FMC API Use Case – The Pinhole Self-Serve Tool v1

Last Updated: 12-June-2017

About This Lab

The goal of this hands-on lab is to display an example of using the FMC's API to accomplish a possible "real world" solution to a problem that otherwise would be intractable.

Example Corp has presented to you and your company a problem they are having. They are asking you for some sort of solution that alleviates the tension between their IT Department and Developer Department.

Example Corp's Developer Department is frustrated with how slow the IT Department is in fulfilling their requests for opening up access through the company firewall so that they can test their application from an "outside host". The IT Department explains that they are understaffed and find it difficult to complete the requests from the Developer Department in a timely manner. Additionally, the IT Department is frustrated with the Developer Department because they rarely provide information as to when they are done with the pinhole that was created in the firewall. Consequently this means that there are a lot of pinholes in the firewall that aren't necessary but no one in the IT Department knows which rules are "in use" and which aren't.

You and your company have created the "Pinhole Self-Serve Tool" application as the solution to Example Corp's problem. The "Pinhole Self-Serve Tool" allows the Developer Department the ability to "create their own firewall rules" on an "as needed" basis with no IT Department intervention. These pinholes are time sensitive and once they expire they will be "automatically" removed.

This dCloud demonstration is your company's presentation of the finished product (with a little bit of looking at the code) to Example Corp.

This lab includes the following Scenarios:

Requirements

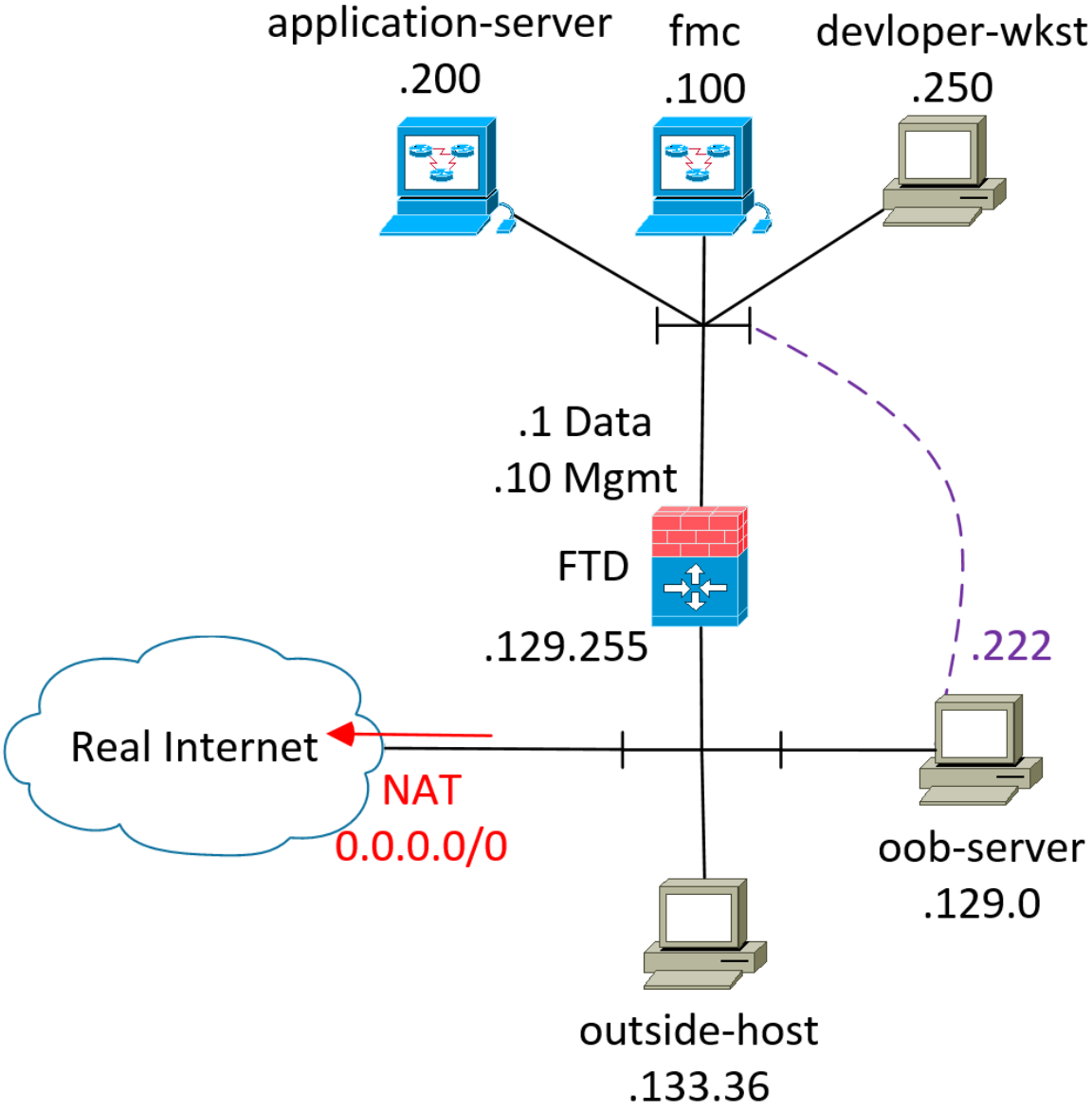
The table below outlines the requirements for this preconfigured demonstration.

Table 1. Requirements

Required	Optional
• An HTML5 capable web browser.	• Laptop with Cisco AnyConnect and an RDP client.

Topology

Example Corp's HQ location consists of a single "inside" network. Outbound traffic is PAT translated using the outside IP of the FTD for the public IP. There is a static NAT entry for the "developer-wkst" however. It uses 198.18.129.129 as its public IP.



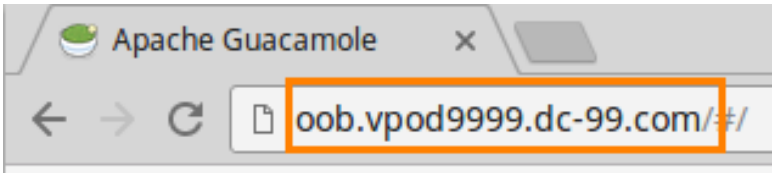
The purple dashed-lines and text indicate the Out-of-Band network that is used to provide RDP/SSH access to the respective workstations/servers within the lab. The purple network is not considered to be a part of the Example Corp topology.

Get Started

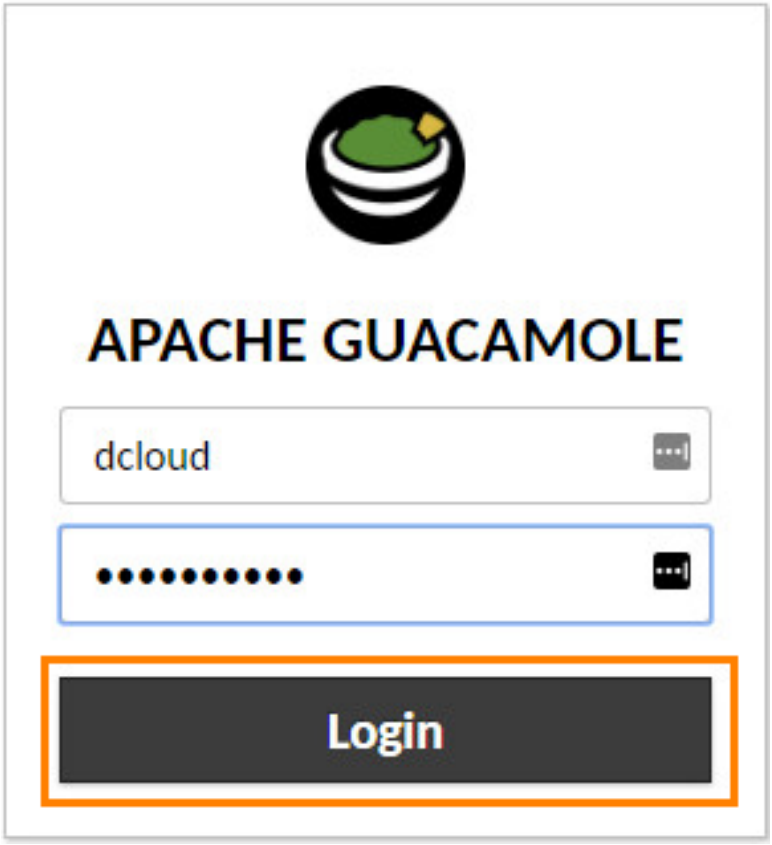
This lab environment is hosted by the dCloud organization within Cisco. You will be using the credentials provided on the dCloud website to establish an HTML5 "tunnel" from your workstation to the lab environment. You will use an HTML5 capable web browser

(most modern day web browser will work) to connect from your workstation to a virtual machine (called oob-server) from which you can access all the other machines within the lab environment. The oob-server machine isn't "inline" in the lab environment and only provides Out of Band access to the machines that are actually "in" the lab. Make sure you are on the correct machine when attempting to perform the actions listed in this lab guide.

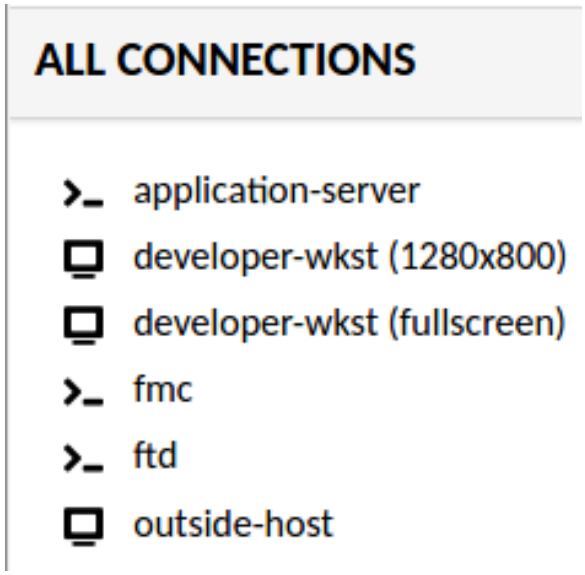
1. Go to http://dcloud.cisco.com and log in using your Cisco CCO credentials.
2. Get your lab's oob URL. It will be on the **Details** page of your dCloud session. (For example: oob.vpod9999.dc-99.com)
3. Using your local web browser access the oob URL. (For example: http://oob.vpod9999.dc-99.com)



4. Log in with the username of **dcloud** and use your **Session ID** as the password.



5. Once logged in you will have a link for each of the devices used in this lab. For **best practice**, right click on the link you want to use and **Open in a new tab**, otherwise you will navigate away from the main page and can't have more than one window open at a time.



