Nmap – TryHackMe Room Walkthrough

-Nandana Silju

Introduction & Objectives:

This report documents the reconnaissance and scanning tasks performed in the TryHackMe **Further Nmap** room. The goals were to explore advanced Nmap techniques—such as slicing scans, timing templates, scripting engines, and output formats—and capture practical output and insights.

Task 1 — Deploy:

In Task 1, the target virtual machine was deployed to serve as the host for all scanning exercises in the Further Nmap room. This machine is intentionally configured for scan-based interaction only, with no exploitation required. Its purpose is to provide a safe, controlled environment for testing Nmap's advanced scanning techniques, script usage, and firewall evasion methods in subsequent tasks.

Task 2 — Introduction:

This section outlines the purpose of the **Further Nmap** room: to deepen familiarity with Nmap's advanced features, including:

- Scanning specific port ranges and services.
- Timing templates (`-T0` to `-T5`) to balance speed and stealth.
- Output options (`-oA`, `-oN`, `-oX`, `-oG`) for saving results in different formats.
- Using the Nmap Scripting Engine (NSE) for vulnerability detection, service enumeration, and OS discovery.
- Searching, installing, and using NSE scripts from both local and online sources.
- Firewall evasion techniques like `-Pn`, packet fragmentation (`-f`), MTU control, scan delays, and checksum tricks.

The introduction emphasizes that the machine provided is for scanning only — no exploitation is required. The goal is to gain hands-on experience using these advanced Nmap switches and interpreting results.

Task 3 — Nmap Switches:

In Task 3, I explored and practiced Nmap's core command-line switches, focusing on different scanning modes, detection capabilities, output formats, and speed settings. Using the help menu (`nmap -h`) and manual pages (`man nmap`), I identified key options such as `-sS` for SYN scans, `-sU` for UDP scans, `-O` for OS detection, and `-sV` for service version detection. Output management was covered with switches like `-oA` (all formats), `-oN` (normal), and `-oG` (grepable). Timing templates (`-T0` to `-T5`) were reviewed for balancing speed and stealth, and port targeting commands like `-p 80`, `-p 1000-1500`, and `-p-` were practiced. Finally, I learned how to run NSE scripts, including specific categories such as `--script=vuln`, preparing me for more advanced Nmap scripting tasks later in the room.

Task 4 — Scan Types Overview:

In Task 4, I reviewed the primary and secondary scanning methods available in Nmap. The three main types—TCP Connect (`-sT`), SYN "half-open" (`-sS`), and UDP (`-sU`)—form the foundation of most port scanning operations, with differences in speed, stealth, and reliability. I also learned about less common but specialized scans such as TCP Null (`-sN`), TCP FIN (`-sF`), and TCP Xmas (`-sX`) scans, which are often used for firewall evasion or stealthier reconnaissance. Additionally, ICMP scanning was covered as a way to check host availability, although it can be blocked by common firewall settings. This section emphasized understanding how each scan type operates, so that the right one can be chosen depending on the environment and objectives.

Task 5 — TCP Connect Scans:

In this task, I explored how TCP Connect scans (`-sT`) operate by completing the full TCP three-way handshake with each target port. This method sends a SYN request and determines the port state based on the response: a SYN/ACK means the port is open, while an RST indicates it's closed (as defined in RFC 9293). If no response is received, the port is considered filtered—usually due to a firewall silently dropping packets. The task also highlighted that firewalls can be configured to send RST packets deliberately,

making accurate scanning more challenging. This scan type is reliable but less stealthy, as it establishes full TCP connections that can easily be logged.

Task 6 — SYN Scans:

This task covered SYN scans (`-sS`), also known as *Half-Open* or *Stealth* scans. Unlike TCP Connect scans, SYN scans send a RST packet immediately after receiving a SYN/ACK, avoiding a full three-way handshake. This makes them faster, less likely to be logged by applications, and able to bypass some older IDS systems. However, they require sudo/root permissions in Linux to craft raw packets and can occasionally disrupt unstable services. Port state detection for closed or filtered ports works the same as in TCP Connect scans, but SYN scans are the default choice when running Nmap with elevated privileges due to their efficiency and stealth.

