may need to troubleshoot issues that arise. You can always refer to the product documentation to verify settings and to troubleshoot. Remember that you still have access to the internet from your local computer if you need other assistance as you progress.

Exercise 1: Work with the Classroom Environment

Objectives:

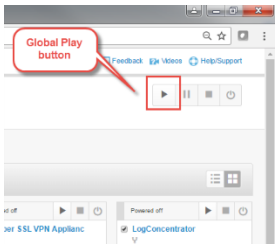
- Get familiar with the virtual environment which houses all the workstations, appliances, user interfaces, and classroom files that you will need to interact with to complete the lab.
- Understand how to navigate Skytap
- Pull up your workstation and log into the NetWitness Web UI
- Note that you will be working from a different Workstation later in the lab (not the jumphost)

Goal: Access the eLab environment, navigate and view the 14 appliances including your Workstations

Getting Started:

1. Access the environment by clicking the link that was provided to you for the On-Demand Lab.
2. Start the machines: Click the **black right-arrow** "Play button" at the **top** of the screen to start all the machines in this environment (instead of having to start one and then wait to start the next one). – see image below.

   WARNING: WAIT 3-5 Minutes for all of the virtual machines to fully boot! If one of them still says "Busy" the lab will fail. – Perfect time for a coffee or tea break.

   Scrolling up and down will reveal that you have 14 machines in your environment. Caution: The machines will auto-suspend after 30 minutes of inactivity. If so, **refresh** your browser to enable the **Global Play button** and re-start them by clicking the Global Play button (shown above).
3. Enter your Windows workstation: Once all of the VMs/boxes are running (green), click the machine entitled "**jumphost**." You are now looking at your Window Workstation. Notice that, at the top of the Desktop screen there is a Skytap toolbar that you can hide with the Up-arrow and show with the Down-arrow

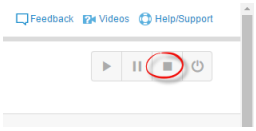
   

4. To navigate to the Linux Workstation (different VM/box in the Skytap environment): On the Skytap toolbar, Click the multi-monitor icon on the left to see all the machines in your environment at any time (you may need to scroll up and down to see all of them).

Your Environment Details:

Important Note: Your environment will suspend itself after ½ hour of inactivity. If you plan on being inactive, it is best to press the Global Stop button (shown below) and restart them when needed.



When you click the "Global" play button (top right of your screen) wait for 5 minutes for all machines turn green and fully boot up. If the "Spinney Wheel" continues, try clicking the browser refresh button. Again, use the Global Start or Stop button (not each machine's Play/Stop button one-by-one)

HINT: Remember this - If the environment seems to not be working, click the Recycle/Reload this page button in the browser

Passwords

All REST API usernames/passwords **`admin/netwitness`**

NetWitness web interface GUI (SA Server –browser) user/pw **`admin/netwitness`**

Linux box username/password **`root/Adm1npass!`**

Appliances:

- Domain Controller - Windows (10.101.240.38)
- Security Analytics/Reporting Engine - Linux (10.101.240.44)
- PacketDecoder - Linux (10.101.240.39)
- packetdecoder2 - Linux (10.101.240.29)
- Packetdecoder3 – Linux(10.101.240.19)
- Log Decoder - Linux (10.101.240.40)
- Packet Concentrator - Linux (10.101.240.45)
- Log Concentrator – Linux (10.101.240.41)
- Broker (10.101.240.46)
- Remote Log Collector – Linux (10.101.240.43)
- Archiver – Linux (10.101.240.37)
- ESA – Linux (10.101.240.36)

REST Ports:

Log Collector: 50101
Log Decoder: 50102
Broker: 50103
Packet Decoder: 50104
Concentrator: 50105
Appliance: 50106
Archiver: 50108
SA Server/Reporting Engine - ESA None

Student files containing code that you will need are located here: For Windows workstation (Jumphost):

C:\Classroom Files__REST_API_Lab_Environment_IPs_Ports_Etc\myLabEnvironmentAndCommands.rtf

For Redhat workstation: Desktop/ClassroomFiles/myLabEnvironmentAndCommands.txt (right-click and open with “Emacs Text Editor”)

Exercise 2: Review of Metrics from with the NetWitness GUI

Objectives:

- Work with the NetWitness Web GUI
- Navigate the Explore View of different appliance types
- Access the REST API visual GUI and compare with the Explore View of the NetWitness GUI
- Identify metrics of the NetWitness product that may be of interest

Goal: Navigate metrics within the GUI by going into the System Stats page of a Packet Decoder. Add a stat as a meter, view a historical chart, and expand and view the stat details. Compare Explore View of the Packet Decoder to the REST GUI.

