Kali_TutorialYT

Full Penetration Testing Course

Cyber Mentor + FreeCodeCamp on YT

Note-taking applications like this are used to take notes of penetration tests, and then use them later on while writing pen-test reports.
The screenshot app used by the teacher is "Greenshot" but, I will be using just the regular Prt. Scrn. button.
The Greenshot app allows users to blurr out important details such as hashes or passwords also.
Pen-test notes have to be organised well.

misc

all miscellaneous commands that aren't in a specific category, or will need a whole category for just one cmd.

history

history command returns the history of the commands you ran

history | grep ping : finds the ping commands used in my history

grep

grep command finds stuff inside a file.

navigatingTheFileSystem

These notes help in navigating around the files using terminal. It is basic linux stuff.

pwd

pwd - Print working directory - Prints current folder location

cd

cd - Change directory - Changes to another folder location

Hit tab twice to see list of elements in current directory.

cd ../ = changes to previous directory/

to change to root, you need to do cd /root not just cd root.

ls

ls stands for list i.e. it lists the contents of your cwd (current working directory)

ls -a = lists out all contents with max details such as permissions etc of every item.

ls -la = ls -a

mkdir

mkdir stands for make-directory which just creates a new directory in your desired location

ex : mkdir ./testFolder = this command creates a new directory in your cwd called "testFolder"

rmdir

rmdir stands for remove-directory.

rmdir testFolder = removes "testFolder" in cwd.

cp

cp stands for copy.

cp new.txt Downloads/file.txt = this cmd will move "new.txt" into Downloads folder and rename it to file.txt

mv

mv stands for move.

mv new.txt Desktop/file.txt = moves new.txt into Desktop and renames to file.txt

locate

locate is used for locating a file.

locate new.txt = locates new.txt in cwd.

man

man stands for manual.

man ls = gives the list of flags that can be used with ls command.

echo

echo cmd is used to create a file.

echo “hello” > hello.txt = creates a file called hello.txt and writes hello in it

echo “hello again” > hello.txt = removes all current content of hello.txt and writes hello again in it

echo “hello world!” >> hello.txt = adds the line hello world! in hello.txt, so hello again stays in the first line.

touch

touch creates a file as well, here you need to use nano to edit the file.

touch hello.txt

nano hello.txt

rm

rm stands for remove, it just removes the file specified

userPermissions

these are all the settings related to users in kali

chmod

chmod is the cmd used for changing the permissions of a user for a file.

a user can have the following permissions = read, write or execute (r, w or x)

in ls -la (or ls -a), we get the permission details of the contents of the cwd.

d stands for directory, r stands for readable, w stands for writable, and x stands for executable.

inside this permissions block, there are three columns, the first one is the set of permissions for the owner, the second one is the set of permissions of the group, and the third one is the set of permissions of the public.

the two columns following the permissions are owner:group columns, owner is the creator of the file, who has all the rights and permissions of that file, group is someone who the owner can give partial or full access of the file to.

chmod +x new.txt = adds executable permission to new.txt

adduser

adduser is the cmd to add a user.
after the cmd, you'll be prompted for general details about the user.

cat

the cat command gives details about the files in a directory, it gives user-specific details. this command displays the contents of one or more files without having to open the file for editing. it gives the hashes as well. a hash usually begins with a $ sign. a hash is the encryption for the password of the user.
these hashes can be passed onto tools like hashcat to crack the password of the user.

cat /etc/passwd = gives details of passwd file with user-specifc data

su

su stands for switch user.

su bob = switches user to bob

sudo

sudo stands for super user do which is basically the admin rights in linux.

sudo apt-get install x.py = installs x.py as super user aka root.

a sudoers file exists which defines the permissions of every user and whether they are super-user or not.

commonNetworkCommands

these are some common networking commands.

ifconfig

ifconfig is similar to ipconfig, it just gives ip details, network details of the ports of the machine.

ifconfig - configure a network interface

iwconfig

iwconfig is just the wireless brother of ifconfig.

iwconfig - configure a wireless network interface

ping

ping cmd sends network packets to specified ip-address

ping - send ICMP ECHO_REQUEST to network hosts

ping 192.168.0.1 = pings the target endlessly.
ping -c 1 192.168.0.1 = pings the target just one time (cuz -c 1 means count 1)
ping -c 1 127.0.0.1 > ip.txt = pings the target and prints the result in ip.txt file.

