Ethical hacking in Cyber Security — by Cyber Mentor

About Pentesting — Part 1

External pentest - o these pentest are the most straight forward and it is looking at an organization's security from the ousside. the methodology for external pentest focuses heavily on what is called OSINT. These pentests tend to last 32-40 hours on average.

Internal Pentest → this is assessing an organization's security from the inside of the network. this means that we somehow breached the parameter perhaps we sent a phishing email and somebody opened and clicked on our link. and now we are inside the network. or maybe we broke into the building and left a dropbox behind.

⇒ what we ethically do on our end is we typically send a laptop out to the client the client plugs that laptop in and we are able to remote into that laptop and perform a network assessment as if we were sitting inside the office.

⇒ the methodology in internal pentest heavily focuses on active directory.

⇒ Impt. to learn about attacks on Active directory.

Web Application pentest → if you want to be web App pentester it is important to know about 'OWASP' guidlines. Methodology focuses on web based attacks and on OWASP.

Wireless Pentest ⇒ Methodology may vary acc. to what type wireless network being used for ex- if they are using a guest network, we might log on to the guest Network and test 'Segmentation.'

means can a guest access internal resources or is the network properly segmented if they are using a pre share key which is what's common in most household. we must test that pre-share key for password strength and see how strong the password and pre share key is.

⇒ if they are using enterprize based network then we open ourselves to variety of new attacks
- need a wireless network adapter.

SSID

Physical pentest & Social Engineering ⇒ In this we are accessing an organizations physical security. Our methodology is going to depend on the tasks and goals that are at hand that are given to us. by the client. In this we are going to the site and trying to break into the building. that can be through cloning badges and that can be through social engineering and picking locks.

⇒ Our methodology also depends on different scenerios and our client goals. client may say we want to see if you can just even get in the building or they might say they I want you to get into the building and find our server closet and take a picture of yourself in our server closet we want to see if you can make it there.

⇒ other side of that is pure social engineering and what i mean by that is doing something like a phishing campaign against an organization and we might do an phishing campaign in combination to the external pentest.

⇒ we might do a like a phishing campaign where we call people and we say hey I'm from I.T can you give me your password or I just sent you a code to your account can you give me the code that I just sent you, we might do a smishing campaign where we send txt messages and see if anybody responds back to us.

Report writing

⇒ To be really good at technical abilities
⇒ To be really good at report writing and communicating effectively (not just to a technical audience but to non-technical audience also)
⇒ Also have to be good at your presentation skills. in a sense that you need to present your findings to a tech and non-tech audience.
⇒ report should be delivered within a week after engagement.
⇒ report should be high-level and technical. So, we have an executive summary for non tech people think about a CEO who might not be a technical person. if they read the report it should be crystal clear about what the issues were and how they should be fixed.

Now, we have a technical findings section. and that's

for the people doing the work. maybe the security engineer, the network engineer. or web app developer depending on who you are working with. they can digest the findings they can say Here's what they found. Here is what the tools they used. here how they do it and here are the recommendations for remediation.

Debrief

⇒ A debrief walks through your report findings. this can be with technical and non technical staff present.

⇒ It gives an opportunity for the client to ask ques and address any concern befor a final report is released.

Networking Refresher

Important read

Introduction

- IP addresses
- MAC Addresses
- TCP, UDP and the three-way Handshake
- Common ports and protocols
- The OSI model
- Subnetting

⇒ Common ports and protocols

- TCP
  - FTP (21)
  - SSH (22)
  - Telnet (23)
  - SMTP (25)
  - DNS (53)
  - HTTP (80) | HTTPS (443)
  - POP3 (110)
  - SMB (139 + 445)
  - IMAP (143)
- UDP
  - DNS (53)
  - DHCP (67, 68)
  - TFTP (69)
  - SNMP (161)
    Simple network management protocol.

FTP → putting file in server and taking from server.

