MODULE 2

ETHICAL HACKING FOOTPRINTING AND RECONNAISSANCE, SCANNING AND ENUMERATION

Foot printing and Reconnaissance: Technical Requirements, Web Searches and Google Hacks, WHOIS Database Records.
Scanning and Enumeration: Exploring Scanning Techniques, Understanding Service Enumeration, Introducing the Nmap Network Scanning Tool.

1.1 ETHICAL HACKING FOOTPRINTING AND RECONNAISSANCE

Hacking computer networks and systems does not start with a click of a mouse in an application to gain access to a target. Although this can occur, it tends to be more opportunistic and not part of a well-thought-out plan. Each phase of the journey uncovers useful information that the attacker can use to not only break in but to stay hidden. But to get there, they need information.

This is where the first phase of most attacks begins, with information gathering and intelligence, often called **footprinting**. In this stage, we passively gain information about a target. What that means is we gather information without the knowledge of the target. This is done primarily through what is called **Open Source Intelligence** (**OSINT**) starting with the simplest of tools, Google. If done correctly, not only will general information be gained about the target but it can overturn potential vulnerabilities and weaknesses. **Reconnaissance** can be employed after this, which involves more obtrusive measures, such as active scanning.

People and organizations generate massive amounts of information—public records, social posts, job listings, leaked datasets, DNS entries, and more—and much of it accumulates online where it can be found, indexed, and correlated. In this chapter we'll introduce practical techniques for harvesting that data (both passively, by searching public sources and archives, and actively, by probing services and registries), then show how to clean, normalize and enrich the raw results so they become useful intelligence rather than noise.

You'll learn how to convert many scattered facts into a coherent profile: who runs a service, what software versions are exposed, where sensitive assets live, and which human targets (admins, developers) might be susceptible to social-engineering approaches.

We'll also cover how defenders use the same toolset defensively: threat modeling their own internet-facing footprint, detecting accidental data exposure, prioritizing patching where public evidence shows real risk, and simulating attacker reconnaissance to close the information gaps attackers exploit. The goal is practical — not voyeuristic — turning vast public data into actionable, privacy-respecting insight that strengthens security posture.

What is footprinting and reconnaissance?

Footprinting is the gathering and recording of information about a target or targets. The information gathered includes but is not limited to names, addresses, phone numbers, business partners, products used, and personal connections, to name a few. Footprinting is primarily a passive task as opposed to reconnaissance, which is more active in nature. **Reconnaissance**, sometimes known as **active footprinting**, is the active scanning and probing of targets. We will get into more detail about scanning in the next chapter. In the meantime, let's continue to look at what information is being gathered.

Once a target has been established, it is time to gather information, and we start with the searching the Internet. As a research tool, the internet knows no bounds, offering limitless information about any topic. However, with this much information, we can be quickly overwhelmed; how do we find the information we need and how do we know the information is relevant and timely? Some of these questions can answered while searching and others will require a little analysis to make the determination. So, let's get started.

Keeping inventory

When gathering information, you will need a place to keep track of it, some form of **inventory**. This not only keeps you from repeating steps but helps to classify what is relevant. Being disorganized will slow the process down and lead to less success. You can use any method or tool to keep track of your information. Spreadsheets are the most common method as they are very flexible and can be adapted to support multiple parts of the overall process. The sheets with tabs can cover items such as names, dates, and links, while other tabs may be adapted to support some of the scanning information later, such as IP address, operating system, and login accounts.

Starting with web searches and ending with footprinting applications, we will be looking at gathering as much information about the target before attempting any of the other steps, including exploitation. It is this initial detective work that makes hackers and pen testers so effective at what they do. Because they have gathered and documented all the information, they are able to quickly pivot and find multiple ways to infiltrate and exploit the target. In many cases, the attackers know more about an organization and how it operates than the internal personnel. Let's get started exploring the tools and techniques of footprinting and reconnaissance.

1.2 WEB SEARCHES AND GOOGLE HACKS

The first tools we're discussing in this chapter are **web searches** and **Google hacks**. You might ask why start with searches? Why not go directly to the website and begin there? While it's tempting to go right where you think the primary information is, this is also one of the first lines of defense for the organization. Because they own the website, they will get information about who is connecting, when they are connecting, what pages they are spending time on, and so on. This information is primarily used by marketing to gather metrics about their brand, focus, and public message. The security teams also use this information to see the same things but from a security perspective. They are interested in what countries requests are coming from, what browsers are being used, and whether there is anything unusual about the requests coming in. We will discuss ways around this later in the chapter.

In the meantime, we are going to focus on the search. There are many search engines available for your use, however, **Google** is the most dominant, and we will focus on its use for gathering information.

When you go on to Google, you are presented with a generic search bar where you can enter what you are looking for and proceed; however, Google has advanced search functions, and if you are aware of them, you can get more focused information than you otherwise would get. These extra functions are referred to as follows:

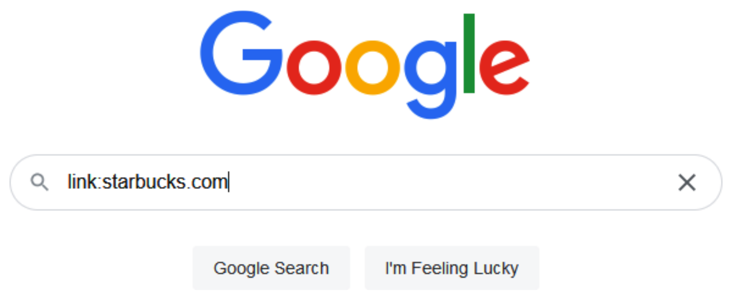
- **Directives**: Google directives are searches where keywords are introduced to the query to change the nature of the search. Examples include Jaguar -car to filter out cars, or site:wikipedia.org arp protocol
- **Google Hacks**: Google Hacks use specific search techniques or search order operations to uncover information.
- **Google Dorks**: Google Dork is a hacker term that refers to using *Google Advanced Search* or a set of search operators to find security holes and information leaks in websites.

This idea of using advanced search functions to find security holes began around 2002 with Johnny Long, who collected Google searches that uncovered vulnerabilities and sensitive information. He labeled these *Google dorks* and stored them in a database called the **Google Hacking Database.** This database is still maintained and can be found at https://www.exploit-db.com/google- hacking-database.

Before we explore Google dorks, we are going to look at Google's advanced search functions, as these are the underpinnings of Google dorks.

Exploring some useful Google hacks

Not including the standard *and*, *or*, and *not*, Google has approximately 38 advanced search functions that you can access by entering them into the Google search window, as shown in the following screenshot:



Example Using a Google dork search function

We'll focus on the most useful functions aka Directives, for information gathering on a target. An item to note when doing searches and the use of quotation marks: in search vernacular, the use of quotation marks means exact or whole phase match, without which the search engine breaks the phrase into separate terms, and the return results may not be relevant to what you are looking for. Let's look at some of these Directives:

- **link**: This finds sites that link to the specified domain. For example, a link can be link:starbucks.com. You can also search for deep links if the path is known or common – for example, link:my-site.com/phpmyadmin.
- **Numrange**: Finds a range of numbers in a query up to 5 digits. At one time, this was considered one of the most dangerous searches. It could be used to harvest phone numbers and credit cards. It still works but limitations have been placed on it.
- **<number>..<number>**: Same as the *Numrange* search without operator – for example, hack 2015..2020.
- **site**: Show your searched term within a specific site – for example, site:darkreading. com Netwire.
- **intitle** and **allintitle**: Shows results with the searched phrase in the title tag of a page. For example, Intitle "Index of" "backup files" returns results with Index of and/or backup files in the title. An example of allintitle index of/admin returns results with Index of and admin in the title tag.
- **allintext**: Locates strings within web pages. An example is allintext: "Index of" "sftp-config.json".
- **inurl and allinurl**: Shows results with your search term in the URL. An example is inurl / intranet/login.php or allinurl /admin/login.php.
- **related**: Shows results that are related to your searched URL, such as related:cnn.com. This information is posted at the bottom of Google results after a search.
- **Info**: Shows information about any searched domain, such as Info:starbucks.com.
- **filetype**: Find documents of the specified type – for example, filetype:pdf policy. This