As someone who is very familiar with the NetWitness product, some initial steps may seem basic to you but the reasons for doing them (to better understand REST) will become apparent as you proceed.

REST is used in order to access and potentially change metrics within the core services of NetWitness, but first, it is important to understand what some of these metrics are and when it may make sense to just access them with the GUI.

1. After all the VMs are running (green), click the one called **jumphost**. This is your Windows workstation desktop and serves as the machine that you would conduct your daily activities on at work.
2. Double-click the **Chrome shortcut** on the Desktop entitled “RSA Security Analytics Login” and login using the default NetWitness credentials (admin: netwitness) If needed, wait 3-5 minutes and reload to give the environment a chance to fully start up. Also, if it says “unsecure” just click through that to proceed.
3. Within the Netwitness GUI, click the **Main Menu->Administration->Health & Wellness**
   Be aware that there are three Packet Decoders in this environment.
4. Click on the **System Stats** Browser tab.
   Notice that you can find and view various metrics in the system here.
5. Click the **Component drop-down** menu and select the option entitled **Decoder**. Then click the **Apply** button.
   Notice on the left column that you are looking at metrics for “packetdecoder3.” At the bottom of the page, you could sift through metrics of the other decoders by viewing pages 2 of 5, 3 of 5 and so on.
6. Toward the right side of the page, click one of the **chart icons** to see a historical visualization of any one of the metrics. Because this is a tutorial environment, many of the charts are static. Notice that you can drag and drop a section of the chart to manipulate the time range viewed and that

you can hover over a point in time on the line or curve to get a value for that particular point in time.

7. Click **the x** to close that chart.

8. Select the statistic at the very top of the page entitled **Assembler packet Bytes**.

9. All the way to the right, click the **Stat Details** button to expand the details.
   You can view details about this particular metric.

10. Click the **Services** tab. Select the Packet Decoder entitled **PacketDecoder**

11. Click the **Gear** icon on the right and choose **View Stats**.
    Here are some Key Stats for this particular Packet Decoder as well as some Service System Information and Host System Information.

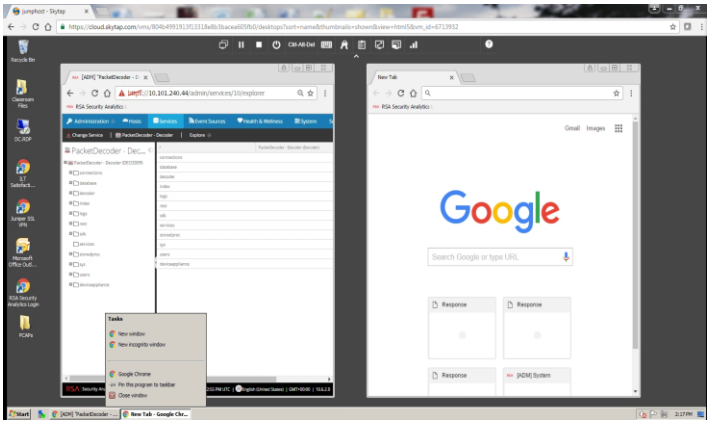
12. All the way on the right side of the screen click the **Charts Stats Tray** to expand it.

13. Click on the **Assembler Packets** stat and then Drag and drop it on the Gauges area.
    Notice that it added this particular statistic as a gauge. Notice the Timeline Charts toward the bottom of the page. Stats can be dragged and dropped here as well.

14. **Scroll down** and notice that there is also a section for Historical Timeline Charts.

15. Now that you have viewed several statistics and metrics within the GUI go back to Services. This time, select the **PacketDecoder** and the **Gear** icon, but go to **View Explore**.

16. Resize the GUI's browser window to make it smaller. Open a NEW browser window (in the system tray, right-click the browser icon, and choose new window). Line up the browser windows as shown in the graphic below.