to differentiate between a "up" machine and "down" machine, you look at the second line of the ping result. which contains stuff like 64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.043 ms

command: to extract just the ip address from a ping result
ping -c 1 127.0.0.1 > ip.txt
cat ip.txt | grep "64 bytes" | cut -d " " -f 4 | tr -d ":"
here, we first pinged the target and stored the result in a text file called ip.txt, then we read the file using "cat", we filtered the file to just find the line which said '64 bytes' using "grep" (we used 64 bytes because its the default size of packets that get pinged to a target), we then cut the line using "cut" where the seperator or delimiter was "-d" and the value of this separator was a whitespace ' ', then we used "-f" to specify that we want the 4th field (because the ip address has 3 whitespaces before it i.e. 3 delimiters before it and the 4th field or element is the ip-address itself), and finally we translated or used "tr" to end the line whenever ':' occurs.

arp

arp -a: gives details of ip and mac address, i.e. return an ip address and its mac address.

arp - manipulate the system ARP cache

netstat

netstat shows you all the ports that are open and what's connected to those ports.

netstat - Print network connections, routing tables, interface sta-
tistics, masquerade connections, and multicast memberships

route

route command returns the details regarding destination, gateway, genmask, flags, metric, ref, use, iface of the host.
it returns kernel ip routing table.

route - show / manipulate the IP routing table

installingUpdating

all software commands are in these notes

updatedb

updatedb stands for updatedatabase, and it just updates the database of all the commands present in kali's list of commands.
so if some command is not working for you in kali, try updatedb once.

apt

apt is used for installing software and packages into your machine

- apt-get install package = installs package
- apt-cache search package = searches for package in database of kali's packages
- apt-cache search package* = searches for everything containing package in its name
- apt policy package = gives details of package, and whether its installed or not.

services

these cmds are used to manage and control system services, start servers etc.

service

service cmd is used to access a service and options related to that service

- service apache2 start = starts apache2 server service
- service ssh start = starts ssh service
- service postgresql start = starts postgresql service
- service ssh stop = stops ssh service

apache2 server can be accessed by going into your computer's local ip address through a web browser and starting the apache2 service.

systemctl

systemctl stands for system control and this cmd is used to manage system services.

- systemctl enable ssh = enables ssh to start automatically upon computer's boot.

scripting

to write a script, we use nano, an save the file as “sh” which means “shell script”

at the top of every script you need to declare what you're running the script in, eg:
    #!/bin/bash = for bash
    #!/bin/python = for python

pythonScripts

Running python files in Kali requires the use of python3 command.

python

python is the command to run python2 commands.
    python -m pyftpdlib -p 21 -w = runs ftp server on port 21

python3

python3 is a python-file running cmd in kali
    python3 helloworld.py = runs helloworld.py
    python3 -m http.server 80 = hosts http server on port 80

sweep.py

```
import subprocess
import datetime
import re
import argparse

def write_result(filename, ping):
    with open(filename, "w") as f:
    f.write(f"Start time {datetime.datetime.now()}")
    for result in ping:
        f.write(result)
    f.write(f"End time {datetime.datetime.now()}")

def ping_subnet(subnet):
    for addr in range(1,255):
    yield subprocess.Popen(["ping, f"{subnet}.{addr}", "-n" , "]"], stdout=subprocesss.PIPE) \
            .stdout.read()                                          \
            .decode()

def main(subnet, filename):
    write_result(filename, ping_subnet(subnet))
```

```
def parse_arguments():
	parser = argparse.ArgumentParser(usage='%(prog)s [options] <subnet>',
					description='ip checker',
					epilog="python ipscanner.py 192.168.1 -f somefile.txt")
	parser.add_argument('subnet', type=str, help='the subnet you want to ping')
	parser.add_argument('-f','--filename', type = str, help='The filename')
	args = parser.parse_args()

	if not re.match(r"{\d{1,3}\.\d{1,3}\.\d{1,3})", args.subnet) \
		or any(a not in range(1,255) for a in map(int,args.subnet.split("."))):
			parser.error("This is not a valid subnet")

	if " " in args.filename:
		parse.error("There cannot be whitespaces in the filename")

	return args.subnet, args.filename
if __name__ == '__main__':
	main(*parse_arguments())
```

