VM-Series for GCP

GCP XFF Blocking Deployment Guide

How to leverage Cloud Armor to restrict traffic based upon XFF headers in GCP

http://www.paloaltonetworks.com

Last Update: July 2018

Table of Contents

Version History

Version number	Comments
1.0	Initial Draft

Overview

When using the HTTP/HTTPS load balancer in the Google Cloud Environment, the load balancer will perform a source NAT on the original client IP, hiding it behind an IP address in the following ranges: 130.211.0.0/22 and 35.191.0.0/16. In addition to randomly changing the IP address used for NAT'ing each connection, IP addresses from these pools are also used for the source of health checks. To compensate for the loss of fidelity, the load balancer inserts an X-Forwarded-For (XFF) header into the request. The XFF header is of format:

X-Forwarded-For: <unverified IP(s)>, <immediate client IP>, <global forwarding rule external IP>, <proxies running in GCP>

And represents a comma-separated list of IP addresses appended by the intermediaries the request traveled through.

As the firewall sees multiple IP addresses, it is unable to distinguish the original source for the purposes of filtering future events in the event a threat is detected. One solution to this is to use a worker node to extract the original source from the XFF and use GCP Cloud Armor to filter the bad actor at the edge.

GCP Cloud Armor is a denial of service/web attack prevention tool that is integrated into the HTTP(S) load balancer and is a good option for mitigating certain types of web-based attacks. More information can be found here:

https://cloud.google.com/armor/

Demo Steps

1) This demo uses detectable threats (in this case SQL Injection attack) between a client browser and the web server to trigger action-oriented log forwarding.
2) The firewall detects the attack and forwards the Threat log to the worker node via an HTTP log forwarding action.
3) The worker node queries the firewall for the URL filtering log based on the sessionID, NAT Source Port, and received time of the detected threat. The response includes the IP address in the X-Forwarded-For HTTP header.
4) The worker node extracts the IP of the attacker from the X-Forwarded-For field.
   The worker node determines the correct rule priority and adds a rule to the Cloud Armor security policy.

Support Policy

This document is released under an as-is, best effort, support policy. The associated scripts should be seen as community supported and Palo Alto Networks will contribute our expertise as and when possible. We do not provide technical support or help in using or troubleshooting the components of the project through our normal support options such as Palo Alto Networks support teams, or ASC (Authorized Support Centers) partners and backline support options. The

underlying product used (the VM-Series firewall) by the scripts or templates are still supported, but the support is only for the product functionality and not for help in deploying or using the template or script itself.

Unless explicitly tagged, all projects or work posted in our GitHub repository (at https://github.com/PaloAltoNetworks/googlecloud) or sites other than our official Downloads page on https://support.paloaltonetworks.com are provided under the best effort policy.

Required Components

This example will use the following components:

- GCP Cloud Armor
- HTTP(S) Load Balancer
- Palo Alto Networks Firewall(s) with URLF and Threat subscriptions (duh)
- TCP Load Balancer
- Linux Webservers
- Linux Worker Node

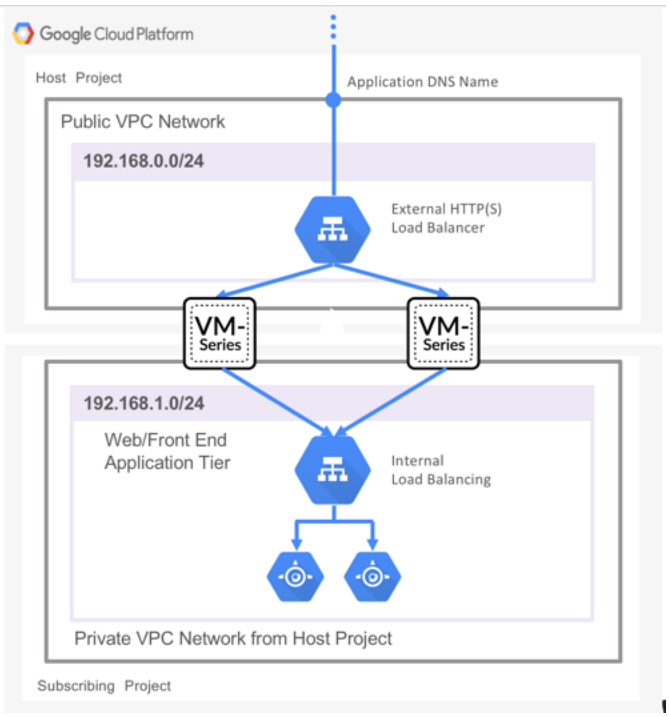
The following table lists the suggested instance sizes for this demonstration:

Instance name	Machine Type
Web Servers	f1-micro
Worker Node	f1-micro
VM Series Firewalls	n1-standard-4

Larger instances may be required/substituted as necessary.

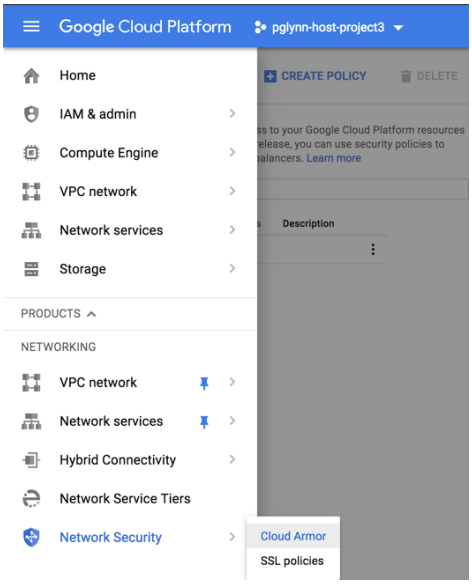
Note: There are costs associated with each machine type launched, please refer to the Google instance pricing page https://cloud.google.com/compute/pricing

It is assumed that the instances have already been deployed, configured, and verified to be passing traffic as expected. The topology used in this example resembles the following:

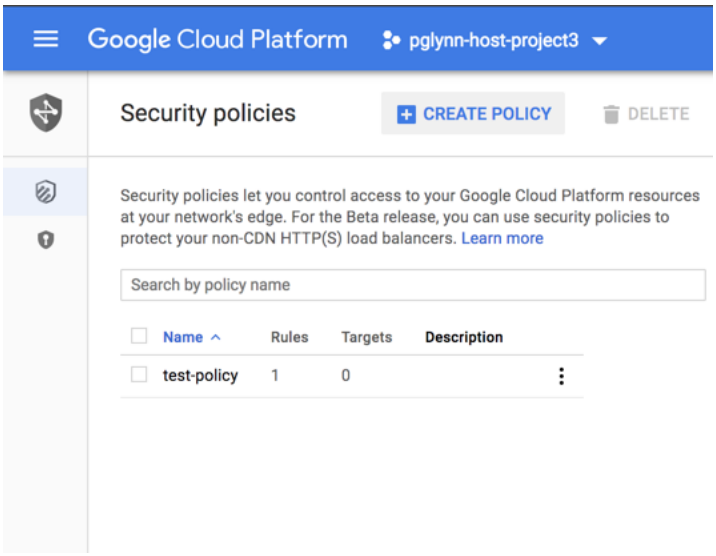


Configure Cloud Armor

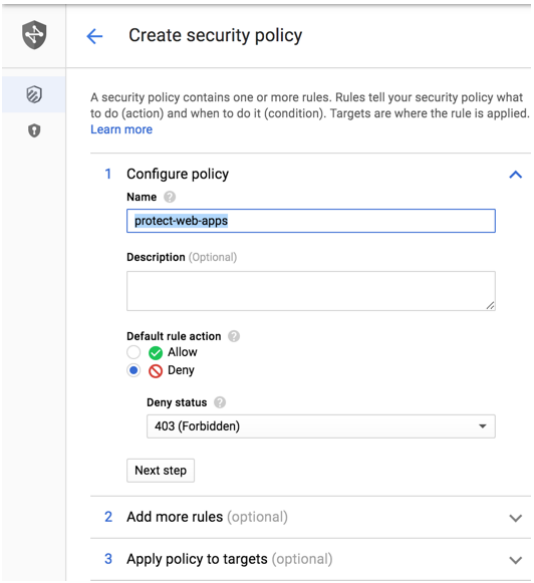
1. Create the initial Cloud Armor security policy. Navigate to **Networking > Network Security > Cloud Armor**.



