IT 351 Cloud Security and DevSecOps

# Lab 9

Fall 2021 – Professor Josh Edgar

Welcome to our ninth and final Lab. This week, we add another simple job – and, for this course, our last job – that will conduct some light penetration testing in the form of network vulnerability scans, enabling us to see if a URL we specify is vulnerable to hackers. The tool, OWASP Zed Attack Proxy, or ZAP, is developed and maintained by the same organization that released the Dependency Check tool we used in Lab 5. As such, we will consolidate them under a single stage, so that they can run back-to-back.

Unlike many of our other Labs, this job stands somewhat on its own due to the complexity associated with getting ZAP integrated with other tools while using GitLab CI/CD. While OWASP ZAP is capable of funneling its reports up to SonarQube just like Dependency Check does, this is difficult to configure and requires building our job with Maven, so we won't actually do it – however, we will place both OWASP jobs under a single GitLab CI/CD stage as if we were preparing to do it. This will also demonstrate to you how multiple jobs can run concurrently under a single stage.

**Before you begin:**

- Your GitLab Runner instance should be started, if you previously stopped it.

**What you'll achieve:**

- Rename your "Dependency Check" stage as an overarching OWASP stage, which will contain both the Dependency Check and ZAP jobs
- Add the OWASP ZAP job to your pipeline and run a pen test against a URL of your choice

**What you'll turn in with this lab:**

- Make a text submission on Canvas with the word "Done" and the URL you chose to pen test, and the professor will check your repo for the completed assignment.

## Task 1: Consolidate OWASP Jobs into a Single Stage

Most of the jobs we've built in this class live within their own stages, keeping them logically separated from one another. A stage can contain multiple jobs, but since stages run consecutively in a pipeline, it's possible that the jobs will either run concurrently or out of order. In this case, since the OWASP jobs can run in whichever order, and it makes sense to group them together, we will consolidate them into a single stage.

1. In VS Code, open your ".gitlab-ci.yml" file. In the top "stages" section, rename your "Dependency Check" stage to simply "owasp" (but **without** quotes).

2. Scroll down to your “owasp-dc” job, and make the necessary adjustment to the stage line in the job code, so that it will now belong to your renamed “owasp” stage.
3. Save your file, but don’t commit or push anything yet.

## Task 2: Create OWASP ZAP Job

We will now add the OWASP ZAP job code below the section for our OWASP Dependency Check job. However, unlike previous Labs, this time I am not giving you the code – this is our final Lab, so I want you to demonstrate that you know how this syntax should work, since we’ve used similar code so many times before! You can follow along with how I build the code in the Panopto walkthrough, but no copy and pasting this time – we’ve done this enough times that hopefully, writing GitLab job code should come to you as second nature by now!

4. Create job code for the OWASP ZAP job. You should probably name it “owasp-zap” (since our other job is “owasp-dc”). Make sure it also aligns to the “owasp” stage you created in Task 1. For this job, we will use the `owasp/zap2docker-stable` image. This time, the script will simply be:
   `zap-full-scan.py -t https://url-of-your-choice.com`
   You can replace the url-of-your-choice bit with any website – Google’s, Marymount’s, it doesn’t matter, as long as it is publicly accessible on the open Internet and supports SSL/TLS traffic (hence the “https” at the beginning).
5. Save, commit and push your changes.

Observe your pipeline as it runs. As we should now be used to, it will take a while to complete, but once your new “owasp” stage kicks off, you should see either your Dependency Check or ZAP job kick off. If we had a really resourceful GitLab Runner with lots of CPU and RAM, these jobs would most likely run concurrently – most of our pipeline would, except for where we specify that an upstream job is needed – but, to keep costs low, we have very small Runners, which is why we frequently see jobs “waiting to be picked by a runner.” Most organizations in the real world either have very powerful Runners, or multiple Runners, allowing their entire pipelines to run much faster.

Once your “owasp-zap” job kicks off, observe the output. It will conduct a penetration test simulation of scans against the URL you specified. It conducts a full scan, which can take several minutes, but if your job code was written correctly, you should see numerous warnings about security flaws with the target website (no website is perfect). Once you see a successful scan output, submit the word “Done” as well as the URL you chose as a **text submission** via Canvas (do not submit the URL as a URL itself).