above is the code of a ip-sweeper python file.
the first function is writing out a file, giving a datetime, for result in ping, its going to write the start time result, then the end time result
now next function, for address in range 1 to 255, we're going to yield a subprocess with Popen (it starts a process), its going to ping the subnet, the address, plus some extra information
now is our main function.
then the function, gives you different details, such as syntax of the script-usage command, plus some file commands
if i don't have a subnet, then the first if statement will execute.
if there's a whitespace then second if statemenet will execute
then the last __name__ part is just a safety measure.

No need to understand everything, but mostly i should be able to understand the concepts such as functions, loops, conditions, rest things obviously need to be googled, such as module functions.

scanner.py

```
#!/bin/python3

import sys #allows us to enter command lines arguments, among other things
import socket
from datetime import datetime

#Define our target
if len(sys.argv) == 2:
	target = socket.gethostbyname(sys.argv[1]) #Translates hostname to IPv4
else:
	print("Invalid amount of arguments.")
	print("Syntax: python3 scanner.py <ip>")
	sys.exit() #allows us to close the program, otherwise it'll crash.

#Add a pretty banner
```

```
print("-"*50)
print("Scanning target {}".format(target))
print("Time started: " + str(datetime.now()))
print("-"*50)

start = 50 #starting port to scan
end = 85 #last port to scan

try:
	for port in range(start,end):
		s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #socket.af_inet is IPv4, and sock_stream is
our port, so value of ipv4 is obtained by inet, and value of port is obtained by sock_stream
		socket.setdefaulttimeout(1) #putting 1 second, but it is a float value
		result = s.connect_ex((target,port)) #returns error indicator, i.e. if any error in connection, then returns
error indicator, else returns 0
		print("Checking port {}".format(port))
		if result == 0:
			print("Port {} is open".format(port))
		s.close()
except KeyboardInterrupt: #if user presses ctrl + c (i.e. closes the program)
	print("\nExiting program.")
	sys.exit()

except socket.gaierror: #if hostname cannot be converted to its ipv4
	print("Hostname could not be resolved.")
	sys.exit()

except socket.error: #if host is down, or any other reason
	print("Could not connect to server.")
	sys.exit()
```

bof.py

```
#!/usr/bin/python
import sys, socket
from time import sleep

buffer = "A" * 100
while True:
	try:
		s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
		s.connect(('192.168.1.1',9999))
		s.send(('TRUN /.:/' + buffer))
		s.close()
		sleep(1)
		buffer = buffer + "A"*100
	except
		print("Fuzzing crashed at %s bytes" % (str(len(buffer))))
		sys.exit()
```

above is the code for a buffer overflow basic fuzzing script, written in python2.
we're declaring the we're calling python2 in the first line
then we're importing modules, that we'll use.
socket module allows to connects to computer's sockets
buffer is a variable is equal to a string of a hundred "A"s
then a while loop is sitting
try block tries to do something and if something else happens then except block starts working
in first line of try:

we're sending a connection, to the ip defined in line 2 of try block, at port 9999, then we're sending a message plus a hundred "A"s, then we're closing, and sleeping for one second, then we're appending buffer to a hundred more "A"s

as long as the connection keeps working, the loop will go on forever, if the connection crashes, then we print out the except block, give details of where the buffer happened, then we're exiting.

bashScripts

ipsweep.sh

first script: ipsweep.sh

```
#!/bin/bash
for ip in `seq 1 254`; do
ping -c 1 $1.$ip | grep "64 bytes" |cut -d " " -f 4 | tr -d ":" &
done
```

this script, uses a for loop to iterate ip from a sequence of 1 to 254, its going to do the following ping command. $1 is user input. the & at the end of the ping command does something called "threading" and basically it just pings all the ip-addresses at once i.e. all ip-addresses 1 to 254 at once, rather than pinging them one at a time, and allows for a quick result.
now to run this script, you write in terminal, the following command:

./ipsweep.sh 192.168.1 = the final part of ip address is the one which will be iterated in the for loop. so 192.168.1.1-254 is the range of ip's being pinged.

