Lab 3: Reconnaissance and Information Gathering

6COSC002W

Ayman El Hajjar

Week 3/4

Requirements and Notes

- **NOTE**- In some activities you need **only** your Kali Linux machine to be connected to the internet.
- In this lab, you need the following VM machines
  1. Kali Linux
  2. OWASP Vulnerable machine

Food for thought: Kali Linux Slogan

Information gathering

One of the most important stages of an attack is information gathering. To be able to launch an attack, we need to gather basic information about our target. So, the more information we get, the higher the probability of a successful attack.

Scanning and identifying services with Nmap

Nmap is probably the most used port scanner in the world. It can be used to identify live hosts, scan TCP and UDP open ports, detect firewalls, get versions of services running in remote hosts, and even, with the use of scripts, find and exploit vulnerabilities.

- We first need to see if the server is answering to a ping or if the host is up:

Is Server Responding?

- nmap -sn 192.168.56.102

In the first command, with the -sn parameter, we instructed Nmap to only check if the server was responding to the ICMP requests (or pings). Our server responded, so it is alive.

- Now that we know that it's up, let's see which ports are open

Open ports?

- nmap 192.168.56.102

The second command is the simplest way to call Nmap; it only specifies the target IP address. What this does is ping the server; if it responds then Nmap sends probes to a list of 1,000 TCP ports to see which one responds and then reports the results with the ones that responded.

- Now, we will tell Nmap to ask the server for the versions of services it is running and to guess the operating system based on that.

Guess Operating System

- nmap -sV -O 192.168.56.102

The third command adds the following two tasks to the second one:

- -sV asks for the banner—header or self identification—of each open port found, which is what it uses as the version
- -O tells Nmap to try to guess the operating system running on the target using the information collected from open ports and versions

Question 1

- **Food for thought**- Put together all the information you gathered for this activity and think about what those information? what can you use them? what did they help you identify? How they can be potentially used at a later stage to help you attack the vulnerable machine?

Identifying active machines

Before attempting a pentest, we first need to identify the active machines that are on the target network range. A simple way would be by performing a **ping** on the target network. Of course, this can be rejected or known by a host, and we don't want that.

- Let's begin the process of locating active machines by opening a terminal window.
- Using Nmap we can find if a host is up or not.
- Nmap has a GUI version called Zenmap, which can be invoked by issuing the command zenmap at the terminal window or by going to Applications | Kali Linux | Information Gathering | Network Scanners | zenmap.

commands for determining network range

- nmap -sP 192.168.56.102

- You can also use Nmap to find a collection of hosts in the same network if they are active or no.

commands for determining network range

- nmap -sP 192.168.56.*

Finding open ports

With the knowledge of the victim's network range and the active machines, we'll proceed with the port scanning process to retrieve the open TCP and UDP ports and access points.

- Let's begin the process of finding the open ports by opening a terminal window

Finding open ports

- nmap 192.168.56.102

- We can also explicitly specify the ports to scan (in this case, we are specifying only the first 1000 ports)

Finding open ports

- nmap -p 1-1000 192.168.56.102

- We can also explicitly specify which port to scan on a specific host as in 1 below or for the whole network active devices as in 2 below.

Finding open ports

1. nmap -p 22 192.168.56.102
2. nmap -p 22 192.168.56.*

- You can output the result to a specific file in a specified format

Finding open ports

1. nmap -p 22 192.168.56.* -oG /tmp/nmap-targethost-tcp445.txt

Fingerprinting for Operating systems

At this point of the information gathering process, we should now have documented a list of IP addresses, active machines, and open ports identified from the target organization. The next step in the process is determining the running operating system of the active machines in order to know the type of systems we're pentesting.

- Let's begin the process of OS fingerprinting from a terminal window
- Using **Nmap**, we issue the following command with the **-O** option to enable the OS detection feature.

OS fingerprinting

- nmap -O 192.168.56.102

Fingerprinting for services

Determining the services running on specific ports will ensure a successful pentest on the target network. It will also remove any doubts left resulting from the OS fingerprinting process.

