**Identify Application Vulnerabilities with Security Command Center**

experimentLabschedule1 hour 15 minutesuniversal\_currency\_alt1 Creditshow\_chartIntroductory

infoThis lab may incorporate AI tools to support your learning.

**GSP1262**



**Overview**

Web Security Scanner (WSS) is one of [Security Command Center's](https://cloud.google.com/security-command-center) built-in services that can be used to identify security vulnerabilities in App Engine, Google Kubernetes Engine (GKE), and Compute Engine web applications.

This service crawls your application, following all links within the scope of your starting URLs, and attempts to exercise as many user inputs and event handlers as possible. It can automatically scan and detect four common vulnerabilities, including cross-site-scripting (XSS), flash injection, mixed content (HTTP in HTTPS), and outdated/insecure libraries.

Web Security Scanner enables early identification of vulnerabilities and delivers very low false positive rates. You can easily set up, run, schedule, and manage security scans.

In this lab, you use Web Security Scanner to scan a Python Flask application for vulnerabilities.

Objectives

In this lab, you learn how to perform the following tasks:

* Launch a vulnerable Python Flask application on a Compute Engine instance.
* Use Web Security Scanner to scan the application and find vulnerabilities.
* Fix the application vulnerability.
* Scan the application again and verify vulnerabilities no longer exist.

**Setup and requirements**

For each lab, you get a new Google Cloud project and set of resources for a fixed time at no cost.

1. Sign in to Qwiklabs using an **incognito window**.
2. Note the lab's access time (for example, 1:15:00), and make sure you can finish within that time.  
   There is no pause feature. You can restart if needed, but you have to start at the beginning.
3. When ready, click **Start lab**.
4. Note your lab credentials (**Username** and **Password**). You will use them to sign in to the Google Cloud Console.
5. Click **Open Google Console**.
6. Click **Use another account** and copy/paste credentials for **this** lab into the prompts.  
   If you use other credentials, you'll receive errors or **incur charges**.
7. Accept the terms and skip the recovery resource page.

**Note:** Do not click **End Lab** unless you have finished the lab or want to restart it. This clears your work and removes the project.

**Scenario**



Cymbal Bank is an American retail bank with over 2,000 branches in all 50 states. It offers comprehensive debit and credit services that are built on top of a robust payments platform. Cymbal Bank is a digitally transforming legacy financial services institution.

Cymbal Bank was founded in 1920 under the name Troxler. Cymbal Group acquired the company in 1975 after it had been investing heavily in Cymbal Group's proprietary ATMs. As the bank grew into a national leader, they put strategic emphasis on modernizing the customer experience both in-person at their branches and digitally through an app they released in 2014. Cymbal Bank employs 42,000 people nationwide and, in 2019, reported $24 billion in revenue.

Cymbal Bank is interested in developing a new banking application for their corporate clients using Google Cloud technology. Application security is critical, and the CTO wants to see how Google Cloud can identify and mitigate application security vulnerabilities. As a Cloud Security Engineer, you are tasked with demonstrating Security Command Center's cutting-edge application vulnerability scanning features.

**Task 1. Launch a virtual machine and create a firewall rule for WSS**

In this task, you set up the infrastructure to demonstrate an application vulnerability to Cymbal Bank's CTO. More specifically, you deploy a virtual machine and open a firewall rule for Web Security Scanner to be able to access the vulnerable application that you intend to deploy.

1. On the Google Cloud console title bar, click **Activate Cloud Shell** (Activate Cloud Shell icon). If prompted, click **Continue**.
2. Create a static IP address that can be used for scanning a vulnerable web application by running the following command:

gcloud compute addresses create xss-test-ip-address --region=us-central1

Copied!

content\_copy

1. Run the following command to output the static IP address you just generated:

gcloud compute addresses describe xss-test-ip-address \

--region=us-central1 --format="value(address)"

Copied!

content\_copy

1. Copy the IP address (a single line of the output) and save it in a notepad.
2. Run the following command to create a VM instance to run the vulnerable application:

gcloud compute instances create xss-test-vm-instance \

--address=xss-test-ip-address --no-service-account \

--no-scopes --machine-type=e2-micro --zone=us-central1-a \

--metadata=startup-script='apt-get update; apt-get install -y python3-flask'

Copied!

content\_copy

The startup script installs python-flask, a Web Application Framework, which is used for running a simple Python application demonstrating cross-site scripting (XSS) vulnerability, which is a common web application security vulnerability.