Exercise 3: Compare Explore with REST and Navigate the REST GUI

Objectives:

- Work with the NetWitness Web GUI
- Work with the REST GUI (in a browser)
- Compare the REST API visual GUI with the Explore View of the NetWitness GUI
- Navigate node paths and Identify metrics of the NetWitness product that may be of interest

Goal: Become familiar with the hierarchy of REST information and ways to access it

Now that you have opened a new Browser window (not a new tab), make a REST call to that same Packet Decoder appliance by typing its IP Address and then the REST port for packet decoders.

1. Type `10.101.240.39:50104` and then press Enter.
2. Enter the default REST username and password of **admin** and **netwitness**

Congratulations on making your first REST call! This time using the REST API GUI which is a browser.

Notice that the directory structure of the Explore View (on the left) and the REST tree and nodes (on the right) are the same. They both have the same directory/tree structure, the same subdirectories/nodes, and the same parameters and values under each.

3. On the left Window (Explore view), in the left pane, click on **database**. The pane on the right of that same window now displays config and stats. Hover over stats and read the description of "A container node for other node types". Now double-click **stats** and notice all the parameters on the left and their corresponding values on the right.

   Service (such as Decoder)
   Has a **Tree** (directory of Nodes)
   Has **Nodes** (such as stats and possibly SubNodes)
   Has **Messages** (such as get, Info, Help) – each is a method or function
   Has **Parameters** (such as capture.rate)
   Has a **Value** (such as 0)

4. On the left pane, again, click **database**. But this time, on the right pane, right-click on **stats** and then click the **Properties** button.
5. Notice how this brings up a Properties box. Click the **down-arrow** to open the drop-down menu. Select the **info** method. Notice how the Message Help box tells you what the info method does. Click the **Send** button and then notice that the Response Output box displays info about the stats node. Now click the **down-arrow** and select the **Help** option. Click **Send**. The output displays things such as the data type and list of supported messages.

Exercise 4: Use REST GUI to build a URL

Objectives:

- Note how the breadcrumbs (queries) from within the NetWitness Web GUI in the Explore View are the basis for the URL strings that you will use within the REST GUI (in a browser)
- Use the REST Properties box to build a URL string for you that you can then use in a browser
- Change the URL string to query different nodes and different parameters

Goal: Become familiar with the URL string and how it relates to the query path

Change your focus to the browser window on the right that is displaying the REST GUI.

1. Click **database**. Now click **stats**. These are the same parameters and values that you saw earlier. In the upper-left, click **the .. (dot dot)** to navigate backwards. Click the **asterisk** within the parentheses next to the stats node.
2. Notice how the Properties box looks very similar to the Explore View. Click the **down-arrow** to open the drop-down menu. Select **info** and click **send**. Notice how the Ouput displays stats/.

   Just above the Output box notice that the REST API URL builder has built the appropriate URI for a call to the stats node and that the message is info and then the content is forced to plain text and that it will time out in 600 seconds.

3. Select this entire string (this **URL**), right-click, and click **Copy**.
4. Navigate to the Address box for this same browser window (where it currently displays **`10.101.240.39:50104/database`** and append the URI that you just copied by pasting it to the end of the existing URL. Eliminate any redundancy (such as the /database).
   Now it should read as follows:

   `10.101.240.39:50104/database/stats?msg=info&force-content-type=text/plain&expiry=600`

5. Press the **Enter** button. This is your second REST call. In this query, you asked the Packet Decoder entitled PacketDecoder to return info to you about the database/stats path. The returned value should say stats/.

6. Go back by clicking the **left arrow** button on the browser. It should display the nodes entitled config and stats.

7. Click the asterisk next to stats.
8. With the ls or “list” method already selected from the drop-down arrow, click the **Send** button. All the parameters for this node are listed in the Output box.

9. In the Output box, scroll down until you find the parameter entitled **`packet.oldest.file.time`**

10. Now, in the URL builder box where it says the following:
`/database/stats?msg=ls&force-content-type=text/plain&expiry=600`

11. Edit the URL by changing the path from /database/stats
To the following:

`/database/stats/packet.oldest.file.time`

The path now points to that parameter.

