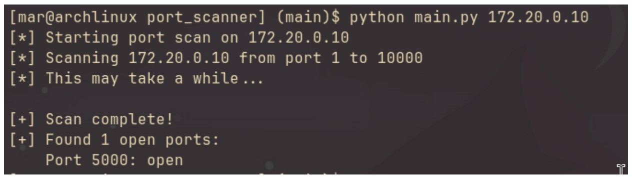
Executive Summary

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.

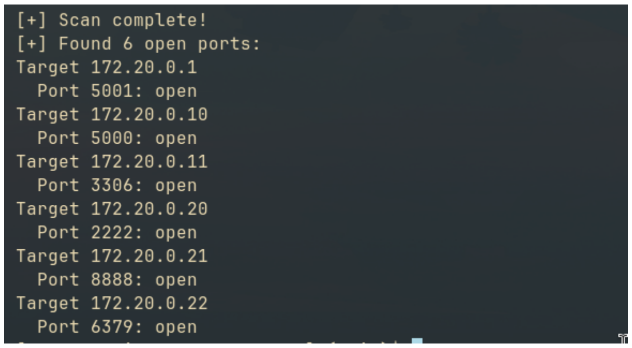


```
[mar@archlinux port_scanner] (main)$ python main.py 172.20.0.10
[*] Starting port scan on 172.20.0.10
[*] Scanning 172.20.0.10 from port 1 to 10000
[*] This may take a while...

[+] 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`

Part 2: MITM Attack

Part 3: Security Fixes

Port Knocking

Honeypot

Remediation Recommendations

Conclusion