- Let's begin the process of service fingerprinting by opening a terminal window
- Open a terminal window and issue the following command

Service fingerprinting

- nmap -sV 192.168.56.102

Spoofing & Decoy Scan

When we are scanning machines that are not ours, we often want to hide our IP (our identity). Obviously, every packet must contain our source address or else the response from the target system will not know where to return to. [1] The same applies to spoofing our IP when using nmap. We CAN spoof our IP address (-S) in nmap, but as a result, any response and any info we are trying to gather will return to the spoofed IP. Not very useful, if we are scanning for info gathering. A better solution is to obfuscate our IP address. In other words, bury our IP address among many IP addresses so that the network/security admin can't pinpoint the source of the scan. Nmap allows us to use decoy IP addresses so that it looks like many IP addresses are scanning the target.

- We can do this by using the -D switch. the D switch will simply give several IP addresses. In another word, different IP addresses will show up on the victim machine including yours but it will be difficult to pinpoint which one is yours.
- Open a terminal window and issue the following command

Service fingerprinting

- nmap -sS 192.168.56.102 -D 10.0.0.1,10.0.0.2,10.0.0.4

UDP scan

Up until this point, all of our scans have been for TCP ports. Some services and ports use UDP to communicate to the outside world. Our previous scan types (-sS and -sT) will not find UDP ports as they are only looking for TCP ports. Some services only run on UDP, such NTP (port 123) and SNMP (port 161). To find these ports and services, we need to do a UDP scan. We can do this with the -sU switch:

[1] https://null-byte.wonderhowto.com/how-to/hack-like-pro-advanced-nmap-for-reconnaissance-0151619/

- To find these ports and services, we need to do a UDP scan. We can do this with the -sU switch

Service fingerprinting

- nmap -sU 192.168.56.102

Reason

Note in the output from the UDP scan above that some ports are reported as open/filtered. This indicates that nmap cannot determine whether the port is open or it is filtered by a device such as a firewall. Unlike TCP ports that respond with a RST packet when they are closed, UDP ports respond with an ICMP packet when they are closed. This can make scans far less reliable, as often the ICMP response is blocked or dropped by intermediate devices (firewalls or routers). Nmap has a switch that will return the reason why it has placed a particular port in a particular state. For instance, we can run the same UDP scan as above with the --reason switch and nmap will return the same results, but this time will give us the reason it has determined the particular state of the port.

Service fingerprinting

- nmap -sU --reason 192.168.56.102

List of targets

Many times we want to scan a list of IP addresses and not an entire subnet. We can use any text editor and create a list of IP addresses and "feed" it to nmap. This is the list of IP addresses I want to scan.

Service fingerprinting

- **gedit scanlist.txt**
- Inside this file put several IP
- Type **nmap -iL scanlist.txt** to scan all IP addresses in the list.
  - Note: You need to be in the same location as your scanlist file for the command to run.

- Nmap is a port scanner, this means that it sends packets to a number of TCP or UDP ports on the indicated IP address and checks if there is a response. If there is, it means the port is open; hence, a service is running on that port.
- In the first command, with the -sn parameter, we instructed Nmap to only check if the server was responding to the ICMP requests (or pings). Our server responded, so it is alive.

- The second command is the simplest way to call Nmap; it only specifies the target IP address. What this does is ping the server; if it responds then Nmap sends probes to a list of 1,000 TCP ports to see which one responds and then reports the results with the ones that responded.
- The third command adds the following two tasks to the second one:
- -sV asks for the banner—header or self identification—of each open port found, which is what it uses as the version
- -O tells Nmap to try to guess the operating system running on the target using the information collected from open ports and versions

In each command we used, you can use **man** command to look at the various options that it can be used with.

- Other useful parameters when using Nmap are:
  - -sT: By default, when it is run as a root user, Nmap uses a type of scan known as the SYN scan. Using this parameter we force the scanner to perform a full connect scan. It is slower and will leave a record in the server's logs but it is less likely to be detected by an intrusion detection system.
  - -Pn: If we already know that the host is alive or is not responding to pings, we can use this parameter to tell Nmap to skip the ping test and scan all the specified targets, assuming they are up.
  - -v: This is the verbose mode. Nmap will show more information about what it is doing and the responses it gets. This parameter can be used multiple times in the same command: the more it's used, the more verbose it gets (that is, -vv or -v -v -v -v).
  - -p N1,N2,...,Nn: We might want to use this parameter if we want to test specific ports or some non-standard ports, where N1 to Nn are the port numbers that we want Nmap to scan. For example, to scan ports 21, 80 to 90, and 137, the parameters will be: -p 21,80-90,137.
  - --script=script_name: Nmap includes a lot of useful scripts for vulnerability checking, scanning or identification, login test, command execution, user enumeration, and so on. Use this parameter to tell Nmap to run scripts over the target's open ports. You may want to check the use of some Nmap scripts at: https://nmap.org/nsedoc/scripts.