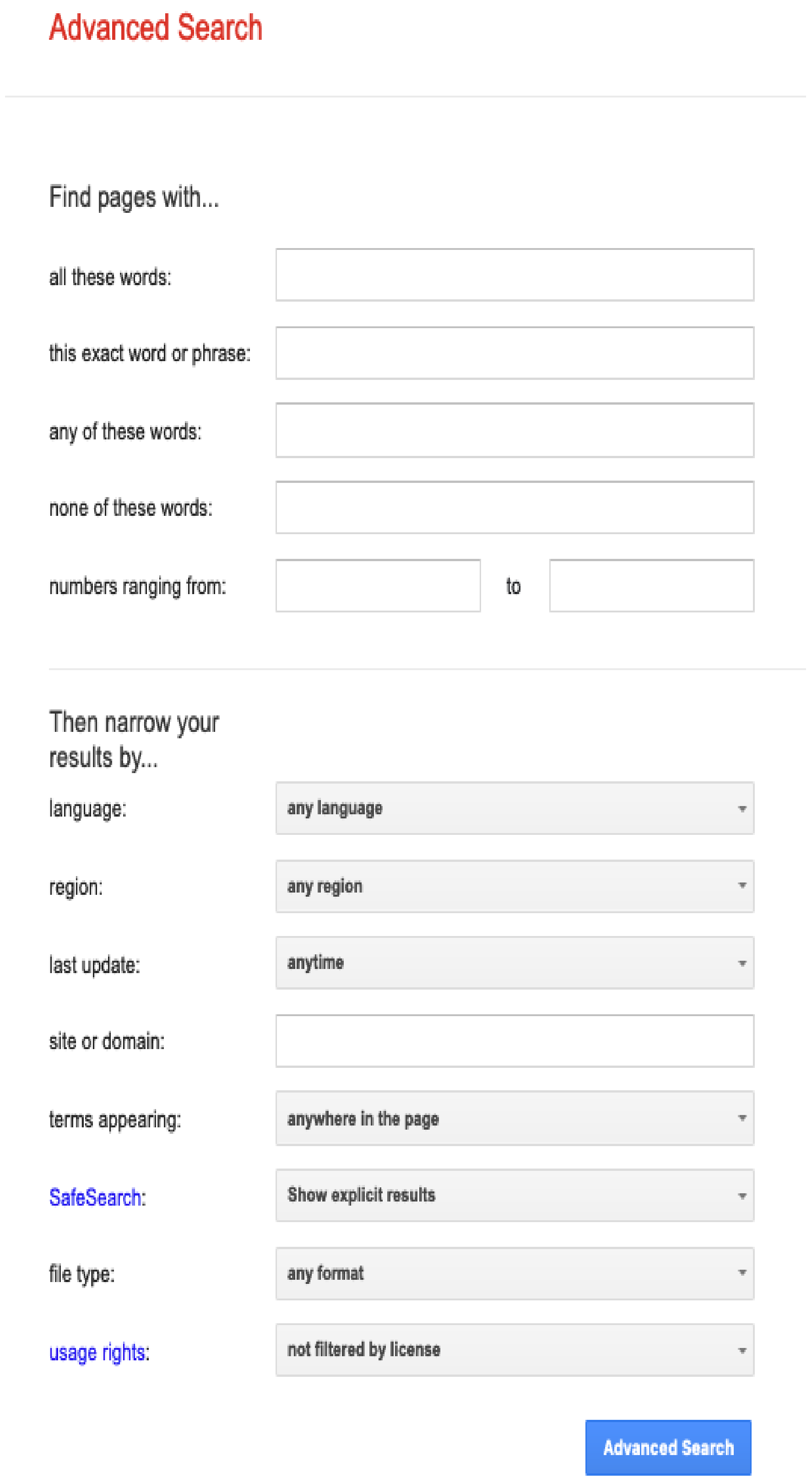
will return PDF documents with the word policy in the title.

- **ext**: Very similar to filetype but we can seek uncommon extensions for more accurate results. An example is ext:log.
- **cache**: Can show cached contents – for example, cache:https://starbucks.com.

Search functions can be joined together to make more complex and focused searches. Let's take a look at a few examples:

- **Finding usernames**: allintext:username filetype:log
- **Finding email addresses**: allintext:email OR mail +*gmail.com filetype:txt
- **Finding SSH private keys**: intitle:index.of id_rsa -id_rsa.pub
- **Finding Putty logs**: filetype:log username putty

These are just a couple of examples of what you can put together. If you can't remember the functions, Google does expose some of these through **Advanced Search**:



Advanced Search

Find pages with...

all these words:	
this exact word or phrase:	
any of these words:	
none of these words:	
numbers ranging from:	to

Then narrow your results by...

language:	any language
region:	any region
last update:	anytime
site or domain:	
terms appearing:	anywhere in the page
SafeSearch:	Show explicit results
file type:	any format
usage rights:	not filtered by license

Google Advanced Search page

This page can be accessed from Google's main search screen by clicking on **Settings** in the lower-right corner.

While it does not expose all the search functions, it does provide a good starting place to begin searching. Now that you know how attackers might be using Google searches to find and exploit information, you might want to know how to stop it, or at least reduce your exposure.

Preventing exploitation through Google searches

There are some things you can do as a defender to make the use of crafted Google searches a little less effective with the company data you are entrusted to protect. The first place to start with is a file called robots.txt. Any web server or service exposed to the internet needs to have this file at the root of the website. The robots.txt file instructs search bots what content on your site(s) it can index. Without this file, the bots will index whatever they encounter. An example of using robots. txt to prevent the indexing of an entire site is as follows:

```
User-agent: *
Disallow /
```

While this is an extreme example, it shows how the robots.txt file can help with securing

```
# robots.txt for https://www.homedepot.com/
User-agent: *
Disallow: /*SiteMapView*
Disallow: /*Navigation?Ns=P_Topseller_Sort|style=List*
Disallow: /*Navigation?Ns=P_Topseller_Sort|style=A*
Disallow: /*CheckoutSignIn*
Disallow: /*OrderItemUpdate*
Disallow: /*THDLogon*
Disallow: /*ShippingMethod*
Disallow: /*OrderItemDisplayViewShiptoAssoc*
Disallow: /*DeliveryCalendar*
Disallow: /*OrderItemAdd*
Disallow: /*MoreViewsPage*
Disallow: /*Search?*
Disallow: /*NCNI-5*
Disallow: /*recordCompareList*
Disallow: /*PLP_Overlay*
Disallow: /*Ntt-*
Disallow: /s/
Disallow: /p/qv/
Disallow: /*OnlineAccount*
```

your website. Here is an excerpt of a robots.txt file from homedepot.com/robots.txt: showing how they broke down what they wanted indexed and what they didn't:

```
Disallow: /*mycart*
Disallow: /*mycheckout*
Disallow: /*usebeta*
Disallow: /canvas/
Disallow: /auth/view/signin?redirect=*
Disallow: /auth/view/createaccount?redirect=*
Disallow: /clickstream-producer/v1/publish
Disallow: /l/search/*
Disallow: /l/detail/*
Disallow: /b/Featured-Products*
Disallow: /compare?*

Sitemap: https://www.homedepot.com/sitemap/d/desktop_sitemap.xml
```

This robots.txt file tells all search bots that they can index the Home Depot website except for certain specified directories. The primary purpose of such exclusions is usually to prevent search engines from indexing areas of a site that contain repetitive, unintelligible, or non-relevant content, which would not add any real value to search results. These might include administrative folders, scripts, temporary directories, or dynamically generated content that could confuse indexing algorithms. From a business and performance perspective, this makes sense—it keeps the website's public presence clean, searchable, and optimized. However, from a security standpoint, this same file can unintentionally act as a roadmap for attackers. Because robots.txt is publicly accessible, it effectively lists out the areas that the site owner does not want indexed. To a malicious actor, these disallowed directories may appear suspicious or worth investigating, as they could potentially lead to restricted or sensitive sections of the site such as admin panels, staging environments, or backup folders. In this way, something designed for search-engine optimization can inadvertently provide reconnaissance clues to those with bad intentions.

To mitigate such exposure, organizations must go beyond the surface level and strengthen their overall security posture. One of the simplest yet most crucial steps is to ensure that the host operating system, running services, and web applications are consistently patched and updated. Unpatched software can contain vulnerabilities that automated bots and scanners can easily exploit. In fact, many bots are programmed to continuously crawl the internet, looking for outdated or misconfigured applications, and to catalog these findings into databases that malicious users later reference when searching for potential targets.

In addition to keeping systems updated, organizations should implement multiple layers of defense. This includes deploying antivirus software to detect and neutralize malware, firewalls to control incoming and outgoing traffic, and Intrusion Detection Systems (IDS) to monitor network behavior for suspicious activity. A Security Information and Event Management (SIEM) system can further strengthen defense by aggregating logs from various sources, identifying anomalies, and alerting administrators to possible security breaches in real time. These tools, when properly configured and monitored, create a security ecosystem that records access attempts, blocks known types of exploits, and provides visibility into what's happening across the infrastructure. Regular security audits are also vital for assessing how much information about an organization is publicly visible. By using the same open-source intelligence (OSINT) methods that attackers rely on—such as advanced Google searches (Google Dorking) and metadata analysis—security teams can identify unintentional data leaks, exposed services, or outdated information before they are discovered by malicious actors.

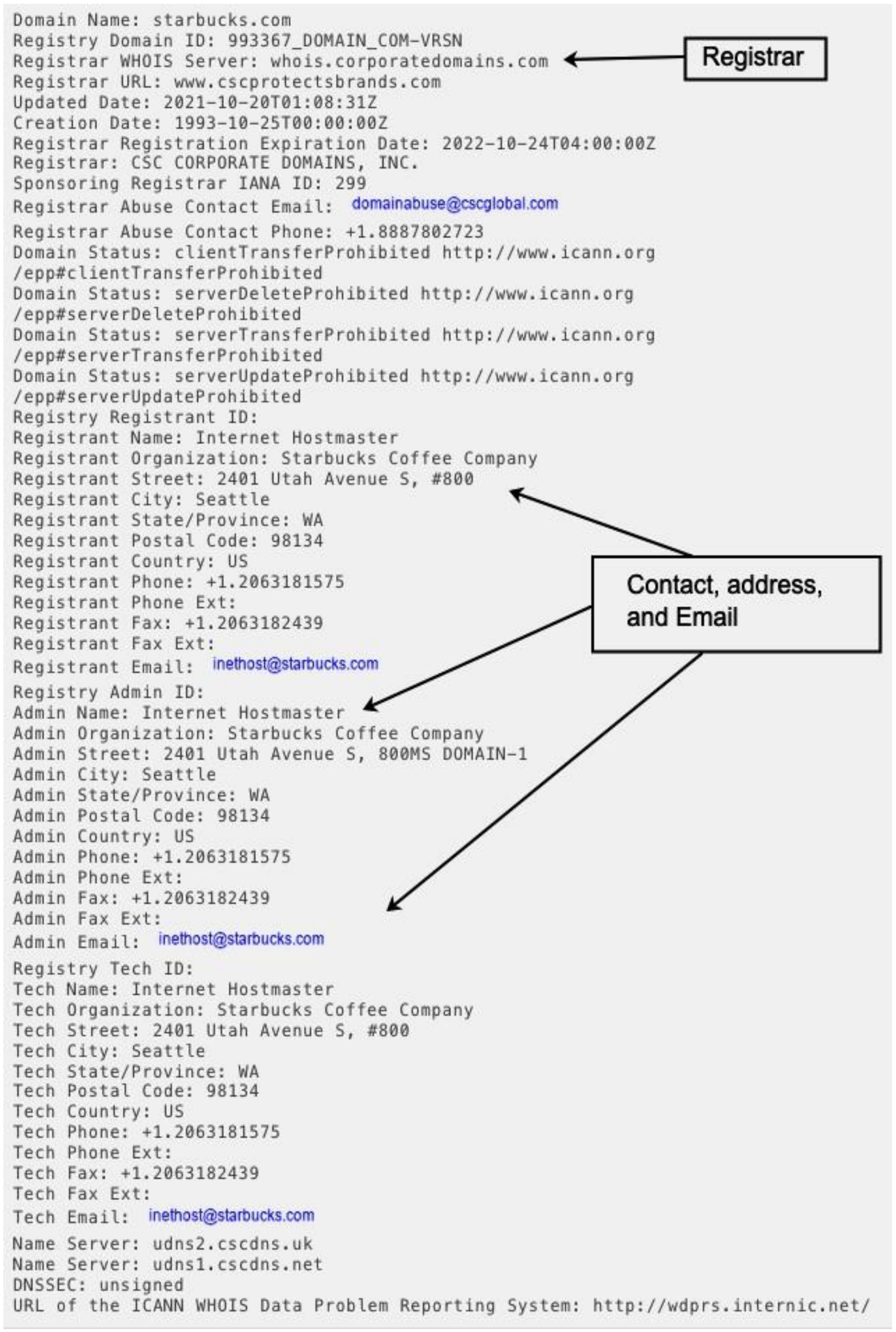
Furthermore, organizations should periodically engage professional penetration testers to conduct simulated attacks under controlled conditions. A penetration test (pen test) mimics the mindset and methods of a real attacker, uncovering weak points that internal teams might overlook. Pen testers use reconnaissance techniques, network scanning, and exploitation attempts similar to those used by actual adversaries, but they do so ethically and with permission. The resulting report gives the organization a detailed understanding of where its defenses are strong and where they need improvement. A well-conducted pen test can reveal information that is leaking unintentionally—from exposed system banners and outdated certificates to misconfigured cloud storage and forgotten subdomains. It allows the organization to patch gaps before a real attacker can exploit them. Essentially, pen testing transforms the hacker's toolkit into a protective instrument, enabling defenders to stay one step ahead.