./ipsweep.sh 127.0.0 > iplist.txt = displays the final result of the pinging, in a file called iplist.txt.

```
#!/bin/bash
if [ "$1" == "" ]
then
echo "You forgot an IP address!"
echo "Syntax: ./ipsweep.sh 192.168.1"
else
for ip in `seq 1 254`; do
ping -c 1 $1.$ip | grep "64 bytes" |cut -d " " -f 4 | tr -d ":" &
done
fi
```

now we added an if statement, asking if the user input 192.168.1 is not present, then we will tell user that hey you forgot something, else we will ping normally.
if statement starts with "if" obv, and ends with "fi" which is "if" backwards.

oneliner-ipsweep

this is just a one-liner version of the ipsweep script we made earlier.

```
for ip in $(cat iplist.txt); do nmap -p 80 -T4 $ip & done
```

this cmd does nmap for every ip in iplist.txt and checks whether port 80 is open or not (-p 80) at a speed of T4. the ampersand (&) runs them all at once, instead of one by one.

breach-parse.sh

```
#!/usr/bin/env bash

if [ $# -lt 2 ]; then
   echo "Breach-Parse v2: A Breached Domain Parsing Tool by Heath Adams"
   echo " "
   echo "Usage: ./breach-parse.sh <domain to search> <file to output> [breach data location]"
   echo "Example: ./breach-parse.sh @gmail.com gmail.txt"
   echo 'Example: ./breach-parse.sh @gmail.com gmail.txt "~/Downloads/BreachCompilation/data"'
   echo "You only need to specify [breach data location] if its not in the expected location (/opt/breach-parse/
BreachCompilation/data)"
   echo " "
   echo 'For multiple domains: ./breach-parse.sh "<domain to search>|<domain to search>" <file to output>'
   echo 'Example: ./breach-parse.sh "@gmail.com|@yahoo.com" multiple.txt'
   exit 1
else
   if [ $# -ge 4 ]; then
      echo 'You supplied more than 3 arguments, make sure to double quote your strings:'
      echo 'Example: ./breach-parse.sh @gmail.com gmail.txt "~/Downloads/Temp Files/BreachCompilation"'
      exit 1
   fi

   # assume default location
   breachDataLocation="/opt/breach-parse/BreachCompilation/data"
   # check if BreachCompilation was specified to be somewhere else
   if [ $# -eq 3 ]; then
      if [ -d "$3" ]; then
         breachDataLocation="$3"
      else
         echo "Could not find a directory at ${3}"
         echo 'Pass the BreachCompilation/data directory as the third argument'
         echo 'Example: ./breach-parse.sh @gmail.com gmail.txt "~/Downloads/BreachCompilation/data"'
         exit 1
      fi
   else
      if [ ! -d "${breachDataLocation}" ]; then
         echo "Could not find a directory at ${breachDataLocation}"
         echo 'Put the breached password list there or specify the location of the BreachCompilation/data as the
third argument'
         echo 'Example: ./breach-parse.sh @gmail.com gmail.txt "~/Downloads/BreachCompilation/data"'
         exit 1
```

```
        fi
    fi

    # set output filenames
    fullfile=$2
    fbname=$(basename "$fullfile" | cut -d. -f1)
    master=$fbname-master.txt
    users=$fbname-users.txt
    passwords=$fbname-passwords.txt

    touch $master
    # count files for progressBar
    # -not -path '*/\.*' ignores hidden files/directories that may have been created by the OS
    total_Files=$(find "$breachDataLocation" -type f -not -path '*/\.*' | wc -l)
    file_Count=0

    function ProgressBar() {

        let _progress=$(((file_Count * 100 / total_Files * 100) / 100))
        let _done=$(((_progress * 4) / 10))
        let _left=$((40 - _done))

        _fill=$(printf "%${_done}s")
        _empty=$(printf "%${_left}s")

        printf "\rProgress : [${_fill// /\#}${_empty// /-}] ${_progress}%%"

    }

    # grep for passwords
    find "$breachDataLocation" -type f -not -path '*/\.*' -print0 | while read -d $'\0' file; do
        grep -a -E "$1" "$file" >>$master
        ((++file_Count))
        ProgressBar ${number} ${total_Files}

    done
fi

sleep 3

echo # newline
echo "Extracting usernames..."
awk -F':' '{print $1}' $master >$users

sleep 1

echo "Extracting passwords..."
awk -F':' '{print $2}' $master >$passwords
echo
exit 0
```