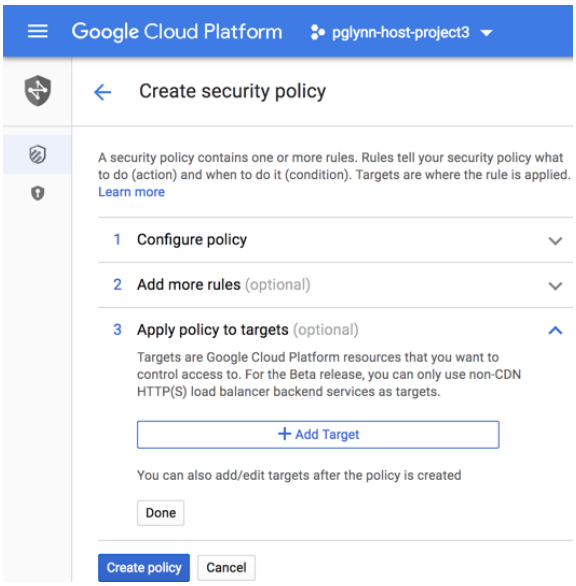
2. Click **CREATE POLICY.**



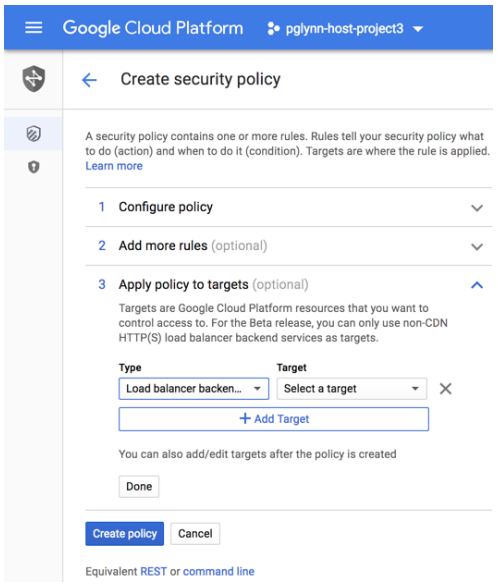
3. Specify a **Name.**



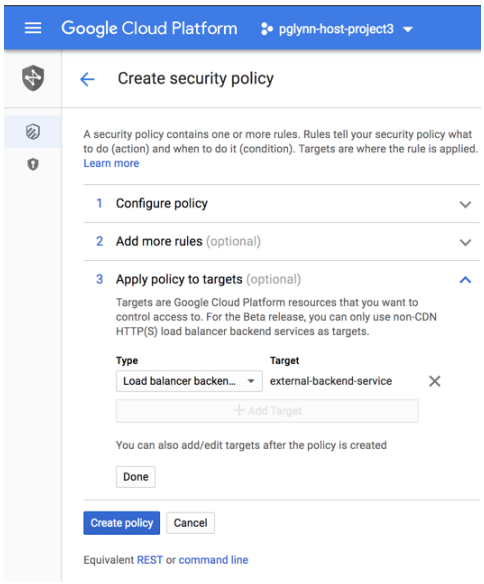
4. Click **Apply policy to targets.**



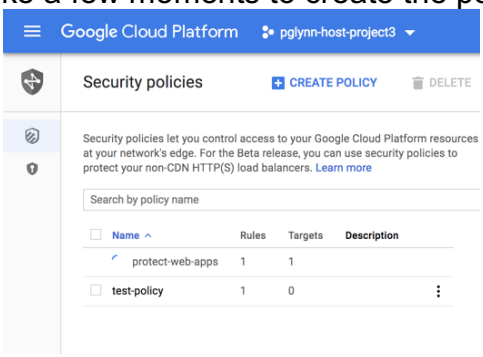
5. Click **+ Add Target**.



6. Select the public HTTP(S) Load Balancer.

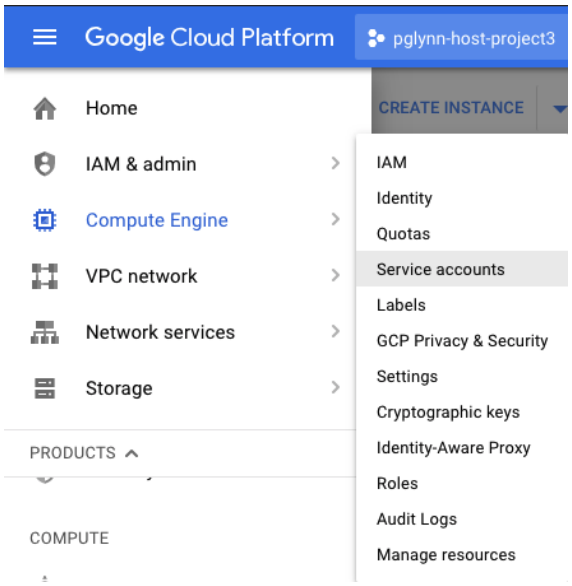


7. Click **Create policy.** It will take a few moments to create the policy.

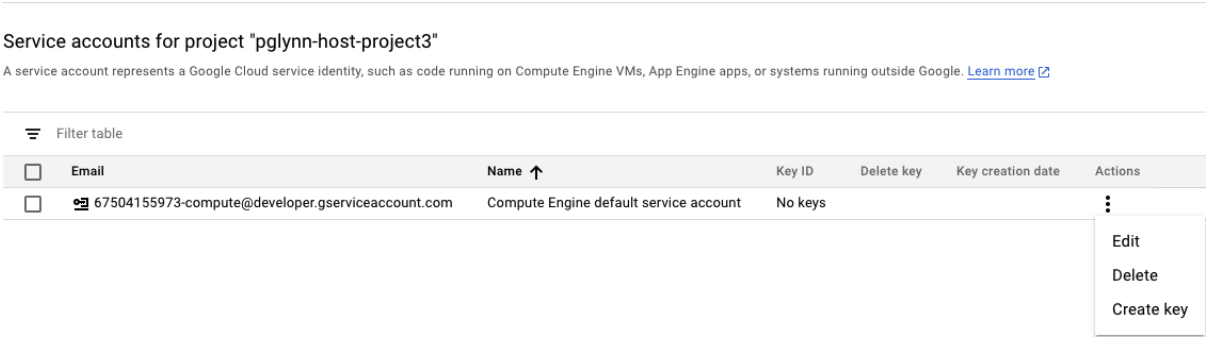


Create a Service Account Key

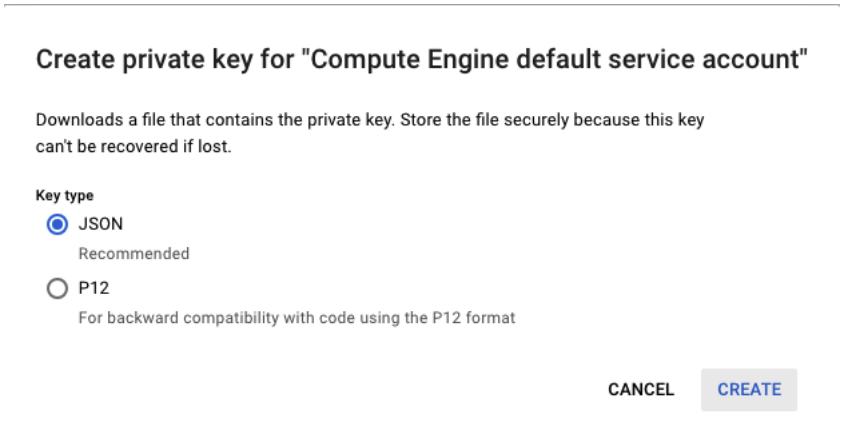
1. Navigate to **IAM & admin > Service accounts.**



2. Click on the vertical ellipses beside the default service account and select **Create key.**



3. Leave the key type as JSON and click **CREATE.**

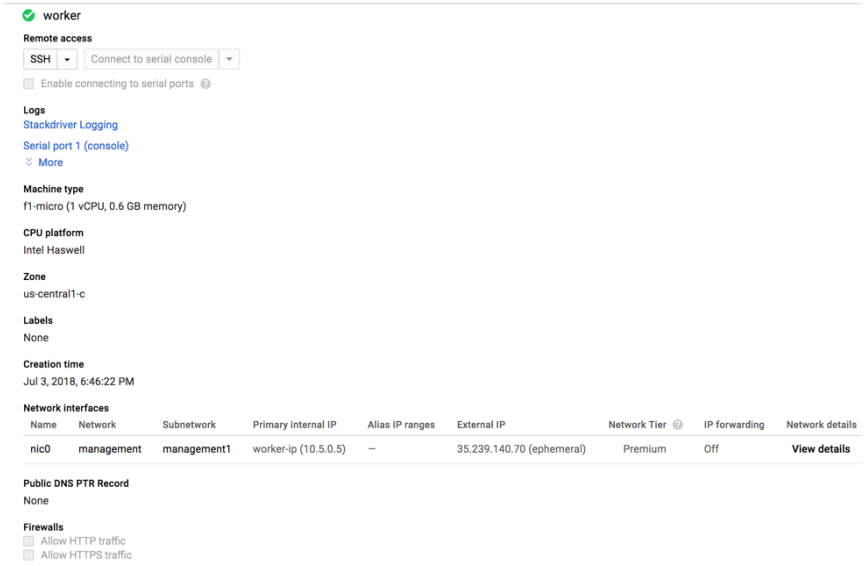


4. Save the key to a secure location as it allows API access to GCP resources.
5. (optional) Rename the key to reflect the service account to which it is attached.