SMB → Based on file sharing (old - 139) (new - 445)

DHCP → provides ip addresses to devices

TFTP → Trivial FTP (utilizes UDP instead of TCP)

Kali linux Overview

__/__/____

- Sudo

Navigating the ~~final~~ file System.

pwd - Point working directory

cd.. → To go out from directory

ls → list of files and folder

cd ~ → goes back to home folder

cd /etc/ → to go into the folder by path

cd kali , cd home → to going into present directory

ls /etc/ → shows list of all files and folders in 'etc'

mkdir heath → to create new directory.

rmdir heath → to remove

ls - la → long all (show all files in detail including hidden files)

Website ⇒ explainshell.com

man ls

echo 'Hi!'	echo 'Hi' > test.txt Cat test.txt

⇒ cp test.txt Downloads
⇒ ls
⇒ ls Downloads
⇒ rm Downloads/test.txt
⇒ ls

⇒ mv test.txt Downloads
⇒ ls
⇒ ls Downloads

⇒ locate test.txt

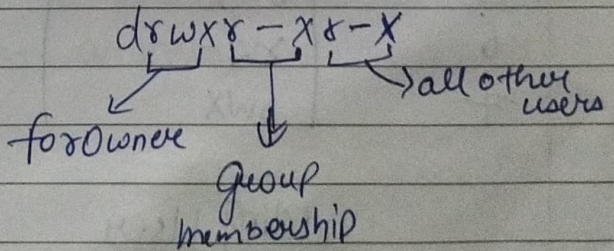
⇒ updatedb (sudo)

⇒ locate test.txt

⇒ Passwd

Users & privileges

→ when we run cmd- ls-la we will see something like this on the left side.



d → directories
r → read
w → write
x → execute

__/__/____

⇒ if we write a script but we cannot execute it until we have full access to do so.

changing permissions

⇒ echo "hello" > hello.txt

⇒ ls -la
(we will not have any execute permissions on hello.txt)

1. ⇒ chmod +rwx hello.txt
or
2. ⇒ chmod 777 hello.txt
⇒ ls -la

Chmod numbers

Number	Permissions	Totals
0	---	0+0+0
1	--x	0+0+1
2	-w-	0+2+0
3	-wx	0+2+1
4	r--	4+0+0
5	r-x	4+0+1
6	rw-	4+2+0
7	rwx	4+2+1

adding User	Switch User
⇒ sudo adduser john	⇒ su john

⇒ sudo cat /etc/shadow

⇒ sudo cat /etc/sudoers

⇒ grep 'sudo' /etc/group
↳ shows who has all privileges

grep → pulling out a particular string out of a file

Common Network Cmds

⇒ ip -a
give detailed info of your current ip address

⇒ ifconfig
is similar as ip-a but it is a old way.

⇒ iwconfig
for wireless connections

⇒ ip n (neighbour)
← alternative →
⇒ arp-a
(To find the ~~broa~~ ip address sends broadcast address)

⇒ ip r (route)
← alternate →
⇒ route
it will show us routing table which will tell us where our traffic is routing

⇒ ifConfig

⇒ Ping 192.168.138.2

Ping cmd Sends icmp requests but not all machines permit icmp traffic. just because we ping a machine and it doesnot respond doesnot mean that its not online. there are machines that have icmp disabled they will not respond to ping requests

⇒ netstat

tells us what open ports and services are there.

Viewing, Editing and Creating files

⇒ echo "hello again" > hey.txt

⇒ cat hey.txt
hello again

⇒ echo "hello again again" >> hey.txt

⇒ cat hey.txt
hello again
hello again again

> → this will overwrite >> → this will add txt

Another way of making new file

⇒ touch newfile.txt

⇒ cat newfile.txt

So, we can use a different type of editor to try and edit this and save the file.
like nano, vim, vi

we will use nano.

⇒ nano newfile.txt

(text editor will open)
type anything in it. → then press ctrl-X
→ Y → Enter to save.

⇒ cat newfile.txt

we also have an graphical notepad. we can access that by

⇒ mousepad newfile.txt

(we can write and edit anything in it. it is like a windows notepad).

⇒ cat newfile.txt

we can also create a newfile using nano

⇒ nano brandnewfile.txt
⇒ cat brandnewfile.txt

Starting and stopping services

⇒ ifconfig

→ copy your own ip address
→ open firefox → navigate to ip address

1. ⇒ sudo service apache2 start
→ now go back to browser → change Https to http and press enter with your ip address on it.

We will get a webpage "Apache 2 Debian Default Page"

if we go at the location of the webpage

var/www/html we will find the page data in our file system.

⇒ sudo service apache2 stop

2. ⇒ python3 -m http.server 80

Ctrl+c → To shutdown the server

3. Suppose we want to start a service whenever we start our machine.

⇒ sudo systemctl enable ssh

⇒ sudo systemctl disable ssh.

"Installing and updating tools"

⇒ sudo apt update && apt upgrade

⇒ Switch user to root
⇒ sudo su -

⇒ sudo apt update && apt upgrade

⇒ apt install cron-daemon-common.