1. Run the following command to open a firewall rule for Web Security Scanner to access a vulnerable application. Note the source ranges from which Web Security Scanner scans applications.

gcloud compute firewall-rules create enable-wss-scan \

--direction=INGRESS --priority=1000 \

--network=default --action=ALLOW \

--rules=tcp:8080 --source-ranges=0.0.0.0/0

Copied!

content\_copy

Click **Check my progress** to verify the objective.

Assessment Completed. VM Creation Successful!

Create the VM with desired configurations

Check my progress

*Assessment Completed. VM Creation Successful!*

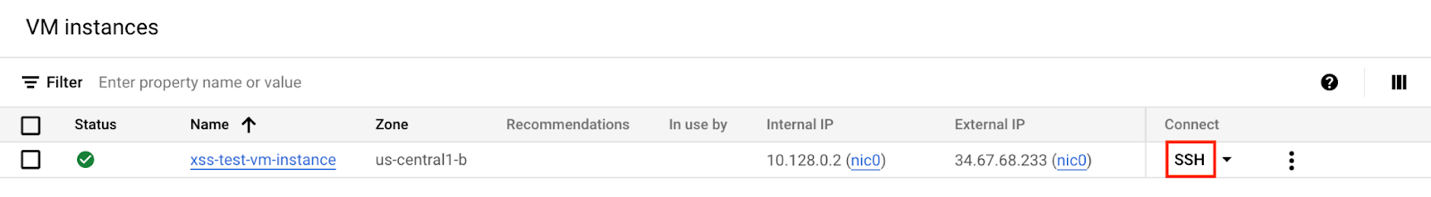
**Task 2. Deploy a vulnerable application to trigger an XSS vulnerability**

In this task, you obtain the application code and introduce a vulnerability for Web Security Scanner to detect. This is in the form of an application, which is a simple form that receives a user's input and outputs it without any changes.

1. In the Cloud console, on the **Navigation menu** (Navigation menu icon), click **Compute Engine** **>** **VM Instances**.

This may take a minute to initialize for the first time.

1. Then click on the **SSH** button next to your instance:



1. A pop-up may appear, asking you to allow SSH in-browser to connect to VMs. Click **Authorize**.

This opens an SSH connection to your VM instance in a new window.

1. In this SSH window (***Not in Cloud Shell***), run the following command to download and extract the vulnerable web application files:

gsutil cp gs://cloud-training/GCPSEC-ScannerAppEngine/flask\_code.tar . && tar xvf flask\_code.tar

Copied!

content\_copy

1. Now run the following command to deploy your application:

python3 app.py

Copied!

content\_copy

1. Soon after, you should receive a message that indicates your application is up and running.

**Output:**

\* Serving Flask app "app" (lazy loading)

\* Environment: production

WARNING: This is a development server. Do not use it in a production deployment.

Use a production WSGI server instead.

\* Debug mode: off

\* Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)

1. Find the static IP address of the VM you copied into your notepad earlier.
2. Replace YOUR\_EXTERNAL\_IP in the URL field below with that IP address, and open the URL in a new browser tab:

http://<YOUR\_EXTERNAL\_IP>:8080

Copied!

content\_copy

**Note:** You can also find the external IP address in the Google Cloud console, where it's listed as a field associated with your VM instance.

A Cymbal Bank corporate banking portal with a web form should appear.