There are two links for the developer-wkst because the FMC GUI requires a minimum of 1280 pixels for the resolution width. If you are on a small screen you might want to use that link. However, in the guide we will be using the "full screen" link.

6. (Optional) This would be a good time to test all the links to make sure all the VMs in the lab topology are working. **Right click** and **Open in a new tab** all the links listed.

Note: Having a new tab open for all the oob links is memory intensive on your computer. You may want to only open a few at a time and close the unused ones when not being referenced in the lab.

Page intentionally left blank.

Scenario 1. The “OLD” Way

Scenario Description

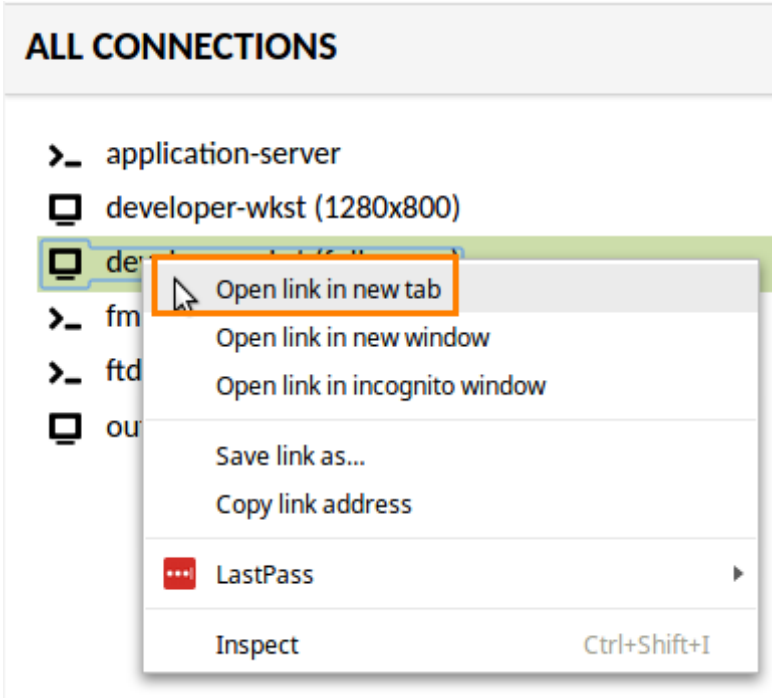
Prior to the “Pinhole Self-Serve Tool” application, the IT Department had to do the following in order to create a pinhole for the Developer Department. Let us pretend the Developer Department issued a support ticket to the IT Department to have port tcp/12345 opened to the developer-workstation (172.16.100.250).

The FTD does have NAT enabled. Thus, the Outside IP address associated with the developer-workstation is 198.18.129.129. Unfortunately, the FMC API does not support configuring NAT rules or NAT Policies yet. Therefore, this solution will only be modifying the Access Control Policy associated with the FTD.

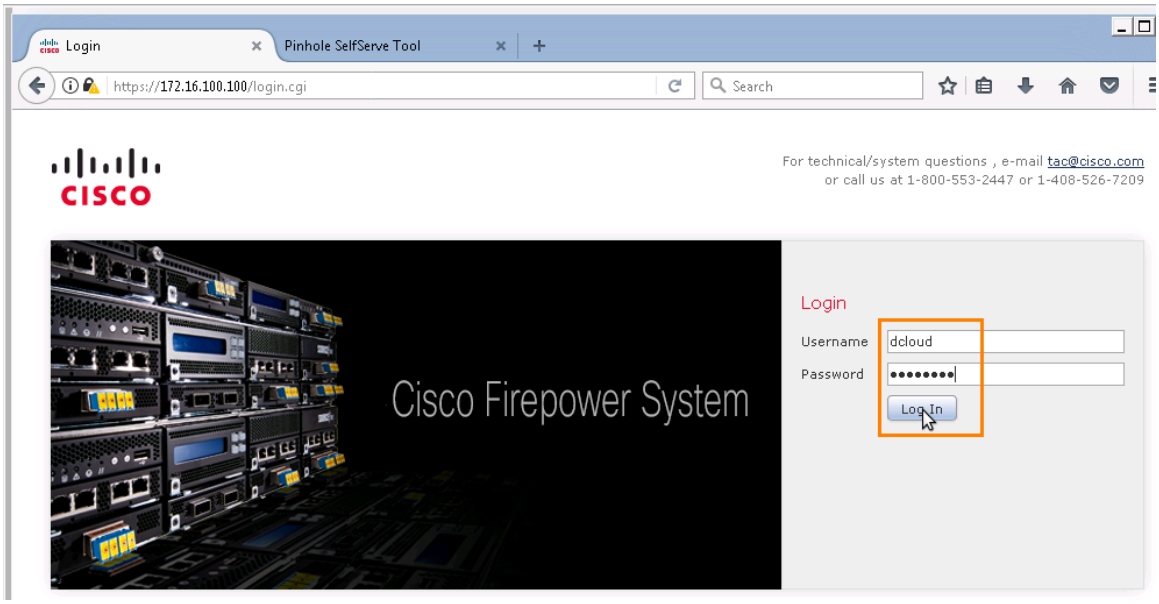
Configure the FTD

In order to save on the number of needed VMs in this demonstration the “developer-wkst” VM will double as the “IT-wkst” too.

1. From the **oob webpage** open a connection to the **developer-wkst** link. The link will automatically log you into that desktop as Administrator/C1sco12345.

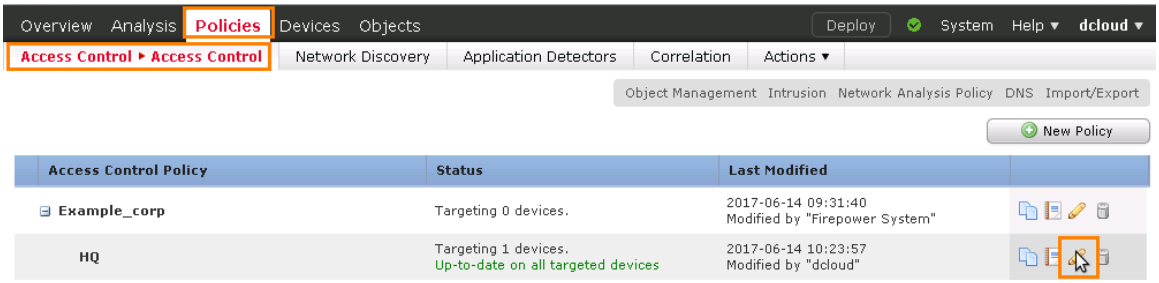


2. Open **Firefox** and login to the FMC website. (The first tab in Firefox should be set to 172.16.100.100. The user/pass is **dcloud/Admin123**.)

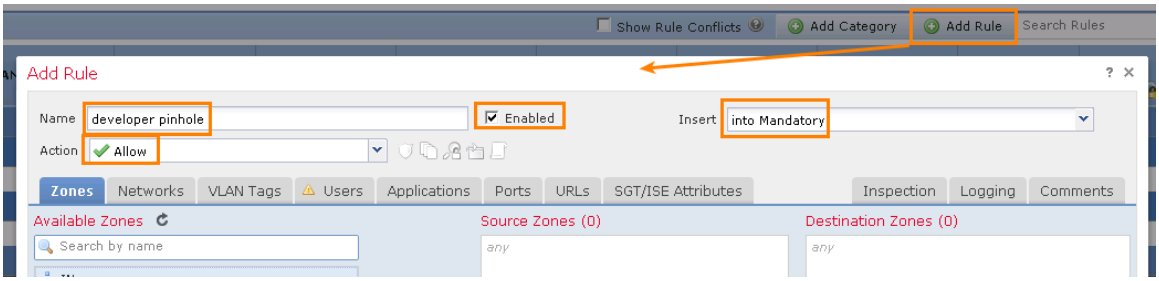


NOTE: The dcloud user has the same rights as Admin except it cannot modify the licensing.

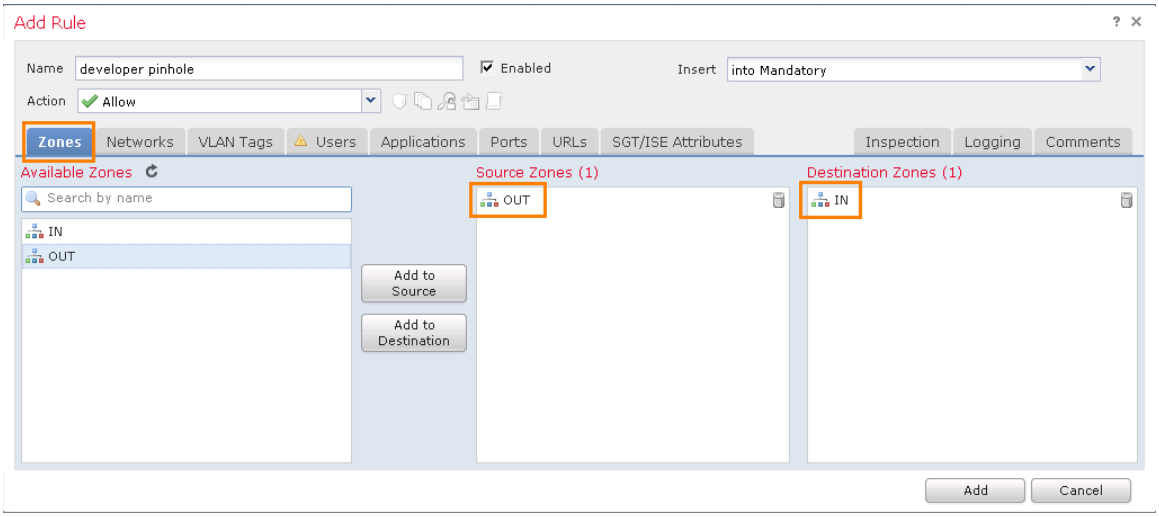
3. Navigate to **Policies > Access Control > Access Control**. Click the **pencil icon** for the **HQ** access control policy to edit that policy.