(Sometimes fully upgrading systems can broke some data so, if we want install any tool, we can do it by using above cmd.)

⇒ Downloading tools using git and github

Just write pimpmykali on browser.
→ You will get a github link.
Click the link → code → copy link

⇒ su kali

⇒ cd /opt

⇒ sudo git clone "paste here"

⇒ cd pimpmykali
⇒ ls

You can find cmd to run tools in their profile. and copy paste them to use.

⇒ Sudo ./pimpmykali.sh

press N

Bash Scripting

Making a ping swipper

⇒ ifConfig

= copying IPaddress

⇒ Ping "Ipaddress"

when we are pinging the address and we're getting some databack.
if we get details back it means the address is active. forex →
64 bytes from 192.168.4.29 : icmp_seq=1 --- ms
this type of reply will be show if the address is active

⇒ Ping 192.168.41.29 -c 1

it will send request only one time.

this is a non-active address
that's why it will not return anything.

(active)

⇒ We will ping our own address^ only one time and then save it in a txt file.

⇒ ping "your ip" -c 1 > ip.txt

⇒ cat ip.txt

if we want to attract only line from the text file we will use grep cmd for that for ex-;

⇒ cat ip.txt | grep "64 bytes"
64 bytes from 192.168.4.29 : icmp_seq = 1 ttl = 128 -- ms.
↑
grep will give whole related to the phrase given.
Now, why we are extracting the line above?
⇒ for building a ping swipper, we need to scan all the networks in a subnet. So, we will gonna ping each and every subnet. then we will extract all the active ip addresses in a txt file.

⇒ we need to grep only the ip address back.

⇒ cat ip.txt | grep "64 bytes" | cut -d " " -f 4

cut ↑ (removing) something from the line
-d ↓ for Cutting delimiter
-f 4 ↓ Count = 4

⇒ Now, we will get the specific ipaddress like this 192.168.4.29:

↑ for removing this Semicolon

⇒ cat ip.txt | grep "64 bytes" | cut -d " " -f 4 | tr -d ":"

192.168.4.29

↑ translate

Now, copy this above line.
then, we will go to the mousepad

⇒ mousepad ipsweep.sh.

Now, we are in mousepad.

#!/bin/bash

// declaring that it's a bash script

⇒ now paste the line you copied above in the next line,

cat ip.txt | grep "64 bytes" | cut -d " " -f 4 | tr -d ":"

change this to this →

ping -c 1 192.168.4.x | ———— ":"

Now, we have write a forloop for continuosly changing the value of last octet of ip address from 1 to 254.
Now, our program will look like ⇒

```
#!/bin/bash
for ip in `seq 1 254` ; do
ping -c 1 192.168.4.$ip | grep ---------- ":" &
done
```

```
./ipsweep.sh $1 $2
$0 ↗    ↑  ↗
```

We can provide diffrent argument values to our program like this.

In above program we will write $1. Instead of 192.168.4, do, now, the program will be like.

```
for ip in `seq 1 254` ; do
ping -c 1 $1.$ip| ---------- ":" &
done
```

it place it above.

⇒ (./ipsweep.sh 192.168.4) dont write it in code

this helps in specifying multiple networks easily.

Now, we will run it.

⇒ chmod +x ipsweep.sh
(it will give execution permission to it.

⇒ ls -la (to check it got the permission)

⇒ ./ipsweep.sh 192.168.4.
(it is cmd to run the program now, it will return all the active ip addresses)

⇒ when we are running our program it is showing unusual things to us if we don't give it any agument like ./ipsweep.sh only.
So, we will ~~make~~ fix it.

⇒ mousepad ipsweep.sh
Now, the new updated program will be:-

```
#!/bin/bash
if ["$1"=="" ]
then
echo "You forgot an IP address!"
echo "Syntax: ./ipsweep.sh 192.168.4"

else
for ip in `seq 1 254`; do
ping -c1 $1.$ip | ____ ":" 8
done
fi
```

Now try and train it

⇒ ./ipsweep.sh 192.168.4

Now, we will run this cmd and store the ips in a txt file
⇒ ./ipsweep.sh 192.168.4 > ips.txt

⇒ cat ips.txt
⇒ we can use this program in nmap Scanning
$ for ip in $(cat ips.txt); do nmap $ip $done
this will run the nmap scanning one by one on every ip and can help in automation of the process

Intro to python

Strings

⇒ open terminal, then make a directory and go into that directory.

⇒ mkdir python
⇒ cd python
clear

⇒ mousepad first.py &

& will make sure that terminal will also available to us when we open mousepad.