1. In the web form, enter the following string:

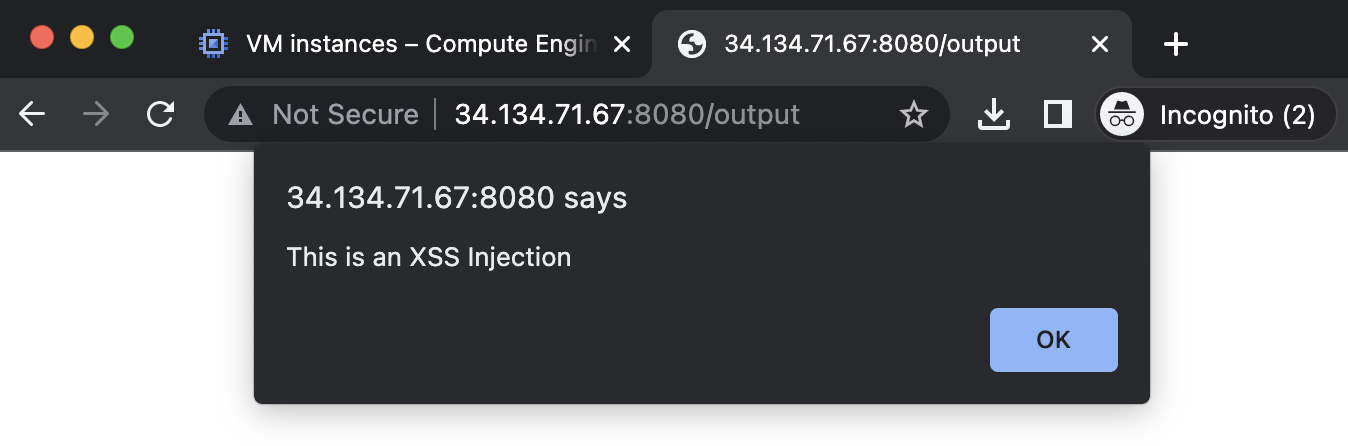
<script>alert('This is an XSS Injection')</script>

Copied!

content\_copy

1. Now click the **POST** button.

You should receive the following alert window.



This is a common vulnerability in web applications: a cross-site scripting vulnerability. Cross-site scripting (XSS) is a vulnerability that enables attackers to run malicious scripts in users' browsers in the context of your application. Your browser interprets a string as a legitimate Javascript and executes it.

An attacker who uses an XSS bug to inject JavaScript into an HTML page gains virtually unlimited access to the logged-in sessions of the victims who visit the page: they may steal user data, tamper with it, change privacy or security settings, or even completely alter the way the product looks and operates. Even more, an XSS vulnerability in one application, no matter how inconsequential, may jeopardize other content within the same domain.

This is one of many application vulnerabilities that Web Security Scanner can help you identify.

Click **Check my progress** to verify the objective.

Assessment Completed!

Download vulnerable web application files on the VM

Check my progress

*Assessment Completed!*

**Task 3. Enable the Web Security Scanner API**

Now that the vulnerable application is launched, it's time to demonstrate Web Security Scanner's abilities to the CTO. But first, you need to configure the API that WSS uses to run.

1. Switch back to the Cloud console browser tab.
2. From the **Navigation menu** (Navigation menu icon), select **APIs & Services** **>** **Library**.
3. In the Search for APIs and services field, type Web Security Scanner and press **Enter**.
4. Select the **Web Security Scanner API**.
5. Click **Enable** to enable the Web Security Scanner API.

Click **Check my progress** to verify the objective.

Step successfully completed. Web Security Scanner API is ready to be used.

Enable the Web Security Scanner API

Check my progress

*Step successfully completed. Web Security Scanner API is ready to be used.*

**Task 4. Scan the deployed application with WSS**

In this task, you configure and set up a scan of the application to check if it finds security vulnerabilities.

1. Open the **Navigation menu** (Navigation menu icon), and select **Security** **>** **Web Security Scanner**.
2. Click **+ New Scan**.
3. In the **Starting URLs** section, the **Starting URL 1** field should be pre-populated with your static IP address.
4. Add the port number **8080**, so that the Starting URL resembles the following:

http://<EXTERNAL\_IP>:8080

Copied!

content\_copy

1. If present, delete **Starting URL 2**.
2. Take a minute to review the remaining fields on the **Create a new scan** screen:

* **Authentication:** a property that can be used to provide application credentials to allow the scanner to authenticate to an app while scanning.
* **Schedule:** a property that can be used to schedule scans to run automatically.
* **Export to Security Command Center:** a property that allows you to automatically export scan configurations and scan results to Cloud Security Command Center after scans are finished.

