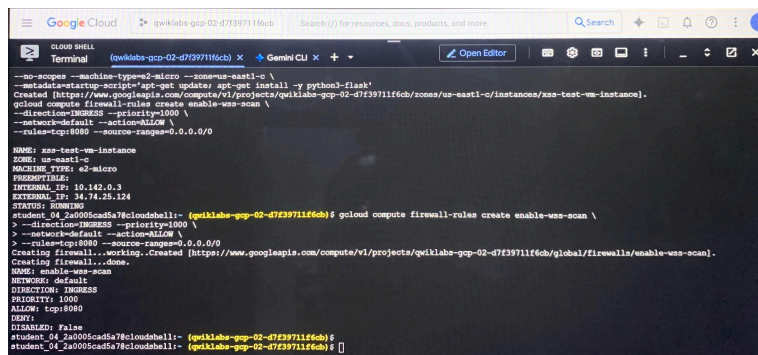


IDENTIFY VULNERABILITIES AND REMEDIATION TECHNIQUES

Cymbal Bank has developed a new banking application for its corporate clients that is set to be hosted and deployed on the new cloud infrastructure. The Chief Information Security Officer (CISO), Javier, wants to prioritize the security of this application before it is launched and customer-facing. My team lead, Chloe, has tasked me with identifying and mitigating any application vulnerabilities for this new application. I'll use the Web Security Scanner in Google Cloud to scan the application for vulnerabilities pertaining to a top OWASP® web application vulnerability known as Cross-Site Scripting (XSS).

Here's how I did this task: **First**, I created a static IP address and launched a virtual machine. **Then**, I deployed the vulnerable application. **Next**, I set up and ran the application. **Then**, I accessed and scanned the application. **Finally**, I fixed the vulnerabilities and re-scanned the application.

I activated the cloud shell terminal and typed in the following command;

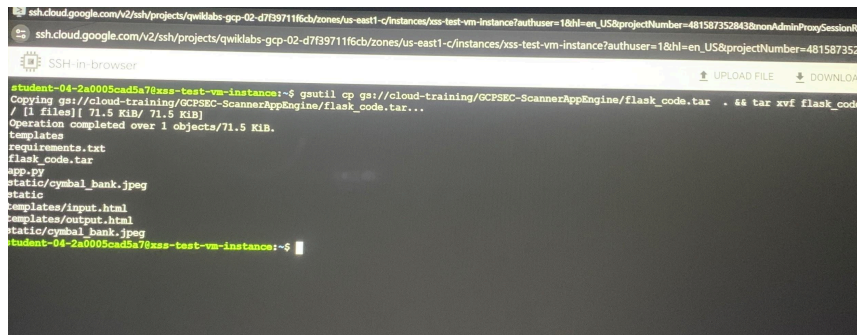


```
Google Cloud | qwiklabs-gcp-02-d7f39711f6cb | Search (/) for resources, docs, products, and more | Search | + | - | x
CLOUD SHELL | Terminal | (quiklabs-gcp-02-d7f39711f6cb) x | Gemini CLI x | + | - | x | Open Editor | + | - | x
--no-scopes --machine-type=e2-micro --zone=us-east1-c \
--metadata=setup-script='apt-get update; apt-get install -y python3-flask'
Created [https://www.googleapis.com/compute/v1/projects/quiklabs-gcp-02-d7f39711f6cb/zones/us-east1-c/instances/xss-test-wm-instance].
gcloud compute firewall-rules create enable-wss-scan \
--direction=INBOUND --priority=1000 \
--network=default --action=ALLOW \
--rules=tcp:8080 --source-ranges=0.0.0.0/0
NAME: xss-test-wm-instance
ZONE: us-east1-c
MACHINE_TYPE: e2-micro
PREEMPTIBLE:
INTERNAL_IP: 10.142.0.3
EXTERNAL_IP: 34.74.23.124
STATUS: RUNNING
student_04_2a005cad5a78@cloudshell: (quiklabs-gcp-02-d7f39711f6cb) $ gcloud compute firewall-rules create enable-wss-scan \
> --direction=INBOUND --priority=1000 \
> --network=default --action=ALLOW \
> --rules=tcp:8080 --source-ranges=0.0.0.0/0
Creating firewall...working...Created [https://www.googleapis.com/compute/v1/projects/quiklabs-gcp-02-d7f39711f6cb/global/firewalls/enable-wss-scan].
Creating firewall...done.
NAME: enable-wss-scan
NETWORK: default
DIRECTION: INBOUND
PRIORITY: 1000
ALLOW: tcp:8080
DENY:
DISABLED: false
student_04_2a005cad5a78@cloudshell: (quiklabs-gcp-02-d7f39711f6cb) $
student_04_2a005cad5a78@cloudshell: (quiklabs-gcp-02-d7f39711f6cb) $
```

This command creates ;

1. A static IP address named xss-test-ip-address in the **REGION** region. This static IP will be used for scanning the vulnerable web application.
2. This command returns the static IP address generated.
3. This command creates a VM instance to run the vulnerable application.