fiveStagesOfHacking

Reconnaissance

Passive

Physical

Social

Web

Host

Active

Scanning & Enumeration

tcp

when we do port scanning, typically we would be scanning the tcp side
some tools only run on udp, some only on tcp
tcp is connection oriented, it has a handshake
its used on applications requiring high reliability
    such as http, telnet, ftp, anything you would need some connection to

udp

it doesn't have a handshake, its connection-less
used on apps that require a fast connection, no high reliability needed for this protocol only fast connection
    apps such as dns, dhcp, snmp

Gaining Access

Maintaining Access

Covering Tracks

reconnaissanceTools

most of these tools are websites that don't require a text-description or manual like linux commands, so just their title is given and you can go on them just enjoy their GUI approach to hacking.
this "stage" of hacking is more of just general knowledge search before you attack a target.
example if you were a robber and you were to steal from a mansion, you would first go to the addresss of that mansion, see all the external features of the mansion, see around the mansion and find some place to part your getaway vehicle, maybe hide in a burger shop etc. so reconnaissance is just that, you gain general info about your target from social media platforms, you gain their bio like name, organisation etc. basically details from linkedin.

targetValidation

WHOIS

findingSubdomains

googleFu

nmap

sublis3r

bluto

crt.sh

theHarvester

theHarvester is an information gathering tool in kali

theHarvester -d tesla.com -l 500 -b all = searches for email and passwords of domain tesla.com upto 500 in length through all sources.

dig

fingerprinting

nmap

wappalyzer

whatweb

builtwith

netcat

dataBreaches

HaveIBeenPwned

emailFinder

emailHunter.io

scanningTools

wireshark

when we use wireshark, we're just listening to the network traffic
if i start capturing eth0 and go to a website on my browser, i can see packets on the wireshark application, which just indicates network traffic.
when we connect to a website, we send a SYN request and we get a SYN ACK reply back from the website, then we send the website an ACK message saying hello, and the website establishes connection
when we send a regular request:

SYN - "Hey is there a port open, youtube.com?"

SYN ACK - "Hey <your ipv4>, yes we have this <xyz> port open"

ACK "Hello!, ok ill connect as <your ipv4>"

when we do a stealth scan:

SYN - "Hey is there a port open, youtube.com?"

SYN ACK - "Hey <your ipv4>, yes we have this <xyz> port open" this way we get an open port of the following website, and put that in nmap and scan it.

RST - "Reset-Just kidding, I dont wanna connect"

nmap

nmap stands for network mapper

Scanning TCP on the network:

nmap -sn 192.168.1.0/24 = scans whether hosts from 192.168.1.0 to 192.168.1.24 are up or down

nmap -T4 192.168.1.1 = T4 is a type of scan speed, with T1 being the fastest, if you go fast, its easier to be detected and miss information, this command however displays what ports of the target are open and the service which they offer.

a -p switch lets you defines the ports you want to scan, if you don't define then nmap will just scan top 1000 common ports, if we only do 1000, we miss a lot of information. so always scan every port in tcp range.

nmap -T4 -p 0-65000 192.168.1.1 = scans ports 0 to 65000 on host with speed t4

sometimes we might find the state of a port to be "filtered" which just means that nmap is not sure whether its open or not, it might be a false positive etc.

cybermentor's default scan:

nmap -T4 -A -p- 192.168.1.1 = we're doing nmap at T4 speed, the -A stands for all and enables OS detection, version detection, script scanning and traceroute, so -A gives as much information as possible, -p- stands for all ports

we can do something called "staging" to improve this scan, in which first we scan quickly, and then scan more technically.

-A does a lot of things at once = heavylifting

nmap -T4 -A -p- 192.168.1.1 -oA client = gives you the result in normal (txt) format, XML format, script kiddie and grepable format. the client words needs to be replaced with the file name.

a -v switch stands for verbose, which gives you a status update every now and then if your scan is very long, if you use -vv then it will cause a greater effect.

Scanning UDP on the network:
nmap -sU -T4 192.168.1.1 = a UDP scan takes a long time with higher probability of false positives.
UDP is not useless. Its important, and you need to look scan UDP atleast once.

