

# **Task 01: Networking Fundamentals, Nmap Scanning, and Automation Scripting**

**CYART – Task 01 Documentation**

Prepared by: **Kalash Mahajan**

Date: **20th January 2026**

## **Introduction**

In modern computing environments, networks form the backbone of communication between systems, applications, and users. Understanding how devices communicate over a network is a fundamental requirement for anyone working in cybersecurity, system administration, or network operations. Core concepts such as **IP addressing**, **ports**, and **communication protocols** determine how data flows across networks and how services are exposed to users and potential attackers.

This document presents the practical implementation of Networking Fundamentals and Nmap-based network scanning. The primary objective of this task is to develop a clear understanding of how networking components operate in a real-world environment and how network visibility can be achieved using industry-standard tools. By performing controlled scans on an authorized target, this task demonstrates how open ports and running services can be identified and analyzed from a security perspective.

**Nmap (Network Mapper)** is one of the most widely used tools in cybersecurity for network discovery and security auditing. In this task, Nmap is used to perform different types of scans, including **SYN** scans, **TCP connect** scans, and **UDP** scans, to observe how various protocols respond and how firewall behavior affects scan results. The output of these scans is then analyzed to identify active services, understand their purpose, and assess potential security risks associated with exposed network services.

Additionally, this task emphasizes the importance of documentation and structured reporting in cybersecurity work. All findings, commands, and observations are documented in a clear and organized manner to simulate real-world reporting practices followed by security analysts and SOC teams. The task also introduces the concept of

automation through scripting, highlighting how repetitive scanning tasks can be automated using Python to improve efficiency and consistency in security operations.

Overall, this document serves as both a technical record of the activities performed and a learning reference that connects theoretical networking concepts with hands-on practical execution.

## Target Description

Target Type: **Local Machine**

Target IP Address: **127.0.0.1 (localhost)**

The authorized scanning target for this task was the local Windows machine, accessed using the loopback IP address 127.0.0.1 (localhost). The loopback address refers to the host system itself and is commonly used for testing and analysis purposes.

Using the local machine ensured that all scans were conducted in a safe and controlled environment, without impacting external networks or devices. This approach aligns with ethical security practices and allows focused analysis of locally running services and open ports.

## Networking Fundamentals

### IP Address:

An IP address uniquely identifies a device on a network and allows systems to locate and communicate with each other. IP addresses can be either public or private, depending on whether they are routable over the internet or restricted to internal networks.

In this task, the loopback address 127.0.0.1 was used. This address refers to the local host and is primarily used for testing and diagnostic purposes. Any traffic sent to this address is handled internally by the operating system, ensuring that the scanning activity remains confined to the local machine and does not interact with external systems.

### Ports:

Ports are logical communication endpoints that allow multiple services to operate simultaneously on a single system. Each network service listens on a specific port number, enabling the operating system to direct incoming traffic to the correct application.

For example, web services typically operate on **port 80 (HTTP)** or **port 443 (HTTPS)**, while file-sharing services may use **port 445 (SMB)**. From a security perspective, each open port represents a potential entry point for attackers. Therefore, identifying open ports and understanding the services bound to them is a critical step in assessing a system's exposure and attack surface.

### **Protocols:**

Network protocols define the rules and procedures for data transmission between devices. Two of the most used transport-layer protocols are TCP and UDP.

**TCP (Transmission Control Protocol)** is a connection-oriented protocol that ensures reliable data delivery through a three-way handshake, acknowledgments, and retransmissions. It is commonly used by services where data accuracy is essential, such as web traffic, file transfers, and remote administration.

**UDP (User Datagram Protocol)** is a connectionless protocol that sends data without establishing a formal connection or guaranteeing delivery. While faster and more efficient, UDP is less reliable and is commonly used for services such as DNS, streaming, and discovery protocols. Due to its nature, UDP services are often harder to scan accurately and may appear as open or filtered in scan results.

### **Basic Network Architecture**

Network architecture defines how devices, services, and security controls are arranged within a network. In a typical environment, end-user devices communicate with services through switches and routers, while firewalls control traffic flow between trusted and untrusted networks.