Now in mousepad

#!/bin/python3 (calling out the directory)

print ("Hello, world!")

Now run it.

⇒ Python3 first.py
it will print Hello World.

⇒ we can add comments also like (using #)

Print string

⇒ we can write string in multiple lines by """ ___ """

__/__/____

⇒ print ("""This string runs
in multiple lines!""")

⇒ print ("This String is " + "concatenated")

we can also concatenate them like that

⇒ print ('\n') # new line

Math

print (50+50) # add
print (50-50) # subtract
print (50*50) # multiply
print (50÷50) # divide.

print (50**2) # exponent
print (50 % 6) # modulo

print (50/6) # division with remainder (or a float)
print (50//6) # no remindee

variables and methods

quote = "All is fair in love and war."

↓ we are storing this string in a variable

print (quote)

Methods

are just functions that are available for a given object now you can think of functions as something built in to python that allows us to do something.

```
print (quote.upper()) # uppercase
print (quote.lower()) # lowercase
print (quote.title()) # title case

print (len (quote)) # counts characters

name = "Heath"  # string
age = 30        # int
gpa = 3.7       # float

print (int(age))
print (int(30.1))
print (int(30.9))

print ("My name is " + name + " and I am " +
       age + " years old.")
```

⇒ this above line will give error because we cannot add string to int directly. In corrected version, we should use str(age).

```
age += 1          (incrementing age by 1)
print (age)

birthday = 1
age += birthday
print (age)
```

Functions

They are like mini programs what they are is an organized block of code that you define and you can call it here instead of repeatedly typing the code.

```
⇒ def who_am_i(): #this is a function without parameters
      name = "Heath"
      age = 30
      print("My name is " + name + " and
          I am " + str(age) + " years
          old.")

= who_am_i()   # calling function
  print(age)

⇒ def add_one_hundred(num): #function with
      print(num+100)          parameters.

  add_one_hundred(100)

⇒ def add(x,y):
      print(x+y)

  add(7,7)

⇒ def multiply(x,y):        we have used return
      return x*y            instead of print.

  multiply(7,7)             It knows that ans is 49
                            but it is not able to print
print(multiply(7,7))        because Print function has
                            not passed in it. So, we will use
```

```
def square_root(x):
    print(x**.5)

square_root(64)
```

```
def nl():  # printing new line
    print('\n')

nl()
```

Boolean Expressions and Relational operators

↓

True and false

```
bool1 = True
bool2 = 3*3 == 9
bool3 = false
bool4 = 3*3 != 9

print(bool1, bool2, bool3, bool4)

print(type(bool1))

bool5 = "True"
print(type(bool5))
```

```
# Relational and Boolean operators
greater_than = 7>5
less_than = 5<7
greater_than_equalto = 7>=7
less than equal to = 7<=7

test_and = (7>5) and (5<7) # True
    In 'and' both statements should be true.
test_or = (7>5) or (5>7) # True
    In 'or' One of the statement should be true (atleast)
                                    atleast.

test_not = not True
# false
```

Conditional Statements (if/else)

```
def drink(money):
    if money >=2:
        return "You have got drink"
    else:
        return "No drink for you!"

print(drink(3))
print(drink(1))

def alcohol(age, money):
    if (age >=21) and (money >=5):
        return "We're getting a drink!"
    elif (age>=21) and (money<5):
        return "Come back with more money"
    elif (age<21) and (money>=5):
        return "Nice toy, kid!!"
    else
        return "No money no age"

print(alcohol(21,5))
print(alcohol(21,4))
print(alcohol(20,5))
print(alcohol(20,4))
```

Lists

```
movies = ["When Harry Met Solly", "The hangover",
          "kick", "The exorcist"]

print(movies[1])  # prints the second item in the list
print(movies[0])  # prints the first item
print(movies[1:3])  # return the first index element and returns all the elements till the second index but not second index. (in above 1 and 2 will be returned only).

print(movies[1:])  # prints all the number from position till the end.

print(movies[:1])  # prints only the element at 0th index.
print(movies[-1])  # returns the last element

print((len(movies)))  # return items count
print(f
movies.append("Jaws")
print(movies)

movies.insert(2, "Hustle")
print(movies)
movies.pop()  # removes the last element
movies.pop(0)  # removes first element
print(movies)

amber_movies = ['Just Go with it', '50 first Dates']
Our_fav_movies = movies + amber_movies
Print(Our_fav_movies)
```

~~bobs~~ grades = [["bob", 82], ["Alica", 90],
["Jeff", 73]]
bobs_grade = grades[0][1]
print(bobs_grade) #2D list
grades[0][1] = 83 #changing bob's grade.
print(grades).