Pre-Defined nmap scipts:
ls /usr/share/nmap/scripts = has a lot of predefined scripts for different tasks in nmap.
nmap --script=all tesla.com = scans tesla.com for all the pre-defined scripts in nmap (not entirely legal)
nmap --script=ssl-enum-ciphers tesla.com = scans tesla.com for all the ciphers and encryption they use.

nessus

During this course, I have used the following username for my nessus account and a temp-mail:
username = johnramboisonnessus
---
nessus is a 3rd party software and doesn't come built-in with kali
you can install it from google and after its done installing, you can log on to your nessus dashboard which shows you details of all the scans you did at https://kali:8834/

when you login to your nessus session, mostly in the start you'll be using the basic network scan and sometimes the advanced scan.
upon clicking on the basic network scan you'll see tabs with settings, credentials and plugins which you can modify as per your needs.
while creating a scan, you can separate your targest with a comma.
you can change the discovery type from port scanner on common ports to port scanner on all ports.

metasploit

metasploit is a module-based tool, there's different modules for different actions, such as scanning, exploiting, cracking etc. so its not just under the scanning-tools category.

to start the metsaploit framework you type:
msfconsole
this will start up the metasploit console

if you type:
search scanner
metasploit will show you a list of all modules that have the keyword "scanner" in them, there are multiple modules for different scanning purposes.

to use a module:
use auxiliary/scanner/portscan/syn

once you enter a module and want to look at its info:
info = basic description of module
info -d = provides full documentation of the module by opening it on your browser.
options = just shows the list of commands that you can use in the module

let's start with using the auxiliary/scanner/portscan/syn module to scan TCP using SYN scan:
once you type in options you'll see a list of commands such as batchsize, delay, jitter, ports etc.
some cmds have a default setting [such as ports (to scan from 1 to 10000)] others dont such as rhosts (remote hosts)

rhosts vs rhost:
rhosts is when you can supply more than one remote host whereas rhost just stands for one remote host, an example for a remote host can be your router.

changing values of options:
set ports 1-65535 = sets the default setting of ports to scan from 1 to 65535 ports
set rhosts 192.168.1.1 = sets one of the remote hosts as 192.168.1.1

using the module:
run
exploit
both the options initiate the module

nikto

nikto is a free built-in kali linux web-scanning tool

scanning a website using nikto:
nikto -h https://youtube.com
nikto -h https://tesla.com
both will scan the specified websites and give details like the ip address, ports, ssl info etc.

burpsuite

burpsuite is a GUI based scanning tool.
no explanation for GUI tools

netdiscover

netdiscover is a scanning tool which finds the ip address in an ip range

exploitationTools

nc

nc stands for netcat
to make a bind shell from your machine to target machine:
nc <target's IPv4> 4445 : tries to establish a bind shell connection to target's ip address over port 4445
nc -lvp 4445 -e /bin/bash or nc -lvp 4445 : listens to port 4445

KioptrixLV1Hack

So these are the notes of the kioptrix machine which we will be hacking. These notes resemble how you are supposed to show to the organisation the notes of your hack or how to report your penetration test.
First you'll have to install Kioptrix on your Virtualisation Software.
Also, you'll have to configure Kali and Kioptrix to run on a NAT Network.
You'll have to go to preferences in VBOX, and create a NAT network, assign it ip address range 192.168.1.0/24.
Now put kali's network adapter type and kioptrix's network adapter type to NAT Network.
Also in the .vmx file of your kioptrix's file directory, the ethernet0.networkName should be set to "nat"
https://www.youtube.com/watch?v=qq76jA7yWfM this youtube video can help in the set up of Kali and Kioptrix in VBOX.

Process

==========
Taking notes: Except this node, all other nodes are part of KioptrixLV1Hack, and they represent, how i would take notes in a real pen-test report.
Example1: if i was given a task by my client-organistaion to hack kioptrix level 1, then these notes would be the ones that i would make to write in my final pen-test report that would be submitted.
Example2: The default web page of kioptrix level 1 is left open which is a vulnerability, so i wrote about it in the "Vulnerabilities" node and added a screenshot as proof along with the IP address of the kioptrix machine.
==========
This part of KioptrixLV1Hack is not part of the actual notes that will be part of your report. This is just the process/ walkthrough of the hack.