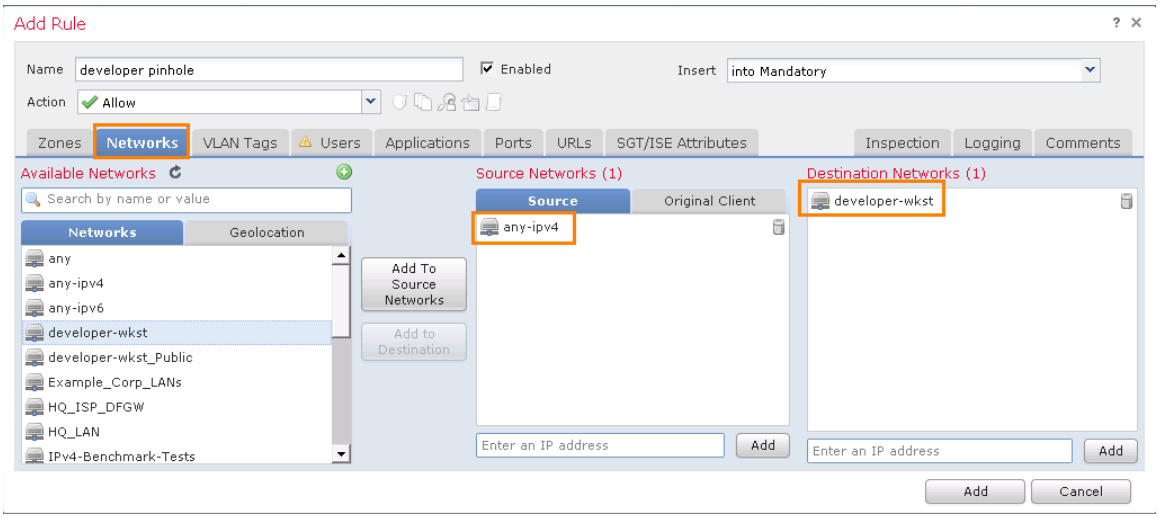
4. Click Add Rule. In the resulting popup fill out the following:
   - Name: **developer pinhole**
   - Enabled: **Checked**
   - Insert: **Into Mandatory**
   - Action: **Allow**



5. On the **Zones tab**, assign **OUT** to the **Source Zone** and **IN** to the **Destination Zone**.



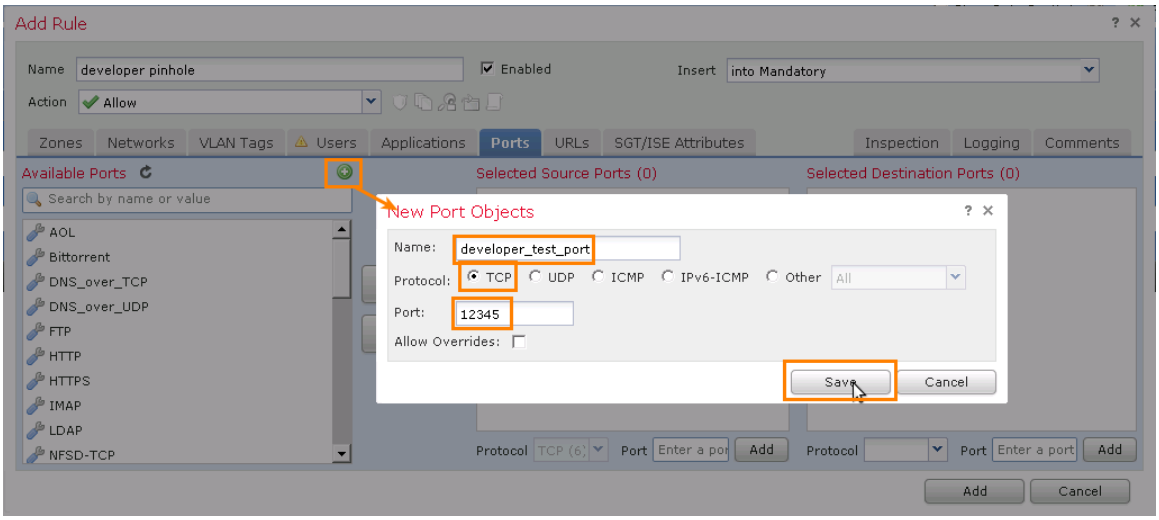
6. On the **Networks tab**, assign **any-ipv4** to the **Source Networks** and **developer-wkst** to the **Destination Networks**.



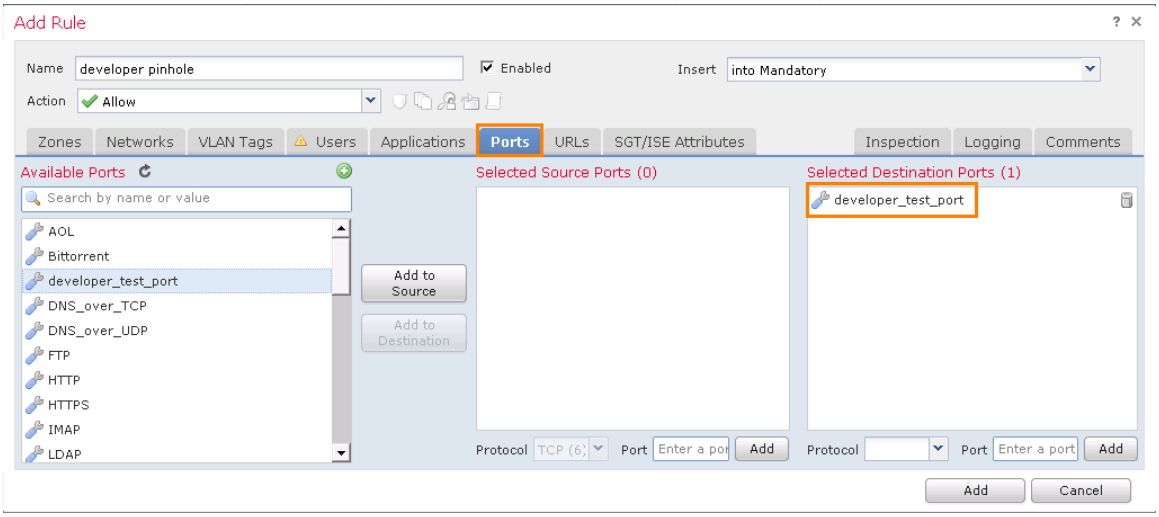
The developer-wkst Network object already existed because it is also used in the NAT policy rules.

7. On the **Ports tab**, click the **green plus icon** to add a custom port. In the resulting popup window fill out the following information and then click **Save**.

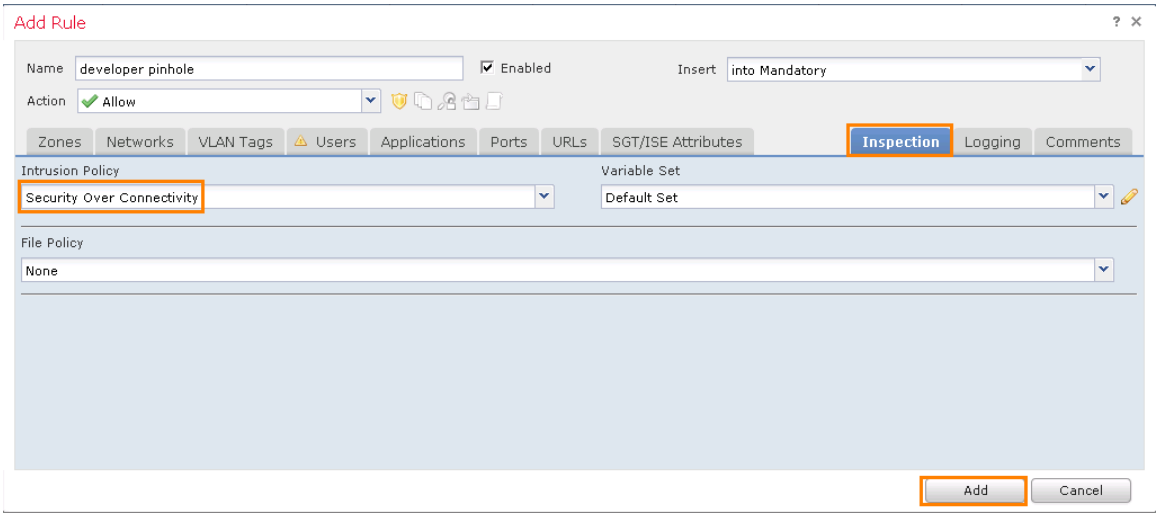
   - Name: **developer_test_port**
   - Port: **12345**



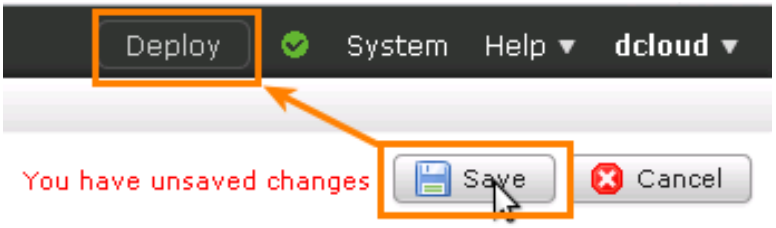
8. Now that the custom port exists, assign **developer-test-port** to the **Destination Ports**.