1. Verify the **Authentication** is still set to **None** and that **Schedule** is set to **Never**.
2. Click **Show More** to investigate the remaining settings.
3. Click **Save** to create the scan.

**Note:** This creates the scan, but do not run it yet. It must currently be run manually since you did not create a schedule yet.

1. Click **Run** to start the scan

**Note:** Given the number of possible tests, this can take a little over **10 minutes** to scan.

1. Return to your SSH session in your separate browser window.

If the session timed out, run the following command to restart your application:

python3 app.py

Copied!

content\_copy

In your SSH Window, you should start to see logs generated similar to the example below—this is Web Security Scanner testing all possible URLs for potential vulnerabilities.

**Output:**

34.29.3.21 - - [23/Mar/2023 23:30:41] "GET /output HTTP/1.1" 200 -

35.184.129.44 - - [23/Mar/2023 23:31:06] "GET /output HTTP/1.1" 200 -

35.184.129.44 - - [23/Mar/2023 23:31:07] "GET /favicon.ico HTTP/1.1" 404 -

34.68.231.45 - - [23/Mar/2023 23:31:09] "POST / HTTP/1.1" 302 -

34.68.231.45 - - [23/Mar/2023 23:31:09] "GET /output HTTP/1.1" 200 -

34.68.231.45 - - [23/Mar/2023 23:31:09] "GET /favicon.ico HTTP/1.1" 404 -

35.184.129.44 - - [23/Mar/2023 23:31:17] "POST / HTTP/1.1" 302 -

35.184.129.44 - - [23/Mar/2023 23:31:17] "GET /output HTTP/1.1" 200 -

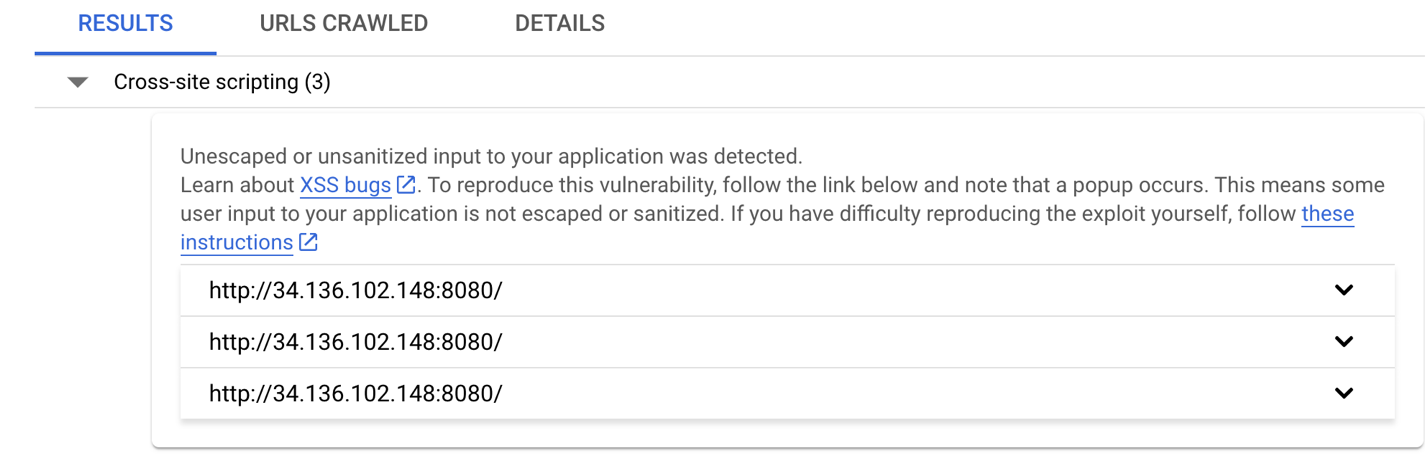
You may see log statements with the following HTTP status codes:

* **200:** A successful request, where the HTTP server answered with an OK response.
* **302:** Indicates a resource is temporarily located elsewhere according to the Location header.
* **404:** indicates that one or more resources were not found.