1. Discover Kioptrix's IP
To do so, you use the following commands or basically you validate your target:
netdiscover = will discover all hosts on your network
netdiscover -i eth0 = will discover all hosts on the interface eth0
netdiscover -r 192.168.1.0/24 = will discover all hosts between 192.168.1.0 to 192.168.1.255
You can use anyone of these commands however the third one will be the fastest
nmap -p- 192.168.1.0/24 = this cmd will work best if you know what all ports of the kioptrix machine are open beforehand. this is also a fast way to discover your victim's ip.

2. Scan Kioptrix's IP

To do so, you can use the following commands/tools:

nmap -p- 192.168.1.104 = so i found that the ip ending with 104 is my victim's ip address, now ill do a basic port scan on that machine to find what all ports of that machine are open.

say, ports 22, 80, 500, 600, 1000, 15000 are open on the machine, then ill run the following scan on those ports for more details on the machine:

nmap -A -T4 -p 22,80,500,600,1000,15000 192.168.1.104 = scans 104 (victim machine, alias given by me), for all the specified ports, with speed -T4, and provides all possible details cuz of the -A flag.

nessus = you can also do a side-by-side nessus-basic-network-scan on the victim machine to provide you with better readability on the vulnerabilities of 104.

nikto -h 192.168.1.104 = scans 104 using nikto. its is good practice to scan your targets using multiple scanners, to find multiple vulnerabilities, maybe one scanner might miss a few vulns, which other scanners can find, and vice-versa.

3. Brute Force the default test page

Now we can also brute force the default test page of kioptrix's apache server using

dirbuster

and changing the options in the tool according to you (its GUI), this will give us the contents of the directories in the website of the test-page, and we might be able to find hidden statistics files as well.

4. Requesting a packet from the server header (Apache/1.3.30 (Unix)..)

Now using the following command:

curl --head 192.168.1.104 = we are pinging the server header of the IP address and we find more information like the ETag of the server and server name along with ssl versions.

5. Checking for error page

Now we check if the default error page of the server is changed or not. To do so, we type:

192.168.1.104/jkjkjkjkjkjkjkjk = here we find that the default error page of the server is not changed and it reveals information such as port number, and the ip address of the server

6. Weak Ciphers Reporting

Next we saw many issues with the SSL of the kioptrix machine, so we went on nmap and typed in:

nmap --script=ssl-enum-ciphers -p 443 192.168.1.104 = scans all ssl ciphers on the victim machine on port 443 using the ssl-enum-ciphers script and then we report the least strenght of the ciphers.

7. Searching up ways to exploit vulnerabilities

Now, to find ways to exploit vulnerabilities you can always search on websites like cve, exploitdb on google. just type in the keyword of the server/protocol you want to exploit and try to find as much info as possible.

Another way of doing this is using the searchsploit command:

searchsploit apache 1.3.20 = searches the locally stored database of exploit db about exploiting apache webserver 1.3.20.

to read the files, just type in cat /usr/share/exploitdb/exploits/(file path given on right of file name)

8. Using metasploit & searchsploit to enumerate SMB/Samba

msfconsole = open up metasploit

use the auxiliary scanner with smb_version and set rhosts to 192.168.1.104, and run the scan

once you do the scan you'll find a Samba version i.e. Samba 2.2.1a

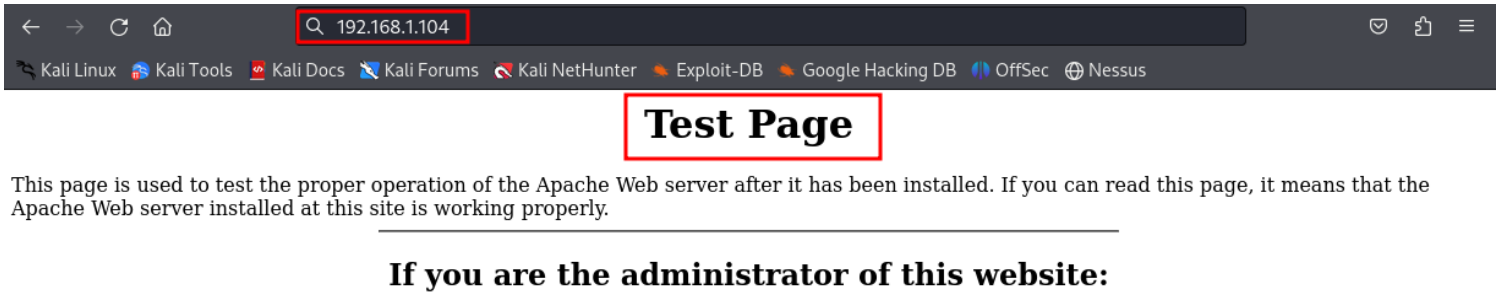
then you can fire up this version on searchsploit and find vulnerabilities, one of them is trans2open which can be performed on metasploit

