# Executive Summary

# Part 1: Reconnaissance

Using the initial starter code to verify socket and docker networking working properly, I was able to discover port 5000 on `172.20.0.10` was open.

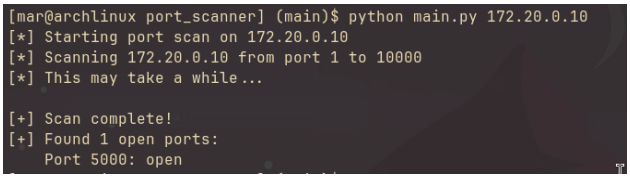


Figure 1: Basic socket test to find port 5000

After implementing some input handling with `argparse`, CIDR handling with `ipaddress`, and threading, I ran `python main.py --target 172.10.0.0/24 --ports 1-10000 --threads 10000` and got the results below.

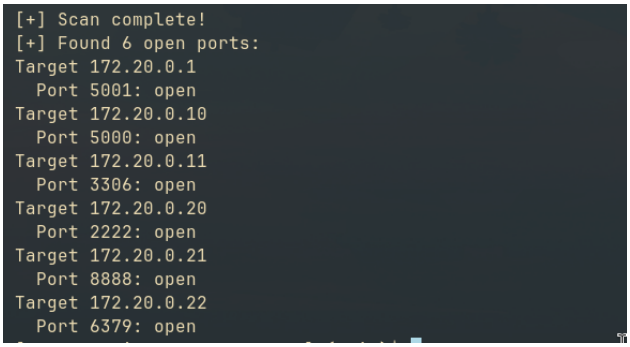


Figure 2: Open ports on the `172.10.0.0/24`

Implementing some level of banner grabbing by going through some common probing techinques onto the same subnet and ports on Figure 3. I'm not certain why `172.20.0.1:2222` appears to allow a socket connection now when Figure 2 doesn't have it open.
Regardless, based on these banners, the services for these ports are

- `172.20.0.1:5001` - `http`
- `172.20.0.10:5000` - `http`
- `172.20.0.11:3306` - `mysql`
- `172.20.0.20:2222` - `ssh`
- `172.20.0.21:8888` - `http`
- `172.20.0.22:6379` - `telnet`

where the last one knowledge that in `telnet`, `get` is a command and can verify by trying to connect via `telnet`, seen in Figure 4.
Accessing the SSH server, credentials are shown when connecting with `ssh`, I was able to read out the flag `FLAG{h1dd3n_s3rv1c3s_n33d_pr0t3ct10n}`.

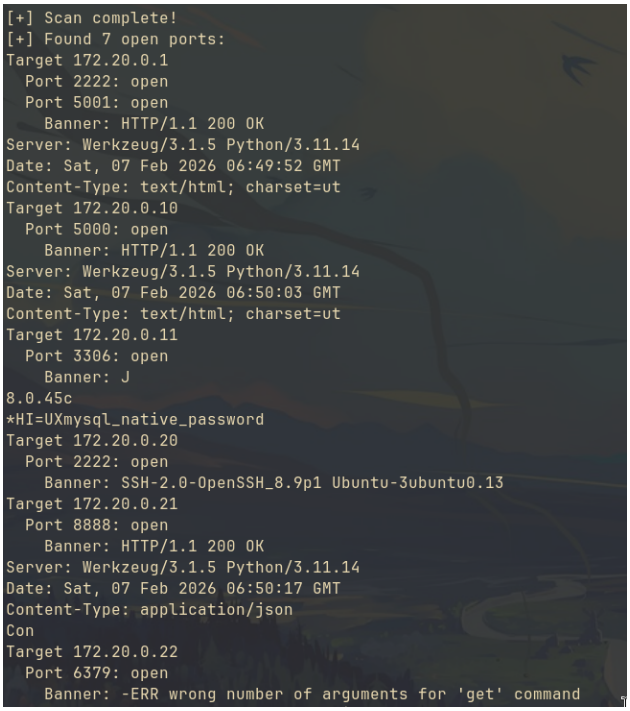


Figure 3: Banner grabbing on `172.10.0.0/24`

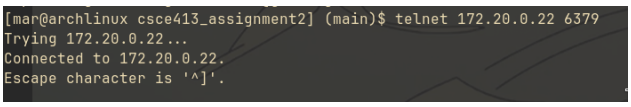


Figure 4: Telnet connection on `172.10.0.22:6379`

Curling these http services, at `172.20.0.21:8888`, it appears to be some sort of api route. Here it says that there's a flag route that also needs a token with hint to intercept network traffic. Based on this description, it likely for part 2 regarding analyzing network traffic.

```
{
  "authentication": {
    "alternative": "?token=<token> query parameter",
    "header": "Authorization: Bearer <token>",
    "hint": "The token can be found by intercepting network traffic...",
    "type": "Bearer token"
  },
  "endpoints": [
    {
      "description": "API information",
      "method": "GET",
      "path": "/"
    },
    {
      "description": "Health check",
      "method": "GET",
      "path": "/health"
```

```
    },
    {
      "description": "Get flag (requires authentication)",
      "method": "GET",
      "path": "/flag"
    },
    {
      "description": "Get secret data (requires authentication)",
      "method": "GET",
      "path": "/data"
    }
  ],
  "message": "This is a hidden API service. Authentication required.",
  "port": 8888,
  "service": "Secret API Server",
  "status": "running",
  "version": "1.0"
}
```


## Part 2: MITM Attack

## Part 3: Security Fixes

### Port Knocking

### Honeypot

## Remediation Recommendations

## Conclusion