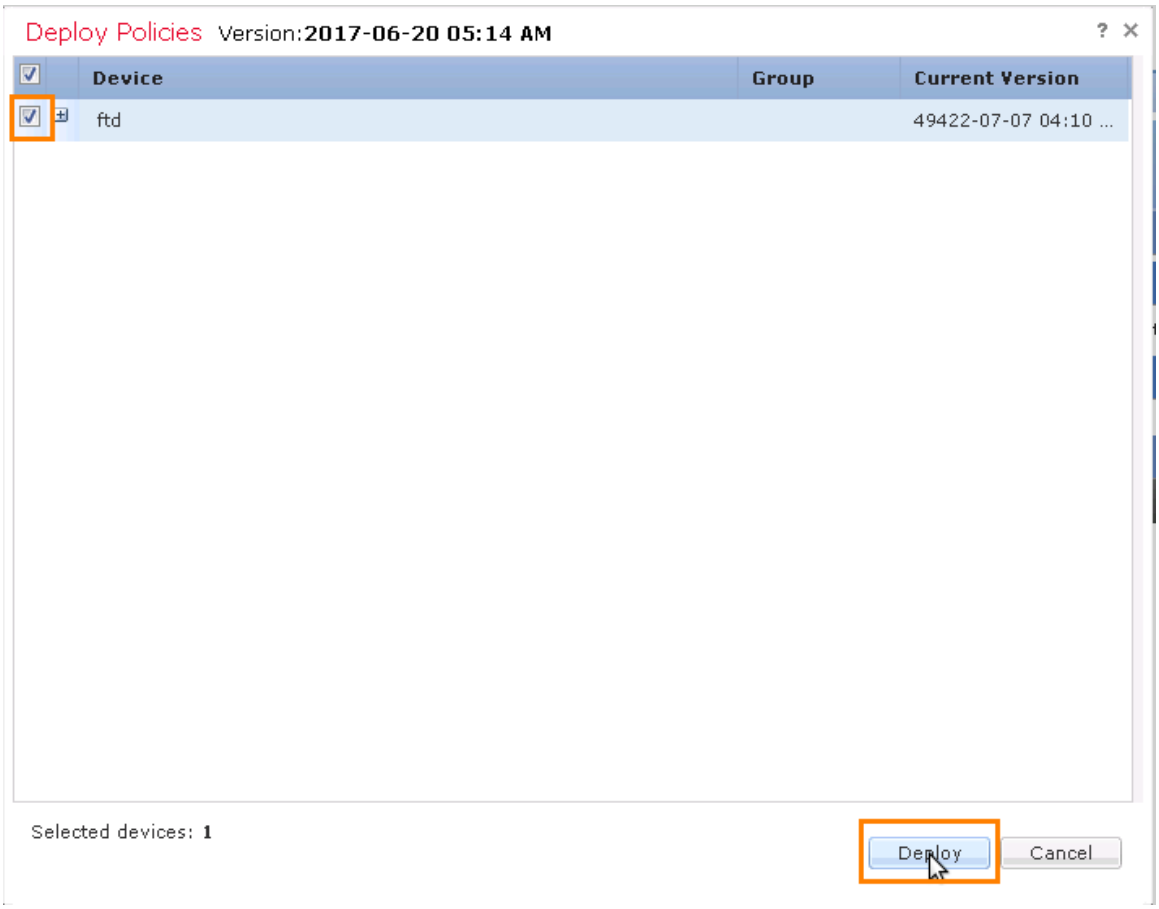
9. On the Inspection tab, select **Security Over Connectivity** for the **Intrusion Policy**. Click **Add** to create this access control policy rule.



10. Click **Save** and then **Deploy**.



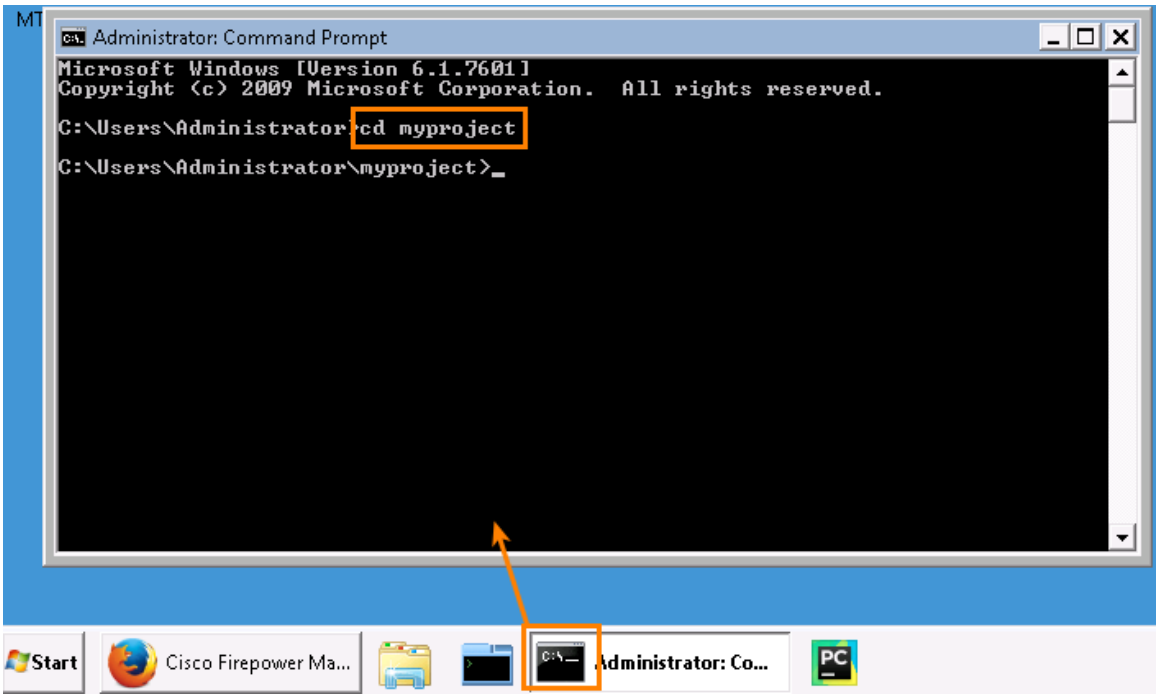
11. In the Deploy popup window **check the box** next to the **ftd** device and then click **Deploy**.



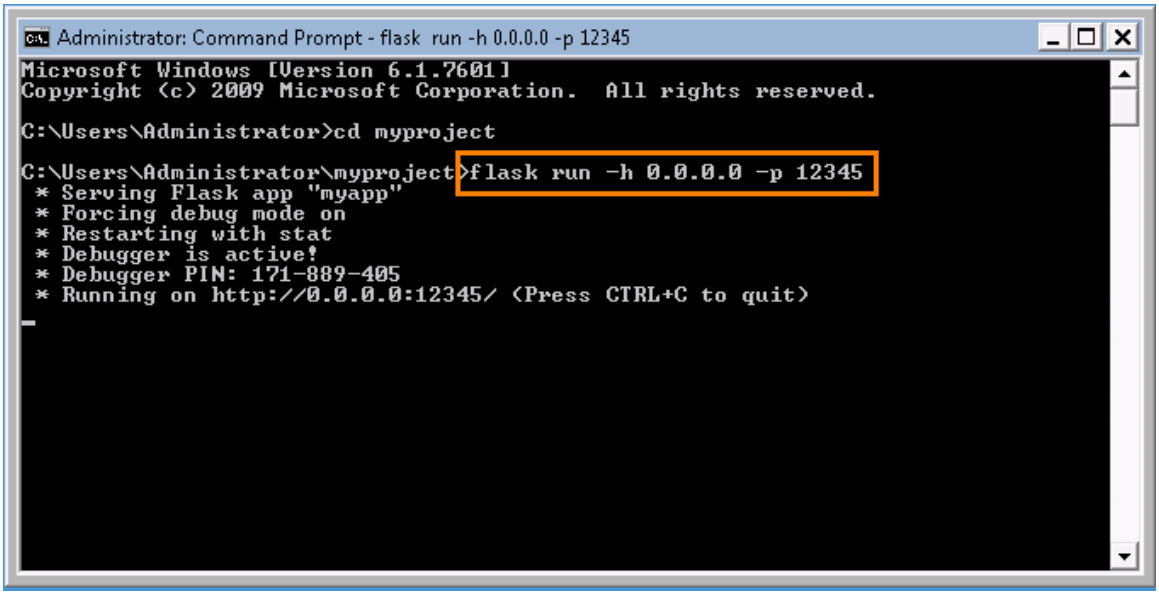
Test the pinhole

While the FMC deploys the configuration to the FTD device, let us set up the developer's test application. Though we are staying on the developer-wkst machine we are switching roles. Assume you are now the developer and you wish to test your application.

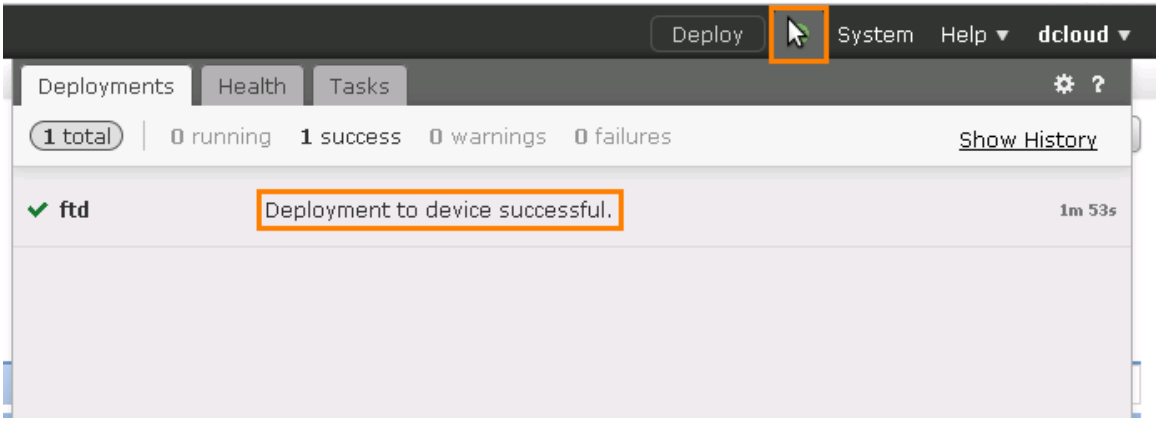
12. Open a **CMD prompt**. Change directory into **myproject**.