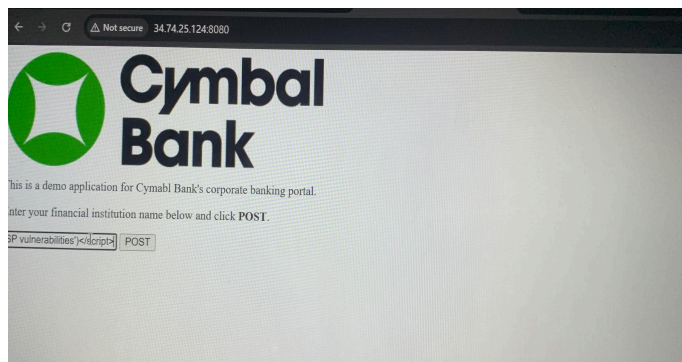
I typed the following in the SSH-in-browser;



```
student-04-2a005cad5a78@ssh-test-vm-instance:~$ gsutil cp gs://cloud-training/GCPSEC-ScannerAppEngine/flask_code.tar .
Copying gs://cloud-training/GCPSEC-ScannerAppEngine/flask_code.tar...
/ [1 files] [ 71.5 KiB / 71.5 KiB]
Operation completed over 1 objects/71.5 KiB.
flask_code.tar
requirements.txt
static/cymbal_bank.jpeg
static
templates/input.html
templates/output.html
static/cymbal_bank.jpeg
student-04-2a005cad5a78@ssh-test-vm-instance:~$
```

This command downloads and extracts the vulnerable web application files.

I tested my application for a vulnerability known as cross-site scripting (XSS). The injected code displayed a message back to the browser. This action by itself is not malicious, however attackers can introduce malicious code into an exploitable application to either steal data from it or implant malware onto the user's device. The alert window opens with the following message: “This is an XSS Injection to demonstrate one of OWASP vulnerabilities”.