Tuples → Do not change, (), immutable

grades = ("a", "b", "c", "d", "e", "f")

we cannot append, pop, change values in tuples
like grades.pop() will not work.
print(grades[1])

looping

# for loops - start to finish of an iterate
Vegetables = ["cucumber", "Spinach", "Cabbage"]
for x in vegetables:
    print(x)

#while loops - execute as long as true
i=1
while i<10:
    print(i)
    i+=1.

Advanced Strings

also immutable

```
my_name = "Heath"
print(my_name[0]) # first letter
print(my_name[-1]) # last letter

sentence = "This is a sentence."
print(sintence[:4])
print(sintence.split()) # delimeter - default is a space

sentence_split = sentence.split()
sentence_join = ' '.join(sentence_split)
print(sintence_join)

quote = "He said, 'give me all your money.'"
print(quote)
quote = "He said, \"give me all your money.\""
Print(quote)

too_much_space = "          hello          "
print(too_much_space.strip())

print("A" in "Apple") # True
print("a" in "Apple") # fake

letter = "A"
word = "Apple"
print(letter.lower() in word.lower()) # improved

movie = "The Hangover"
print("My favorite movie is {}.".format(movie))
print("My favorite movie is %s." % movie)
print(f"My favorite movies is {movie}.")
        ↑ mainly used.
```

__/__/__

Dictionaries

- key/value pairs {}

```
drinks = {"Old monk": 8, "Royal stag": 10, "Bud": 7}
    # drink is the key price is the value
print(drinks)

employees = {"finance": ["Bob", "Linda"], "IT":
        ["Gene", "Louise",], "HR": ["Jimmy",
         "Mort" ]}
print (employees)

employees['Legal'] = ["Mr. fond"]  # adds new
print (employees)                  key-value pair

employees.update ({"sales": ["Andie", "Ollie"]})
                    # adds new key-value pair
print (employees)

drinks['old monk'] = 9
print (drinks)

print (drinks.get ("Royal stag"))
```

Importing Modules

⇒ mousepad importing.py$ # creating Python file to write Script.

⇒ #!/bin/python3
⇒ import sys # System functions and parameters
⇒ from datetime import datetime as dt

⇒ print (sys.version)
⇒ print (dt.now()).

Sockets

→ can be used to connect two nodes together

⇒ mousepad S.py &

⇒ #!/bin/python3

→ we are gonna use sockets to connect ports and ip addresses

⇒ import socket

⇒ Host = '127.0.0.1'
⇒ PORT = 7777 # taking any randomport to connect

⇒ S= socket.socket(socket.Af_INET, socket.SOCK_STREAM) # af_Inet is IPv4, Sock_stream is a port.
⇒ s.connect ((HOST, PORT))

⇒ In this case, we are connecting our own IP address (loopback address) to a random port 7777.

Netcat

→ it allows to connect to open ports or establish a listener on an open port so.

run in Cmd line (Terminal)
⇒ nc -nvlp 7777
↑
opening a l-listener on a p-port. (7777)

Note ⇒ there has to be two ports connecting together one on each side in a network.

Building a port Scanner

⇒ mousepad scanner.py & (Terrible port scanner)

→ #!/bin/python3

⇒ import sys
⇒ import socket
⇒ from datetime import datetime

Python3 scanner.py 192.168.1.1 only two arguments allowed not less, not more

# define our target
if len(sys.argv) = 2:
# translates hostname to ip address (IPv4)
target = socket.gethostbyname(sys.argv[1])
else:
print("Invalid amount of arguments.")
print("Syntax: python3 scanner.py <ip>")

⇒ it will go out and do port scanning on the IP address provided.

#add a pretty banner

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

we are gonna try something if it works it's okay
Otherwise we will throw exceptions :-

Total no. of ports = 65,535
there is no need to scan all those ports which will make our program slow.
prefered scan is from 50 to 85.

```
try:
    for port in range(50, 85):
        s = socket.socket(socket.AF_INET,
                          socket.SOCK_STREAM)
        socket.setdefaulttimeout(1)
        # so, if it responds back or doesn't respond
        back with in a second we're just going to move
        on, we don't want to ~~wait~~ kept waiting.
        result = s.connect_ex((target, port))
        if result == 0:
            print(f"port {port} is open")
        s.close()
```

error indicator

(if a port is open error result return 0 and if a port is closed it returns 1)

before we run it, there are some exceptions that we need to consider.

```
except KeyboardInterrupt:
        print ("\n Exiting Program.")
        sys.exit()

except socket.gaierror:
        print ("Hostname could not resolved.")
        sys.exit()

except socket.error:
        print ("could not connect to server.")
        sys.exit()
```

Note. When we are on a virtual machine and we run cmd ifConfig and then that ip address that is shown there, that is not our true ip address.

