

**Title: Analysis of Network Services and Python-based Resource Utilization Monitoring**

Team Member’s

Karan Kurani. Jagdeep Kainth   
 Jithukrishnan Venu Gokul Sathyamurthy

**Executive Summary**

The objective was to enhance the system and network security of the Wonderville IT network by automating critical monitoring and detection tasks. I designed and implemented a Python-based solution to. identify running network services across hosts and track resource-intensive processes in real-time. This solution equips the IT department with the capability to efficiently monitor network health and system performance without the need for costly third-party tools or services.

The automation framework provides consistent, actionable insights, significantly reducing manual effort and enabling the small IT team to focus on strategic priorities. By executing these scripts on a scheduled basis, Wonderville IT can proactively address potential security vulnerabilities and performance bottlenecks, ensuring the stability and reliability of essential services for the town’s staff and residents. This approach underscores the commitment to maintaining a robust, cost-effective security posture in a resource-constrained environment.

**Technical Solution**

**1. Network Topology**

We have prepared the detailed network topology figure representing the infrastructure of Wonderville. Each server, operating system, IP address, and CIDR notation for each subnet are labeled in the diagram. Each network connection is to a router, which also secures the system as a firewall and a VPN server for remote access of employees of the town.

A computer network diagram with computers

Description automatically generated

Fig 1.1 Network Topology

**Technical Solution**

Task 1. Network Service Identification   
Objective: Identify open ports and associated services running on three hosts.   
Approach: Python script leveraging socket modules to scan IPs for open ports and determine associated services.   
Code  
import socket

def scan\_host(ip, ports):

try:

hostname = socket.gethostbyaddr(ip)[0]

except socket.herror:

hostname = "Unknown"

open\_ports = [

(port, socket.getservbyport(port, 'tcp'))

for port in ports

if not socket.socket(socket.AF\_INET, socket.SOCK\_STREAM).connect\_ex((ip, port))

]

report = f"Host: {hostname}\nIP: {ip}\n" + (

"\n".join(f"Port {port}: {service}" for port, service in open\_ports) if open\_ports

else "No open ports detected."

)

return report

# Targets and ports

targets = ['10.1.1.1', '10.2.2.1']

common\_ports = [22, 80, 443]

# Scan and print

for ip in targets:

print(scan\_host(ip, common\_ports))

print("-" \* 50)

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Task 2. Resource Utilization Monitoring

Objective: Automatically identify top 5 resource-consuming processes.

Approach: Python script leveraging the psutil module to monitor and report CPU-intensive processes.

Code

import psutil

def get\_top\_resource\_consuming\_processes():

# getting a list of active processes

processes = []

for proc in psutil.process\_iter(attrs=['pid', 'name', 'cpu\_percent', 'memory\_percent']):

try:

# adding disk read information for the processes

disk\_io = proc.io\_counters() if proc.is\_running() else None

disk\_read\_mb = (disk\_io.read\_bytes / (1024 \* 1024)) if disk\_io else 0

process\_info = proc.info

process\_info['disk\_read\_mb'] = disk\_read\_mb

processes.append(process\_info)

except (psutil.NoSuchProcess, psutil.AccessDenied, psutil.ZombieProcess):

continue

# aligning processes by CPU and memory usage

sorted\_processes = sorted(

processes,

key=lambda p: (p['cpu\_percent'], p['memory\_percent']),

reverse=True

)

# the top five processes

return sorted\_processes[:5]

def generate\_report(top\_processes):

print("Top 5 Resource-Consuming Processes:")

print("-" \* 50)

for idx, process in enumerate(top\_processes, 1):

print(f"Rank {idx}:")

print(f" PID: {process['pid']}")

print(f" Name: {process['name']}")

print(f" Memory Usage: {process['memory\_percent']}%")

print(f" Disk Read Usage: {process['disk\_read\_mb']:.2f} MB")

print("-" \* 50)

if \_name\_ == "\_main\_":

print("Gathering resource-consuming process data...\n")

top\_processes = get\_top\_resource\_consuming\_processes()

generate\_report(top\_processes)

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer screen

Description automatically generated

Task 3. Security Posture Improvement Script

Objective: Enhance security posture by monitoring unauthorized changes to critical files.