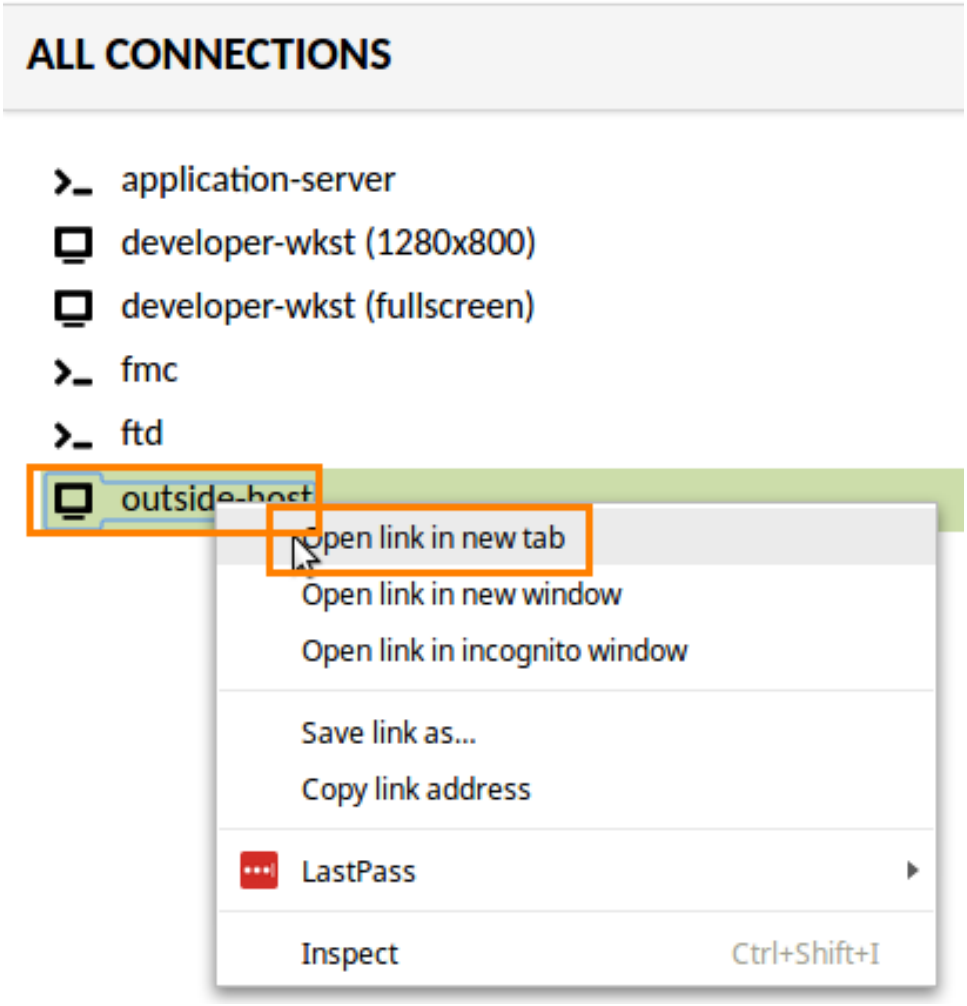
13. Issue the command: **flask run –h 0.0.0.0 –p 12345**. This will start the developer's application (a simple we page for our demonstration purposes) on port 12345 (the port opened in the ftd device).



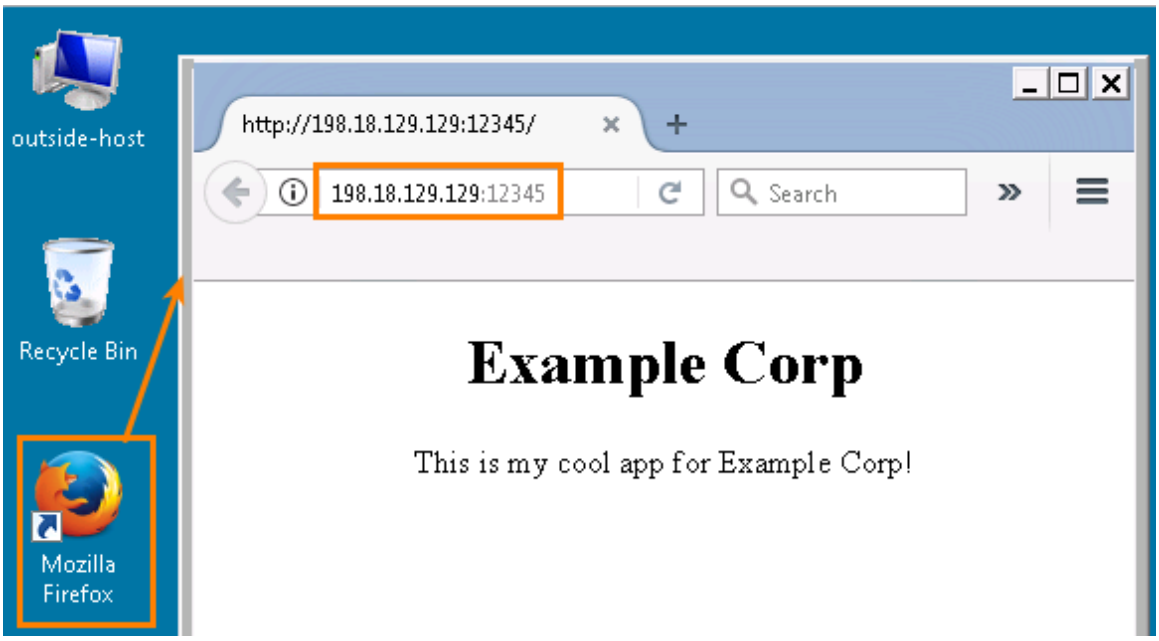
14. Ensure that the FMC has finished deploying the changes to the FTD. In the FMC GUI click the Message Center icon (the circle next to the Deploy button). On the Deployments tab the status of the deployment will be shown. Only continue when the status reports it has completed successfully.



15. From the **oob-server** list, open a connection to the **outside-host** machine.



16. On **outside-host** open up **Firefox** and access **http://198.18.129.129:12345**.

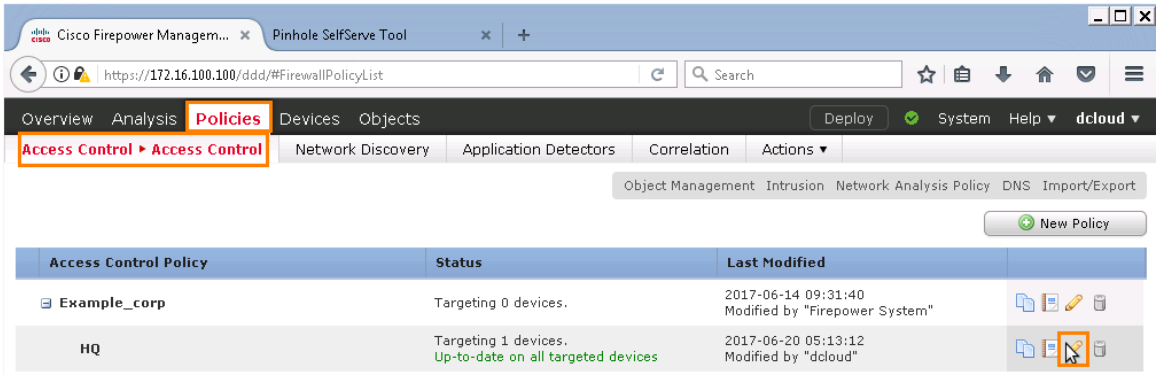


Clean up the FTD

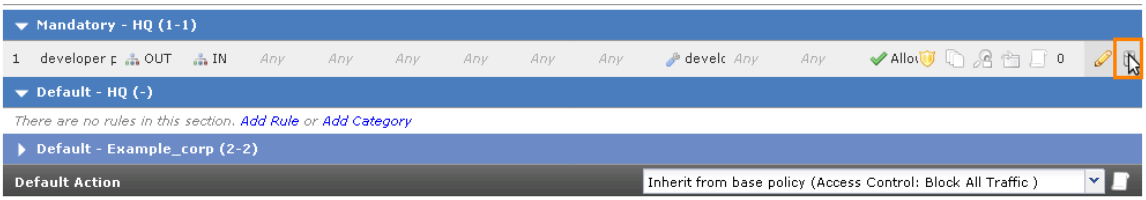
With the developer's testing of their application from the "outside" it is time to remove the pinhole. This step rarely gets done but "in a perfect world" the Developer Department would issue another support ticket to the IT Department to close the previously opened pinhole.

This, of course, assumes that everyone involved remembers the proper information in order for the IT Department to be able to identify the correct information to delete/remove.

17. On the **developer-wkst**, access the FMC GUI. **Navigate to Policies > Access Control > Access Control**. **Edit** the **HQ** policy.