Check out the [HTTP Status and Error Codes](https://cloud.google.com/storage/docs/json_api/v1/status-codes#standardcodes) documentation for more information.

While the scan is running, feel free to explore the **Results**, **URLs Crawled**, and **Details** tabs. You can also check out this [getting started video](https://www.youtube.com/watch?v=1BengAd2_cI), or this [vulnerability scanning video](https://www.youtube.com/watch?v=ai5Hr5zkn50) to learn more about Web Security Scanner.

1. When the scan is done running, the **Results** tab should indicate the cross-site vulnerabilities.



The Web Security Scanner was able to scan all starting URLs and detect the XSS vulnerabilities in Cymbal Bank's application. The ability to automate the detection of these critical vulnerabilities is a major benefit for security-minded organizations like Cymbal Bank.

Click **Check my progress** to verify the objective.

Step successfully completed. The Web App vulnerabilties are existing after the first scan.

Run a Web Security Scanner scan and detect application vulnerabilities

Check my progress

*Step successfully completed. The Web App vulnerabilties are existing after the first scan.*

**Task 5. Correct the vulnerability and scan again**

Now that you have demonstrated Web Security Scanner can detect a XSS vulnerability, you remediate the vulnerability and run the application scan again.

1. Return to your SSH window that's connected to your VM instance.
2. Stop the running application by pressing **CTRL + C**.
3. Edit the **app.py** file using the nano editor by running the following command:

nano app.py

Copied!

content\_copy

1. Locate the two lines that set the output string:

# output\_string = "".join([html\_escape\_table.get(c, c) for c in input\_string])

output\_string = input\_string

Copied!

content\_copy

1. Remove the # symbol from the first line and add it to the beginning of the next line (*ensure that you indent your code properly!*)

Your final lines must resemble the following:

@app.route('/output')

def output():

output\_string = "".join([html\_escape\_table.get(c, c) for c in input\_string])

# output\_string = input\_string

return flask.render\_template("output.html", output=output\_string)

Copied!

content\_copy

**Note:** html\_escape\_table is a dictionary that contains one-to-one pairings of special HTML characters like "<" to their text representation. You use this table to escape special HTML characters so your form ingests and interprets submissions as raw text only. You can refer to this [What is HTML Escape?](https://www.lambdatest.com/free-online-tools/html-escape#:~:text=What%20is%20HTML%20Escape%3F,HTML%20entities%20to%20plain%20text) documentation for more information.

1. Now type **CTRL+X**, **Y**, and **Enter** to save your changes.
2. Re-run the application:

python3 app.py

Copied!

content\_copy

1. Return to the Google Cloud console browser tab (you should still have the Web Security Scanner page open):
2. Click **Run** at the top of the page.

In your SSH Window, you should start to see logs where Web Security Scanner tests application URLs for potential vulnerabilities.

**Output:**

34.29.3.21 - - [23/Mar/2023 23:30:41] "GET /output HTTP/1.1" 200 -

35.184.129.44 - - [23/Mar/2023 23:31:06] "GET /output HTTP/1.1" 200 -

35.184.129.44 - - [23/Mar/2023 23:31:07] "GET /favicon.ico HTTP/1.1" 404 -

34.68.231.45 - - [23/Mar/2023 23:31:09] "POST / HTTP/1.1" 302 -

34.68.231.45 - - [23/Mar/2023 23:31:09] "GET /output HTTP/1.1" 200 -

34.68.231.45 - - [23/Mar/2023 23:31:09] "GET /favicon.ico HTTP/1.1" 404 -

35.184.129.44 - - [23/Mar/2023 23:31:17] "POST / HTTP/1.1" 302 -

35.184.129.44 - - [23/Mar/2023 23:31:17] "GET /output HTTP/1.1" 200 -

1. While you are waiting for the results of the scan, login to the URL http://<EXTERNAL\_IP>:8080 using your browser in a separate tab.

The web form displays once again.