Task 7 — UDP Scans:

UDP scans (`-sU`) work differently from TCP scans because UDP is a stateless protocol with no handshake. Nmap sends packets to target ports and waits for a response. If there's **no response**, the port is marked **open|filtered** (it could be open or blocked by a firewall). If a **UDP response** is received, the port is marked open. Closed UDP ports typically respond with an **ICMP "port unreachable"** message, allowing Nmap to label them as closed. UDP scanning is significantly slower than TCP scanning — scanning 1000 ports can take 20+ minutes — so `--top-ports <number>` is often used to limit scans to the most common UDP ports. For known services, Nmap may send service-specific payloads to improve detection accuracy.

Task 8 — NULL, FIN, and Xmas Scans:

NULL (`-sN`), FIN (`-sF`), and Xmas (`-sX`) scans are stealth-focused TCP scanning methods that avoid the SYN flag to help bypass certain firewalls.

- **NULL scans** send packets with no flags.
- **FIN scans** send packets with the FIN flag set.
- **Xmas scans** send packets with PSH, URG, and FIN flags, resembling a "lit" packet in Wireshark.

For **closed ports**, the expected RFC 793 behavior is a **RST** response; for **open ports**, there's typically no reply (which can also indicate filtering, making

results open|filtered). Some systems — notably **Microsoft Windows** and many Cisco devices — send RSTs to all malformed packets, making all ports appear closed.

These scans are primarily used for **firewall evasion**, though modern IDS can often detect them.

Task 9 – ICMP Network Scanning (Ping Sweep):

When approaching a target network in a black box assessment, one of the initial objectives is to determine which IP addresses are active. This process is known as network mapping and can be performed using **ICMP Network Scanning** in Nmap.

A **ping sweep** sends ICMP echo requests to multiple IP addresses within a specified range. Any host that responds is identified as alive. Although this method is not always accurate due to firewalls or ICMP blocking, it serves as a useful baseline for reconnaissance.

In Nmap, a ping sweep is executed with the `-sn` switch, which disables port scanning and focuses on host discovery. The tool uses ICMP echo requests and, depending on privileges, may send ARP requests on local networks. Additionally, Nmap sends TCP SYN packets to port 443 and TCP ACK (or SYN) packets to port 80 to help identify active systems.

For scanning the entire **172.16.x.x** network with a **/16 netmask**, the CIDR notation command is:

nginx

CopyEdit

```
nmap -sn 172.16.0.0/16
```

This command checks all possible IP addresses from **172.16.0.0** to **172.16.255.255**, marking any that respond as active hosts.

Task 10 – Nmap Scripting Engine (NSE) Overview:

The **Nmap Scripting Engine (NSE)** significantly enhances Nmap's capabilities by allowing automation of advanced scanning, enumeration, and even exploitation tasks. NSE scripts are written in the **Lua** programming

language and can be leveraged for purposes ranging from vulnerability detection to brute force attacks.

The NSE library contains multiple script categories, each serving different objectives:

- **safe** – Non-intrusive scripts that do not negatively affect the target.
- **intrusive** – Potentially disruptive scripts, unsuitable for production environments.
- **vuln** – Designed to identify vulnerabilities in the target system.
- **exploit** – Attempts to actively exploit detected vulnerabilities.
- **auth** – Bypass or test authentication mechanisms.
- **brute** – Perform brute-force attacks to uncover credentials.
- **discovery** – Gather additional information about the network or services.

While NSE is extremely powerful for reconnaissance and security testing, the choice of script category must match the engagement scope and rules of engagement. For example, **intrusive** scripts should be avoided in live production networks due to the risk of disruption.

Key Points:

- NSE scripts are written in **Lua**.
- **Intrusive** scripts can negatively impact a target and should be avoided in production environments.

Task 11 – Working with the NSE (Report Overview):

In **Task 11**, I learned how to work with the Nmap Scripting Engine (NSE), particularly focusing on the `ftp-anon.nse` script, which checks for anonymous login access to an FTP server.