So, for that we will open our windows machine copy your default gateway

then write cmd in linux terminal:

⇒ python3 Scanner.py 192.168.4.1
(your default gateway)

⇒ So, now our Scanner will check each and every port for that default gateway.

it will show you all the open ports.

User Input

- Create a new file. → mousepad input.py

```
#!/bin/python3

name = input("Enter your name: ")
drink = input("what's your fav drink: ")
print("Hello " + name + "! have a " + drink + ".")
```

run it by ⇒ python3 input.py

making a mini calculator

```
x = input("Give me a num: ")   # 3
y = input("Give me another: ") # 2
```

print (x+y) it will gives us 32 because it is taking input as a string. So, either we need to take its integer or float. So, new program will be like.

```
x = float(input("Give me a number:"))
y = float(input("Give another: "))

print(x+y)
```

now, run it python3 input.py.

Now, asking user for the operation they want to done for two numbers.

⇒

```
#!/bin/python3

x = float(input("enter first num: "))
y = float(input("Enter second num: "))
o = input("Give me the operator: ")

if o == "+":
    print(x+y)
elif o == "-":
    print(x-y)
elif o == "/":
    print(x/y)
elif o == "*":
    print(x*y)
elif o == "**" or o == "^":
    print(x**y)
else:
    print("Unknown Operator")
```

2^2 = 2**2 = 2²

Save it & run it by ⇒ python3 input.py

Reading and writing files

⇒ creating a txt file months.txt

⇒ mousepad months.txt

We will write months

1. January
2. febrawry
}
12. December

Make a new file

⇒ mousepad readwrite.py

```
#!/bin/python3
months = open('months.txt')
print(months)
print(months.mode)
print(months.readable())
months.close()
months.read()
print(month.readline())
print(month.readline())
```

To close the file.

it will output all the months (or text written in that txt file)

Gives first line as output only

↑ Gives 2nd Line as output if used second time.

Python3 readwrite.py

diff. of s from above

if we use ⇒ print(month.readLines())
It will print all the text as in form list and in one line.

if we use it second time ⇒ print(month.readlines())
it will give us empty list. [] like this.

if we use months.seek(0), pointer goes back to 0th index position

Ex.
```
print(month.readLines())
months.seek(0)
print(months.readLines())
```

↑ how it will print both times the whole list.

For else we can also use for loop

```
for month in months:
    print(month)

months.close()
```

it will print like

January
(gap b/w)
febrary
↓
December

if we would want it like → January ← without gap
we can use
print(month.strip())
febrary
|
December.

r → read w → write a → append

```
#!/bin/python3

days = open('days.txt', "a")
```

↑ gives permission to append (add) to the existing ~~str~~ string.

"w" → overwrite on the existing string
"r" → gives permission to read only.

```
days.write("Monday")
```

↑ then change it to "\n tuesday"

```
days.close().
```

⇒ Python3 readwrite.py
⇒ cat days.txt.

Classes and Object

⇒ mousepad Employees.py &

```
Class Employees :
    def __init__(self, name, department, role,
                 salary, years_employed):
        self.name = name
        self.department = department
        self.role = role
        self.salary = salary
        self.years_employed = years_employed.
```

methods

Init function ⇒ • all classes have a func. called the Init function and this is always executed when the class is being initiated.

• we are gonna use this Init func to assign value to object properties.

⇒ Now, we will make a new python file.

⇒ mousepad Newemployees.py &

⇒ #!/bin/python3

Now, we will import the class we have built.

⇒ from Employees import Employees

So, we can make couple of employees here.

```
e1 = Employees("Bob", "Sales", "Director of Sales",
               100000, 20)
```

e2 = Employees ("Linda", "Executive", "(10", 150000, 10)

print (e1.name)
print (e2.salll)

So, we can build upon more things in our class - Employees.
like if we want to make another function ⇒

```
def eligible_forretirement (self):
    if self.year_employed >= 20:
        return true
    else:
        return false
```

Now, in ouremployees file.

print (e1.eligible_for_retirement())

Building a shoe budget tool.
→ using classes and objects.
and we are gonna see what shoes we can afford based on how much money we have.