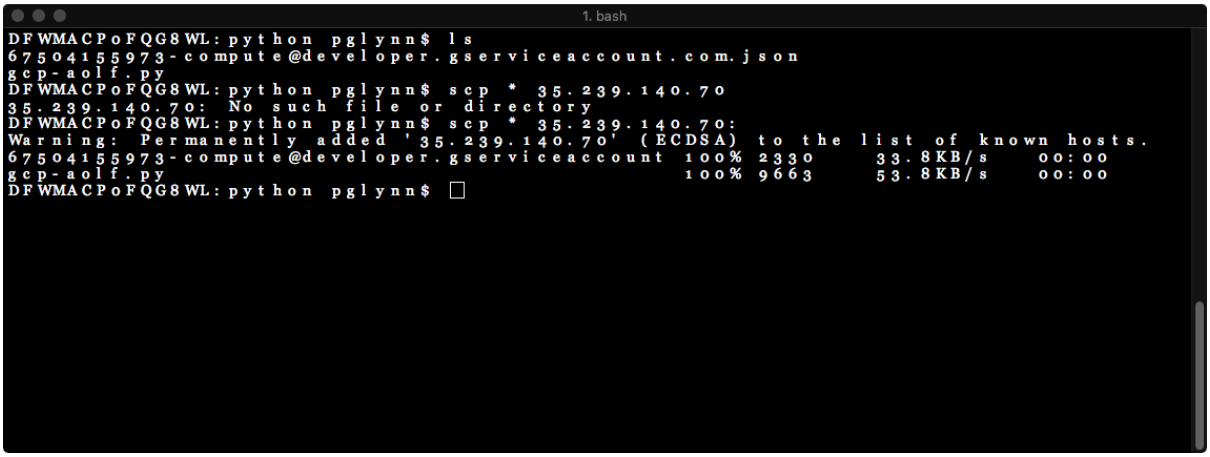
Prepare the Worker Node

1. Deploy a worker node with the following settings:

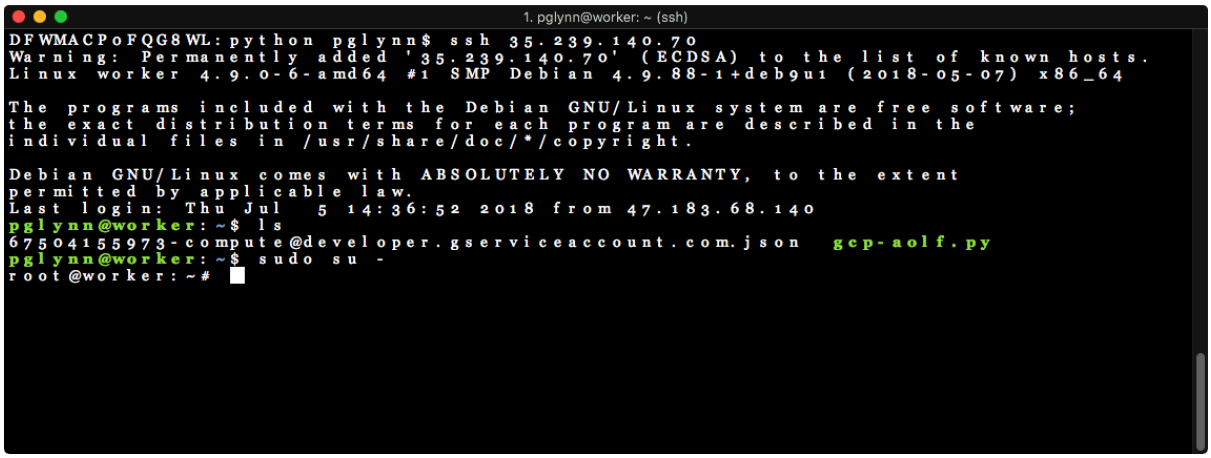
   Machine type: F1-micro
   Network: FW management subnet
   Internal IP: static
   External IP (optional): ephemeral



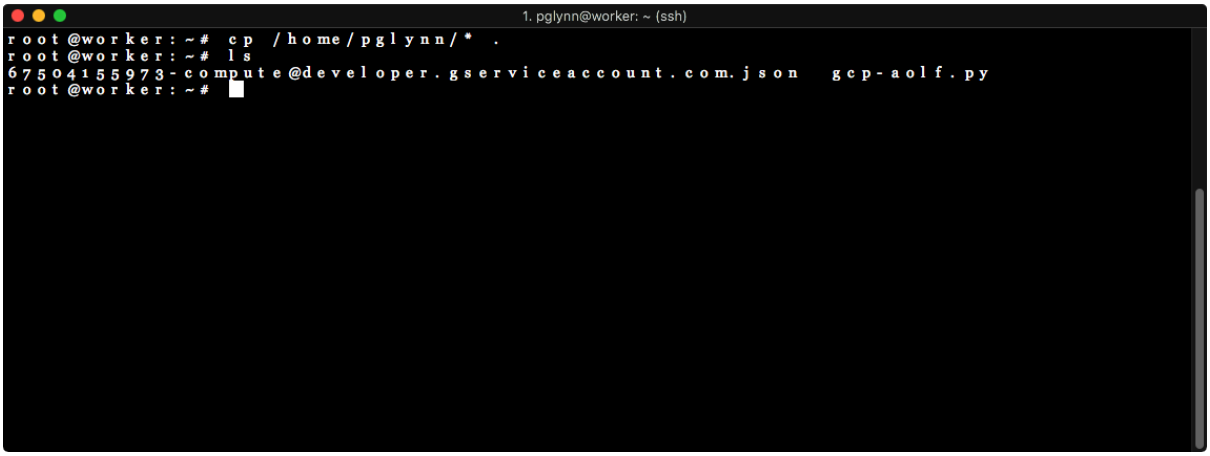
2. Copy the service account key and Python code to the worker node.