Although it's the most popular, Nmap is not the only port scanner available and, depending on varying tastes, maybe not the best either. There are some other alternatives included in Kali Linux, such as:

- unicornscan
- hping3

- masscan
- amap
- Metasploit scanning modules

An example of one of them below. If you cannot fine it, you can install it by typing **sudo apt-get install amap**

- Using **amap**, we can also identify the application running on a specific port or a range of ports, as shown in the following example.

Service fingerprinting

- amap -bq 192.168.56.102 20-1000

Using Nmap scripting Engine (NSE) for Reconnaissance

The Nmap scripting engine is one of Nmap's most powerful and, at the same time, most flexible features. It allows users to write their own scripts and share these scripts with other users for the purposes of networking, reconnaissance, etc.[2] These scripts can be used for:

- Network discovery
- More sophisticated and accurate OS version detection
- Vulnerability detection
- Backdoor detection
- Vulnerability exploitation

Locate Nmap Scripting Engine

Nmap is installed on most pentesting distributions. To locate Nmap scripting Engine, you need to move to the script where they are installed.

Nmap Scripting Engine location

- cd /usr/share/nmap/scripts

Question 2

- **Food for thought**- Can you categorize the various scripts ? which script you think will be useful when gathering information about our OWASP Vulnerable machine?

[2] https://null-byte.wonderhowto.com/how-to/hack-like-pro-using-nmap-scripting-engine-nse-for-reconnaissance-0158681/

Example: Identifying a web application Firewall

- Nmap includes a couple of scripts to test for the presence of a WAF. Let's try some on our vulnerable-vm.

Detecting WAF

- nmap -p 80,443 --script=http-waf-detect 192.168.56.102

You can also check if WAF is installed on other websites when you are connected to the Internet. Make sure vulnerable machine should be off when you connect to the Internet.

fingerprinting WAF

- nmap -p 80,443 --script=http-waf-fingerprint www.tryhackme.com/

- Now, let's try the same command on a server that actually has a firewall protecting it. Here, we will use the university website.

Detecting WAF

- nmap -p 80,443 --script=http-waf-detect www.tryhackme.com/

- We can do both as well.

Detecting WAF

- nmap -p 80,443 --script http-waf-detect,http-waf-fingerprint www.tryhackme.com/

- **A web application firewall (WAF)** is a device or a piece of software that checks packages sent to a web server in order to identify and block those that might be malicious, usually based on signatures or regular expressions.

- We can end up dealing with a lot of problems in our penetration test if an undetected WAF blocks our requests or bans our IP address. When performing a penetration test, the reconnaissance phase must include the detection and identification of a WAF, **intrusion detection system (IDS)**, or **intrusion prevention system (IPS**). This is required in order to take the necessary measures to prevent being **blocked** or **banned**.

- we will use different methods, along with the tools included in Kali Linux, to detect and identify the presence of a web application firewall between our target and us.

Question 3

- **Food for thought-** What is the difference between detect waf and fingerprint waf scripts?

Question 4

- **Food for thought-** What is the difference between detect waf and fingerprint waf scripts?

Mapping the network

Further information gathering tools can be used such as threat assessment with **Maltego** or mapping the network with **casefile**. Explore those tools and use resources from the internet to try to collect all the information and tools you have gathered or used using those software. Other software also are pre installed in Kali Linux.

Determining network range

With the gathered information obtained by the previous tool, we can now focus on determining the IP addresses range from the target network. in this section, we will explore dmitry, at tool that allows to determine the network range.