then you can come on metsaploit and search trans2open, and open the trans2open exploit for linux machines since our kioptrix machine is linux, and we can get ourself a vulnerability through metasploit.

Vulnerabilities

Default Web Page (Low)

192.168.1.104



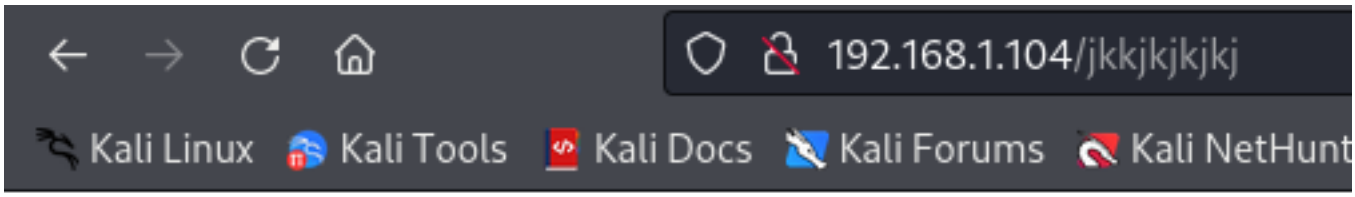
Server Header Info Disclosure (Low)

192.168.1.104

```
HTTP/1.1 200 OK
Date: Sat, 16 Mar 2024 19:27:21 GMT
Server: Apache/1.3.20 (Unix)  (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
Last-Modified: Thu, 06 Sep 2001 03:12:46 GMT
ETag: "8805-b4a-3b96e9ae"
Accept-Ranges: bytes
Content-Length: 2890
Connection: close
Content-Type: text/html
```

Default 404 Info Disclosure (Low)

192.168.1.104



Not Found

The requested URL /jkkjkjkjkj was not found on this server.

Apache/1.3.20 Server at 127.0.0.1 Port 80

Weak Ciphers (Low)

192.168.1.104

```
|       TLS_RSA_WITH_RC4_128_MD5 (rsa 1024) - F
|       TLS_RSA_WITH_RC4_128_SHA (rsa 1024) - F
|     compressors:
|       NULL
|     cipher preference: client
|     warnings:
|       64-bit block cipher 3DES vulnerable to SWEET32 attack
|       64-bit block cipher DES vulnerable to SWEET32 attack
|       64-bit block cipher DES40 vulnerable to SWEET32 attack
|       64-bit block cipher RC2 vulnerable to SWEET32 attack
|       Broken cipher RC4 is deprecated by RFC 7465
|       Ciphersuite uses MD5 for message integrity
|       Export key exchange
|       Insecure certificate signature (MD5), score capped at F
|_  least strength: F
```

SMB Anonymous Login Possible (Low)

192.168.1.104

```
Server does not support EXTENDED_SECURITY  but 'client use spnego = yes' and 'client ntlmv2 auth = yes' is se
t
Anonymous login successful
Password for [WORKGROUP\kali]:

        Sharename       Type      Comment
        ---------       ----      -------
        IPC$            IPC       IPC Service (Samba Server)
        ADMIN$          IPC       IPC Service (Samba Server)
Reconnecting with SMB1 for workgroup listing.
Server does not support EXTENDED_SECURITY  but 'client use spnego = yes' and 'client ntlmv2 auth = yes' is se
t
Anonymous login successful

        Server               Comment
        ---------            -------
        KIOPTRIX             Samba Server

        Workgroup            Master
        ---------            -------
        MYGROUP              KIOPTRIX
```