Making a new python file.

⇒ mousepad shoes.py &

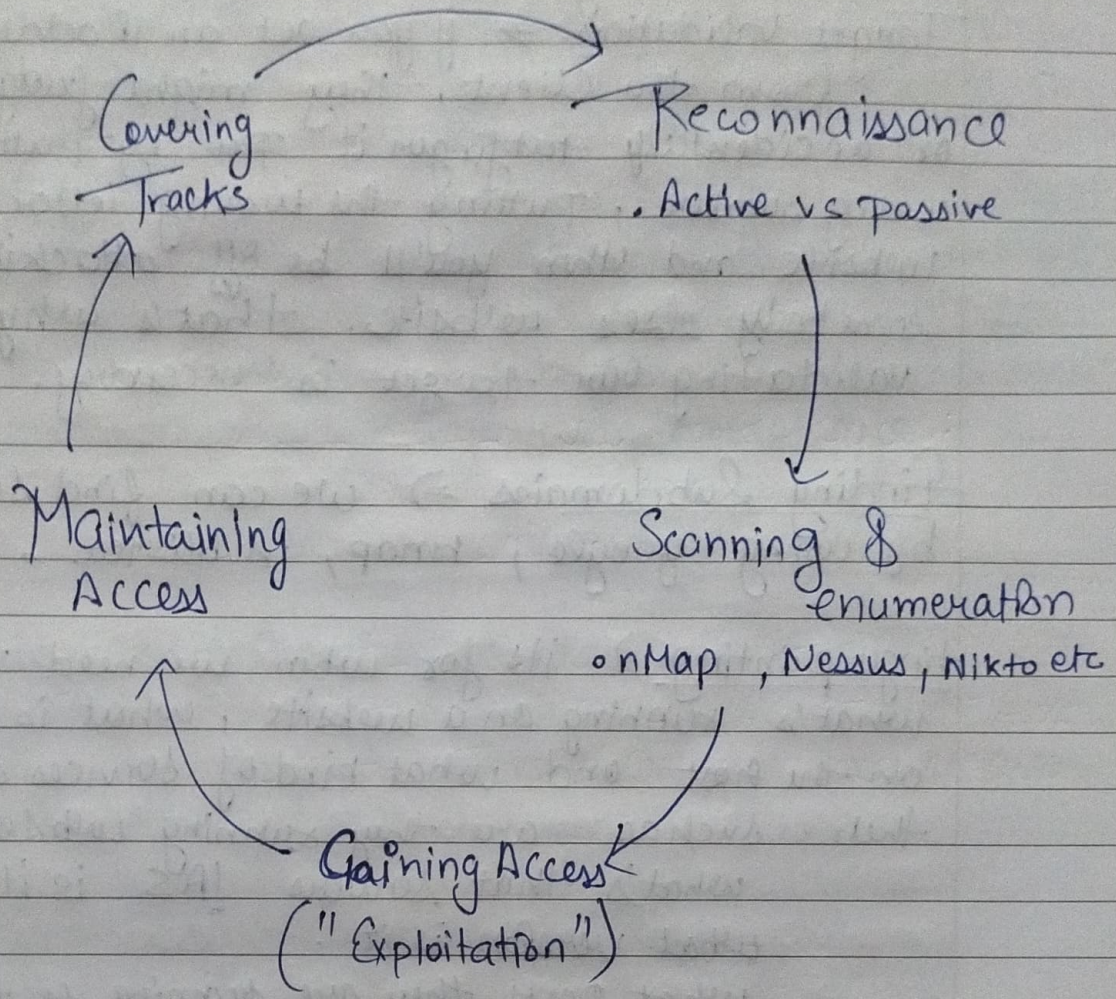
```
class Shoes:
    def __init__(self, name, price):
        self.name = name
        self.price = float(price)

    def budget_check(self, budget):
        if not isinstance(budget, (int, float)):
            print('Invalid entry. please enter a
                  number')
            exit()
    def change(self, budget):
        return(budget - self.price)

    def buy(self, budget):
        self.budget_check(budget)

        if budget >= self.price:
            print(f'You can cop some
                  {self.name}')
            if budget == self.price:
                Print('You have exactly enough money.')
            else:
                Point(f'You can buy and have ${
                self.change(budget)} left over')
        exit('Thanking for using')
```