Understanding basic network architecture helps security analysts interpret scan results correctly. For example, firewalls may block or filter certain ports, affecting scan visibility, while internal services may only be accessible within trusted network boundaries. This task demonstrates how system configuration and defensive controls influence network scan outcomes.

## Nmap Installation and Verification

Nmap was downloaded and installed from the official website <https://nmap.org>, ensuring the tool was obtained from a trusted and up-to-date source. After installation, the setup was verified to confirm that Nmap was correctly installed and accessible from the system's command line interface.

The following command was executed to verify the installation and check the installed version:

```
nmap -v
```

Successful execution of this command confirmed that Nmap was properly installed and ready for use. Verifying the installation before performing scans is an important step to ensure tool reliability and avoid execution or configuration issues during security assessments.

## Nmap Scans Performed

To analyze the network exposure of the authorized target, multiple Nmap scan types were executed. Each scan was selected to observe different aspects of network behavior and service availability. The combination of TCP and UDP scans provides a more complete view of the system's attack surface.

### SYN Scan (TCP Half-Open Scan)

Command Used:

```
nmap -sS 127.0.0.1 -oN syn_scan.txt
```

The SYN scan is a half-open TCP scan that initiates a connection by sending a SYN packet but does not complete the full TCP handshake. Because the connection is never fully established, this scan is considered more efficient and less intrusive than a full TCP connect scan.

This scan is widely used by security professionals as it allows rapid identification of open TCP ports while generating minimal interaction with the target system. In this task, the SYN scan was used to identify actively listening TCP services and establish a baseline view of open ports on the local machine.

### TCP Connect Scan

Command Used:

```
nmap -sT 127.0.0.1 -oN tcp_scan.txt
```

The TCP connect scan performs a full TCP three-way handshake to determine whether a port is open. Unlike the SYN scan, this method relies on the operating system's network stack to establish the connection, making it more detectable and slower. This scan was included to compare results with the SYN scan and observe how fully established connections are handled by the system. TCP connect scans are useful in environments where SYN scans are restricted or when administrative privileges are unavailable.

## **UDP Scan**

Command Used:

```
nmap -sU 127.0.0.1 -oN udp_scan.txt
```

The UDP scan is used to identify services running over the User Datagram Protocol. Since UDP is connectionless and does not provide acknowledgment responses, UDP scanning is inherently slower and less definitive than TCP scanning.

In many cases, UDP ports may appear as open or filtered due to firewall behavior or lack of response from the service. This scan was performed to identify potential UDP-based services such as name resolution, discovery, and VPN-related services that could otherwise be overlooked in TCP-only scans.

## **Scan Findings**

The Nmap scans provided visibility into the active network services on the local machine by identifying TCP and UDP ports that responded to scan probes. The results reflect both the services currently listening on the system and the effect of operating system and firewall behavior on scan visibility.

### **Understanding Port States**

*Open:*

A port is marked as open when an application on the system is actively listening for incoming connections on that port. This indicates that a service is running and reachable from the scanning host.

*Open / Filtered:*

This state indicates that Nmap could not determine whether the port is open or blocked

by a firewall. This result is common in UDP scans because UDP does not provide acknowledgment responses, and firewalls may silently drop packets.

### TCP Scan Findings

The TCP SYN and TCP connect scans identified the following open TCP ports on the local machine. These ports represent services that are actively listening for TCP connections.

Port	Protocol	Service
80	TCP	HTTP
443	TCP	HTTPS
135	TCP	MSRPC
445	TCP	SMB
902	TCP	VMware Service
912	TCP	VMware Service

Both the SYN scan and TCP connect scan produced consistent results, confirming the presence of these services and demonstrating the reliability of TCP-based scanning techniques.

### UDP Scan Findings

The UDP scan revealed several ports in an open or filtered state. Due to the connectionless nature of UDP, these results indicate potential UDP-based services or filtered responses rather than definitive open ports.