Key Insight: The `ftp-anon.nse` script accepts an optional argument:

CopyEdit

`ftp-anon.maxlist`

- This parameter controls the maximum number of files to return in the directory listing. By default, the limit is set to 20 entries, but this can be adjusted—or disabled entirely using a value of `0`, or unlimited listing when verbosity is enabled.

Task 12 – Missing Dependency Issue in Node.js Backend:

During the execution of the backend project using the `npm start` command, the application failed to run due to a missing dependency error. The terminal output indicated:

javascript

CopyEdit

```
Error [ERR_MODULE_NOT_FOUND]: Cannot find package
'express' imported from ...
```

This error occurs when the required Node.js module (`express`) is not installed or is missing from the `node_modules` folder. The issue can be resolved by running:

nginx

CopyEdit

```
npm install express
```

This ensures the dependency is added to the `node_modules` directory and listed in the `package.json` file. Once installed, re-running `npm start` should allow the backend server to start without errors.

Task 13 – Firewall Evasion Techniques in Nmap:

In this task, the focus was on identifying and bypassing firewall configurations that can interfere with host discovery and scanning. A common scenario encountered in Windows environments is the blocking of ICMP packets by default firewall settings. This prevents both manual `ping` operations and Nmap's default host discovery from succeeding, resulting in the target being marked as inactive.

To address this, Nmap's `-Pn` switch was highlighted, which bypasses the ICMP check and treats all specified hosts as alive, allowing scans to proceed. While this can effectively bypass ICMP restrictions, it may significantly increase scan time for inactive hosts.

Additional firewall evasion options were also covered, including:

- **`-f`** – Packet fragmentation to evade detection by firewalls or intrusion detection systems.
- **`--mtu <number>`** – Custom maximum transmission unit size for finer packet control.
- **`--scan-delay <time>ms`** – Delay between packets to avoid triggering time-based firewall rules.
- **`--badsum`** – Generates packets with invalid checksums to elicit responses from firewalls.
- **`--data-length <number>`** – Appends arbitrary random data to packets for obfuscation purposes.

These techniques collectively enhance the ability to bypass network defenses and gather information even in restricted environments.

Task 14 – Practical Nmap Scanning and Analysis:

In this practical exercise, various Nmap scanning techniques were applied to the target machine to validate concepts covered in previous tasks. Initial reconnaissance determined that the host did not respond to ICMP echo requests, indicating either ICMP blocking or filtering at the firewall level.

An **Xmas scan** was then conducted across the first 999 ports, revealing that all scanned ports were classified as **open|filtered**. The scan output explained that this status occurs when no response is received, preventing Nmap from differentiating between open and filtered states.

A **TCP SYN scan** over the first 5000 ports identified **five open ports**, providing a clearer picture of available services. A follow-up **TCP Connect scan** against port 80 was monitored in Wireshark to observe the full three-way handshake process in action.

Finally, the **ftp-anon** NSE script was executed against port 21. The script confirmed that anonymous FTP login was possible, granting access to the FTP service without authentication.

These results demonstrate the combined use of host discovery, stealth scanning, and targeted script execution to effectively profile a host's network exposure even when traditional probing methods, such as ICMP pings, are blocked.

Task 15 – Conclusion:

The Further Nmap module concluded with an emphasis on the importance of continuous learning and referencing official documentation. While the tasks covered advanced scanning techniques, firewall evasion, NSE scripting, and practical enumeration, the room highlighted that Nmap's capabilities extend far beyond what was explored here.

The official Nmap documentation remains the most comprehensive reference, offering detailed explanations, usage examples, and an up-to-date script library. While memorizing the documentation verbatim is unnecessary, familiarity with its structure ensures rapid access to relevant commands and options when tackling real-world scenarios.

Through the combined application of theoretical knowledge and hands-on exercises, this module reinforced the utility of Nmap as an essential tool for reconnaissance, vulnerability discovery, and targeted service interaction during penetration testing.