12. Next alter the URL string from a **`?msg=ls`**
to **`?msg=get`** (be sure to leave the question mark in place)

13. Copy the entire URL by right-clicking and selecting **Copy** (Ctl-C will likely not work)

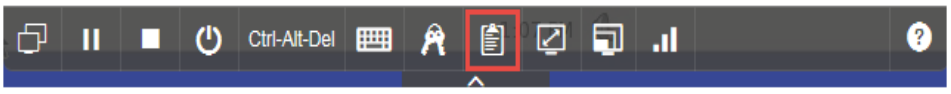
14. Navigate to the Address bar of the browser, with /database selected, just after the port number, right-click and select Paste. The URL should look like this:

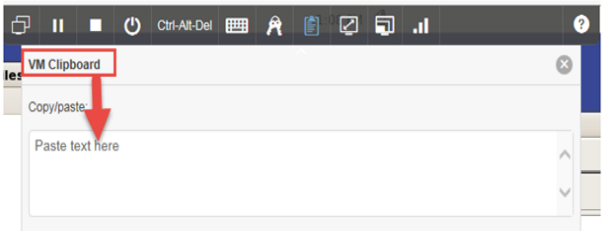
`10.101.240.39:50104/database/stats/packet.oldest.file.time?msg=get&force-content-type=text/plain&expiry=600`

15. Press the **Enter** button.
This is the date/time of the oldest packet on that particular Packet Decoder (Remember, we have three of them in this environment)

Write down that date/time here _____ .

HELPFUL HINT: It may come in handy – especially during a later exercise (Exercise 7 and beyond within the Linux Red Hat Workstation) – to know that Skytap copy and pasting can be somewhat difficult (right click and Copy tends to work where Ctl-C does not). Skytap itself has a Clipboard/buffer what can temporarily store text but this only works if you do it twice (copy into the Clipboard, copy out of the Clipboard – then do that same procedure AGAIN)





Exercise 5: Manipulate the URL to change the packet decoder and change the output

Objectives:

- Change your REST call to access metrics on a different appliance (but of the same type)
- Manipulate the URL string so that the Web Service sends an output in a different format

Goal: Become familiar with the hierarchy of REST information and ways to access it

1. In the browser URL, change the IP address to the Packet Decoder entitled packetdecoder2 which has an IP Address of **`10.101.240.29`** Next press **Enter**. Provide that web service with your default REST credentials (admin/netwitness). This is the time of the oldest packet on that Packet Decoder.
   Make a note of this date/time here _____ .

2. Now change it to **`Packetdecoder3 – Linux(10.101.240.19).`** Enter your **creds** and press **Enter** and view the returned information.
   Make a note of this date/time here _____ .

3. Now change the output from plain/text to **application/json** (JavaScript Object Notation). In the URL, just replace **`text/plain`** with **`application/json.`** Press **Enter**
   Notice that the web service on that Packet Decoder has returned a format that would be suitable for an application written in JavaScript.

4. Change the output from **`application/json`** to **`text/xml.`** Press Enter.
   Similarly, the output is now in xml.

5. So far, you have used the visual Graphic User Interface (a browser) of the REST API. As you saw, in the tutorial example, there were three different values for the oldest packet time. What would happen if you needed to get these values updated for every Packet Decoder every week? Instead of bookmarking so many decoders, it might make better sense to have a script that does it for you and populates a report.
   Before you navigate to a Linux workstation to write such a script, first do the following:

   Now edit your REST API call in the Address box so that the URL is just the IP Address and the port number for the Packet Decoder entitled **`PacketDecoder -10.101.240.39:50104`**

6. Navigate the tree and its nodes to the path/parameter of **`decoder/stats/capture.rate`**
   What is the current capture rate? _____ . You will work with this same parameter in an upcoming exercise.

   That concludes this exercise.

Exercise 6: Use the URL string in a Command Line Interface with curl

Objectives:

- Work with REST using a Command Line Interface (CLI)
- Make a REST call using curl
- View the results that the appliance's Web Service sends back to you

Goal: Become familiar with accessing REST in a non-GUI-based way

For this exercise, you will be using a different workstation. Instead of the jumphost (Windows 2008 machine), you will navigate over to the **Red Hat Enterprise Linux box**.