Port	Protocol	Service
137	UDP	NetBIOS
500	UDP	ISAKMP
1900	UDP	UPnP
4500	UDP	NAT-T
5353	UDP	mDNS
5355	UDP	LLMNR

UDP scanning required significantly more time to complete compared to TCP scans, highlighting the inherent challenges of identifying UDP services and the impact of firewall rules on scan accuracy.

### Comparison of TCP and UDP Results

The TCP scans provided clear and deterministic results, allowing accurate identification of active services. In contrast, the UDP scan results were less definitive due to the lack of response mechanisms in the UDP protocol. This comparison demonstrates why comprehensive network assessments often require multiple scan types to obtain a more complete understanding of system behavior.

## Service Analysis

### HTTP (Port 80)

HTTP (Hypertext Transfer Protocol) is an application-layer protocol used for transmitting web content between clients, such as web browsers, and servers. It operates over TCP and is commonly used to deliver websites and web-based applications. In the scanned environment, the presence of HTTP indicates that a web service is actively listening on the local machine.

One key characteristic of HTTP is that it transmits data in plain text, meaning that information such as requests, responses, headers, and session identifiers are not encrypted by default. As a result, HTTP traffic can potentially be intercepted or modified if transmitted over untrusted networks. From a security perspective, HTTP services are also closely tied to application-layer risks, as vulnerabilities often arise from insecure coding practices rather than the protocol itself.

Common weaknesses associated with HTTP-based services include improper input validation, weak authentication mechanisms, and insecure session management. These issues can lead to web-based attacks such as cross-site scripting (XSS), SQL injection, and session hijacking if the underlying application is not properly secured.

### SMB (Port 445)

SMB (Server Message Block) is a network protocol primarily used in Windows environments to enable file sharing, printer access, and inter-process communication between systems. It allows users and applications to access shared resources over a network as if they were local to the system.

In a typical enterprise environment, SMB plays a critical role in day-to-day operations; however, it is also one of the most commonly targeted services during internal network attacks. SMB services are often leveraged for system enumeration, credential access, and lateral movement once an attacker gains initial access to a network.

Exposed or misconfigured SMB services can allow attackers to exploit protocol weaknesses, abuse weak credentials, or move laterally across systems. Historically, vulnerabilities in SMB have been widely exploited to spread malware and ransomware within networks, making it a service that requires strict access controls, regular patching, and continuous monitoring.

## Security Risks Identified

Based on the Nmap scan results, several network-accessible services were identified on the local machine, which collectively contribute to the system's exposed attack surface. The TCP scans revealed multiple actively listening services, while the UDP scan indicated the presence of several background discovery and communication protocols.

The detection of **HTTP (port 80)** alongside **HTTPS (port 443)** indicates that a web service is accessible over both encrypted and unencrypted channels. While HTTPS provides secure communication, the availability of HTTP introduces the possibility of *unencrypted data exchange* if misused or improperly redirected.

The scans also confirmed that **MSRPC (port 135)** and **SMB (port 445)** services are active. These services are integral to Windows system functionality but are frequently abused in *Windows-based attacks* when exposed or insufficiently restricted. Their presence highlights the importance of limiting access to trusted networks and ensuring that only necessary services are enabled.

Additionally, the UDP scan reported several ports in an open or filtered state, including services related to name resolution, device discovery, and VPN communication. Although these results do not conclusively confirm open ports, they demonstrate how UDP-based services can be difficult to assess and may remain accessible depending on firewall behavior and system configuration.

Overall, the scan results emphasize that even a local system can expose multiple services by default. Regular network scanning, service review, and access control enforcement are necessary to ensure that only required services remain accessible and that unnecessary exposure is minimized.

## Network Diagram

A network diagram was created using draw.io to visually represent the Nmap scanning activity performed during this task. The diagram illustrates the scanning host, the authorized target system, and the direction of the scans executed.

The diagram shows the **Nmap scanner (local machine)** initiating SYN, TCP connect, and UDP scans toward the **Local Windows host (127.0.0.1)**. It also summarizes the open TCP services and UDP services identified during the scan, providing a clear visual overview of the system's exposed network services.