Upon completion of the scan in the **SSH-in-browser** window, I viewed logs being generated as Web Security Scanner tests all possible URLs for potential vulnerabilities.

```
SSH-in-browser
192.178.11.99 - [28/Sep/2025 20:57:58] "GET / HTTP/1.1" 200 -
192.178.11.99 - [28/Sep/2025 20:57:58] "GET /.git/config HTTP/1.1" 404 -
192.178.11.99 - [28/Sep/2025 20:57:58] "GET /.svn/wc.db HTTP/1.1" 404 -
192.178.11.99 - [28/Sep/2025 20:57:59] "POST / HTTP/1.1" 302 -
192.178.11.100 - [28/Sep/2025 20:57:59] "POST /.git/config HTTP/1.1" 404 -
192.178.11.100 - [28/Sep/2025 20:57:59] "POST /.svn/wc.db HTTP/1.1" 404 -
192.178.11.99 - [28/Sep/2025 20:57:59] "GET /output HTTP/1.1" 200 -
192.178.11.100 - [28/Sep/2025 20:57:59] "GET /.git/config HTTP/1.1" 404 -
192.178.11.100 - [28/Sep/2025 20:57:59] "GET /output/.git/config HTTP/1.1" 404 -
192.178.11.100 - [28/Sep/2025 20:57:59] "GET /.svn/wc.db HTTP/1.1" 404 -
192.178.11.99 - [28/Sep/2025 20:57:59] "GET /output/.svn/wc.db HTTP/1.1" 404 -
4.125.215.194 - [28/Sep/2025 20:58:02] "GET / HTTP/1.1" 200 -
74.125.215.196 - [28/Sep/2025 20:58:03] "GET /static/cymbal.bank.jpeg HTTP/1.1" 200 -
74.125.215.196 - [28/Sep/2025 20:58:03] "GET / HTTP/1.1" 200 -
74.125.215.195 - [28/Sep/2025 20:58:04] "GET /static/cymbal.bank.jpeg HTTP/1.1" 200 -
74.125.215.196 - [28/Sep/2025 20:58:04] "GET / HTTP/1.1" 200 -
74.125.215.196 - [28/Sep/2025 20:58:04] "GET /static/cymbal.bank.jpeg HTTP/1.1" 200 -
74.125.215.195 - [28/Sep/2025 20:58:05] "GET / HTTP/1.1" 200 -
74.125.215.194 - [28/Sep/2025 20:58:06] "GET /static/cymbal.bank.jpeg HTTP/1.1" 200 -
4.125.215.196 - [28/Sep/2025 20:58:11] "POST / HTTP/1.1" 302 -
4.125.215.196 - [28/Sep/2025 20:58:11] "GET /output HTTP/1.1" 200 -
4.125.215.196 - [28/Sep/2025 20:58:12] "POST / HTTP/1.1" 302 -
4.125.215.196 - [28/Sep/2025 20:58:12] "GET /output HTTP/1.1" 200 -
4.125.215.195 - [28/Sep/2025 20:58:15] "POST / HTTP/1.1" 302 -
4.125.215.195 - [28/Sep/2025 20:58:15] "GET /output HTTP/1.1" 200 -
4.125.215.195 - [28/Sep/2025 20:58:15] "POST / HTTP/1.1" 302 -
4.125.215.194 - [28/Sep/2025 20:58:16] "POST / HTTP/1.1" 302 -
4.125.215.194 - [28/Sep/2025 20:58:16] "GET /output HTTP/1.1" 200 -
4.125.215.196 - [28/Sep/2025 20:58:17] "GET /output HTTP/1.1" 200 -
4.125.215.196 - [28/Sep/2025 20:58:18] "GET /output HTTP/1.1" 200 -
4.125.215.195 - [28/Sep/2025 20:58:19] "GET /output HTTP/1.1" 200 -
4.125.215.194 - [28/Sep/2025 20:58:19] "GET /output HTTP/1.1" 200 -
```

To fix the XSS vulnerability, I validated the output string variable in the nano code editor. The output string is the processed output of the user-supplied web form input.

```
SSH-in-browser
GNU nano 7.2 app.py
# limitations under the license.
import flask
app = flask.Flask(__name__)
input_string = ""

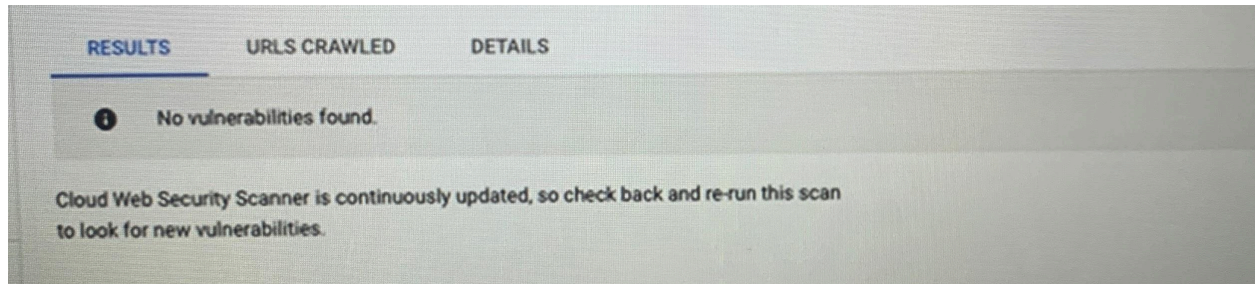
html_escape_table = {
    "&": "&amp;",
    '"': "&quot;",
    "'": "&apos;",
    ">": "&gt;",
    "<": "&lt;",
}

@app.route('/', methods=["GET", "POST"])
def input():
    global input_string
    if flask.request.method == "GET":
        return flask.render_template("input.html")
    else:
        input_string = flask.request.form.get("input")
        return flask.redirect("output")

@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)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)
```

Then I re-scanned to ensure no vulnerabilities were left.



Through this lab, I gained practical experience in scanning for application vulnerabilities. I learned the importance of a security analyst's ability to scan for application vulnerabilities, which is essential for helping identify and address potential weaknesses, managing risks, meeting compliance requirements and ultimately, maintaining a robust security posture to protect an organization's asset and reputation. In this lab I completed one of the fundamental aspects of proactive cybersecurity strategies.