HINT: Troubleshooting – if your environment auto-suspends after 30 minutes of inactivity (again just click your browser's **Refresh** and then click the **Global Play button**), AND if you get an error of the network being down (eth0 down), just click System->Administration->Network, select eth0 and click 'Activate.'

1. Using the Skytap tool bar, click on the **left-most icon** to view all the machines in the environment.

Click View all VMs (14) and scroll down if needed, and click on the **Red Hat Linux** box.
WARNING: WAIT 1-2 Minutes for it to fully boot! The screen will change from red to blue.
(if you are on a Windows machine and need to shrink it, just press Ctl and the - key).
Log in using the following credentials**: root/Adm1npass!** After logging in, click the **Refresh** button for the browser to make the Skytap button-set appear, then expand the screen by clicking the **icon** shown circled in the graphic below in the Skytap menu.

Navigate to the following location on your Redhat workstation's Desktop: **ClassroomFiles** and then open **myLabEnvironmentAndCommands.txt** by right-clicking it and open it by clicking the "**Emacs Text Editor**" option.

2. curl is an http utility that enables Unix and Windows users (who have added it to their Windows environment) to communicate over http. In this case, at your Red Hat workstation, you will open a Command Line Interface and use curl to "get" information from a Packet Decoder located at a different IP Address using the curl command. You will supply a URL or REST call to curl and the Packet Decoder will return a value to your command line screen. In other words, the very same URL that you used in the browser earlier in the lab, you will now supply to curl.
Open the CLI by clicking **Applications** on the menubar. Then hover over **Accessories** and click the **Terminal** option to open a command window.

3. Type the following command exactly with no carriage returns (Unix is case-sensitive):

```
curl -u admin:netwitness
"http://10.101.240.39:50104/decoder/stats/capture.rate?msg=get&force-
content-type=text/plain&expiry=600"
```

HINT: Copy/Past/Edit by opening the Classroom Files folder on the Desktop, right-click the file entitled "myLabEnvironmentAndCommands.txt," and open with "Emacs txt Editor" to view all the code, default REST ports, IPs of the machines in your environment, and more. To copy/paste/edit syntax/code be aware that Skytap/Linux will require you to right-click Copy from the selected text in the editor then, in the Command window, right-click and paste. Also be aware that the code that you have selected and pasted may refer to a different IP, REST port, path etc. than what *you* are trying to query. Again, you can try using the Skytap Clipboard icon (but remember to copy/paste twice!)

(Note: there is a space between the word netwitness and double quotation marks for the http URL).

4. Next press **Enter**. Type the username/password of **admin/netwitness** and Press **Enter**.
   The expected result is **0**.

   You have now made a REST "get" call to the Packet Decoder and specified that you wanted the result to print to screen in plain text using a Terminal or Command Line without the use of a browser. Although you have successfully extracted a value, you have not saved it anywhere.

   That concludes this exercise.

Exercise 7: Write a script that uses curl and saves the output to a file on disk

Objectives:

- Programmatically make a REST call and save the plain text outputted value to disk.
- Trap the returned values (not just displaying output in a GUI or on the CLI window)

Goal: Edit curl script text file (instead of using the Unix text editor entitled vi) to make changes to a REST query and then run that script from the command line and then view the resulting output that has been saved to a file on your Red Hat workstation's disk.

Review the curl script entitled **myCurlScript.sh.txt** in the **ClassroomFiles** folder on the Red Hat **Desktop**.

In this exercise, your goal is to change the script so that it will "get" the Concentrator's Meta bytes volume for the last hour (**meta.bytes.last.hour**) and put that value into a text file entitled **decoderOutput.txt** and then to couple that value with today's date (from the Red Hat system).

Edit the scripts URL string to reflect the information that you are looking for. The **meta.bytes.last.hour** parameter can be found on the appliance type of **Concentrator** (see your lab guide for the IP Address of the Packet Concentrator). Concentrators have an appliance specific **port of 50105.** The node that contains that parameter and value is the **database/stats** node.

1. Open a file called **myCurlScript.sh.txt** found on the Desktop in the ClassroomFiles folder, change a line of code within it, and then **save it** back down with the same file name in order to overwrite the initial string of code that it contained.
2. Adjust the script to "get" the parameter that you want and then save it (as the same name myCurlScript.sh.txt)
3. Open the CLI terminal window again.
4. Type `ls -l`
5. Type `cd Desktop/`
6. Type `cd ClassroomFiles`
7. Type `ls -l`

The first thing to note is that the sample scripts and files have a .txt appended to the end of the files and this will have to be removed for them to be able to run.