The generated diagram is included separately as task01\_network\_diagram.png and is referenced below.

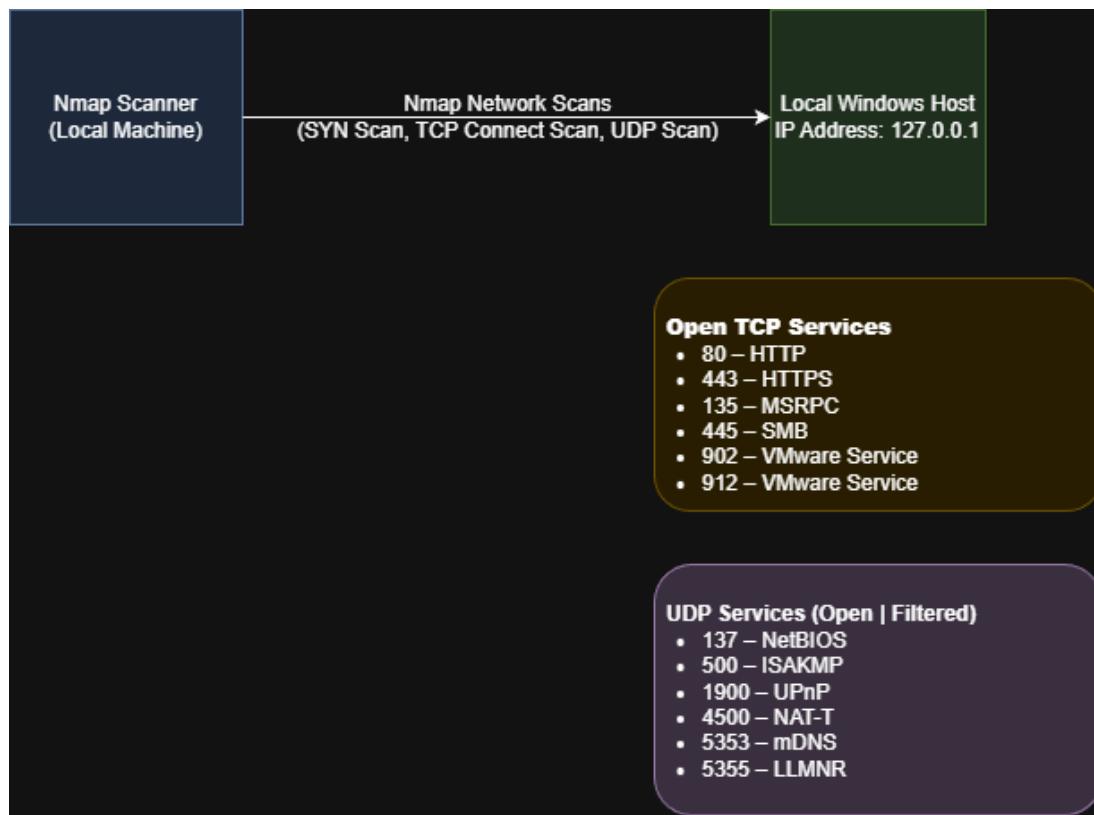


Figure 1 Network Diagram

## Key Learnings

This task reinforced the importance of approaching network scanning not as a tool-driven activity, but as an analytical process that combines protocol behavior, system configuration, and security context. By performing multiple scan types on the same

target, it became clear how different transport protocols and scan methodologies influence visibility, accuracy, and interpretation of results.

One key learning was understanding how TCP and UDP fundamentally differ from a detection and assessment perspective. While TCP-based scans provided deterministic results, UDP scans highlighted the ambiguity introduced by connectionless communication and firewall behavior. Interpreting open versus open or filtered states required an understanding of how operating systems and security controls handle unsolicited traffic rather than relying solely on scan output labels.

The task also emphasized that exposure does not always imply misconfiguration. Several identified services were expected in a Windows environment, demonstrating the need to distinguish between necessary system functionality and unnecessary service exposure. This distinction is critical in real-world security operations, where analysts must prioritize findings based on context rather than raw port counts.