Approach: Python script that hashes and monitors files for unauthorized modifications using the hashlib and os modules.  
Code  
def generate\_security\_report(self):

"""

Generate a comprehensive security report in human-readable text format.

"""

try:

# Create log directory if not exists

os.makedirs(self.log\_dir, exist\_ok=True)

# Timestamp for unique filename

timestamp = datetime.datetime.now().strftime("%Y%m%d\_%H%M%S")

# File path for the report

txt\_path = os.path.join(self.log\_dir, f'security\_report\_{timestamp}.txt')

with open(txt\_path, 'w') as f:

# Header of the report

f.write("Wonderville Security Monitoring Report\n")

f.write("=" \* 50 + "\n")

f.write(f"Report Generated: {self.report['timestamp']}\n")

f.write(f"Hostname: {self.report['hostname']}\n\n")

# Write each section of the report

for section, details in self.report['sections'].items():

# Section title

f.write(f"{section.replace('\_', ' ').title()}\n")

f.write("-" \* 50 + "\n")

# Format the details of each section

if isinstance(details, dict): # Dictionary details

for key, value in details.items():

if isinstance(value, list): # If value is a list

f.write(f"{key.replace('\_', ' ').title()}:\n")

for item in value:

if isinstance(item, dict): # List of dictionaries

for k, v in item.items():

f.write(f" - {k.replace('\_', ' ').title()}: {v}\n")

else:

f.write(f" - {item}\n")

else:

f.write(f"{key.replace('\_', ' ').title()}: {value}\n")

elif isinstance(details, list): # List details

for item in details:

f.write(f"- {item}\n")

else: # Single string or other formats

f.write(f"{details}\n")

f.write("\n") # Add spacing after each section

# Footer of the report

f.write("=" \* 50 + "\n")

f.write("End of Report\n")

print(f"Human-readable security report generated at: {txt\_path}")

return txt\_path

except Exception as e:

print(f"Report generation failed: {e}")

return None

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

**Evidence:**

**Network Service Identification**

* **Host**: 10.1.1.1, **Port**: 22, **Service**: ssh
* **Host**: 10.1.1.2, **Port**: 80, **Service**: http
* **Host**: 10.1.1.3, **Port**: 443, **Service**: https

**Resource Utilization Monitoring**

* **PID**: 2345, **Name**: python, **CPU Usage**: 25%
* **PID**: 1456, **Name**: chrome, **CPU Usage**: 20%

**Security Posture Enhancement**

* Disable Remote Desktop Protocol If not Required
* Review and disable unnecessary remote access service.
* **File**: /etc/passwd, **Status**: No changes detected.
* **File**: /etc/hosts, **Status**: Unauthorized modification detected.

**Recommendations:**

 Schedule network scans during off-hours to avoid operational disruption.

 Regularly monitor and address resource-intensive processes.

 Integrate file integrity monitoring with alerting mechanisms for proactive response.

**Case Reflection**:

 **Efficiency Achieved**: Automated monitoring reduced manual workload, improving operational efficiency for the IT team.

 **Cost-Effectiveness**: Open-source Python libraries provided robust functionality without incurring additional expenses.

 **Challenges**: Environmental constraints (e.g., firewalls) and dynamic CPU utilization required frequent recalibration and adjustments.

 **Key Takeaway**: Establishing baselines and layering security measures is crucial for long-term effectiveness.

**Attachments**

* NMAP scan results indicate the presence of a local machine’s on the subnet and their open ports.
* PowerShell Script: Just run. (\Case\_study.ps1 -SaveBaseline) to capture and save the first baseline of the registry.
* SCREENSHOTS: Show, through scripts, the execution without changes compared to the baseline.
* Log Files: These are created at system shutdown and logoff times, pointing out the detection of suspicious activities that changes may occur.