We cannot execute these files right now because they are only read/write. Do a few things to make them executable and then validate that they can run.

Solution:

"http://10.101.240.45:50105/database/stats/meta.bytes.last.hour?msg=get&force-content-type=text/plain&expiry=600"

8. In the CLI, copy the curl script to one with a .sh extension by doing the following:
   Type **`cp myCurlScript.sh.txt myCurlScript.sh`**

   (If the system asks you "overwrite existing file"?, Type an answer of y for "yes").
   Now you can see that you have two versions of that file.

   You have the myCurlScript.sh.txt which you can easily edit in the "Emacs" tool. It is a simple text editor found on your Red Hat Desktop) to make a "template" or "known good copy"

   Then you can come back to the Command Line and change the mode or ability to execute the file with the **.sh** extension, that you will run a little later.

9. Now change the mode on the myCurlScript.sh file to executable.

   Type **`chmod +x myCurlScript.sh`**

10. Type **`ls -l`** to do a long listing of the files
    Notice that the file is now executable (-rwxr-xr-x)

11. Run the current file.
    Type **`./myCurlScript.sh`**

    The unix results of running the .sh file are returned and give you the number of bytes received and the entire curl transaction.

12. Type **ls -l** to list the files in this directory and notice the decoderOutput.txt file. This is the one that the myCurlScript.sh script wrote the results to. In other words, the results of the REST query have been written to the file decoderOutput.txt.
13. Cat that file to see its contents by typing the following:

    `cat decoderOutput.txt`

    And press **Enter**

The expected value of this Concentrator's meta.bytes.last.hour is the last entry in the decoderOutput.txt file for this metric. It is an ever-changing numeric value that may be five digits.

If you run the .sh again and cat the decoderOutput.txt, you will see two entries because the script tells it to append results to the end. So the expected result would be the first date and time and your metric and then the next date and time and your metric again.

(Solution: The URL that your script should have is shown below:)
`http://10.101.240.45:50105/database/stats/meta.bytes.last.hour`

That concludes this exercise.

Exercise 8: Write a simple Python script that pulls IPs from a .csv file

Objectives:

- Automate the process of making a REST call from within a Python script that would then allow you to manipulate the returned values with further calculations
- Write a .csv file that contains the IPs of all the Packet Decoders in the environment (You would normally write a Python script for each appliance type and a corresponding .csv for it)

Goal: Work with a Python script that makes a REST call. Write a .csv that will supply the IPs to the Python script. Run the script and note the returned values. 1. Create a .csv file. 2. Edit the script to make a simple REST call, 3. Save it down 4. Go to the CLI and convert it to .sh, enable it as an executable, 5. Run it and make note of the returned values.

Use a simple Python script (Overview)

1. Go to the Desktop of your Red Hat box, open the folder entitled **ClassroomFiles.**
2. Next, double-click the file **rest_script.py.txt**. Do not open the file that ends with .py but instead open the one that ends with .txt. Within the "Emacs" text editor, review the script.
3. The dialog asks to 'Run' or 'Display' -> Choose 'Display'

Even if you don't know Python, you can see that, when you save it down it will be stored in this Red Hat's binary directory. The script itself, imports the Python modules needed to work with URL requests. It then defines a variable that takes some parameters and prints any returned results to the command line interface. In the "main" logic, it defines some variables that correspond to a username and password. Recall that REST has a default username and password that you can supply at runtime. Normally, you would not want these hard coded into any script. For tutorial simplicity the script does however, contain them. The script then "includes" IPs that will be supplied to it from a Comma Separated Values file as an input to the script. Finally, it runs a "for" loop that contains the port number for a Packet Decoder, in this case, and a REST call. The REST call is a "get" call not a "set" call and, in this case, pulls the parameter entitled capture.received in plain text from whatever Packet Decoders you specify in the .csv file.

In other words, it will do a "get" call on all of the Packet Decoders and will return a value for each.

In order for your script to work, you will need to create a .csv file that houses all the IP Addresses of the Packet Decoders, in this case.