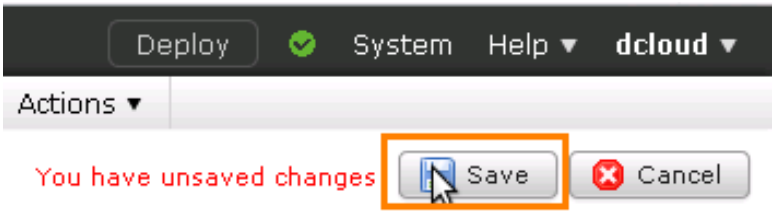


18. Delete the access control policy rule created earlier. Click the **trashcan icon** for the rule.

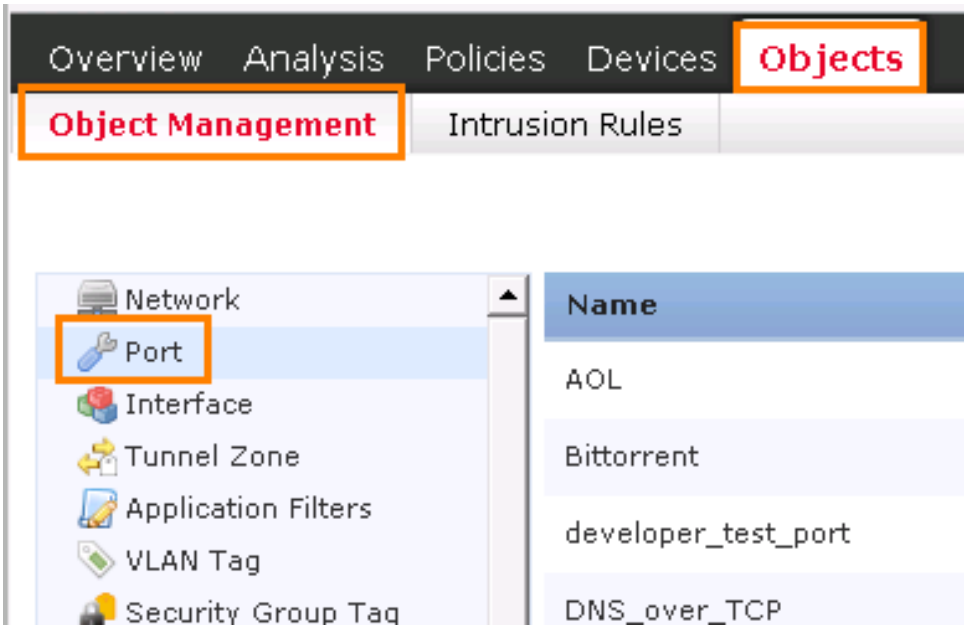


19. Confirm the deletion by clicking **Yes** in the popup window.

20. Click **Save** to complete the deletion.

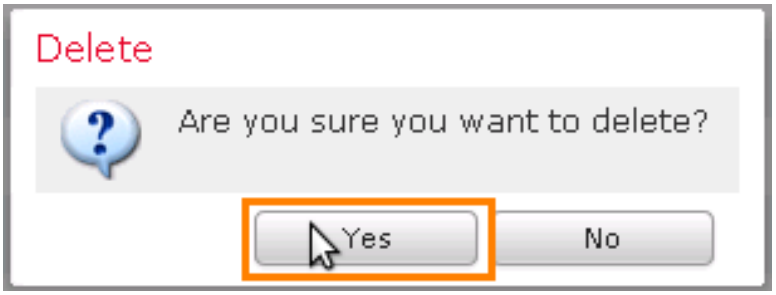


21. Navigate to **Objects > Object Management**. Select **Port** from the menu in the left side column.



22. Find the **developer_test_port** object and click the **trashcan icon** to delete it.

23. Click **Yes** to confirm deletion in resulting popup window.



24. Normally you would then need to deploy these changes to the FTD device. To save time we are skipping that step, as it will be done for us during the next scenario.

Notice that we did not delete the developer-wkst Network object. This is because that object is also used by the NAT Policy.

Scenario Summary

This scenario showed an example of what a Security engineer needs to do in order to create “pinholes” in the company firewall in order to allow access from the “outside” to the developer’s application. As you can see the process is slow and error prone. Additionally, the cleanup process is prone to either not happen or leave unused objects in the FMC.

Page intentionally left blank.

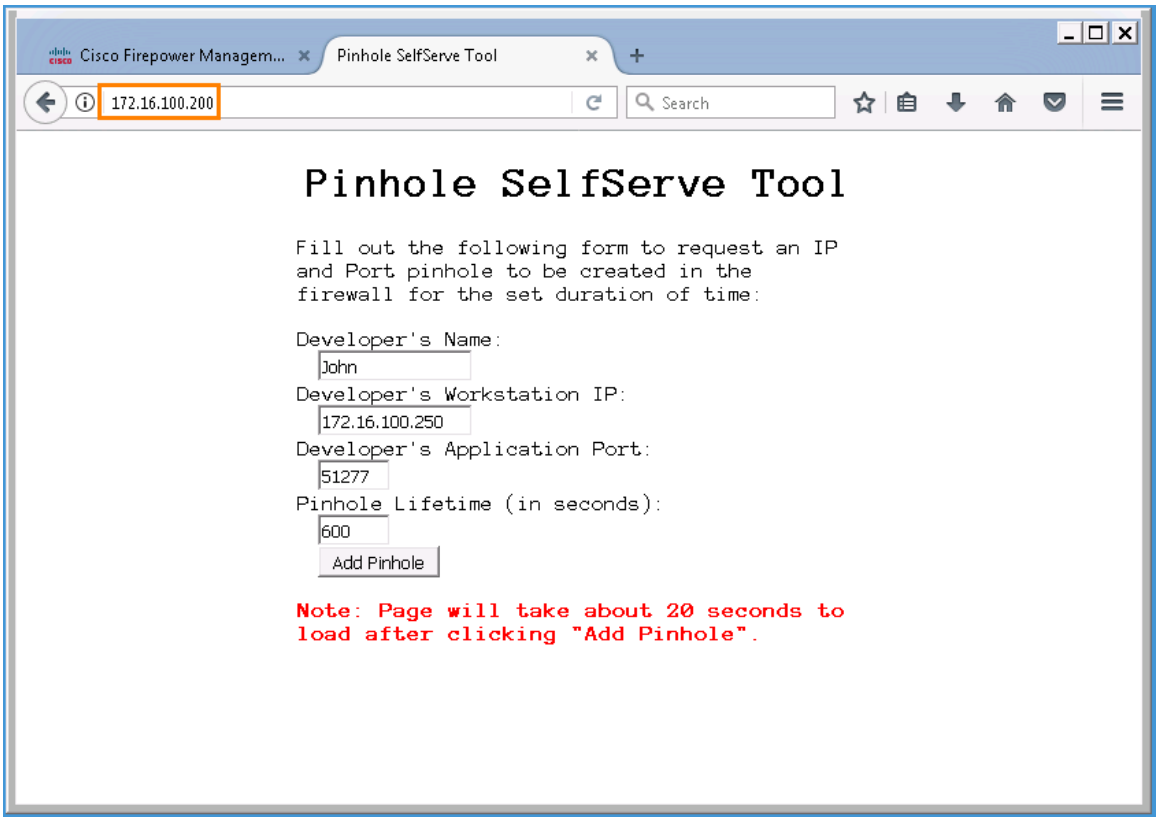
Scenario 2. Using the Pinhole Self-Serve Tool

Scenario Description

The “OLD” way is fraught with delays, miscommunication, and lost information. It is now time to present the “NEW” way for the Developer Department to test their application from the outside-host.

Configure the FTD

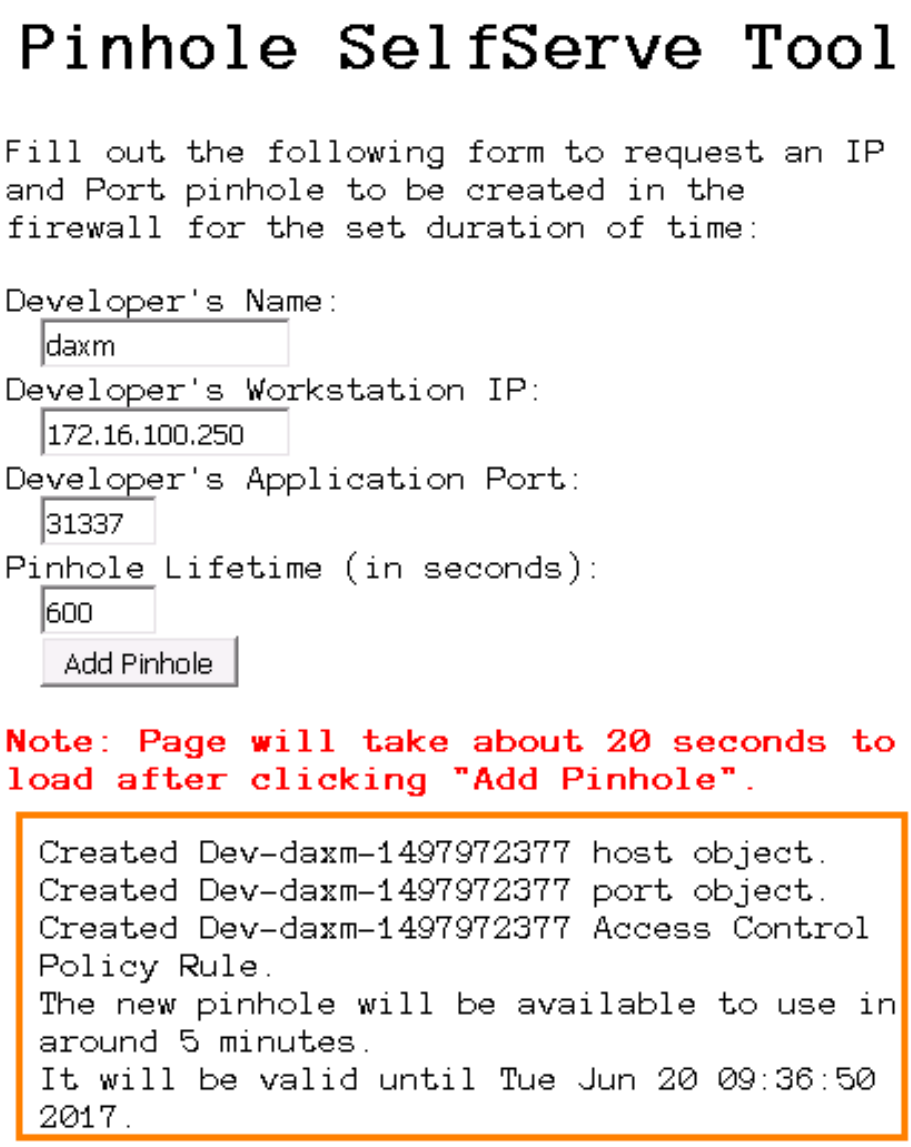
1. On the **developer-wkst** open up a new tab in **Firefox** and access the Pinhole Self-Serve Tool website (**http://172.16.100.200**). (The second tab should already be open to that URL).