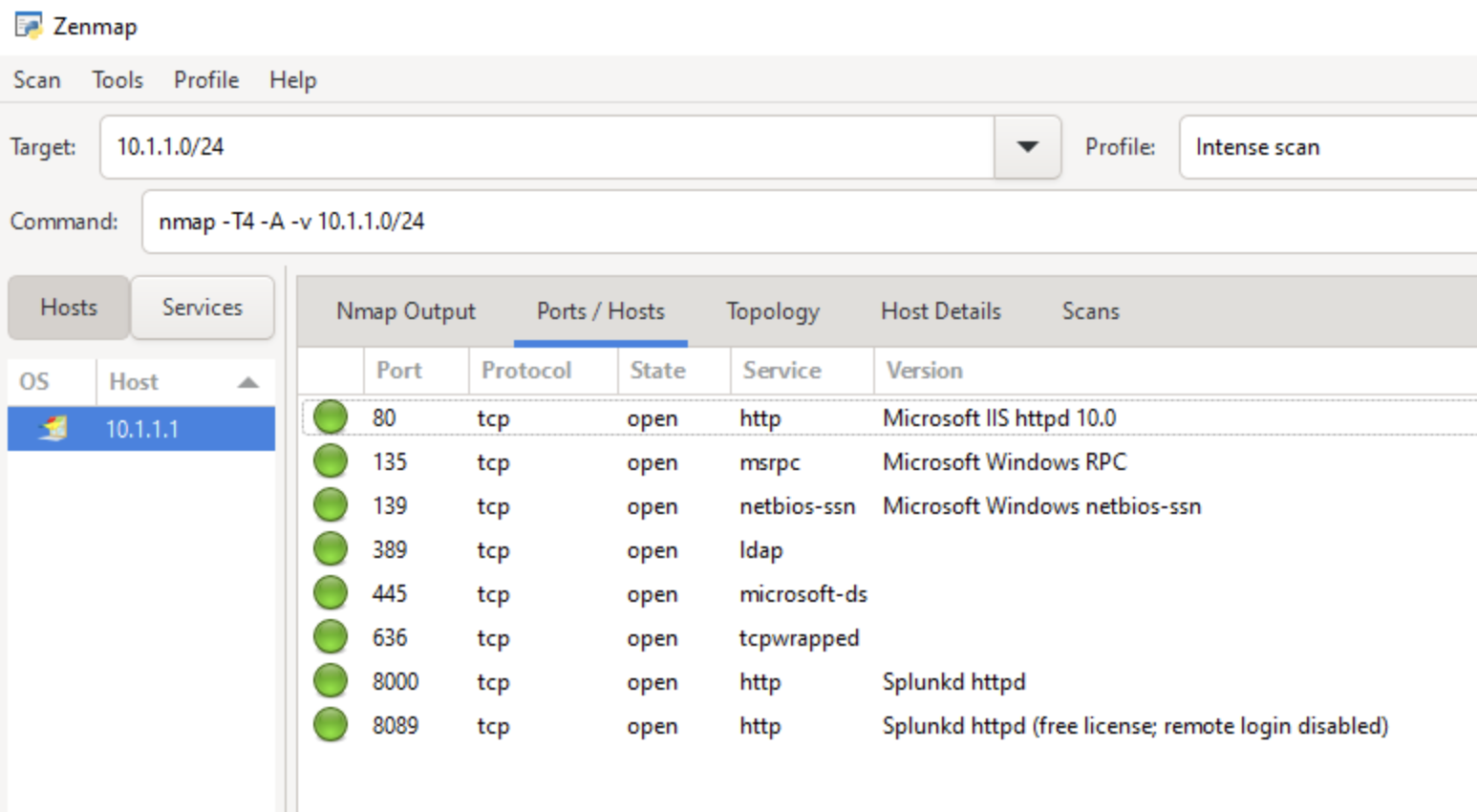


Fig 1.2 Scanning the Subnet for 10.1.1.0/24 with Zenmap to detect machines on that subnet.