- Let's begin the process of determining the network range by opening a terminal window.

commands for determining network range

- dmitry -wnspb 192.168.56.102 -o /root/Desktop/dmitry-result

Reconnaissance

Information from simply looking at code

- Looking into a web page's source code allows us to understand some of the programming logic, detect the obvious vulnerabilities, and also have a reference when testing, as we will be able to compare the code before and after a test and use that comparison to modify our next attempt.
- Lets look at the source code of an application and arrive at some conclusions from that. We will look specifically at an application called WackoPicko.
- WackoPicko has been developed as a *real* web application with following features: [3]

[3] https://www.aldeid.com/wiki/WackoPicko

  - **Authentication**: WackoPicko provides personalized content to registered users.
  - **Upload Pictures**: When a photo is uploaded to WackoPicko by a registered user, other users can comment on it, as well as purchase the right to a high-quality version.
  - **Comments on Pictures**: Once a picture is uploaded into WackoPicko, all registered uses can comment on the photo by filling a form
  - **Purchase Pictures:** A registered user on WackoPicko can purchase the high-quality version of a picture.
  - **Search**: The search feature offers the possibility to filter pictures by looking for strings in the tags of the images
  - **Guestbook**: A guestbook page provides a way to receive feedback from all visitors to the WackoPicko web site.
  - **Admin Area**: WackoPicko has a special area for administrators only, which enables the creation of new users.

- With the source code we can discover the libraries or external files that the page is using and where the links go. Also, as can be seen in the preceding image, this page has some hidden input fields. The selected one is **MAX_FILE_SIZE**;

- this means that, when we are uploading a file, this field determines the maximum size allowed for the file we are uploading. So, if we alter this value, we might be able to upload a file bigger than what is expected by the application; this represents an important security issue.

Taking advantage of robots.txt

- One step further into reconnaissance, we need to figure out if there is any page or directory in the site that is not linked to what is shown to the common user. For example, a login page to the intranet or to the **content management systems (CMS)** administration. Finding a site similar to this will expand our testing surface considerably and can give us some important clues about the application and its infrastructure.

- We will use the **robots.txt** file to discover some files and directories that may not be linked to anywhere in the main application.

- Browse to http://192.168.56.102/vicnum/

- Now we add robots.txt to the URL.

- This file tells search engines that the indexing of the directories jotto and cgi-bin is not allowed for every browser (user agent). However, this doesn't mean that we cannot browse them.

- Let's browse to http://192.168.56.102/vicnum/cgi-bin/
- We can click and navigate directly to any of the Perl scripts in this directory.
- Let's browse to http://192.168.56.102/vicnum/jotto/
- Click on the file named jotto. Jotto is a game about guessing five-character words; could this be the list of possible answers? Check it by playing the game; if it is, we have already hacked the game!

How these tools works

robots.txt is a file used by web servers to tell search engines about the directories or files that they should index and what they are not allowed to look into. Taking the perspective of an attacker, this tells us if there is a directory in the server that is accessible but hidden to the public using what is called "security through obscurity" (that is, assuming that users won't discover the existence of something, if they are not told about it).

Finding Files and folders

DirBuster is a tool created to discover, by brute force, the existing files and directories in a web server. We will use it in this recipe to search for a specific list of files and directories. We will use a text file that contains the list of words that we will ask DirBuster to look for.

- Create a text file **dictionary.txt** containing the following:
  - info
  - server-status
  - server-info
  - cgi-bin
  - robots.txt
  - phpmyadmin
  - admin
  - login
- save the file
- Navigate to **Applications** ⇒ **Kali Linux** ⇒ **Web Applications** ⇒ **Web Crawlers** ⇒ **dirbuster**.
- On the DirBuster's window, set the target URL to **http://192.168.56.102**
- Set the number of threads to 20.

- Select **List based brute force** and click on **Browse**.
- In the browsing window, select the file we just created (**dictionary.txt**).
- Uncheck the **Be Recursive** option.
- Click on **Start**.
- If we go to the **Results** tab, we will see that DirBuster has found at least two of the files in our dictionary: **cgi-bin** and **phpmyadmin**. The response code 200 means that the file or directory exists and can be read. PhpMyAdmin is a web-based MySQL database administrator; finding a directory with this name tells us that there is a DBMS in the server and it may contain relevant information about the application and its users.

How this tool works

DirBuster is a mixture of crawler and brute forcer; it follows all links in the pages it finds but also tries different names for possible files. These names may be in a file similar to the one we used or may be automatically generated by DirBuster using the option of "pure brute force" and setting the character set and minimum and maximum lengths for the generated words.
To determine if a file exists or not, DirBuster uses the response codes from the server. The most common responses are listed, as follows:

- **200. OK:** The file exists and the user can read it.
- **404. File not found:** The file does not exist in the server.
- **301. Moved permanently:** This is a redirect to a given URL.
- **401. Unauthorized:** Authentication is required to access this file.
- **403. Forbidden:** Request was valid but the server refuses to respond.