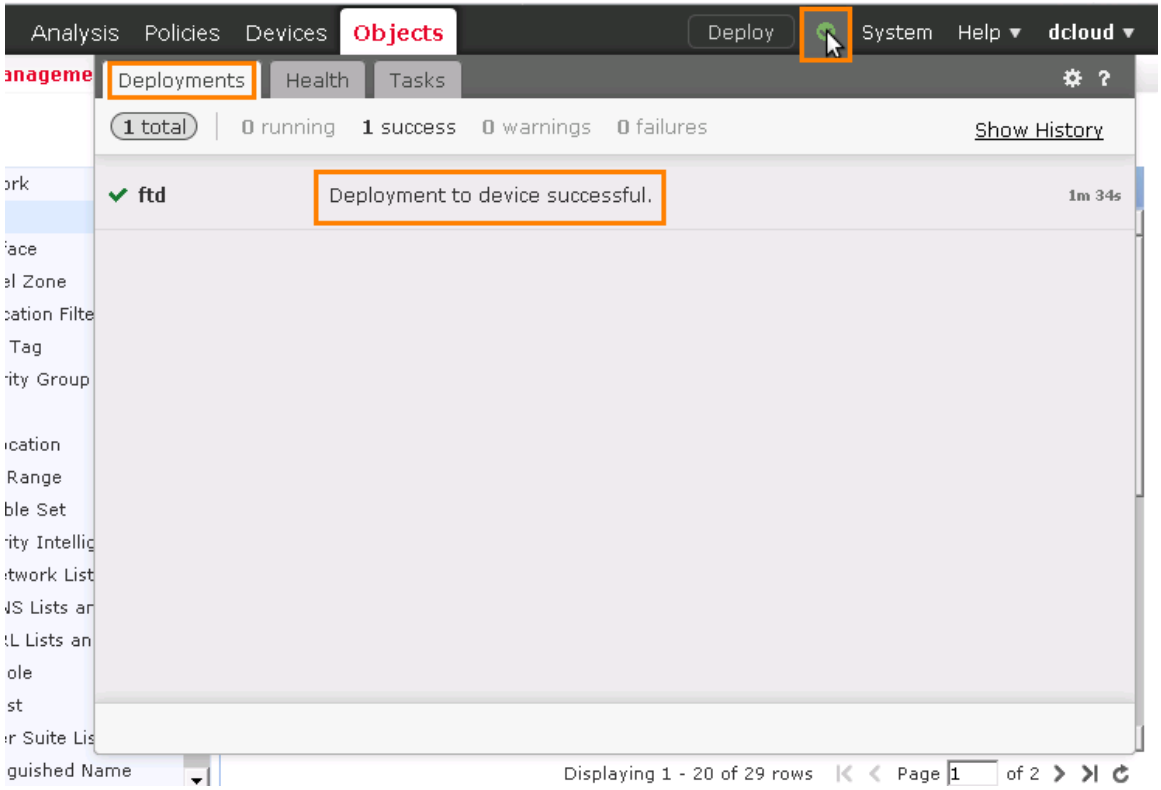
2. For the purposes of this lab the fields will automatically fill with valid options. Modify them if you wish. Take special note of the Developer's Application Port value as we will need to reference it soon. Click **Add Pinhole** when you are ready.

Don't set the **Pinhole Lifetime** value too large or you won't see the "clean up" functionality later in this lab.

3. The Pinhole Self-Serve Tool will connect to the FMC via its API, configure network and port objects as well as an access control rule; just like we did manually in the previous scenario. The Pinhole Self-Serve Tool will then deploy those changes to the FTD device automatically. When the tool is finished, it will provide some summary information describing what was done.

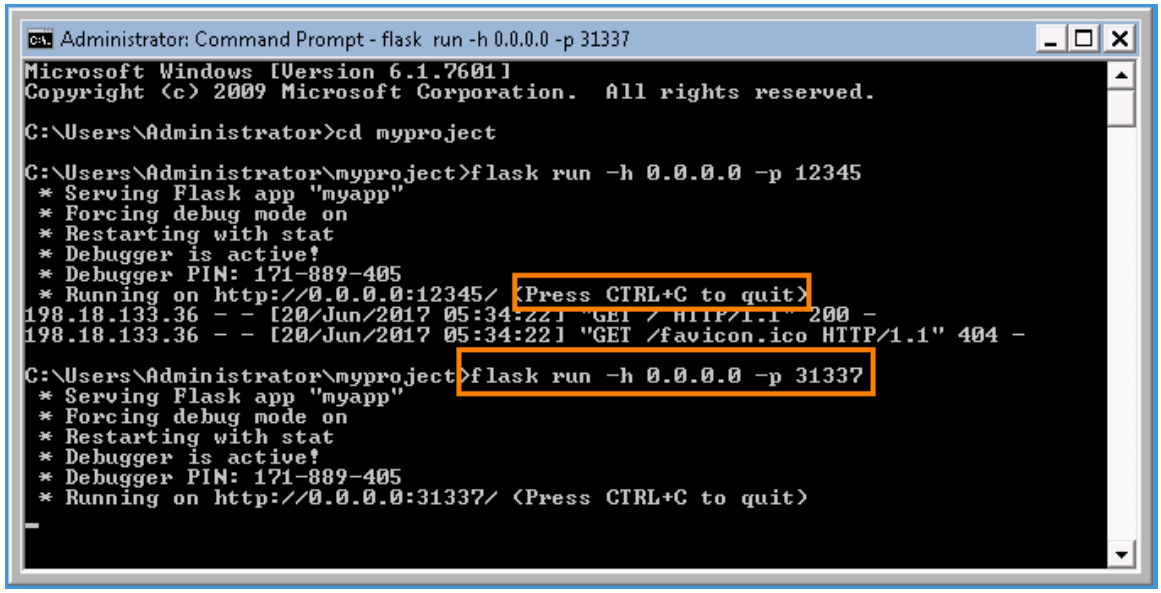


4. As mentioned in the summary the configured pinhole will not be available until the FMC is done deploying the configuration to the FTD. We can check on its status by accessing the FMC GUI and clicking on the Message Center icon again. Ensure that the deployment is complete before continuing the lab.

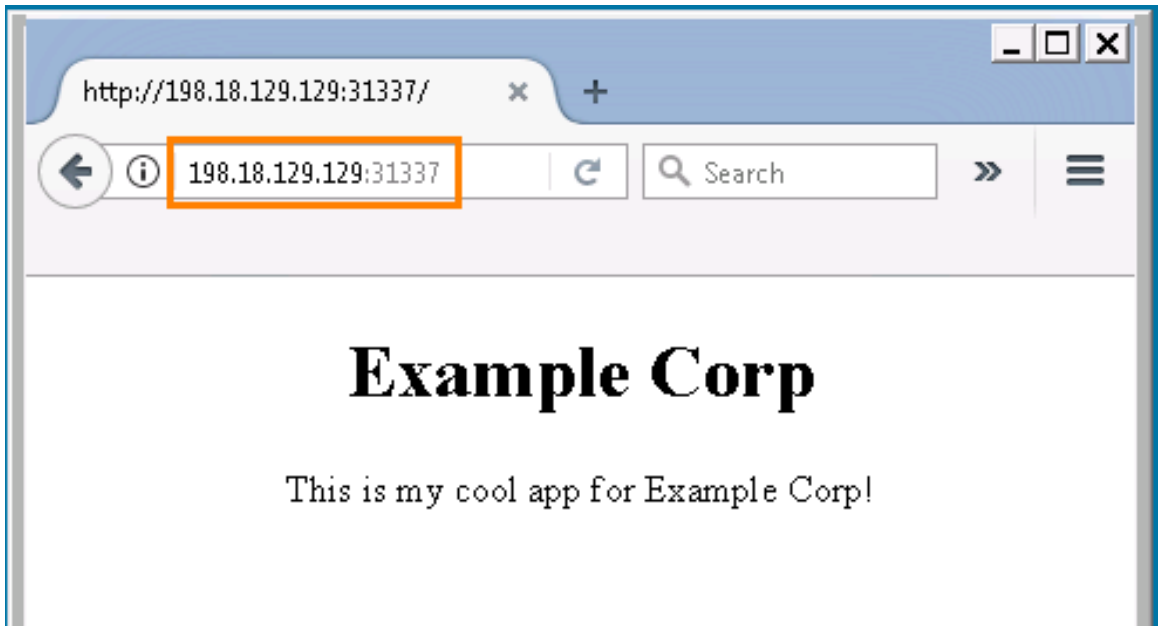


Test the pinhole

5. Now that the pinhole is open we need to configure the developer's application to use the opened port. Open the **CMD** prompt that is running the **Flask** application. Press **Ctrl-c** to stop the program. Then issue the command **flask run –h 0.0.0.0 –p 31337** (use your port if it is different) to restart Flask on that port.