3. Connect to the worker node and “su” to root.

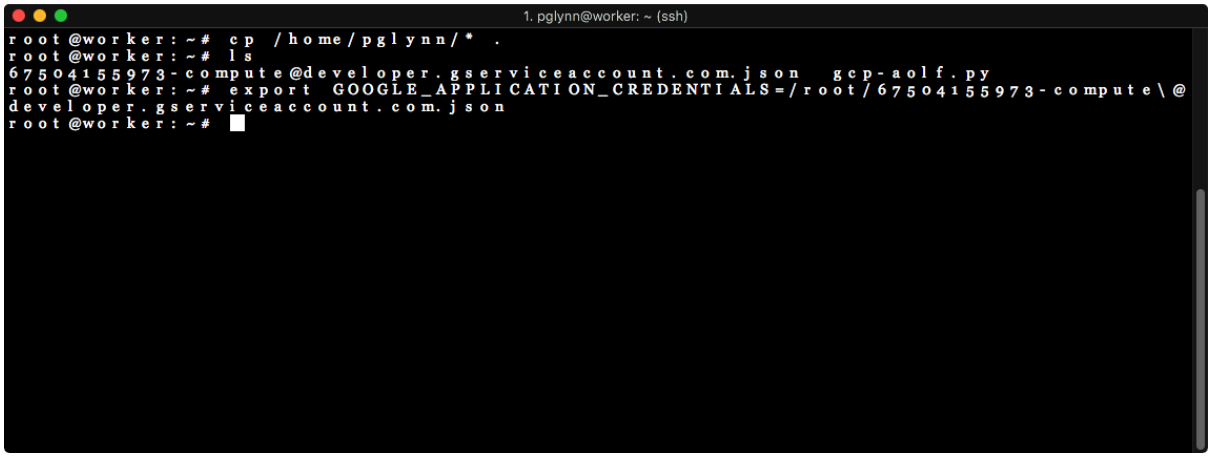


4. Copy the Python script and service account key to root's home directory

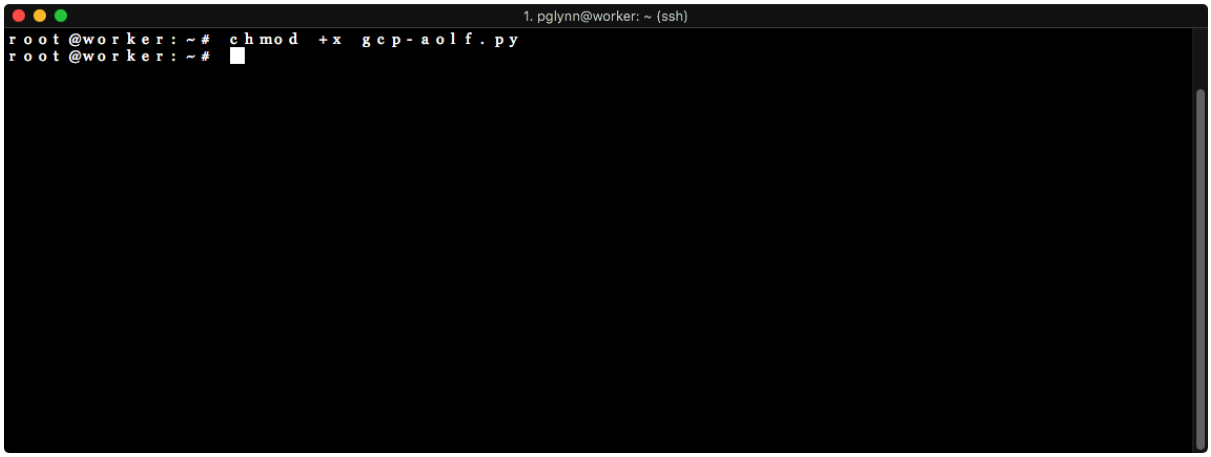


5. Create an environment variable pointing to the service account key. This is necessary as the Python script will need the key to authenticate to the GCP environment. The format is:

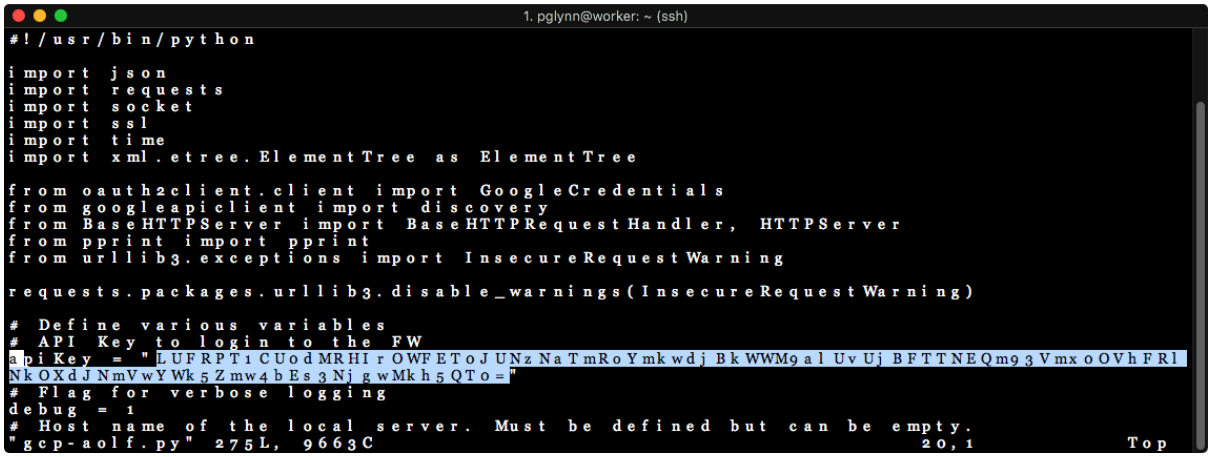
export GOOGLE_APPLICATION_CREDENTIALS=/path/to/service_account_key.json



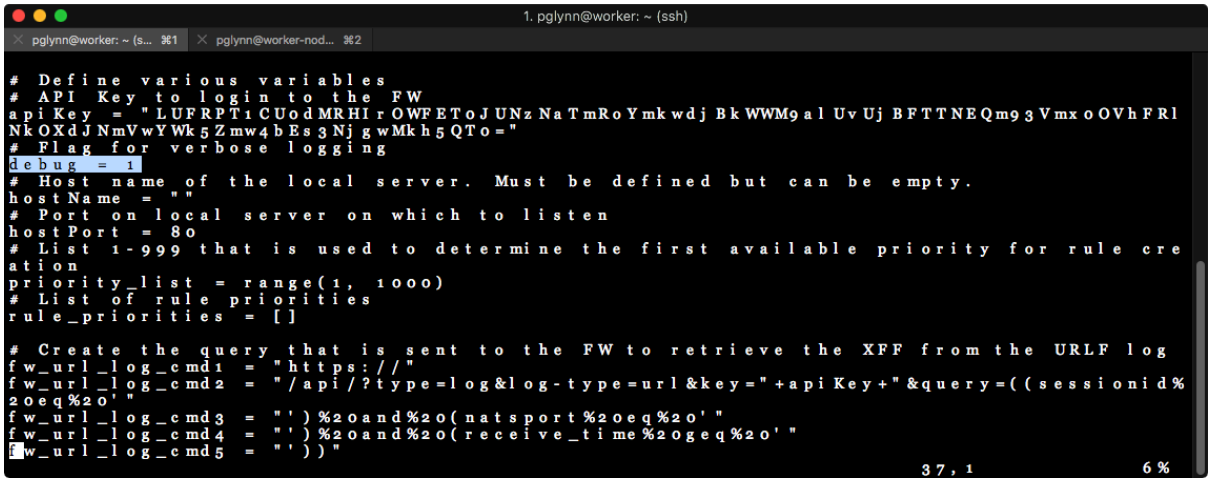
6. Add execute permission to the Python script.



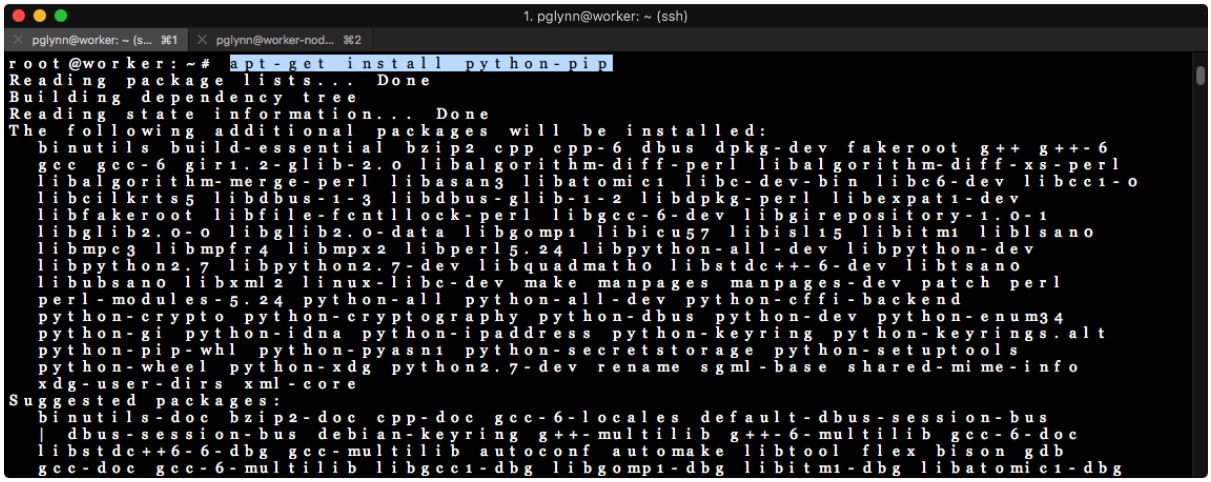
7. Edit the Python script and replace the FW API key with the one specific to your implementation.