1.3 WHOIS DATABASE RECORDS

The **WHOIS** database is a database repository of information pertaining to domain registrations on the internet. It contains details about domain owners, their contact information, registration and expiration dates, and the domain's associated name servers. This database is maintained by domain registrars and overseen by international governing bodies such as the **Internet Corporation for Assigned Names and Numbers** (**ICANN**). While WHOIS data has historically been publicly accessible, evolving privacy concerns have led to changes where some domain registrars allow for the protection of the registry information of domain owners by offering privacy protection, where the data either isn't available or the provider uses their information instead of the actual registrant. There is still a great deal of information available.

To gather more detailed information about a domain, we can look it up using lookup services or directly access one of several WHOIS databases distributed around the world. When a domain (be it a website or service) is exposed to the internet by name, it has to be registered by a registrar. The registrar gathers detailed information about the domain and people associated with it. The registrar then contacts a **Regional Internet Registry** (**RIR**), which maintains the WHOIS database to make the update. So, *what exactly is in the WHOIS database and why is it good for reconnaissance*?

A WHOIS record contains information about the registrar and IP addresses, when it was created and is due for an update, as well as administrative contacts and name servers. More on this later. Let's take a look at an actual WHOIS record:



```
Domain Name: starbucks.com
Registry Domain ID: 993367_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.corporatedomains.com
Registrar URL: www.cscprotectsbrands.com
Updated Date: 2021-10-20T01:08:31Z
Creation Date: 1993-10-25T00:00:00Z
Registrar Registration Expiration Date: 2022-10-24T04:00:00Z
Registrar: CSC CORPORATE DOMAINS, INC.
Sponsoring Registrar IANA ID: 299
Registrar Abuse Contact Email: domainabuse@cscglobal.com
Registrar Abuse Contact Phone: +1.8887802723
Domain Status: clientTransferProhibited http://www.icann.org
/epp#clientTransferProhibited
Domain Status: serverDeleteProhibited http://www.icann.org
/epp#serverDeleteProhibited
Domain Status: serverTransferProhibited http://www.icann.org
/epp#serverTransferProhibited
Domain Status: serverUpdateProhibited http://www.icann.org
/epp#serverUpdateProhibited
Registry Registrant ID:
Registrant Name: Internet Hostmaster
Registrant Organization: Starbucks Coffee Company
Registrant Street: 2401 Utah Avenue S, #800
Registrant City: Seattle
Registrant State/Province: WA
Registrant Postal Code: 98134
Registrant Country: US
Registrant Phone: +1.2063181575
Registrant Phone Ext:
Registrant Fax: +1.2063182439
Registrant Fax Ext:
Registrant Email: inethost@starbucks.com
Registry Admin ID:
Admin Name: Internet Hostmaster
Admin Organization: Starbucks Coffee Company
Admin Street: 2401 Utah Avenue S, 800MS DOMAIN-1
Admin City: Seattle
Admin State/Province: WA
Admin Postal Code: 98134
Admin Country: US
Admin Phone: +1.2063181575
Admin Phone Ext:
Admin Fax: +1.2063182439
Admin Fax Ext:
Admin Email: inethost@starbucks.com
Registry Tech ID:
Tech Name: Internet Hostmaster
Tech Organization: Starbucks Coffee Company
Tech Street: 2401 Utah Avenue S, #800
Tech City: Seattle
Tech State/Province: WA
Tech Postal Code: 98134
Tech Country: US
Tech Phone: +1.2063181575
Tech Phone Ext:
Tech Fax: +1.2063182439
Tech Fax Ext:
Tech Email: inethost@starbucks.com
Name Server: udns2.cscdns.uk
Name Server: udns1.cscdns.net
DNSSEC: unsigned
URL of the ICANN WHOIS Data Problem Reporting System: http://wdprs.internic.net/
```

Example WHOIS information for Starbucks.com

In the preceding example, we see the registrant information for starbucks.com. This includes the company address, phone number, and email. Additionally, it contains information about when the domain was registered and when its expiration date is. In some cases, the information contained here provides insights that might not be found anywhere else, such as an alternative email address or phone number that isn't published on the official website. Attackers may be able to use this information for social engineering or other search parameters to add to their footprinting campaign.

Now that you have seen a WHOIS record, the question comes to mind of how you access this information.

Accessing WHOIS information

For a WHOIS query or domain lookup, we primarily go to one of many third-party brokers, such as WHOIS.net, lookup.icann.org, and domaintools.com. They will provide basic WHOIS information for collection; however, DomainTools provides better, more robust information that the others do not. They also offer a paid subscription where you can get even more information such as what the website looked like at different times, reverse lookups, and monitoring tools. You can also go directly to the RIRs. There are five RIRs divided into regional coverage of the globe, as follows:

- **American Registry for Internet Numbers** (**ARIN**), which is found at https://www.arin. net/: This RIR covers North America including the US, Canada, and the Caribbean islands
- **The Réseaux IP Européens Network Coordination Centre** (**RIPE**), which can be found at

https://www.ripe.net/: This RIR covers Europe, the Middle East, and parts of Central Asia

- **The Asian Pacific Network Information Centre** (**APNIC**), which you can find at https:// www.apnic.net/: This RIR covers the Asia-Pacific region
- **The Latin American and Caribbean Internet Address Registry** (**LACNIC**), which is found at

https://www.lacnic.net/: This RIR covers Latin America and most of the Caribbean

- **AfriNIC**, which is at https://afrinic.net/: This RIR covers all of Africa

Once you have reviewed the WHOIS record, you may now have a part of your targeting inventory with names, addresses, IP addresses, and name servers. All that information is straightforward except name servers. So, what are name servers and why are they important?

Understanding the name server entry

Nameservers are the DNS servers that hold the records for the domain you are targeting. Besides just knowing what servers hold the records, you can also derive some of the underlying services driving the website such as a service provider (for example, **AT&T** or **Amazon Web Services** (**AWS**)) or even **self-hosting**.

Here is an example excerpt from WHOIS records showing AWS and AT&T service providers:

- **AWS**: The following record shows they are using AWS as a service provider:

```
Name Server: ns-1086.awsdns-07.org
Name Server: ns-1630.awsdns-11.co.uk
Name Server: ns-47.awsdns-05.com
Name Server: ns-576.awsdns-08.net
```

Example AWS name servers

- **AT&T**: Here is an example excerpt from a WHOIS record showing they are using AT&T as a service provider. You'll notice the number of name servers is larger:

```
Name Server: ns5.attdns.net
Name Server: ns3.attdns.com
Name Server: ns4.attdns.com
Name Server: ns6.attdns.net
Name Server: ns2.attdns.com
Name Server: ns1.attdns.com
```

Example AT&T name servers

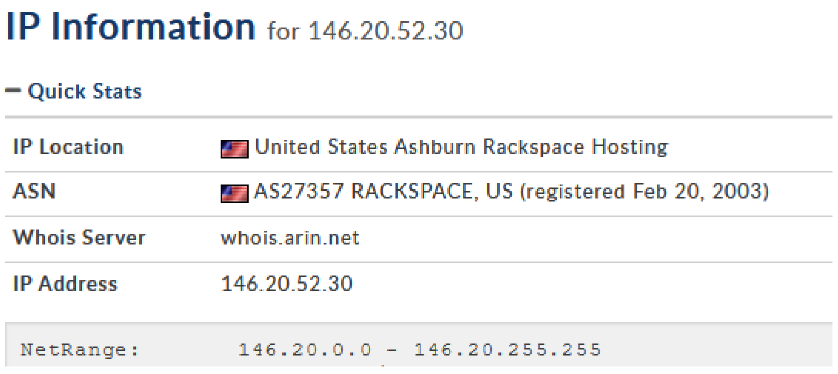
This could mean more services are being hosted and or there is load balancing technology in place.