Password Profiling

- With every penetration test, reconnaissance must include a profiling phase in which we analyze the application, department or process names, and other words used by the target organization. This will help us to determine the combinations that are more likely to be used when the need to set a user name or password comes to the personnel.
- We will use CeWL to retrieve a list of words used by an application and save it for when we try to brute-force the login page.

word dictionary from WackoPicko

- cewl -w cewl_WackoPicko.txt -c -m 5 http://192.168.56.102/WackoPicko/

- Now, we open the file that CeWL just created and see a list of "word count" pairs. This list still needs some filtering in order to discard words that have a high count but are not very likely to be used as passwords; for example, "Services", "Content", or "information".
- You can edit the file to remove words you dont want.

How this tool works

CeWL is a tool in Kali Linux that crawls a website and extracts a list of individual words; it can also provide the number of repetitions for each word, save the results to a file, use the page's metadata, and so on.

- **Crunch**: This is a generator based on a character set provided by the user. It uses this set to generate all the possible combinations. Crunch is included in Kali Linux.
- **Wordlist Maker (WLM)**: WLM has the feature of generating a word list based on the character sets and it can also extract words from text files and web pages (http://www.pentestplus.co.uk/wlm.htm).
- **Common User Password Profiler (CUPP)**: This tool can use a word list to profile the possible passwords for common user names and download word lists and default passwords from a database (https://github.com/Mebus/cupp).

John the ripper

John the Ripper is perhaps the favourite password cracker of most penetration testers and hackers in the world. It has lots of features, such as automatically recognizing the most common encryption and hashing algorithms, being able to use dictionaries, and brute force attacks; thus, enabling us to apply rules to dictionary words, to modify them, and to have a richer word list while cracking without the need of storing that list. This last feature is the one that we will use in this recipe to generate an extensive dictionary based on a very simple word list.

- We will use the word list generated in the previous step, Password profiling with CeWL, to generate a dictionary of possible passwords.
- John has the option of only showing the passwords that he will use to crack a certain password file. Let's try it with our word list:

word dictionary from WackoPicko

- john --stdout --wordlist=cewl_WackoPicko.txt

- Another feature John has, as mentioned before, lets us apply rules to modify each word in the list in various ways, in order to have a more complete dictionary:

word dictionary from WackoPicko

- john --stdout --wordlist=cewl_WackoPicko.txt --rules

- As you can see in the result, John modified the words by switching cases, adding suffixes and prefixes, and replacing letters with numbers and symbols (leetspeak).

- Now we need to do the same but send the list to a text file instead, so that we can use it later:

word dictionary from WackoPicko

- john --stdout --wordlist=cewl_WackoPicko.txt --rules > dict_WackoPicko.txt

- john --stdout --wordlist=cewl_WackoPicko.txt --rules > dict_WackoPicko.txt

How this tool works

CeWL is a tool in Kali Linux that crawls a website and extracts a list of individual words; it can also provide the number of repetitions for each word, save the results to a file, use the page's metadata, and so on.

- Although John the Ripper's aim is not to be a dictionary generator, but to efficiently use word lists to crack passwords (and it does it very well); its features allow us to use it to expand existing lists and create a dictionary that is better adapted to the passwords used by modern users.

- In this example, we used the default rule set to modify our words. John's rules can be defined in its configuration file, located in Kali Linux in **/etc/john/john.conf**

- More information about creating and modifying rules for John the Ripper can be found at: http://www.openwall.com/john/doc/RULES.shtml

Enumeration

Enumeration is a process that allows us to gather information from a network. We will examine DNS enumeration technique. DNS enumeration is the process of locating all DNS servers and DNS entries for an organization. DNS enumeration will allow us to gather critical information about the organization such as usernames, computer names, IP addresses, and so on.

Note

- You should not do this on any website without permission

- **DNS enumeration** is a tool that is pre installed on Kali Linux.
- Kali Linux tools are pre-installed in bin folder of the user.

commands for enumeration

- cd /usr/bin
- ./dnsenum --enum www.hackthissite.org

- We should get an output with information like host, name server(s), mail server(s), and if we are lucky, a zone transfer.
- Explore the various options using the **man** command to see the additional options. Those options can help if you are looking for something specific.