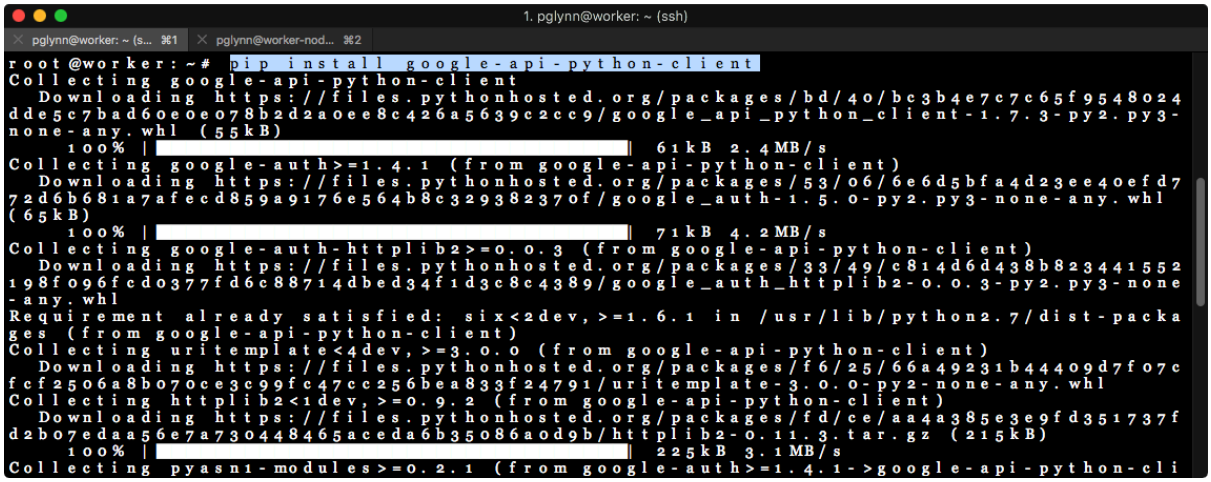
8. (optional) To monitor script execution or for debugging purposes, set the debug flag to “1”.



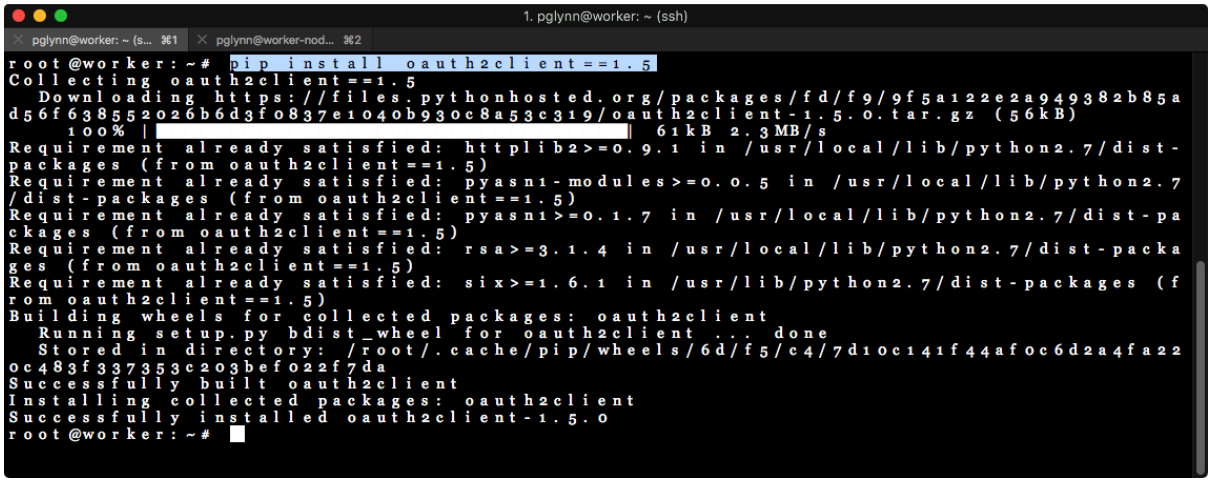
9. Install pip.



10. Install the Google API Client Library.

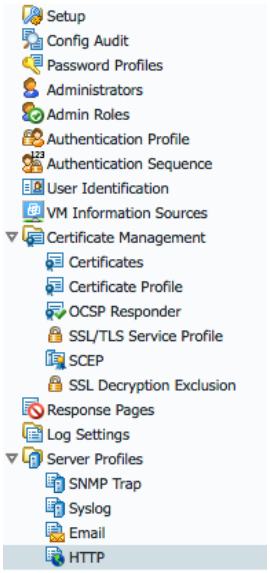


11. Install the OAuth client.



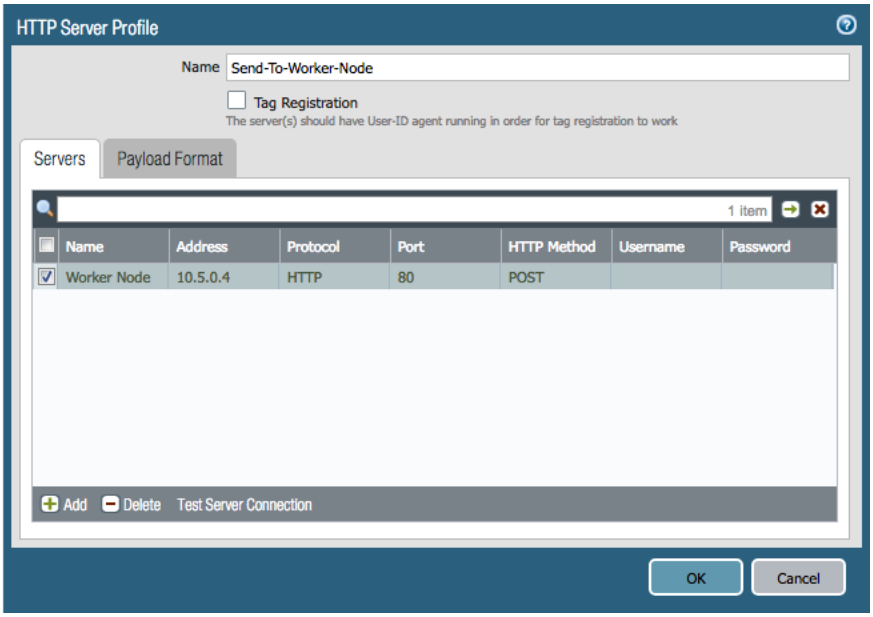
Configure the Firewalls

1. Login to the firewall and navigate to **Device > Server Profiles > HTTP.**



2. Add a new server profile with the following parameters:

   Name: free-form text (e.g. Worker Node)
   Address: Internal IP of the worker node
   Protocol: HTTP
   Port 80
   HTTP Method: POST

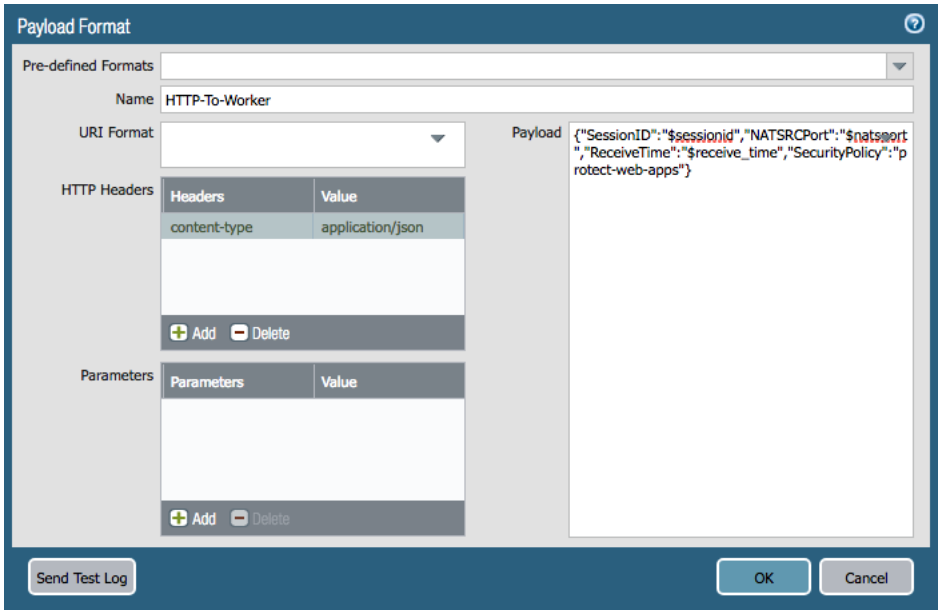


3. Under **Payload Format**, edit the log type for **Threat** and create a new payload format:

Name: Free-form text
Content-type: application/json
Payload:

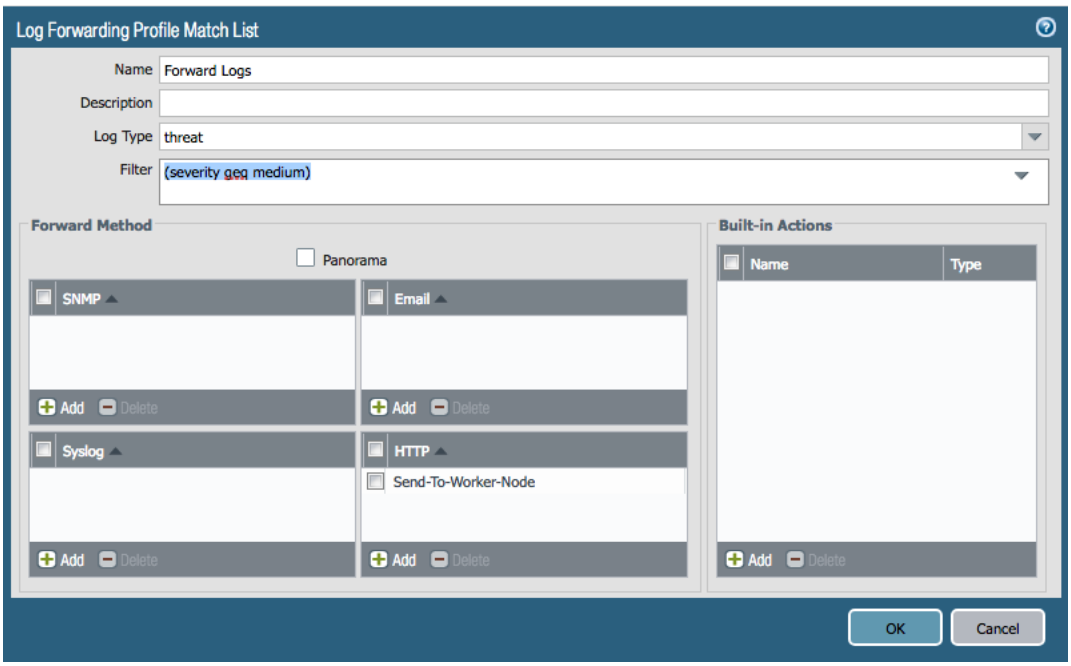
{"SessionID":"$sessionid","NATSRCPort":"$natsport","ReceiveTime":"$receive_time","SecurityPolicy":"protect-web-apps"}

(replace "protect-web-apps" with the name of the Cloud Armor security policy created earlier!)

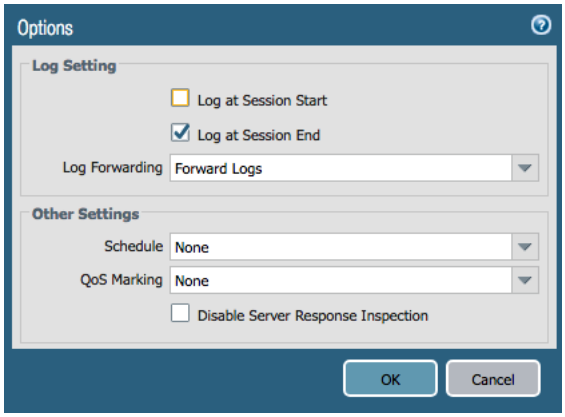


4. Navigate to **Objects > Log Forwarding** and add a new log forwarding profile with the following parameters:

   Name: Free-form text
   Log Type: Threat
   Filter: (severity geq medium)
   Forward Method: HTTP > (your HTTP Sever Profile)



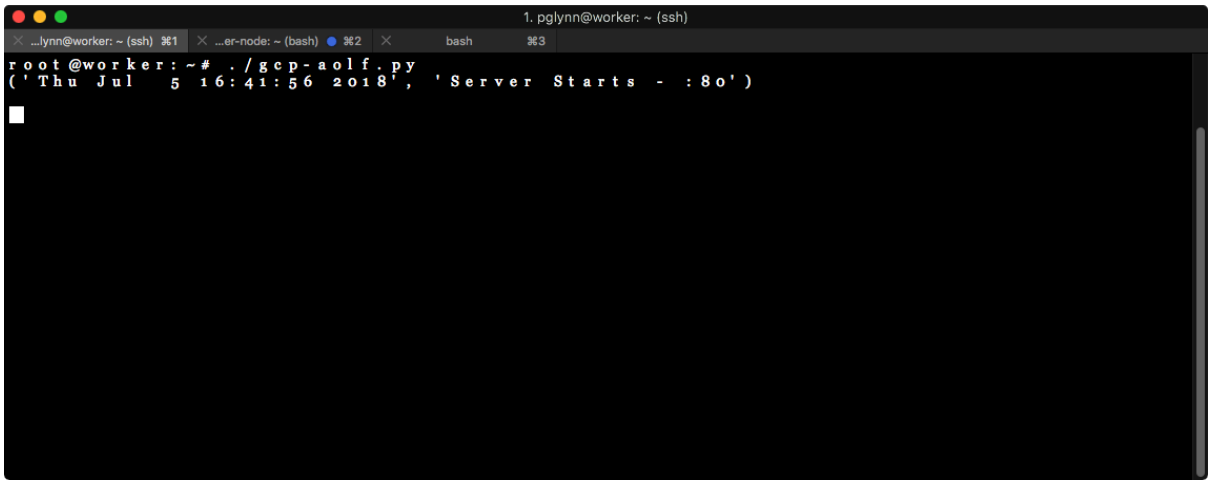
5. Edit the policy permitting web traffic from the untrust/internet side of the FW and add the log forwarding profile to the **Options.**



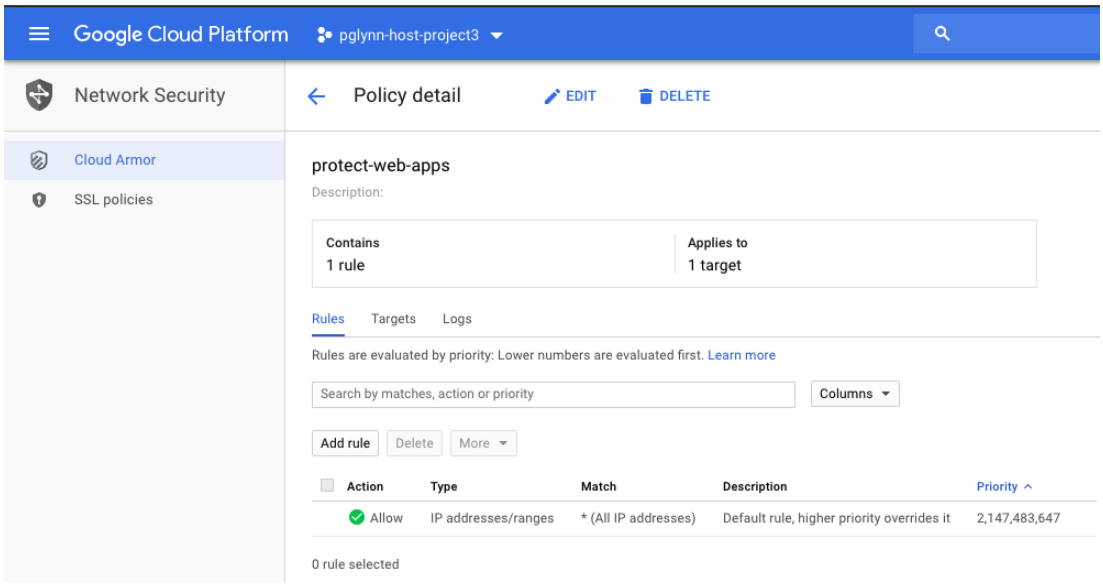
6. Commit the changes and replicate to the other FW.