- **Self-hosting**: The following is an example excerpt from a WHOIS record showing they are likely self-hosting. The entry references a single domain, and there are only two name servers, which is the minimum recommended number for failover and redundancy:

```
Name Server: GIDN1.FPL.COM
Name Server: LCDN1.FPL.COM
```

Example of possible self-hosting name servers

Now that you have investigated a WHOIS record, you can use the IP addresses you gained to geolocate the server and services using the same tools. In the next figure, we are using DomainTools (research. domaintools.com) to pull the geolocation of one of the name servers we discovered by IP address, as shown in the following screenshot:



IP Information for 146.20.52.30

– Quick Stats

IP Location	United States Ashburn Rackspace Hosting
ASN	AS27357 RACKSPACE, US (registered Feb 20, 2003)
Whois Server	whois.arin.net
IP Address	146.20.52.30

```
NetRange:        146.20.0.0 - 146.20.255.255
```

Example geolocation by IP address with DomainTools

WHOIS and DomainTools are not the only sources of information. There are many others available, some of which are government information sources while others are blogs and social media posts. Now, let's look at some of those sources.

Third-party sources of intel

Depending on the size of the target and whether they are a publicly traded company, there may be more places or types of information you can look for and gather. One place to start, at least for publicly traded companies, is with their financial statements. In the US, companies are required to file reports about the financial health of the company. In these reports, they disclose other types of information, including company direction as well as executive staff and board members. You can find these reports in the **Electronic Data Gathering, Analysis, and Retrieval** (**EDGAR**) database on the *US Securities and Exchange Commission* website at https://www.sec.gov/edgar/searchedgar/ companysearch.html. The report that usually contains the most information is the company's annual report, also known as the **10-K**. Let's take a look at some other areas for collecting intelligence.

Sources for collecting intelligence

Collecting intelligence through online sources involves the systematic gathering and analysis of information from various digital channels, including websites, forums, news articles, and databases. This process aims to extract valuable insights, patterns, and trends to inform decision-making across diverse fields such as cybersecurity, business strategy, or governmental policy. One note of caution: validate the information collected, as it could be out of date or just wrong. Let's look at some common online sources of this information.

1.4 ETHICAL HACKING SCANNING AND ENUMERATION

After the footprinting phase, scanning and enumeration mark a transition from broad visibility-gathering to methodical interrogation. Footprinting gives you the lay of the land — domain names, public IP ranges, software versions hinted at by public pages, and people who might be targets for social engineering — but it rarely tells you what actually responds, how it responds, or what lives behind firewalls and NAT. Scanning begins to answer those questions by actively or passively probing the target's infrastructure to determine which hosts are alive, which ports accept connections, and which protocols speak a recognizable language. Enumeration then takes the raw results of scanning and turns them into structured lists: services and their versions, user and share names, application endpoints, and other discrete assets that can be reasoned about, correlated, and prioritized. Importantly, this phase is iterative and cyclical: new intelligence discovered during enumeration often expands the scope and focus of further scans, and a successful breach of a perimeter system can turn an external scan into an internal discovery exercise where previously unreachable hosts and services are enumerated for lateral-movement potential. The dual goals here are pragmatic (find the real attack surface) and analytic (turn noisy probes into reliable intelligence), and both must be pursued with careful timing, tool selection, and interpretation to avoid misleading results or unintended disruption.

Technically, scanning divides into passive and active approaches, and each has trade-offs. Passive scanning — collecting information without directly interacting with a host in a way that generates new traffic — includes parsing public records, indexing honeypots or internet-wide scans, and monitoring public telemetry feeds. It is stealthy, low-risk, and often the first choice when you don't want to alert defenders or breach legal authorization boundaries. Active scanning, by contrast, sends crafted packets to solicit responses; it includes ping sweeps, TCP SYN/ACK handshakes, UDP probes, banner grabbing, and more advanced fingerprinting techniques that infer OS and service versions from timing, packet minutiae, and protocol quirks. The classic tools (Nmap, masscan, ZMap) exemplify different philosophies: Nmap favors accuracy and flexible discovery scripts, masscan and ZMap emphasize speed for Internet-wide sweeps. Effective scanners combine several methods — for example, a TCP SYN scan to find open ports, followed by targeted probes (HTTP HEAD, TLS handshake, or SMTP EHLO) to collect banners and version strings. That gathered data must then be interpreted carefully: banner strings can be misleading, services may be fronted by proxies or load balancers, and advanced defensive devices intentionally obfuscate or alter replies to disrupt fingerprinting. Timeouts, retry logic, and rate-limiting become practical concerns because aggressive scans can trigger IDS/IPS rules or inadvertently cause denial-of-service on fragile endpoints; conversely, overly conservative timing can miss transient services. Enumeration extends this by iterating lists — host lists, port lists, user lists — and applying application-specific queries (SMB null sessions, SNMP community string probes, LDAP queries, database banner examinations) to pull out structured information: usernames, shared folders, API endpoints, or certificate chains. Skilled operators combine automated enumeration scripts with manual inspection to validate noisy findings, weed out false positives, and identify the truly exploitable surface.

It helps to remember, in the middle of all these mechanics, what the scanning and enumeration phase is trying to deliver: a prioritized, actionable map of the target environment. With that in mind, the specific objectives of network scanning are worth restating and then unpacking, because they form the checklist against which you should judge both coverage and risk:

- To identify hosts on a network
- To identify open and closed ports on each host
- To identify the operating systems place – that is, Windows, Linux, or Mac
- To identify the services running on a network
- To identify the processes running on a network
- To identify the presence of security systems such as firewalls and antivirus
- To identify system and network architecture
- To identify vulnerabilities

Each item on this list maps to different techniques and outputs. Identifying hosts uses ARP sweeps on a LAN, ICMP and TCP SYN scans across routed networks, or passive DNS/NetFlow analysis from logs to infer asset activity. Discovering open and closed ports is the most literal form of scanning and provides the scaffolding for targeted enumeration: open ports tell you where to focus application-level probes (HTTP on 80/443, SSH on 22, RDP on 3389), while closed or filtered results reveal defensive presence (stateful firewalls or ingress ACLs). OS identification leverages fingerprinting methods that analyze TCP/IP stack behaviors, TTL values, and specific protocol signatures; accurate OS fingerprinting is critical because exploitation often targets OS-specific vulnerabilities. Service identification and process discovery are deeper: they use banner grabs, direct protocol negotiation, or authenticated queries to enumerate versions and running binaries; on internal networks, agent-based tools (EDR) or authenticated scans can reveal precise process lists and patch levels. Detecting security systems — firewalls, IDS/IPS, WAFs, endpoint AV — is both defensive reconnaissance and a safety check; understanding the protection landscape helps you avoid triggering alerts and informs how you test. System and network architecture (subnets, VLANs, domain structure, trust relationships) emerges from correlating routing information, traceroutes, DNS records, and LDAP or Active Directory enumeration. Finally, identifying vulnerabilities synthesizes all the above: version strings and configurations are matched against vulnerability databases, prioritized for exploitability (is the vulnerable service externally reachable? is authentication required?), and triaged by potential impact. Together, these goals make scanning and enumeration a discipline of both breadth and depth: broad to cover the attack surface and deep to convert surface hits into meaningful risks.

Execution at scale demands discipline, tooling, and governance; defensive teams as well as pentesters must harden their methodology and controls. From an operational perspective, automated scanners should be run with policies that respect business availability and legal boundaries: schedule high-noise scans for maintenance windows, use credentialed scans where possible to reduce false positives, and maintain rigorous logging so results are auditable. Enumeration should include normalization and enrichment steps (standardizing hostnames, resolving IP aliases, augmenting with asset ownership and criticality data) so findings are actionable for patch management and incident response teams. Post-scan workflows are equally important: verified vulnerabilities should feed into ticketing systems with remediation owners, compensating controls, and timelines; ambiguous results require manual validation or follow-up scans with altered probe techniques. Ethical and legal considerations must govern every scan — always operate with explicit authorization, a clear scope, and rules of engagement that prohibit disruptive tests (like DoS) unless explicitly approved. For defenders, repeated internal enumeration is part of threat-hunting and exposure management: continuous scanning helps surface shadow IT, forgotten services, and drift from hardened baselines. For offensive teams, the goal is not merely to catalog but to validate exploitability and demonstrate impact in a controlled manner so organizations can prioritize fixes. Ultimately, the power of scanning and enumeration lies not in the volume of data collected but in the quality of interpretation: turning noisy probe results into a prioritized roadmap for defense that reduces attack surface, speeds remediation, and improves resilience against real adversaries.

1.5 EXPLORING SCANNING TECHNIQUES

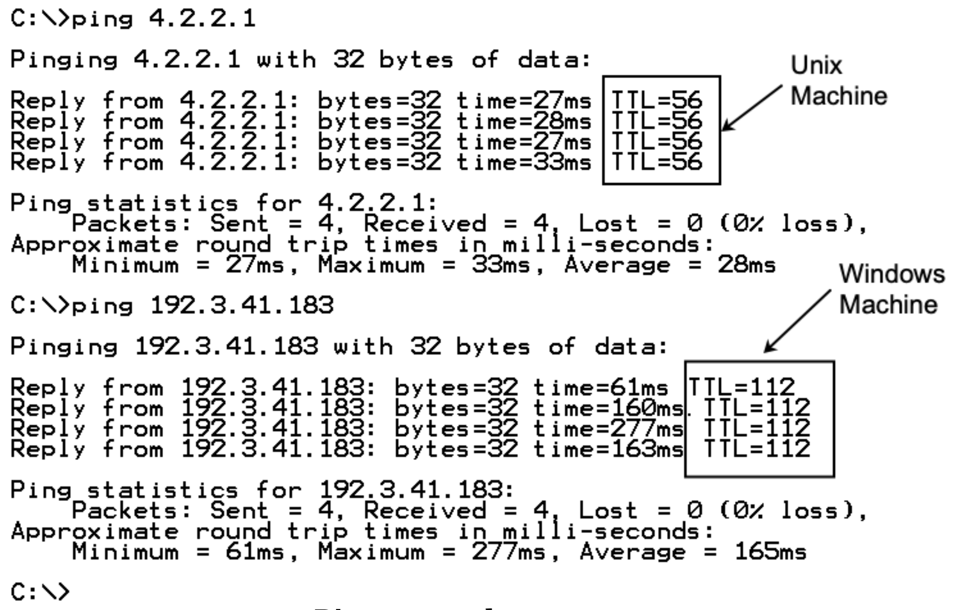
There are several scans and scan types that can be performed, including a port scan, a host scan, and even a vulnerability scan, but before we get into that, let's look at the most basic ways to determine whether a machine is on a network and reachable. One of the first ways to determine whether a machine is reachable is to use the built-in utilities of ping and Traceroute that are included with most operating systems.