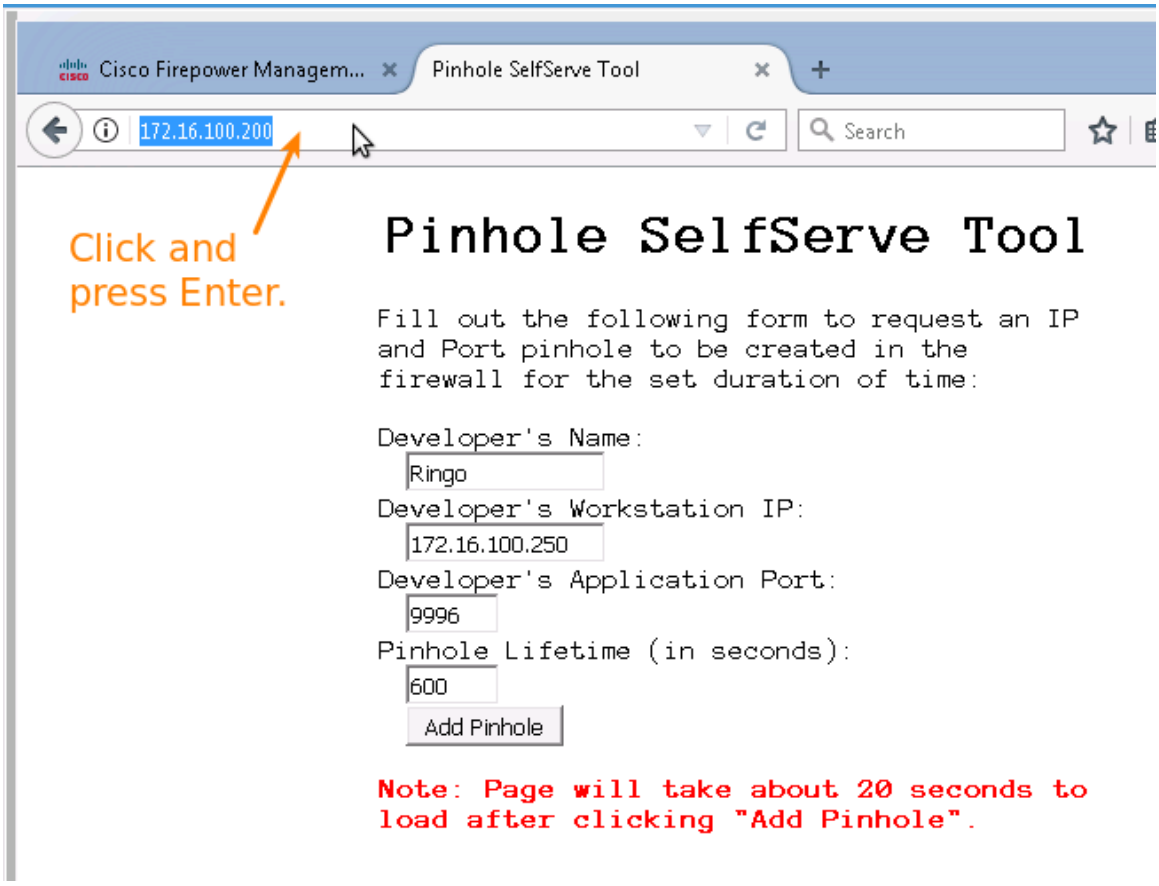
6. On the **outside-host** machine, access the new URL, **http://198.18.129.129:31337** (use your port if different).



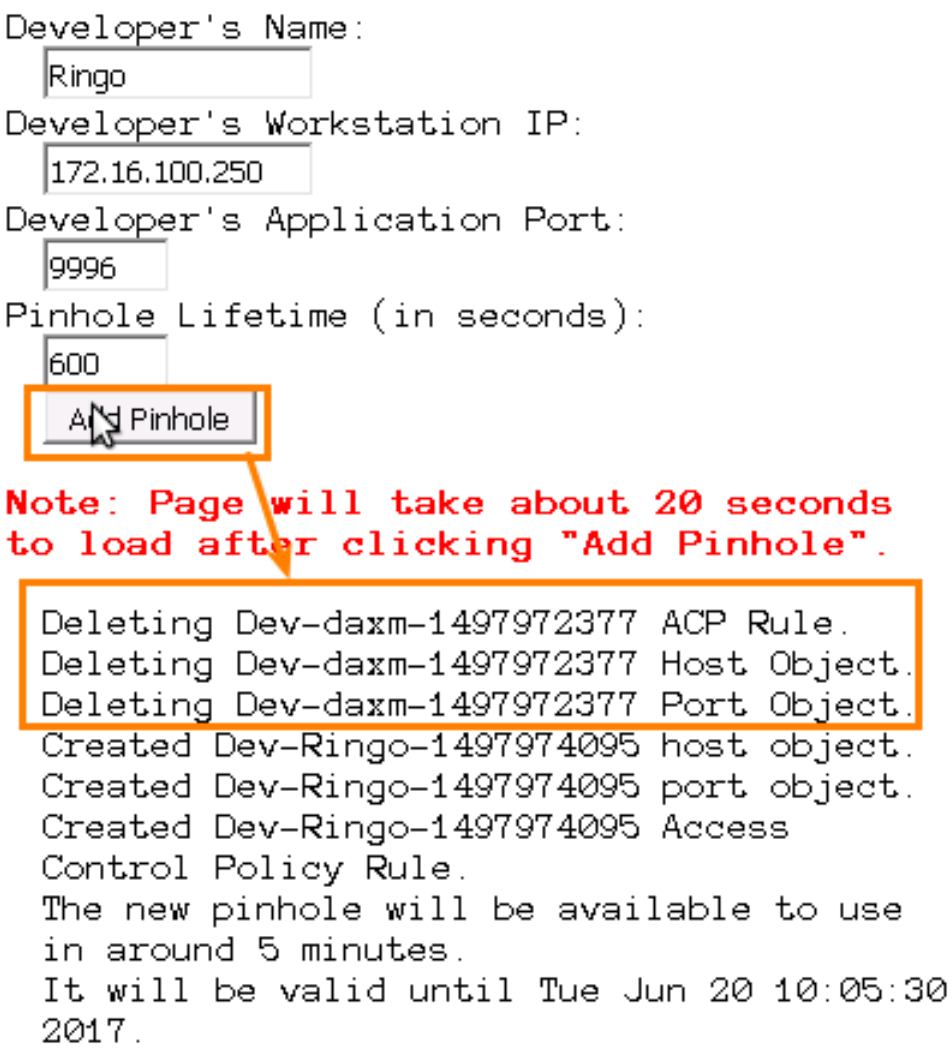
Clean up the FTD

Clean up is easy. Each time the Pinhole Self-Serve Tool runs it checks all the existing rules and objects to see if anything is past its expiration date. So each time a new pinhole is created it is also cleaning out old pinholes! In the lab this clean up code is only associated with the Add Pinhole function. However, it could easily be set up to run at regular intervals on the server to clean up expired entries outside of when adding new ones.

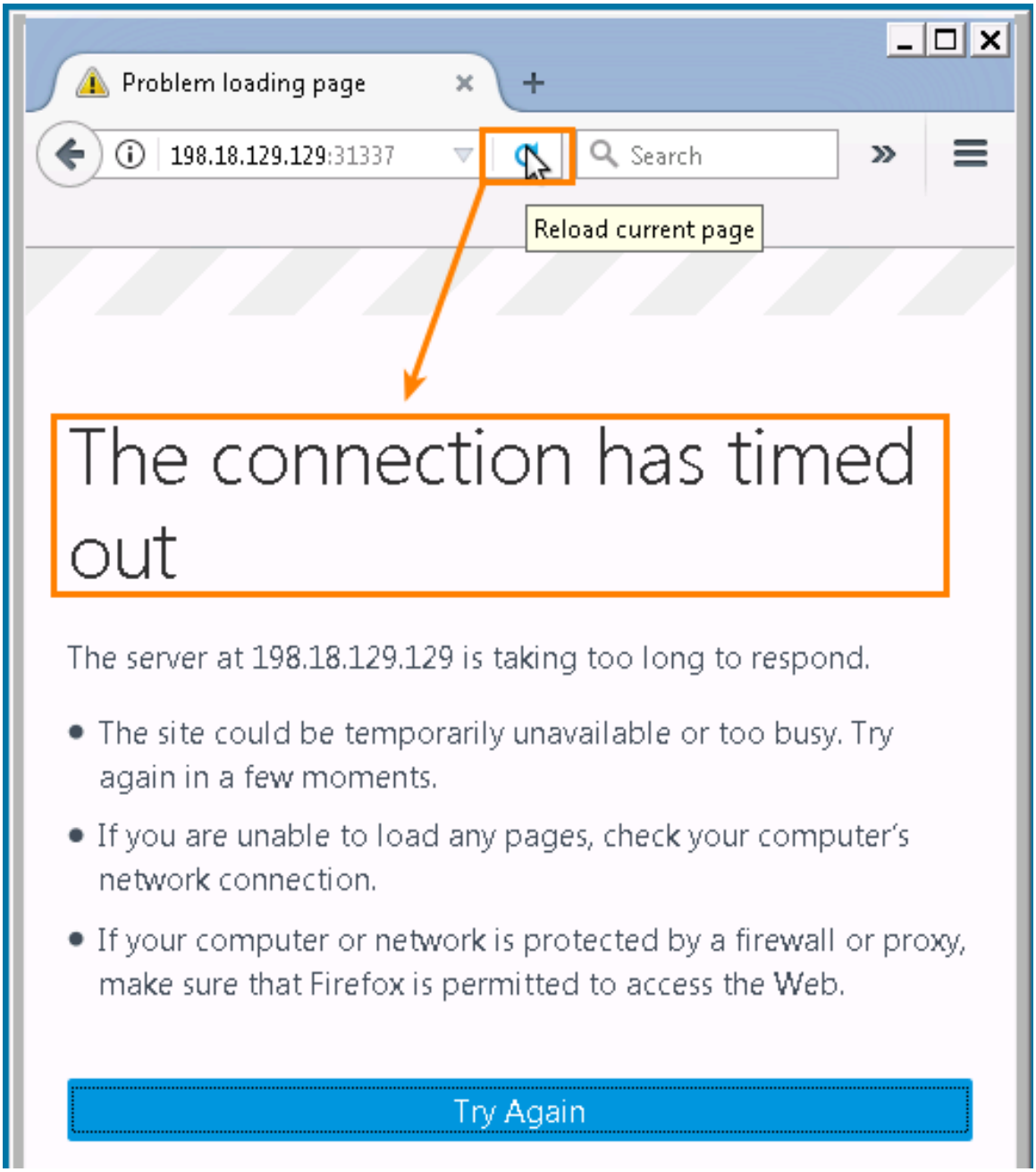
7. Return to the **Pinhole Self-Serve Tool webpage** on the **developer-wkst**. Refresh the page by clicking in the address bar area and then press Enter on you keyboard.



8. If enough time has passed (10 minute by default) click the **Add Pinhole** button again. Be sure to NOT use the same port used in the last created pinhole. When the page refreshes note the summary information and that the expired entries are removed.



9. Once the FMC is done deploying the changes to the FTD the previously open port no longer works. On the outside-host try to refresh the web page. (The timeout of the page can take some time.)



Scenario Summary

Using the power of the FMC's API functionality the workload of the IT Department's staff was reduced while simultaneously providing more timely responsiveness to the Development Department's needs. Additionally there is no communication confusion nor un-used pinholes left open.

Congratulations!
You have completed this lab!

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **www.cisco.com/go/offices.**