Test/Verify

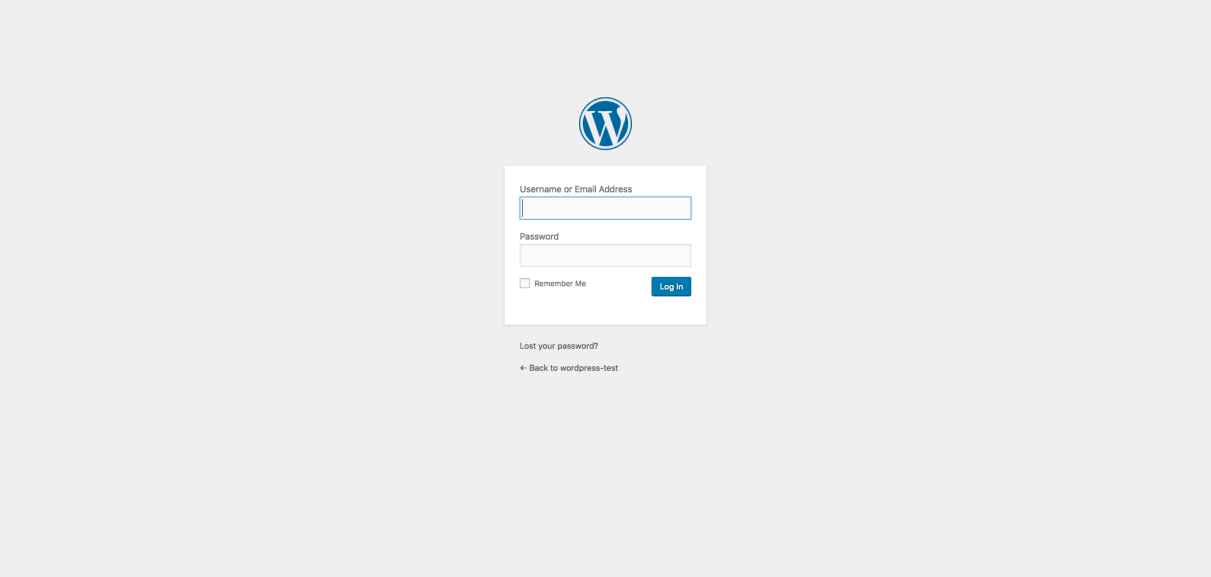
1. Launch the Python script from the worker node. You will have to be root if launching it manually as the script listens on a well-known port (TCP/80).



2. Check the Security Policy. In a production environment, there may be multiple rules blocking/permitting access.



3. Navigate to the web page (we are using a wordpress server for this example).

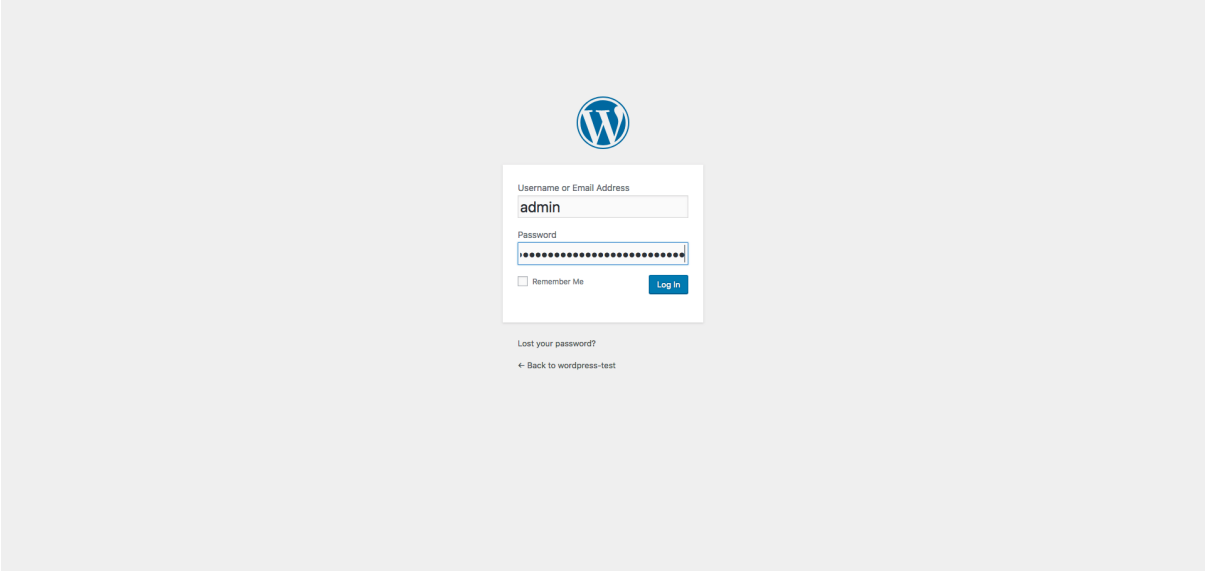


4. Input a valid username and for the password, simulate an XSS or SQL Injection attack. Examples include:

   <script>alert("HI")</script>

   %' or '0'='0

   1' or '1' = '1



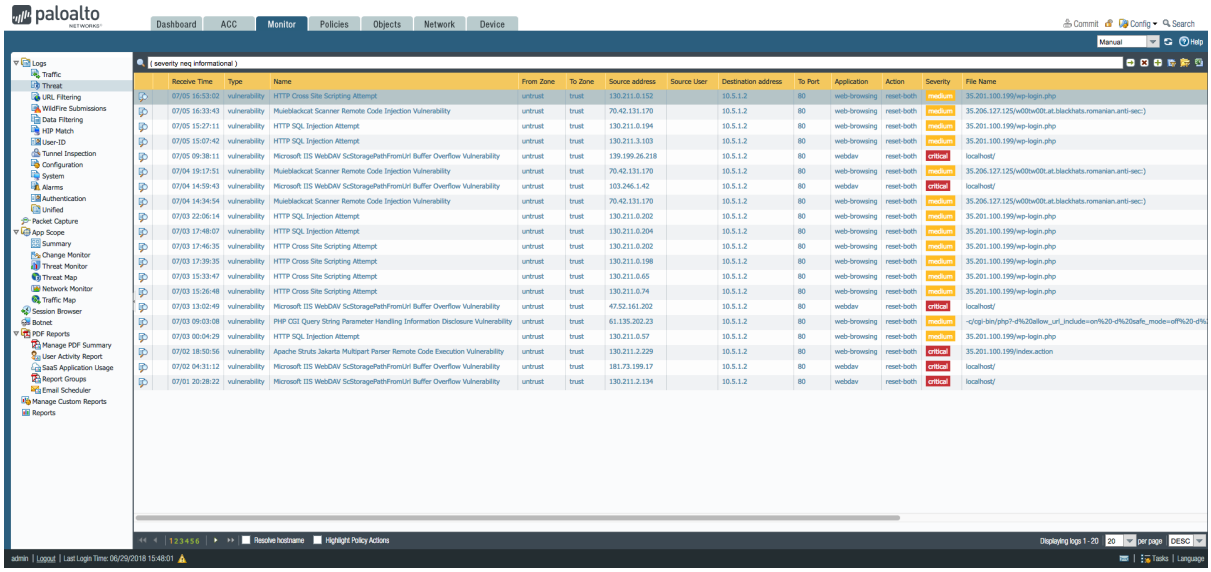
5. The firewall should block the attempted login.

Error: Server Error

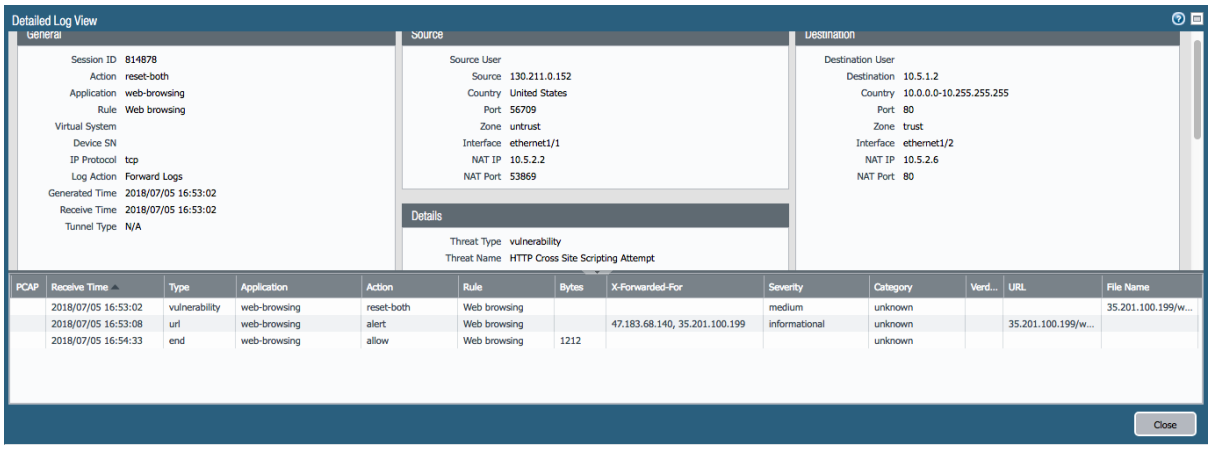
The server encountered a temporary error and could not complete your request.