Ping

Network engineers learned long ago they needed a way to determine whether remote systems were online. They created the **Internet Control Message Protocol** (**ICMP**) as a means to support network troubleshooting. At the core of the ICMP protocol is the utility called **Ping**. There are several functions of ping; however, the primary ones that most are familiar with are ICMP_ECHO_REQUEST, also known as **Type 8**, and ICMP_ECHO_REPLY, also known as **Type 0**. Ping is a very simple utility that can be run by using the ping command, followed by either an IP address or a domain name. An example of this would be either ping 192.168.100.54 or pinging a domain name such as ping cnn. com. This tells the machine to send an ICMP Echo Request (**Type 8**). Hopefully, the system at the other end is available and can respond with a reply, which would be an ICMP Echo Reply (**Type 0**).

The ping utility is quite useful and provides a great deal of information beyond just informing you that a machine is present and responding. For one, it provides the response time to diagnose whether there are communication problems on a network. The other interesting feature of ping is the last element, called **Time to Live** (**TTL**), which is the measurement of the number of hops or routers your machine traverses to get to the system at the other end. Each time the ping request traverses a router, the TTL number is decremented by one.

It is interesting to note that Microsoft and Unix systems implement this differently by using a different starting number. Unix machines start with a TTL of 64, and Microsoft Windows machines start with a TTL of 128. The following figure shows examples of a ping request and replies with different TTL values:



```
C:\>ping 4.2.2.1

Pinging 4.2.2.1 with 32 bytes of data:

Reply from 4.2.2.1: bytes=32 time=27ms TTL=56
Reply from 4.2.2.1: bytes=32 time=28ms TTL=56
Reply from 4.2.2.1: bytes=32 time=27ms TTL=56
Reply from 4.2.2.1: bytes=32 time=33ms TTL=56

Ping statistics for 4.2.2.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 27ms, Maximum = 33ms, Average = 28ms

C:\>ping 192.3.41.183

Pinging 192.3.41.183 with 32 bytes of data:

Reply from 192.3.41.183: bytes=32 time=61ms TTL=112
Reply from 192.3.41.183: bytes=32 time=160ms TTL=112
Reply from 192.3.41.183: bytes=32 time=277ms TTL=112
Reply from 192.3.41.183: bytes=32 time=163ms TTL=112

Ping statistics for 192.3.41.183:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 61ms, Maximum = 277ms, Average = 165ms

C:\>
```

Ping examples

Using the previous example, it can be determined that the IP address of 4.2.2.1 is a Unix machine and is approximately eight hops, or routers, away. Additionally, the IP address of 192.3.41.183 is a Microsoft Windows machine and is approximately 16 hops, or routers, away. The key takeaway here is by using a built-in utility, we were not only able to determine whether a system was responding but were also able to fingerprint the operating system at the other end.

Now for the bad news—this is a well-known technique for fingerprinting a system. In response, defenders have been implementing methods to prevent this from outside the network, making ping somewhat limited in its usefulness for reconnaissance. However, that does not mean everyone's defenses have been updated and that you should not try it. Additionally, ping is rarely prevented once you have established a foothold in a network and operate on the inside. Because of this, executing a ping at scale is possible; this is also known as a ping sweep, Let's take a look at this.

Ping at scale

Ping works great on a couple of systems, but how can you use it on a large network? It is time- consuming to ping every possible host address by running the ping command individually. Using an application or script to run the ping command at scale is often referred to as a **ping sweep**. This technique is used to identify which systems are available and which are not on a large scale. The ping sweep was first developed by chaining the ping command to a list of addresses; later, applications such as Nmap were developed to automate the process. A ping sweep, as the name implies, will ping a batch of devices and help an attacker determine which ones are active. Ping sweeps aid in network mapping by polling network blocks or lists of IP addresses or domains.

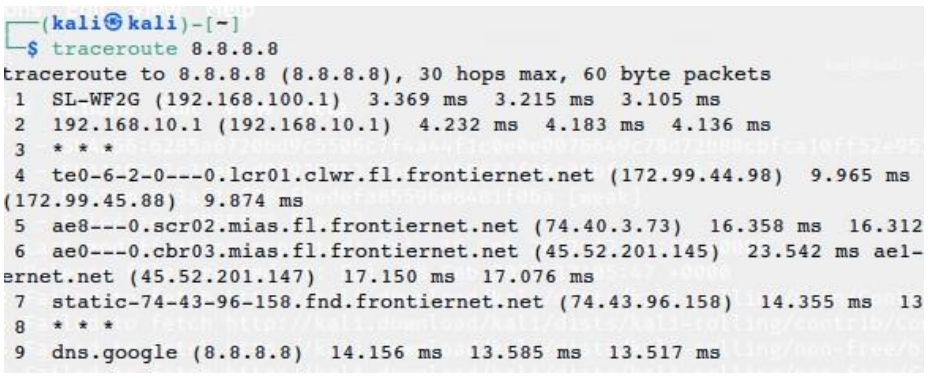
So, how do you scale a ping to perform a sweep? You could write your own script or utility that takes in a list pinging each system in the list. Alternatively, you could use one of the many tools available. Several vendors offer ping tools that provide various levels of functionality and extra features, such as ping sweep ability. Here are a few:

- **Solarwinds** (*commercial*): This offers network monitoring and ping sweeps to determine if systems are up or down
- **Advanced IP Scanner** (*free*): A Windows GUI tool designed to easily sweep networks
- **Fping** (*free*): A command-line tool for Linux to perform ping sweeps at scale
- **Angry IP Scanner** (*free*): A Windows GUI tool to perform quick ping sweeps and port scans
- **Nmap/Zenmap** (*free*): A Linux/Windows tool to perform ping sweeps, port scans, and so on

These are just a few of the tools out there to perform ping sweeps and more. We will focus primarily on the Nmap network scanning tool that is part of Kali Linux installation, and we will show you how to use it in the *Introducing the Nmap network scanning tool* section. First, let's look at the other built-in utility, Traceroute, and see how we can use it as part of our network scanning process.

Traceroute

Traceroute, or **Tracert** if you're using a Windows machine, is a utility designed to map out the routers between you and the target host. It does this by creating specially crafted ping packets in which the TTL value is increased by one until it gets to the target host. As previously discussed, for every router the ping packet goes through, the TTL is decremented by one. If the value of the TTL goes to zero, an error message is sent back to the host that initiated it. The utility then adds one more to the TTL value until it completes the process. Each time it traverses a router, the information is displayed on the screen:



```
┌──(kali㉿kali)-[~]
└─$ traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  SL-WF2G (192.168.100.1)  3.369 ms  3.215 ms  3.105 ms
 2  192.168.10.1 (192.168.10.1)  4.232 ms  4.183 ms  4.136 ms
 3  * * *
 4  te0-6-2-0---0.lcr01.clwr.fl.frontiernet.net (172.99.44.98)  9.965 ms
(172.99.45.88)  9.874 ms
 5  ae8---0.scr02.mias.fl.frontiernet.net (74.40.3.73)  16.358 ms  16.312
 6  ae0---0.cbr03.mias.fl.frontiernet.net (45.52.201.145)  23.542 ms ae1-
ernet.net (45.52.201.147)  17.150 ms  17.076 ms
 7  static-74-43-96-158.fnd.frontiernet.net (74.43.96.158)  14.355 ms  13
 8  * * *
 9  dns.google (8.8.8.8)  14.156 ms  13.585 ms  13.517 ms
```

Traceroute to Google's DNS server

In the preceding screenshot, we can see an example Traceroute to Google's DNS server at 8.8.8.8. The packet exits the internal network at 192.168.100.1 and traverses seven routers before reaching the target system. Note that two of the routers have * * * instead of names or IP addresses. This typically means that these routers have been configured not to return their information to the traceroute utility.

Now, you might be asking how this utility is useful to the attacker; it is actually useful in two ways:

- In the mapping of the perimeter, the last routers in a traceroute display will be part of the target network. In other words, we may be able to map out devices such as the external router and firewall between the perimeter and the target device.
- Once the attacker has gained a foothold on a network, especially a large network, they

can use this utility to map out the router infrastructure as part of the attack.

In summary, both ping and traceroute are highly useful tools for attackers and the defending ethical hacker. Firstly, this is because the utility is on every machine, meaning it doesn't have to be downloaded or installed. Secondly, the tools are used by administrators and technicians to troubleshoot a network as they were designed to do. If the network is not blocked internally, it likely will not be monitored in any significant way. With a little bit of time and patience, an attacker can map out an entire network using readily available tools, without alerting anyone to their presence. Now that we have discussed identifying hosts online and enumerating both hosts and routers, let's look at identifying open ports and services.

1.6 UNDERSTANDING SERVICE ENUMERATION

After finding out what systems are available and responding, the next step is to find out what services are available. Services such as email and web servers open ports for communication. The most straightforward way to find them is to perform a **port scan**.

Before we explore port scans, it is important to understand ports and how they work.

Introducing ports

There are a total of 65,535 TCP and 65,535 UDP ports available on any given system. It might seem that you would need to memorize all the ports; however, of these ports, the first 1024, also known as the **well-known ports**, are the ones commonly associated with specific services. An example of this is web servers, which commonly operate using port 80 on a system. A list of these port assignments can easily be found with a quick internet search; however, the **Internet Assigned Numbers Authority** (**IANA**) organization maintains the official list, which you can find at https://www.iana.org/. One more thing to note is that the list is more of a guideline than a rule. Engineers can configure services to use different ports for whatever reason. Each port identified will be associated with a specific process on a machine that either sends or receives information at any time. If a port scan returns ports that are not immediately recognizable, those port numbers should be investigated further to determine their nature. Now that we have some background information on what ports are, let's look at how port scans work.

