

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.

```
[mar@archlinux port_scanner] (main)$ python main.py 172.20.0.10
[*] Starting port scan on 172.20.0.10
[*] Scanning 172.20.0.10 from port 1 to 10000
[*] This may take a while...
[+] Scan complete!
[+] Found 1 open ports:
    Port 5000: 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.

```
[+] Scan complete!
[+] Found 6 open ports:
Target 172.20.0.1
    Port 5001: open
Target 172.20.0.10
    Port 5000: open
Target 172.20.0.11
    Port 3306: open
Target 172.20.0.20
    Port 2222: open
Target 172.20.0.21
    Port 8888: open
Target 172.20.0.22
    Port 6379: open
```

Figure 2: Open ports on the 172.10.0.0/24

Implementing some level of banner grabbing by going through some common probing techniques 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}`.

```
[+] Scan complete!
[+] Found 7 open ports:
Target 172.20.0.1
  Port 2222: open
  Port 5001: open
    Banner: HTTP/1.1 200 OK
Server: Werkzeug/3.1.5 Python/3.11.14
Date: Sat, 07 Feb 2026 06:49:52 GMT
Content-Type: text/html; charset=ut
Target 172.20.0.10
  Port 5000: open
    Banner: HTTP/1.1 200 OK
Server: Werkzeug/3.1.5 Python/3.11.14
Date: Sat, 07 Feb 2026 06:50:03 GMT
Content-Type: text/html; charset=ut
Target 172.20.0.11
  Port 3306: open
    Banner: J
8.0.45c
*!H=UXmysql_native_password
Target 172.20.0.20
  Port 2222: open
    Banner: SSH-2.0-OpenSSH_8.9p1 Ubuntu-3ubuntu0.13
Target 172.20.0.21
  Port 8888: open
    Banner: HTTP/1.1 200 OK
Server: Werkzeug/3.1.5 Python/3.11.14
Date: Sat, 07 Feb 2026 06:50:17 GMT
Content-Type: application/json
Con
Target 172.20.0.22
  Port 6379: open
    Banner: -ERR wrong number of arguments for 'get' command
```

Figure 3: Banner grabbing on 172.10.0.0/24

```
[mar@archlinux csce413_assignment2] (main)$ telnet 172.20.0.22 6379
Trying 172.20.0.22...
Connected to 172.20.0.22.
Escape character is '^]'.
```

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