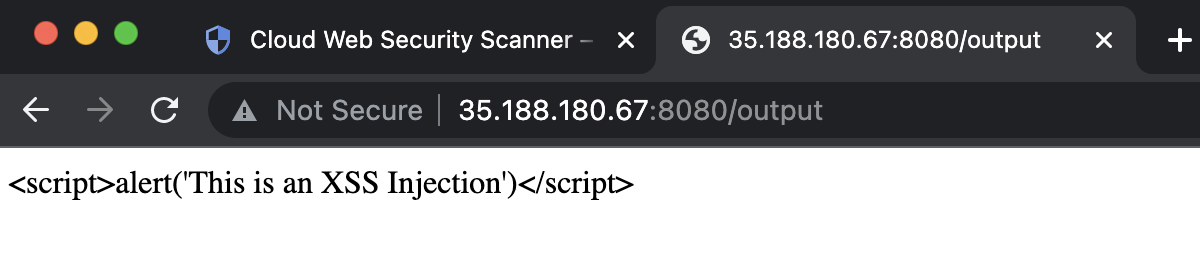
1. In the web form, enter the same string that you entered in before:

<script>alert('This is an XSS Injection')</script>

Copied!

content\_copy

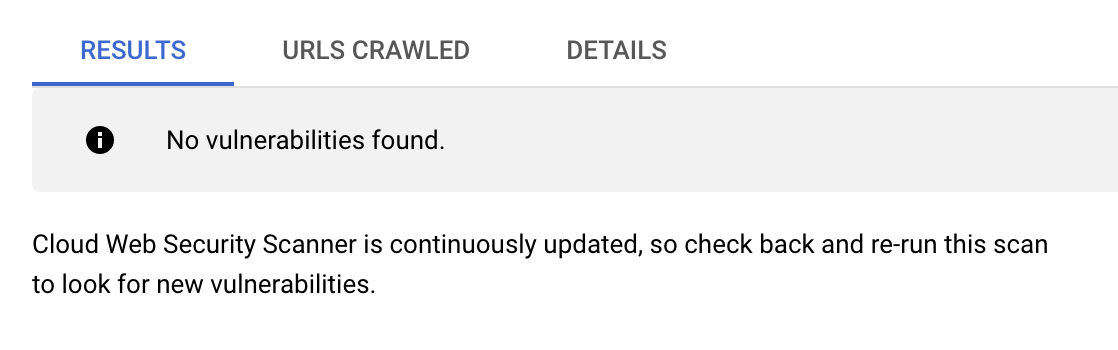
1. Now click the **POST** button.
2. Verify that this time you receive the following string in the browser:

**Note:** Although this technique works in this simple scenario, for proper protection of your web application you need to use more advanced techniques and frameworks which are out of scope of this lab.

Explore the links below for more resources:

* [Angular Security](https://angular.io/guide/security)
* [Google XSS Game](https://xss-game.appspot.com/)
* [OWASP Juice Shop](https://owasp.org/www-project-juice-shop/)

1. Return to the Google Cloud console, where you left off on the Web Security Scanner page.
2. Click **Run** at the top of the page to re-scan your application.
3. Soon after, you should notice that the results yield no more XSS vulnerabilities.



Click **Check my progress** to verify the objective.

Step successfully completed. The Web App vulnerabilties are fixed.

Correct vulnerabilities and rescan your application using Web Security Scanner

Check my progress

*Step successfully completed. The Web App vulnerabilties are fixed.*

**Congratulations!**

You have successfully demonstrated to the Cymbal Bank CTO how to identify and remediate XSS vulnerabilities with Google Cloud's powerful Web Security Scanner solution.

**End your lab**

When you have completed your lab, click **End Lab**. Google Cloud Skills Boost removes the resources you’ve used and cleans the account for you.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

* 1 star = Very dissatisfied
* 2 stars = Dissatisfied
* 3 stars = Neutral
* 4 stars = Satisfied
* 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.

Google Cloud training and certification

...helps you make the most of Google Cloud technologies. [Our classes](https://cloud.google.com/training) include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. [Certifications](https://cloud.google.com/certification/) help you validate and prove your skill and expertise in Google Cloud technologies.

**Manual Last Updated November 28, 2024**

**Lab Last Tested November 28, 2024**

Copyright 2022 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.