How do port scans work?

Port scans are designed to interrogate a port to determine whether it is responding and available for communication on a network. Let's look at the three components involved:

1. In normal communications with a service, a machine initiates communication through TCP. The TCP protocol uses what are known as flags, or a combination of flags, to control communications. To establish communication, it uses a special set of flags known as a **three-way-handshake**. The three-way-handshake is a series of packets (three to be exact) that establishes communication. The first packet, which comes from the requesting machine, is a **synchronize** (**SYN**) packet. This can be compared to someone picking up a phone and dialing a number; the ringing is the requester attempting to talk to someone or synchronize with them.
2. The next part of the communication chain is **synchronize and acknowledge** (**SYN-ACK**). In keeping with the phone analogy, this would be when the person at the other end answers the phone and says hello.
3. The final packet in the three-way handshake is **ACK**. This last part is equivalent to the caller responding to the *hello* prompt and beginning the conversation.

This is how the TCP three-way-handshake works, and it is used to establish reliable communication. In addition, once communication has been established, other flags are used to control communications, which are as follows:

- **Reset (RST)**: This means the port does not accept the requests.
- **Finish (FIN)**: This means the conversation is over and the session is torn down. Both the sender and receiver will send the FIN packets to gracefully terminate the connection.
- **Push (PSH)**: This means the sent data passes directly to the application instead of being buffered.

- **Urgent (URG)**: This means the sent data will be processed immediately.

Now that you know about communication flags, it can be understood that pentesters, attackers, and the like can manipulate the communication process and flags by crafting TCP packets, designed to gain information about running applications or services. As stated, one of the mechanisms that port scanning relies on is the use of flags. Common or popular scans designed for TCP port scanning include the following:

- **TCP connect scan**: This type of scan is reliable because it requires the completion of the TCP handshake; however, it is also the easiest to detect. If the port is open and available, it will reply with SYN/ACK; if the port is closed or non-responsive, it will reply with RST/ACK.
- **TCP SYN scan**: This scan is also known as a half-open scan because the TCP handshake is not completed. The incomplete handshake still provides enough data to attackers to let them know whether a system is up and running while trying to be stealthy and evade security systems.
- **TCP FIN scan**: This scan attempts to detect a port by sending a request to close a nonexistent connection. This type of attack is enacted by sending a FIN packet to a target port; if the port responds with RST, it signals a closed port. This technique is usually effective only on Unix devices.
- **TCP NULL scan**: In this scan, specially designed packets are crafted where the TCP flags are not set. Since this breaks the rules to TCP communication, it is an attempt to get a response from a system and possibly disclose whether the port is really open or closed.
- **TCP ACK scan**: This scan sends a crafted packet with an ACK flag set, in order to trip an **access control list** (**ACL**) response and let the attacker know the port is available but an ACL is in place.
- **TCP Xmas tree scan**: This scan functions by sending packets to a target port with flags set in combinations that are illegal or illogical (for example, FIN, PSH, and URG). The results are then monitored to see how a system responds. Closed ports should return

RST.

The TCP protocol offers tremendous capability and flexibility that can be manipulated in an attacker's favor. Unfortunately, the UDP protocol, which is used often for streaming things such as video content and music, does not offer the same capabilities. This is largely because of how the protocol is designed and what it is used for. UDP, sometimes referred to as the unreliable protocol, can be thought of as a best-effort communication and, as such, uses none of the flags and offers none of the feedback that is provided with TCP, making it much more difficult to manipulate and port-scan.

At this stage, an attack has adopted a more interactive and aggressive approach. There are many tools available that can be used to map open ports and identify services running on servers in a target network.

Port scanning issues

Some precautions need to be taken when scanning networks; this applies to both attackers and defenders:

- The first is not taking a network down and breaking things, as an unusable network is not useful.
- The next thing for defenders to consider is not creating more work by triggering alerts and tickets to resolve. Take some time to map out an approach to port scanning or network mapping, and apply the appropriate compensating controls, making all parties involved aware of what the plan is and its purpose.
- Some other aspects to be aware of are as follows:

- **False positives**: Some software will use port numbers registered to other software, which can cause false alarms when port-scanning. This can lead to setting up unnecessary blocks of legitimate programs. Once an item of interest comes up, take some time to investigate it further before applying controls and blocks.
- **False negatives**: Port scanning can be intensive on the machine performing an operation.

This can sometimes exhaust resources, creating false negatives because the machine can no longer properly process the data being returned to it.

- **Heavy traffic**: Scanning port scans especially can create high utilization on a network, introducing latency. With this in mind, it is usually recommended to perform the scanning outside normal business hours if it is going to be a large, encompassing scan.
- **A system crash**: Scans have been known to crash systems, especially vulnerability scan attempts that can render a system inoperable. As stated previously, carefully plan out scans and what is being done, and make the relevant parties aware.
- **Unregistered port numbers**: Many port numbers in use are not registered, which complicates the act of identifying what software uses them.

While port scanning is an effective tool for an ethical hacker or attackers, there are some defenses that can be used to either thwart or make a scan difficult or less reliable. Let's discuss some of these.

Scanning countermeasures

There are more techniques than can be covered here; however, the following list includes some techniques that can employed to prevent an attacker from acquiring information from a port scan, starting with the firewall:

- **A deny-all rule**: This basic firewall rule blocks all traffic to all ports unless such traffic has been explicitly approved.
- **An Intrusion Detection System (IDS)**: Introduce an IDS behind the firewall that reviews traffic to identify scans.
- **Testing the firewall**: Test and verify the firewall's capability and rules to detect and block undesirable traffic.
- **A port scan test**: Use the same tools as the attacker to scan and test, with the goal of gaining a better understanding of what the scan looks like and what controls detect and alert on it.

Once past the perimeter, many attackers find an internal network to be soft and less secure.

This is usually because, for business reasons, security controls are relaxed, allowing a business to operate unhindered. There are some techniques that can be performed on the internal network that can help, including the following:

- **Segmenting the network**: Breaking a network into logical segments makes broad scans more difficult.
- **ACLs on internal routers**: ACLs can be used to block or drop traffic based on rules very similar to what would be implemented on the firewall.
- **Implement a SIEM or alerting system**: The use of a **Security Information and Event Management** (**SIEM**) system can and detect network behavior such as scans.
- **Performing internal scans for dedicated systems**: When using applications that attackers use, such as Nmap, false positives can occur. To avoid confusion, perform these operations on dedicated systems that can quickly be identified, and consider implementing rules or exceptions for these systems to prevent unnecessary alerts.
- **Endpoint protection systems**: In recent years, software for monitoring an endpoint has been released, allowing security personnel to not only monitor the network but to also be able to assess and alert about a specific machine. This software has rules to detect common attack vectors, such as port scans, and not only block them but also alert staff to make them aware of the behavior.

While they may not be able to specifically stop it, alerts can be managed to detect it early. These are just some of the countermeasures and techniques that can be implemented to mitigate scanning. This will be covered more in depth when we discuss Windows and Linux system security in *Chapters 5* and *6*.

Because every tool cannot be covered here, it is necessary to limit the discussion to those tools that are widely used and well known. We will focus on Nmap; however, any tool that provides the same functionality, including scanning and enumeration, would be applicable here. Let's now look at the Nmap scanning tool.

1.7 INTRODUCING THE NMAP NETWORK SCANNING TOOL

The Nmap tool is a commonly used tool for most penetration testers and ethical hackers. It is constantly updated by an active group of contributors to this open source project. Nmap is primarily a port scanner, showing which TCP and UDP ports are open on a target system.

However, Nmap is not just a port scanner. It also provides numerous other features, including ping sweeps, operating system fingerprinting, and tracerouting. It can even be expanded with the **Nmap Scripting Engine** (**NSE**) to become a general-purpose vulnerability scanner as well. We'll look at each of these features.

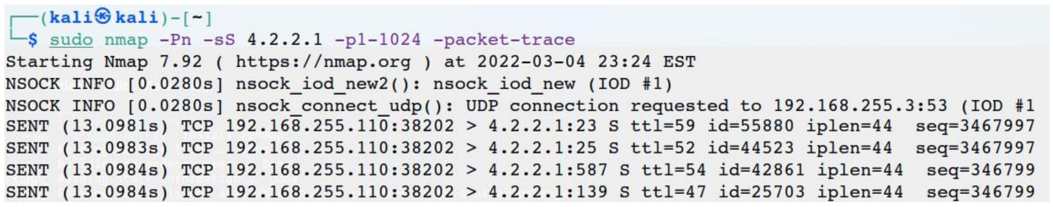
When using Nmap, it can be helpful to have the tool itself display a summary of the packets that it sends. The command switch to invoke this is the –packet-trace switch. It displays various status messages on its output, including some of the calls it makes to the operating system such as connect() sent or received, the protocol used (TCP or UDP), and the source and destination IP addresses and ports. It also displays other header information, such as the IP TTL and the TCP sequence number.

```
Sudo nmap -Pn -sS 4.2.2.1 -p 1-1024 –packet-trace
```

The Nmap command to perform packet trace would look like this:

Let's break this command down to understand it better:

- The -Pn parameter indicates that we don't want to ping the target system; we just want to scan it.
- The -sS parameter indicates that we want a SYN scan.
- The -p 1-1024 parameter tells Nmap to scan ports 1 to 1024 only.
- The –packet-trace option makes Nmap display the status and packet summary information. The screenshot, for reference, is as follows:



```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -Pn -sS 4.2.2.1 -p1-1024 -packet-trace
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-04 23:24 EST
NSOCK INFO [0.0280s] nsock_iod_new2(): nsock_iod_new (IOD #1)
NSOCK INFO [0.0280s] nsock_connect_udp(): UDP connection requested to 192.168.255.3:53 (IOD #1
SENT (13.0981s) TCP 192.168.255.110:38202 > 4.2.2.1:23 S ttl=59 id=55880 iplen=44  seq=3467997
SENT (13.0983s) TCP 192.168.255.110:38202 > 4.2.2.1:25 S ttl=52 id=44523 iplen=44  seq=3467997
SENT (13.0984s) TCP 192.168.255.110:38202 > 4.2.2.1:587 S ttl=54 id=42861 iplen=44  seq=346799
SENT (13.0984s) TCP 192.168.255.110:38202 > 4.2.2.1:139 S ttl=47 id=25703 iplen=44  seq=346799
```

An Nmap example Nmap with -packet-trace

If a user forgets to invoke Nmap with the –packet-trace option, they can turn it on after invoking Nmap by hitting the *P* key. If they no longer wish to see the packet trace, they can use the *shift* + *P*, to turn off packet tracing.