Please try again in 30 seconds.

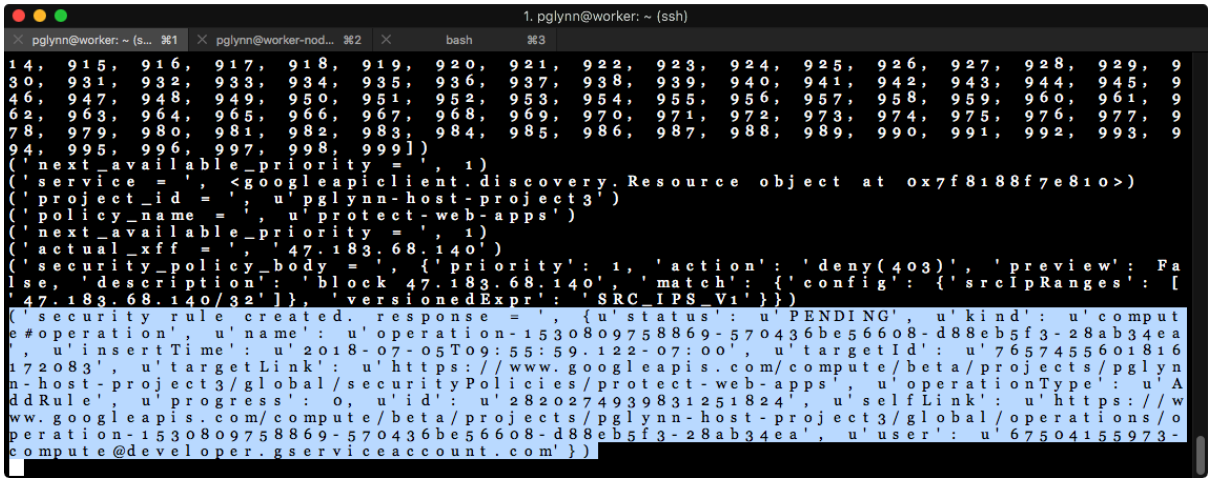
6. A Threat log event will be generated.



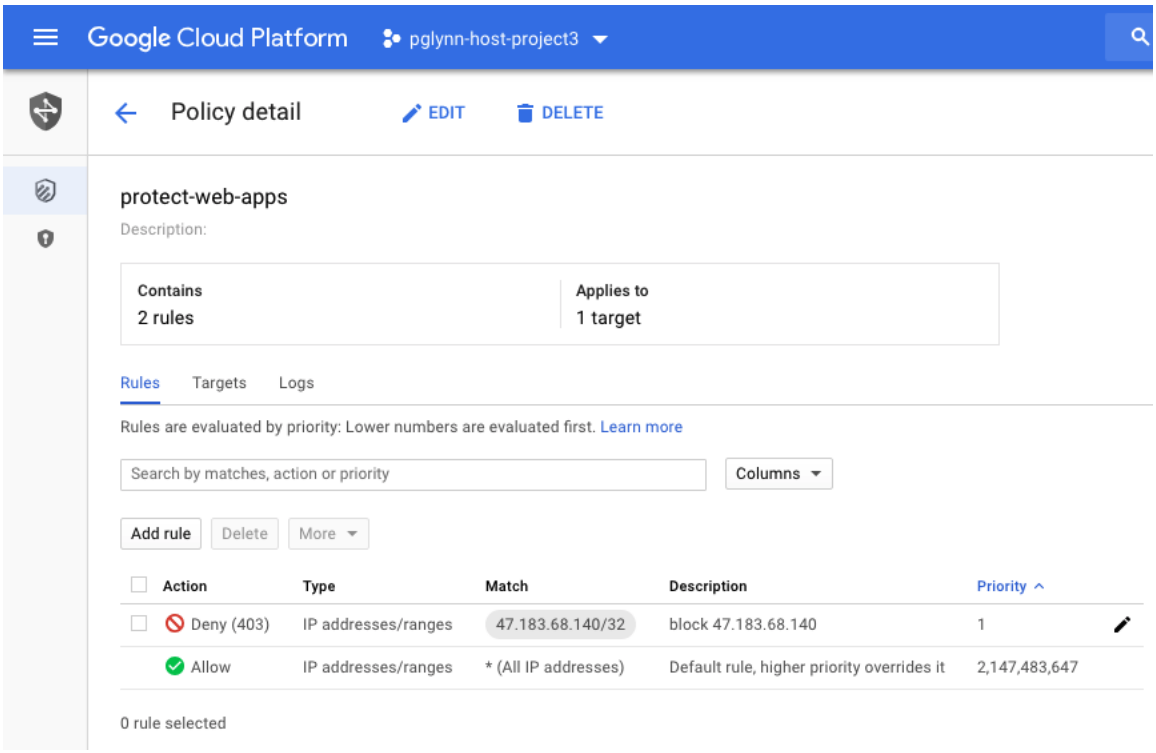
7. Details on the log entry will show the Traffic, URL, and Threat logs as well as the log forwarding action.



8. If debugging is enabled, you will see a large amount of output culminating in the acceptance of the request to create a new rule.



9. Check the security policy after a few moments (a browser refresh may be required).

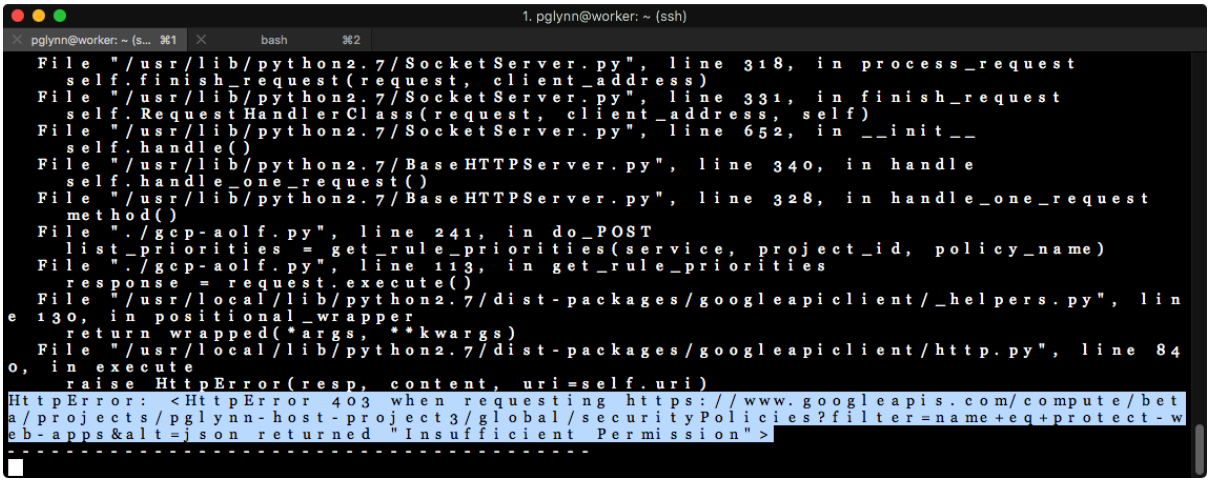


10. Verify by re-attempting the XSS/SQL Injection attempt from the browser. You should see a **403 Forbidden** error.
11. When done, interrupt the Python script with <CRTL>-<C>.

Troubleshooting

- For the script to be able to execute, it needs to load two python libraries: google-api-python3-client and oauth2client==1.5. If those two libraries are not installed prior to running the script, it will exit with an error.

- If the service account does not have the correct permissions or the authentication key has not been loaded, the script will run but fail when attempting to query the GCP environment.



For more details on the API calls, including required IAM permissions, see:

https://cloud.google.com/compute/docs/reference/rest/beta/securityPolicies/list

https://cloud.google.com/compute/docs/reference/rest/beta/securityPolicies/addRule