Wallarm Filtering Node Summary

Step1;

## Set Up FastAPI Backend

1. I opened VSCode and create a project folder:  
 mkdir wallarm-fastapi-demo && cd wallarm-fastapi-demo && code .  
  
2. Create the backend app in 'src/main.py':  
 from fastapi import FastAPI  
 app = FastAPI()  
  
 @app.get('/demo')  
 async def root():  
 return {'message': 'Hello from FastAPI backend!'}

A screenshot of a computer

AI-generated content may be incorrect.  
  
3. Add requirements.txt with fastapi and uvicorn  
4. Add Dockerfile to build the app using Python 3.10

Step 2

Create a docker-compose.yml file in the root folder.  
2. Define services for Wallarm WAF and FastAPI backend.  
3. I set environment variables including your WALLARM\_API\_TOKEN, from the wallarm documentation  
4. Run 'docker-compose up --build' to deploy both services.

Step 3

1. Run this code to simulate the attacks - docker run --rm -v ./reports:/app/reports --network wallarm-demo\_wallarm-net -it wallarm/gotestwaf --url=http://wallarm-svc:80/demo --blockStatusCodes 200

2. Verify output for blocked attacks through the Wallarm cloud platform and compare with the report.

Results

A screenshot of a computer

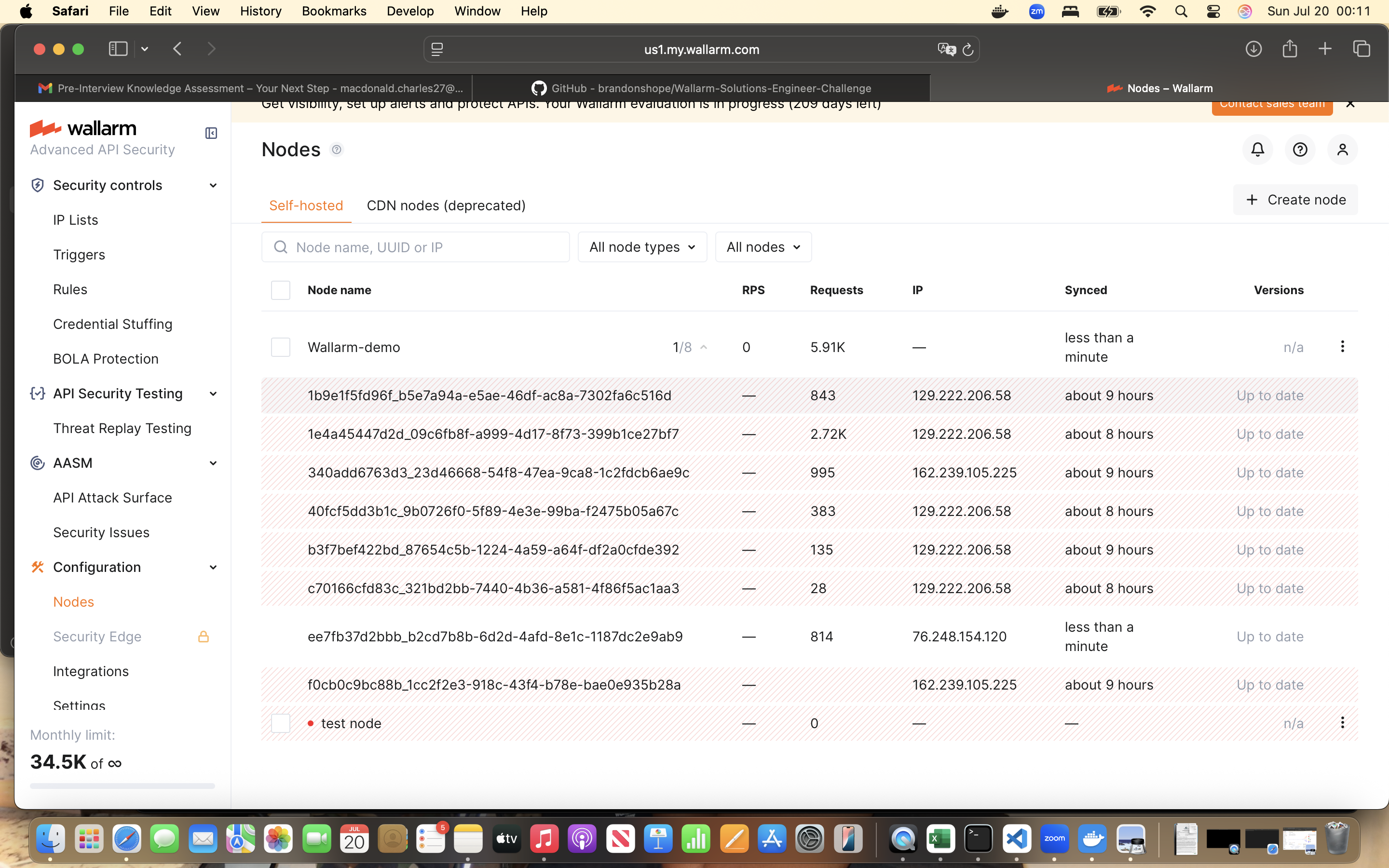
AI-generated content may be incorrect.

This shows the FASTAPI is sending the requests through to the Wallarm Platform.

A screenshot of a computer

AI-generated content may be incorrect.

The node flagged 97 attacks in monitor mode in GoTestWaf attacks on the Wallarm Cloud Dashboard.



This shows that the Node token used in the Wallarm cloud platform is working.

A screenshot of a computer

AI-generated content may be incorrect.

This shows the container was created and it ran successfully. (the chosen deployment is docker)

A screenshot of a computer

AI-generated content may be incorrect.

**This is the GoTestWaf container**:

**docker run --rm -v ./reports:/app/reports --network wallarm-demo\_wallarm-net -it wallarm/gotestwaf --url=http://wallarm-svc:80/demo --blockStatusCodes 200**

This command runs GoTestWAF inside a Docker container to simulate attack traffic through the Wallarm Filtering Node and test its effectiveness. The tool sends requests to the FastAPI backend via Wallarm and logs the results, treating HTTP 200 responses as blocked attacks for evaluation purposes. The test results are saved in a local reports folder for review and documentation. These reports summarize which attacks were detected, blocked, or passed through, helping you evaluate the effectiveness of your Wallarm WAF deployment

I have attached the pdf generated and this will show the results but here is a snapshot of the results. Which shows the initial 108 requests that went through the node, soon after that the 708 requests didn’t go through. What I think is the reason the WAF is not handling all attack categories- which may require specific policies to be enable in the Wallarm UI. I am certain the traffic is being routed correctly through to Wallarm because I fixed the traffic is going through the correct port.