Additionally, while a scan is running, hitting any key will print the current status to the screen, showing the elapsed time, the number of hosts completed, the number of hosts up and running, and the number of hosts currently being scanned in parallel. Other keys that can be used during a running operation include the *V* key to increase verbosity and the *D* key for debug modes.

Another operational control that Nmap supports is the controlling of scanning, including randomness and speeds. Let's explore this in the next section.

Controlling Nmap scan speeds

Scanning speeds can be controlled from the command line by adding a -T parameter, followed by a number from 0 to 5. As these numbers increase, so does the scan speed, and they correspond to the following modes:

- T0 (*Paranoid mode*): This mode is designed to scan slowly in order to avoid common detection rules used by IDS systems and SIEMs. Nmap does this by sending packets approximately every five minutes. Additionally, no packets are sent in parallel with a Paranoid scan.
- T1 (*Sneaky mode*): This mode sends packets every 15 seconds, and as with Paranoid, no parallel packets are sent.
- T2 (*Polite mode*): This mode sends a packet every 0.4 seconds; this mode also does not send packets in parallel. This mode is good when the network capability and load is in question. In other words, it helps to prevent you from impeding or crashing a network.
- T3 (*Normal mode*): This mode is the default mode and really doesn't need to be changed if this is the desired mode. It is designed to run quickly without overwhelming a network while providing quick feedback. This mode, as well as the ones that follow, all use parallel scanning to manage and return results quickly.

- T4 (*Aggressive mode*): This mode will never wait more than 1.25 seconds between responses before sending the next packet. Nmap documentation suggests using -T4 for "reasonably modern and reliable networks."
- T5 (*Insane mode*): This mode waits no more than 0.3 seconds for a response to each probe. Documentation suggests this should only be used on very fast networks. Additionally, at this pace, more errors in results as well as tripping security defenses are likely.

An example of setting the scan speed might look like this – sudo nmap -Pn -T5 -p1-500 192.168.255.2.

Nmap also offers finer controls beyond the six built-in speeds if needed. These settings are not used often; however, if Nmap is used by security personnel, manipulation of these fields might add a distinctive enough signature, where security personnel could distinguish between a sanctioned scan and an attack, using the tool with default settings. The finer-grained Nmap timing options include the following:

- --host_timeout: The maximum time in milliseconds spent on a single host before moving on.
- --max_rtt_timeout: The maximum time to wait for a probe response before retransmitting or timing out. The default is 9,000 milliseconds.
- --min_rtt_timeout: To speed up a scan, Nmap measures the timing response from a target and lowers its timeouts to match that target's network behavior, speeding up a scan but possibly missing responses on networks with high variability.
- --initial_rtt_timeout: This value sets the initial timeout for probes, which will be lowered automatically as Nmap measures the network performance of a target. The default is 6,000 milliseconds.
- --max_parallelism: This option sets the number of probes that Nmap will send in parallel.
- --scan_delay: This value sets the minimum time that Nmap waits between sending probe packets.

An example of setting a scan speed might look like this – sudo nmap –host-timeout 1 192.168.255.0/24.

Outputting results

Nmap supports server output parameters and functions. This type of output becomes very useful for attackers and defenders for inventory and categorization. It also makes it useable as input for later scans, reports, and other types of tools that can leverage output. The output types and additional functions are listed as follows:

- -oN/-oX/-oS/-oG <file>: An output scan in the following formats – normal, XML, s|<rIpt kIddi3, and the Grepable format, followed by a given filename
- -oA <basename>: Output in three of the major formats at the same time
- -v: Increases the verbosity level; for even more verbosity, use -vv
- -d[level]: Set or increase the debugging level, they range from 1 to 9
- --open: Only show open (or possibly open) ports
- --packet-trace: Show all packets sent and received
- --iflist: Print host interfaces and routes (for debugging)
- --log-errors: Log errors/warnings to the normal-format output file
- --append-output: Append to, rather than overwrite the specified output
- --resume <filename>: Resume an aborted scan
- --stylesheet <path/URL>: Use an XSL style sheet to transform XML output to HTML
- --webxml: Reference a style sheet from nmap.org for more portable XML
- --no-stylesheet: Prevent associating an XSL style sheet with XML output

An example of using the output function would be sudo nmap -Pn 192.168.255.2 -p1-1024 -oX scantarget.xml. This command will produce an XML file called scantarget.xml, which can then be used by other tools, including custom-written programs, to perform further operations. One useful method that defenders have is using the output to keep a running inventory of the machines and ports in use. This way a more accurate list of ports and services can be produced for a quick review, without the need to perform a scan or some other reconnaissance. Now that we have looked at output, let's look at extending Nmap beyond its core functions with the NSE.

The NSE

The NSE has numerous goals that really extend the capabilities of Nmap beyond mere port scanning and OS fingerprinting. These goals include the following:

- Utilizing Nmap's efficient multithreaded architecture to send arbitrary messages and receive responses in parallel to and from multiple targets
- Creating an environment so that a development community can write and release free scripts that can easily be incorporated into scans by all Nmap users
- Supporting network discovery options that augment Nmap's port scanning and OS fingerprinting features, including Whois lookups and DNS interrogation
- Enhancing version detection functionality beyond *probing and matching* to look more deeply into the interaction with a target
- Performing vulnerability scanning of target systems to find configuration flaws and other issues
- Detecting systems that have been infected with malware or backdoors based on their network behavior
- Supporting the exploitation of given flaws to gain access to a target machine or its information; this is not functionality integrated into Nmap

Nmap scripts are written in the **Lua** programming language. Lua is regarded as a flexible and extremely fast scripting environment. Its interpreter is free, cross-platform, and has a very small footprint. Lua has been incorporated into a number of projects, including the open-source IDS Snort, Wireshark, and RSA's Netwitness to parse network traffic.

Invoking NSE can be done in several ways:

- The first way involves using the -sC switch, followed by the target and port. This will run the scripts in the default category, described in the next bullet point. An example of how to run this is sudo nmap -sC 192.168.255.2 -p1-1024, which runs the scripts over the first 1,024 ports of the target address.
- The next way to invoke NSE is through categories. Script developers have divided the scripts into several categories, which are as follows:

 safe: These scripts are designed to have minimal impact on a target.

 intrusive: These scripts are the type that may leave log entries, attempt to guess passwords, resulting in lockouts, or have other impacts on a system.

 auth: The scripts in this category are focused on authentication routines and include password guesses and bypass tests.

 malware: These scripts are used to check for the presence of malware and backdoors on a target. Activities here include checking to see whether a common backdoor port is listening and operational. Other activities include interrogating ports to solicit responses that are associated with malware.

 version: These scripts attempt to get version information of services listening on ports. While this function is available natively in Nmap, these scripts offer more comprehensive checks.

 discovery: The scripts in this category are used to get more information about the network environment. These can include DNS lookup and the use of other network protocols that present network information.

 vuln: These scripts are used to determine whether a target has a known security flaw, such as an unpatched or out-of-date version or a misconfiguration.

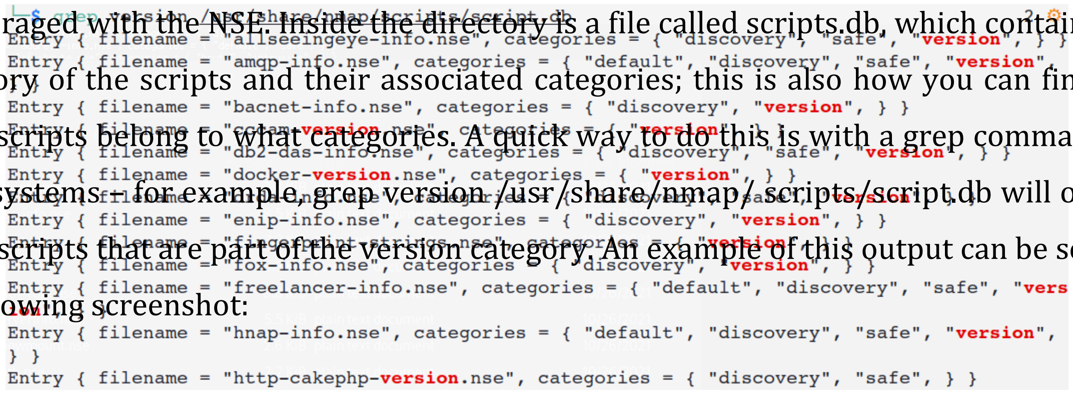
 external: The external category will leverage the use of internet-based third-party databases and systems to pull additional information. These can include things such as Whois lookups and geolocating.

 default: This last category includes the same scripts that are run when the -sC switch is used.

An example of using a NSE category for a scan would be sudo nmap --script=safe 192.168.255.2. This will execute all the scripts categorized as safe against the target. You can also involve multiple categories by separating them with a comma. Applying this to previous example would be sudo nmap --script=safe,default 192.168.255.2. Just like before, this will execute all the scripts in the safe as well as the default categories. Lastly, the NSE script selection supports the and, or, and not Boolean expressions by using quotes and parentheses to separate the logic; an example would be sudo nmap --script="(discovery and safe) and not smb*" 192.168.255.2 Here, both the discovery and safe scripts are run, except any that start with smb.

- The last way to invoke NSE is by leveraging a specific script. The format for this is similar to using the category filter, except instead of using a specific category, a script name is supplied. An example of using a specific script would be sudo nmap –script=http-auth.nse 192.168.255.2 -p80. This will execute the http-auth.nse script against the target.