Task I: Create a CSV file with all the IP Addresses of all of the Packet Decoders (in this case).

As long as the .csv file has the devices that correlate with the appliance-type port that is in your query, then it will work. Thus, if you are looking for meta from Concentrators, then the port number will need to be the REST port number for Concentrators, and the IPs in the .csv file will need to be IPs of the

Concentrators in the environment. If one of the IPs in the .csv file is NOT a Concentrator the script will still run but will return a "does not exist" and will then move on. It won't break anything but it is not ideal.

The name of your Python script is rest_script.py and, from a naming convention stand-point in the field, it would normally be good to keep things "clean" by having .csv files and python scripts distinctly named and populated based on the appliance type. Brokers IPs would go in a myBroker.csv and would be included in a myBroker.py script and so on.

For this tutorial though, just keep the file names as they are. Thus you will have mydevices.csv and rest_script.py.

1. In the ClassroomFiles folder, open the **mydevice.csv** file. Type in a **list of the appliances' IP Addresses**. For this task it will be three **Packet Decoders**. In this case, since we want to get the oldest packet time from all of the Packet Decoders in our environment, type the following:

   Type each IP and then press **Enter** to start the next IP on its own line (this is how Unix will understand the delimiting of the IPs – with a Line Return not a comma)

   ```
   10.101.240.39
   10.101.240.29
   10.101.240.19
   ```

   These IPs represent the packet decoders entitled **PacketDecoder, packetdecoder2, and Packetdecoder3** respectively.

2. Click the **Save** button to save your .csv.

3. Go back to your CLI and type **ls -l** to confirm that your file is listed.

   Your **mydevices.csv** is now going to be an input into the a python script that you created entitled **rest_script.py**

   However, instead of "getting" the **capture.received** value for each Packet Decoder, your goal is to get the current amount of uptime that each Packet Decoder has been running. For this you will need to "get" the **/sys/stats/uptime** value.

Task II: Edit the rest_script.py.txt script

1. Change the REST call by editing the URL string. Change it from `/decoder/stats/capture.received` to `/sys/stats/uptime.`

   After you run this Python script, the returned results will tell you how long each of the Packet Decoders (whose IPs you have specified in the .csv file) have been running.

2. Save this text file. Click **File ->Save**.

3. Open the **Command Line** terminal.

4. Do an **ls -l** to make sure that you can see your file called **rest_script.py.txt.**

5. Change it from **a .txt** to a **.py** file extension.
   Type **`cp rest_script.py.txt rest_script.py`**

   (If the system asks you "overwrite existing file"?, Type an answer of **y** for "yes").

6. Do an **`ls -l`** and see that you have two versions of that file.
   If it isn't already enabled as an executable, change the mode on the **rest_script.py** file to executable.

7. Type **`chmod +x rest_script.py`**

8. Type **`ls -l`** to do a long listing of the files
   Notice that the file is now executable `(-rwxr-xr-x)`

9. Run the current file.
   Type **`./rest_script.py`**

10. Type a username of **admin** and press Enter

11. Type a password of **netwitness** and press Enter

    Notice that those three Packet Decoders returned the expected results in which you can see the uptime of each of those core appliances.

It should look something like this within your Console window:

```
9905, 2 hours 45 minutes 5 seconds
76529,21 hours 15 minutes 29 seconds
76512,21 hours 15 minutes 12 seconds
```

That concludes this exercise.

Exercise 9: Write a Python script that pulls the oldest packet time from every Packet Decoder, does a calculation, and provides you with the retention duration for each one.

Objectives:

- Perform additional logic on values (within the script) that have been pulled programmatically from NetWitness
- Automate the process of making a REST call from within a Python script
- Write a .csv file that contains the IPs of all the Packet Decoders in the environment

Goal: Fulfil management's need to know how long packets are being retained by the organization across the enterprise. Work with a Python script that makes a REST call, performs a calculation, and returns the answers to your business question. Specifically, pull the oldest packet time from each Packet Decoder, convert each to a number of days that the oldest packet has been retained and thus be able provide the duration for all Packet Decoders whenever you run the script.