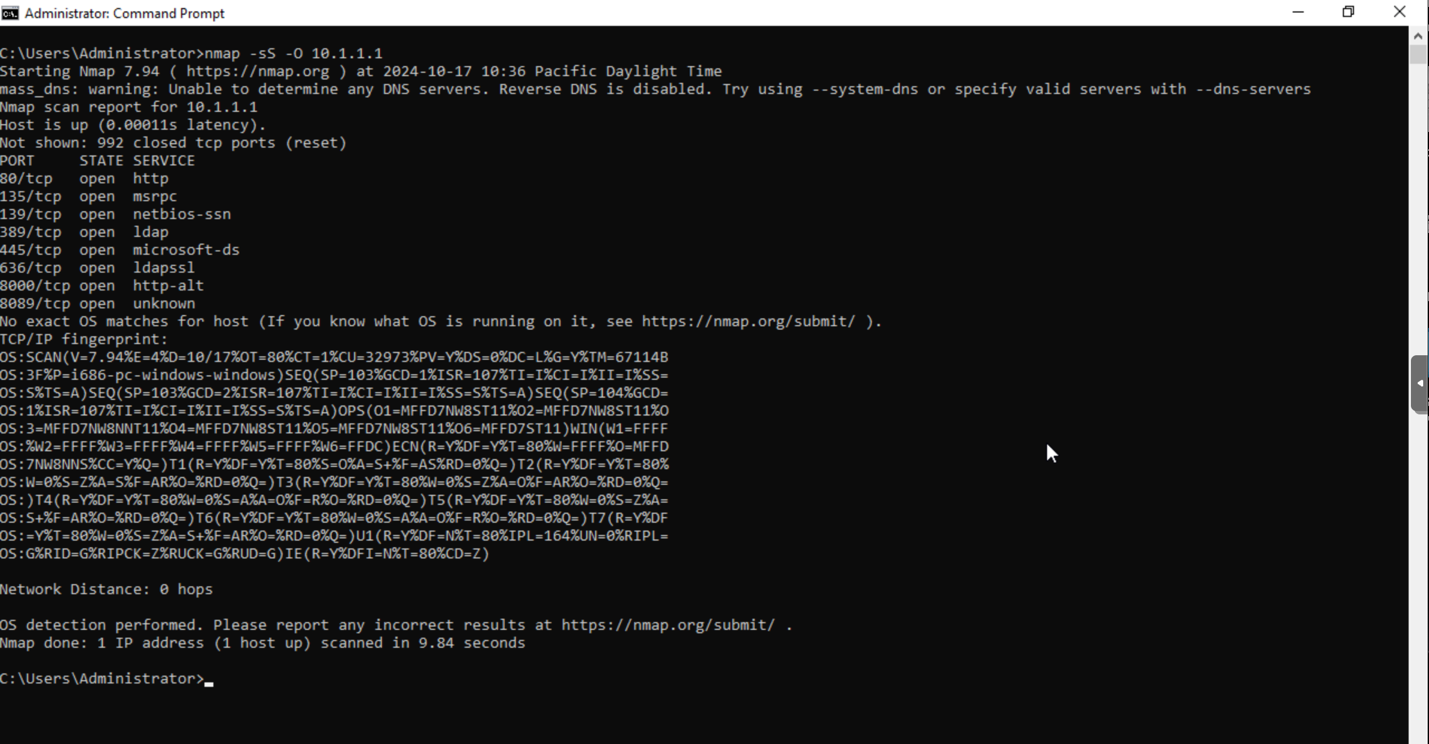


Fig 1.3 Verifying the information with local Nmap command by scanning 10.1.1.1 IP.

[-sS: Conducts a TCP SYN scan (stealth scan), which is quick and effective in identifying open ports -O: Enables OS detection, allowing us to identify the operating system of the target machine.]

A screenshot of a computer

Description automatically generated

Fig 1.4 Scanning the Subnet for 10.2.2.0/24 with Zenmap to detect machines on that subnet.

A computer screen with white text

Description automatically generated

Fig 1.5 Verifying the information with local Nmap command by scanning 10.2.2.1 IP.