The next obvious questions are, where are these scripts, and how do I access them? On Windows systems, the scripts directory is located under the Nmap installation directory. On Linux machines, the scripts directory is typically located in the /usr/share/nmap/scripts directory. At the time of writing, there are 600+ scripts that can be leveraged with the NSE. Inside the directory is a file called scripts.db, which contains the inventory of the scripts and their associated categories; this is also how you can find out which scripts belong to what categories. A quick way to do this is with a grep command on Linux systems – for example, grep version /usr/share/nmap/ scripts/script.db will output all the scripts that are part of the version category. An example of this output can be seen in the following screenshot:



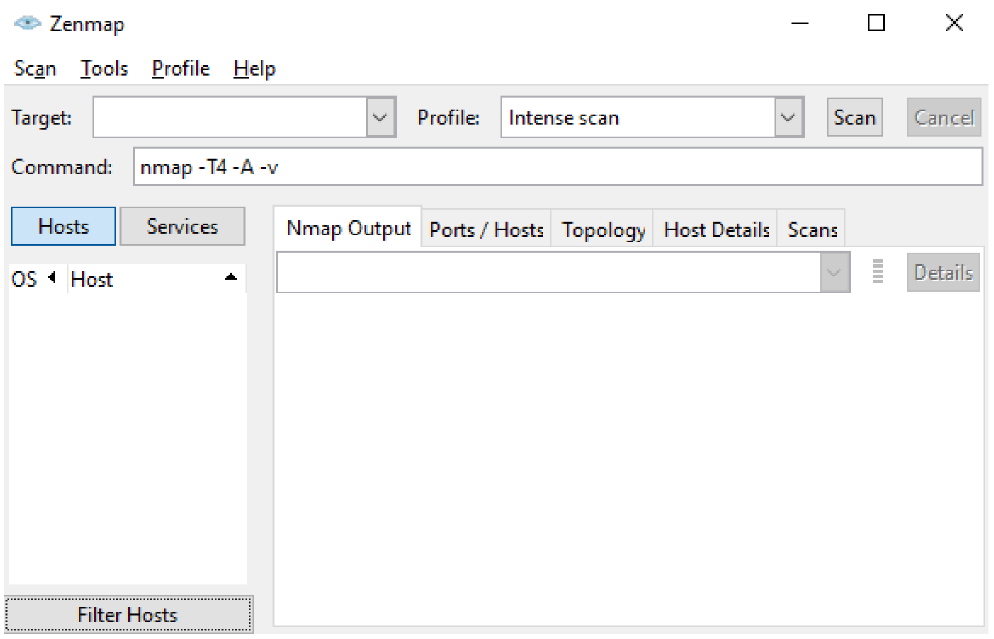
```
└─$ grep version /usr/share/nmap/scripts/script.db
Entry { filename = "allseeingeye-info.nse", categories = { "discovery", "safe", "version", } }
Entry { filename = "amqp-info.nse", categories = { "default", "discovery", "safe", "version",
} }
Entry { filename = "bacnet-info.nse", categories = { "discovery", "version", } }
Entry { filename = "cccam-version.nse", categories = { "version", } }
Entry { filename = "db2-das-info.nse", categories = { "discovery", "safe", "version", } }
Entry { filename = "docker-version.nse", categories = { "version", } }
Entry { filename = "drda-info.nse", categories = { "discovery", "safe", "version", } }
Entry { filename = "enip-info.nse", categories = { "discovery", "version", } }
Entry { filename = "fingerprint-strings.nse", categories = { "version", } }
Entry { filename = "fox-info.nse", categories = { "discovery", "version", } }
Entry { filename = "freelancer-info.nse", categories = { "default", "discovery", "safe", "vers
ion", } }
Entry { filename = "hnap-info.nse", categories = { "default", "discovery", "safe", "version",
} }
Entry { filename = "http-cakephp-version.nse", categories = { "discovery", "safe", } }
```

Displaying category scripts from Nmap's Script.db

Another thing to note is that scripts can belong to more than one category. Custom scripts written for the NSE engine can also go into the scripts directory for easy access. The script.db file will have to be updated to support the custom scripts associated categories. However, note that any changes made to this folder will likely be overwritten when new updates come out. Now that we have discussed all things from the Nmap command line, let's take a look at the Nmap GUI.

The Nmap GUI

At noted, Nmap has both a command line and a GUI interface. The GUI interface, called **Zenmap**, supports several functions and configurations. The main screen when Nmap starts up can be seen in the following screenshot:



Command: nmap -T4 -A -v

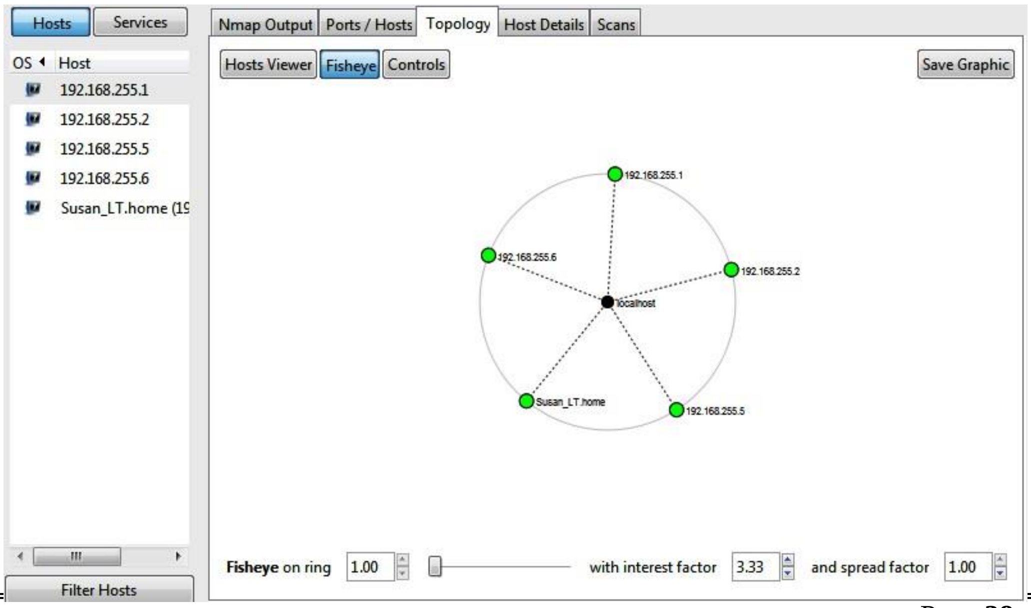
Nmap's main screen

Some of the functions include the **Target** field, which supports machines names, single IP addresses, and subnets. In the **Profile** dropdown, 10 preconfigured scan types are set, including the following:

- **Intense scan**: This profile adds -T4 -A -v to the scan target
- **Intense scan plus UDP**: This profile adds -sS -sU -T4 -A -v to the scan target
- **Intense scan, all TCP ports**: This profile adds -p 1-65535 -T4 -A -v to the scan target
- **Intense scan, no ping**: This profile adds -T4 -A -v -Pn to the scan target
- **Ping scan**: This profile adds -sn to the scan **target**
- **Quick scan**: This profile adds -T4 -F to the scan **target**
- **Quick scan plus**: This profile adds -sV -T4 -O -F –version-light to the scan target
- **Quick traceroute**: This profile adds -sn –traceroute to the scan target
- **Regular scan**: This profile does not add any switches to the scan **target**
- **Slow comprehensive scan**: This profile adds several switches, including -sS -sU -T4 -A -v -PE -PP -PS80,443 -PA3389 -PU40125 -PY -g 53, to the scan target Now let's take a brief look the network mapping feature of Zenmap.

Mapping the network

One feature that Zenmap offers that isn't available in the command line is the **Topology** function, where Zenmap constructs a visual network map. The following is an example of what the Zenmap **Topology** view looks like:



Fisheye on ring 1.00 with interest factor 3.33 and spread factor 1.00

Nmap's Topology feature

While the GUI interface is convenient to use, it is less flexible than the command-line version, mostly because you cannot wrap other commands and scripts, such as scheduled jobs, output redirection, or manipulation, into a set of tasks.

Now that we have learned about scanning and enumeration, let's take some time to put this information into practice with a lab, using the Nmap application.

Lab – Scanning and enumeration

To begin, start up your lab environment, including the Kali Linux and the Metasploitable machines set up in the previous lab. When all the machines are booted up, log in to your Kali machine with the account name kali and password kali.

Open a Command Prompt and run the following nmap commands with the switches that have been provided. This exercise will show ping sweeps and port scans of both the TCP and UDP ports. For an explanation of how the switches work, use Nmap's reference guide (http://nmap.org/book/ man-briefoptions.html).

To help you understand this better, each of the following questions is followed by the command or commands to run to get the answer:

1. Explain what the -sn switch does:

Run the sudo nmap -sn 192.168.255.0/24 command

2. Explain what the –Pn switch does:

Run the sudo nmap –Pn 192.168.255.2 command

Run the sudo nmap -Pn 192.168.255.3 command

3. Explain what the -sS switch does:

Run the sudo nmap -sS 192.168.255.2 command

Run the sudo nmap -sS 192.168.255.3 command

4. Explain what the -sT and -sU switch does:

Run the sudo nmap –sT 192.168.255.2 command

Run the sudo nmap –sU 192.168.255.2 command

Run the sudo nmap –sT 192.168.255.3 command

Run the sudo nmap –sU 192.168.255.3 command

5. Explain what the -O switch does:

Run the command sudo nmap –O 192.168.255.2 command

Run the command sudo nmap –O 192.168.255.3 command

6. Explain what the -sV switch does:

Run the sudo nmap -sV 192.168.255.2 command

Run the sudo nmap -sV 192.168.255.3 command

7. Explain what the -A switch does:

Run the sudo nmap -A 192.168.255.2 command

Run the sudo nmap -A 192.168.255.3 command

8. Explain what the -p1-1024 switch does:

Run the sudo nmap –Pn 192.168.255.2 -p1-1024 command

Run the sudo nmap -Pn 192.168.255.3 -p1-1024 command