⇒ Now, make a newfile name shoepurchase.py

⇒ mousepad shoepurchase.py &

```
#!/bin/python3

from shoes import shoes:

low = Shoes('And 1s', 30)
medium = Shoes('Air force 1s', 120)
high = Shoes('Off whites', 400)

try:
    shoe_budget = float(input('what is your budget'))
except ValueError:
    exit('Please Enter a number')

for shoes in [high, medium, low]:
    shoes.buy(shoe_budget)
```

five Stages of ethical Hacking



Passive Recon Overview

- ~~I~~nformation Gathering

⇒ Physical Information ⇒ the info we can get by going on the location and gather information. ex → looking for the smoking area outside particular company office and interacting with their employees and tailgating with them inside the office.

→ Job information ⇒ looking for companies employee online. and we might want to know their title, ph. no., their manager, we should also look for pictures. any kind of picture that can give info about Job.

Listen to Darknet Diaries

__/__/__

assessment.

Web/Host if we get a web/host, we will be doing ⇒

Target validation ⇒ if you get an IP address from the client, they might fudge it or accidentaly fatfinger it by putting the wrong number, putting the wrong letter in the website and then you'll be off attacking somebody else's website. that's why validating our target is necessary.

finding Subdomains ⇒ we can find subdomains by using google, nmap, sublister.

fingerprinting ⇒ its for when we need to know what's running on a website, what is running on the host and what kind of services are out there. such as are they running web server, what's that servers IAS is it Apache what version is it.
what ports they are running on the machine is FTP open, and what version.

Note ⇒ when we use tools and started scanning websites and our target it becomes active scanning unless its is passive.

Data breaches ⇒ we're talking about breached incident happens in past that have leacked data. that are like Home depot, Equifax, Linkedin all breaches that are out there that have credentials dumped and those credentials become available to us and we try to utilize them to gain access.

Target Validation → whois, nslookup, dnsrecon

finding Subdomain → Google fu, dig, Nmap, Sublister, bluto, crt.sh

finger printing → Nmap, Wappalyzer, What-Web, Built with, Netcat

Data Breaches → Have I been Pawned, Breach-parse, We Leak info.

Identifying Our target ⇒

before we begin reconnaissance we have to establish our client to attack.

Go to Bug Crowd ⇒ "Tesla"

→ Programs (if you dont find Tesla, you can perform Information Gathering on any program)

Discovering Email Addresses ⇒

Kali linux → firefox → hunter.io → search any company name → like tcm-sec.com / tesla.com
→ You will find email address of different employees acc. to their job titles.

1. hunter.io 2. Phonebook.cz

3. Voilanorbert.com 4. Clearbit.com
Clearbit connect

⇒ After that, we will take the email and we can verify this at tools. verifyemailaddress.io if email is working or not.

1. ⇒ tools. verify email address. io

emailHippo

2. ⇒ email-checker.net/validate

⇒ Don't Underestimate "forgot Password".

Suppose you want to find the owner of a particular email address or to whom that email address is linked. you can go to gmail login and write "pleasedonthackmesirplz @gmail.com" and then forgot password and then click on try another way. you will get some email-id like. "h****@tc****.com"

So, this is the email id of the person whose ~~that is~~ account is linked to that account that we have tried to login.

"pleasedonthackmesirplz @gmail.com"

Breached Credentials - part 1

⇒ Gathering Breached credentials w/ Breach Parse

⇒ go to github and search

https://github.com/hmaverickadams

⇒ click on breach-parse
You can run it by using cmd's

⇒ cd /opt/ breach-parse /
⇒ ~~opt~~ ./ breach-parse.sh @tesla.com tesla.txt # we have entered the file

⇒ Storing data in tesla.txt

⇒ cat tesla-

⇒ gedit tesla-master.txt (it will show the saved email-and passwords)

Breach-parse → is a tool which helps us in gather information ~~on~~ like credential stuffing. it collects the leak data from the website that stores that data. A tool for parsing breached passwords. This tool searches in a whole data and basically gives us the data we need.
for ex → . Email ids with their passwords.

Breached Credentials Part 2

Hunting

Go to Dehashed.com (Paid)

Suppose, we got a hashed password somehow of someones email address like.

Email : bob@tesla.com
Hashed password : bztdOAjUWGU=

We will go to "hashes.org" so, we can search hashes on that.
to dehash them but sometimes it won't works.
So, now, we will go to google again and search for hash password on google.

Part 1 Complete

Part - 2 ⇒