Within this case-based scenario, Your Manager, at XYZ Corporation, has asked you to identify the oldest packet times on all Packet Decoders in the enterprise and needs the results in order to ensure that they are being retained for a required minimum of 4 weeks. You have decided to write a Python script that will get the results for you.

You already know that the parameter entitled **packet.oldest.file.time** will yield the values that you are interested in extracting from NetWitness and that it is located on every Packet Decoder under the `/database/stats` node.

1. On the Desktop of your Red Hat box, open the folder entitled **ClassroomFiles**. Next, double-click the file entitled **rest_packet-retention.py.txt** (not the file that ends with .py) to open it in the "Emac" text editor.

The rest_packet_retention.py.txt script is very much like the Python script that you worked with in a previous exercise except this one does some calculation to take a date/timestamp and calculate the difference between it and the Red Hat system's current date/time. Thus it is yielding the duration.

(system timestamp – oldest packet timestamp = duration) In other words, Raw retention of packets for one of the Packet Decoders is 50 days 20:40:02, for example.

2. Change the URL string in the script to the correct path/parameter. This should result in a standard REST "get" query from within a Python script (That could later be modified to write to a file, spreadsheet, third-party application, etc).

3. Go to your Command Line Interface and copy your **rest_packet_retention.py.txt** over to your **`rest_packet_retention.py`**

4. If the system asks you if you want to overwrite? Answer **y** for "yes" and press Enter
5. Do an **ls -l** if you wish and a chmod if needed

```
chmod +x rest_packet_retention.py
```

6. **Run** your Python script.

How long are your packets being retained on all of your Packet Decoders?

Your results should look something like the following:

Host 10.101.240.39 Packet Oldest time: 2016-10-24 17:10:49 Raw Retention 50 days, 20:40:02

Host 10.101.240.29 Packet Oldest time: 2016-12-07 17:10:49 Raw Retention 6 days, 17:40:4

Host 10.101.240.19 Packet Oldest time: 2016-12-07 20:52:00 Raw Retention 6 days, 16:58:5

Congratulations, you have just programmatically extracted data from NetWitness to solve a business problem! You've ensured that this platform is meeting your organization's security requirements. And, you have created a repeatable process that can now be performed again at any time by just running your script!

That completes this exercise.

.

Exercise 10: Research metrics that exist within NetWitness from the REST GUI

Objectives:

- Identify a repeatable and easy way to sift through nodes, sub-nodes, and parameters in order to identify the single parameter in which you may be interested
- Work with the depth property
- Be able to perform a data dump of all parameters of all nodes of all devices

Goal: Use the REST Properties box (Right-click on a Node's asterisk), use the List method and Depth property to view the output which lists all devices, nodes, sub-nodes, and parameters in NetWitness.

Now that you have used various methods of accessing metrics within NetWitness, you may want a way to find what those metrics are and where they are located without having to endlessly poke around in the Explore View within the GUI.

1. From within your Red Hat workstation, Click **Applications ->Internet ->Firefox** to open a browser. In the URL Address box, navigate to the Packet Decoder at the IP Address; **10.101.240.39** with a REST port of **50104**

2. Enter the default **credentials** and press Enter.

3. Click the **asterisk** next to the node entitled **database** (to go into the Properties)

4. Every REST node has a "depth" command. Click the Properties drop-down arrow. In the drop-down menu, select **ls** as the message. In the Parameters box type **depth=5** and press the **Send** button. The Output box displays all of the path/parameters for the database node.

5. To make it more human readable, select the entire URL from the URL builder (between the Message Help box and the Output box) and **copy it (Right click ->copy)**

6. Now paste **(Right-click ->Paste)** the URL into the browser's Address box and press **Enter.** Be careful to not set a double slash between the port number and the word database!

   You can see an organized list of all the sub-nodes for database grouped together and for each, you can see all the parameters. This way you can more easily sift through and find the parameter that you are interested in without having to click up and down node structures and through sub-nodes while hunting for one.

That completes this exercise and the lab. Feel free for "extra credit" to have the rest_packet-retention.py.txt script write the results to a file, or for greater accuracy, use REST to pull and correlate the timestamp of the appliance with its parameter value. If you enjoyed this On-Demand lab and eLearning, please take a few moments to complete the course survey for "RSA NetWitness Using the REST API" back in the eLearning browser. Hope you learned a lot about the possibilities of using the REST API.