Additionally, documenting each step of the process highlighted the value of clear, structured reporting in cybersecurity workflows. Accurate documentation ensures that scan results can be reviewed, reproduced, and validated by other analysts, which is essential in SOC environments and security assessments.

Overall, this task strengthened the ability to translate raw network scan data into meaningful observations, reinforcing a mindset focused on visibility, interpretation, and risk-aware decision-making, rather than tool execution alone.

## Python Automation

To extend the manual scanning process and demonstrate basic automation capabilities, a Python script was developed using the python-nmap library. The objective of this automation was to replicate a standard Nmap SYN scan in a repeatable and structured manner while generating a readable scan report automatically.

The script acts as a lightweight automation layer over the Nmap tool, allowing scan execution and result extraction without manually running commands each time. This approach reflects real-world security workflows, where repetitive scanning tasks are often automated to improve efficiency and consistency.

## **Overview of the Automation Script**

The Python script, named nmap\_automation.py, performs the following high-level functions:

1. Accepts a target IP address or hostname as user input.
2. Executes a TCP SYN scan (-sS) using Nmap.
3. Extracts scan results programmatically, including:
  4. Host IP address
  5. Host state
  6. Open TCP ports
  7. Associated service names and versions (if available)
8. Generates a structured text-based scan report for documentation and review.

The automation ensures that scan results are captured in a consistent format, reducing manual effort and minimizing the risk of transcription errors during reporting.

## **Scan Execution and Data Extraction**

The script initializes the Nmap scanner using the PortScanner object provided by the python-nmap library. Once the target is provided, the script triggers a SYN scan, mirroring the manual scan performed earlier in the task.

After the scan completes, the script iterates through the discovered host data and extracts relevant information from the scan results. Only ports in an open state are recorded, ensuring the report remains focused on active services.

This method demonstrates how raw scan data can be programmatically filtered and structured, which is a key requirement in automated security assessments and SOC tooling.

## **Automated Report Generation**

Upon completion of the scan, the script generates a report file named scan\_report.txt.

The report includes:

1. Scan timestamp
2. Target IP or hostname
3. Host state
4. List of open TCP ports
5. Service names and detected versions (if available)
6. Scan completion confirmation

The generated report serves as an evidence artifact that can be reviewed independently or attached to assessment documentation. This mirrors common practices in security operations, where automated scan outputs are stored for auditing, validation, or further analysis.

### Purpose and Practical Value

The automation component of this task highlights the importance of combining security tools with scripting to improve operational efficiency. While manual scanning is useful for learning and validation, automation enables scalability and repeatability, especially in environments where frequent assessments are required.

- This script demonstrates the ability to:
- Translate manual security tasks into automated workflows
- Integrate external security tools with Python
- Produce structured, analyst-friendly output suitable for reporting

### Files Generated

Python Script: *nmap\_automation.py*

Automated Scan Report: *scan\_report.txt*

Both files are included as part of the task deliverables and were generated from a successful test run against the authorized local machine.

### Summary

The Python automation component successfully complements the manual Nmap scanning performed earlier in the task. By automating scan execution and report generation, this section demonstrates foundational scripting skills and reinforces how automation supports effective and scalable security operations.

### Conclusion

This task successfully demonstrated the practical relationship between networking fundamentals, network scanning techniques, and basic automation. By applying concepts such as IP addressing, ports, and transport-layer protocols, and validating them through hands-on Nmap scans, the task provided clear visibility into how systems expose services and how those services can be identified and analyzed.

The manual execution of different scan types, combined with structured analysis and visualization, reinforced the importance of understanding not only what services are visible on a system, but why they appear and how protocol behavior and system configuration influence scan results. The inclusion of automation through Python further highlighted how repetitive security tasks can be streamlined while maintaining accuracy and consistency.

Overall, the task emphasized that effective security assessment is not limited to tool usage alone, but requires contextual interpretation, proper documentation, and the ability to scale processes through automation. These skills are essential for real-world security operations, continuous monitoring, and maintaining a secure system posture.