

Binghamton University, Watson School of Engineering

Stage 2:

Prevention against 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  
October 21st, 2024

## Contents

1. Setting up the Environment for Prevention .....	
1.1 Recap of the Evil Twin Attack .....	2
1.2 Prevention Objective .....	2
1.3 Host Machine Setup (Attacker and Victim).....	2
2. Theoretical Prevention Techniques .....	
2.1 Hardening the Windows Wireless Configuration .....	3
2.2 Implementing Strong Authentication and Encryption Techniques.....	4
2.3 Using Rogue Access Point Detection Tools.....	4
2.4 User Education and Awareness .....	5
2.5 Configuring MAC Address Filtering and Network Segmentation.....	5
2.6 Disabling Wi-Fi Protected Setup (WPS).....	6
2.7 Advanced Wireless Intrusion Prevention Systems (WIPS).....	6
2.8 Leveraging VPN for Secure Communication.....	6
3. Evil Twin Attack Prevention Implementation (Practical).....	
3.1 Overview of Practical Script Implementation .....	7
3.2 Python Script for Prevention .....	8
3.3 Script Explanation and Functionality .....	9
3.4 Testing and Results .....	9
4. Conclusion and Future Work .....	12
5. Bibliography .....	13

# **1. Setting up the Environment for Prevention**

## **1.1 Recap of the Evil Twin Attack**

In the first stage of this project, I successfully performed an Evil Twin attack to exploit a vulnerability in wireless networks. Using two USB Wi-Fi adapters with the Atheros AR9271 chipset on Kali Linux, I created a rogue Wi-Fi access point, named "FreeWiFi," to trick a Windows victim into connecting. Using tools like `airmon-ng`, `aireplay-ng`, and a Python script hosting a fake captive portal, I captured sensitive user credentials, including usernames and passwords. The attack worked by exploiting the inability of most devices to distinguish between a legitimate and a rogue access point based solely on the SSID.

This second phase aims to address the vulnerabilities identified during the first stage by implementing theoretical and practical prevention methods against the evil twin attack.

## **1.2 Prevention Objective**

The objective is to implement preventive measures that can minimize the risk of a successful Evil Twin attack. This involves both theoretical approaches to secure wireless networks as well as practical, automated techniques to recognize and respond to potential threats in real-time.

## **1.3 Host Machine Setup (Attacker and Victim)**

For this stage, the attacker's machine runs Kali Linux, while the victim's machine is Windows OS, and is monitored for any suspicious Wi-Fi connections.

## 2. Theoretical Prevention Techniques

### 2.1 Hardening the Windows Wireless Configuration

In order to make your wireless access point and network configuration less vulnerable to attacks, you need to harden your wireless network. Prevention can include measures such as:

- **Disabling SSID Broadcasting:** Disabling SSID broadcast prevents the network name from being displayed publicly, which reduces the chances of an attacker identifying and mimicking the network.
- **Changing Default SSIDs and Passwords:** Default credentials are well-known to attackers from the manufacturers. Changing these makes the network harder to identify and exploit.
- **Use Complex SSIDs and Passwords:** Unique and complex SSIDs reduce the chance of attackers mimicking easily recognizable names, such as "FreeWiFi", "PublicHotspot", etc.
- **Reducing Transmission Power:** Limiting the power of the Wi-Fi signal can reduce the coverage area, making it more difficult for attackers outside the intended area to detect and attack the network.
- **Create Network Profiles:** Set up network profiles for known and trusted SSIDs. These profiles will contain specific security settings that limit the possibility of inadvertently connecting to a malicious network.

Windows devices often have an automatic connection feature that can make them vulnerable to Evil Twin attacks.

- **Disable Auto-Connect:** Ensure the Windows machine does not automatically connect to any open network by modifying the wireless adapter settings. Turn off the "Connect automatically" option for all non-trusted Wi-Fi networks.
- **Forget Unsecured Networks:** Regularly forget all networks not needed by the user, particularly open or public networks. This prevents the Windows machine from inadvertently connecting to a rogue access point created by an attacker using Kali Linux.

These measures are foundational in preventing Evil Twin attacks by making the network more difficult to impersonate.

## 2.2 Implementing Strong Authentication and Encryption Techniques

Using strong authentication and encryption protocols is crucial for preventing unauthorized access to wireless networks:

- **WPA3 Encryption:** When possible, use WPA3 encryption to secure the wireless network. WPA3 provides significantly stronger encryption standards, making it more challenging for Kali Linux tools to successfully launch an Evil Twin attack.
- **802.1X Authentication with RADIUS Server:** Implementing 802.1X with a RADIUS server adds an extra layer of authentication, ensuring that devices are properly verified before gaining access to the network. This prevents rogue devices from easily connecting.
- **Disabling Legacy Protocols:** Ensure that the router and Windows machine are configured to use WPA2 or WPA3 only. Disabling older protocols like WEP or WPA will prevent attackers from using simpler, brute-force methods available in Kali Linux to compromise the network.
- **Using Strong, Unique Passwords:** Secure passwords, which are long and complex, are essential to prevent brute-force attacks or guessing of network credentials.
- **Enable 2FA for Network Access:** Implement two-factor authentication for sensitive networks. Even if an attacker creates a rogue access point and tricks the victim into connecting, they would still need to pass an additional authentication step, reducing the risk.

These encryption and authentication techniques make it more challenging for an attacker to gain unauthorized access, thus preventing potential Evil Twin attacks.

## 2.3 Using Rogue Access Point Detection Tools

Regular monitoring and auditing of the network can help identify potential security breaches early on, while detection tools are used to identify rogue access points, hence preventing an evil twin attack:

- **Install a Rogue Access Point Detection Tool:** Utilize tools on the Windows machine to detect suspicious SSIDs that may be acting as rogue access points. Applications like **Kismet**, and **AirSnare** can identify rogue access points and alert users if they attempt to connect to an unknown or suspicious SSID.
- **Windows Firewall:** Configure the Windows firewall to block traffic from unknown sources. The firewall will help prevent communication with the rogue access point, mitigating the damage even if a connection occurs.
- **Use Windows Defender ATP:** Utilize Windows Defender Advanced Threat Protection (ATP) to monitor network behavior and block unusual activity that could be linked to a rogue access point.

- **Automated Alerts:** Setting up alerts for unusual activities, such as de-authentication packets or rogue SSIDs, can help network administrators respond to potential attacks quickly.

The combination of these tools and regular monitoring gives us insight into network security, allowing us to prevent attacks like Evil Twin in advance.

## 2.4 User Education and Awareness

User behavior is often the weakest link in network security. Educating users to recognize potential risks and adopt safe practices is crucial:

- **Educate Users to Avoid "Free" Wi-Fi:** Users must be educated on the risks of connecting to "Free Wi-Fi" networks. They should learn how to verify the legitimacy of an SSID before connecting, particularly in public areas.
- **VPN Usage:** Educate users to always use a Virtual Private Network (VPN), especially when connecting to public Wi-Fi. This ensures that even if they mistakenly connect to an Evil Twin network, their data is encrypted, reducing the effectiveness of data theft attempts by Kali Linux attackers.
- **Disable Wi-Fi Adapter:** Encourage users to turn off the Wi-Fi adapter on their Windows machine when they are not actively using the internet. This prevents automatic connections to potentially malicious access points.

## 2.5 Configuring MAC Address Filtering and Network Segmentation

- **Whitelist MAC Addresses:** Implement MAC address filtering on the router, which allows only pre-approved MAC addresses to connect to the network. While MAC spoofing is possible in Kali Linux, this adds a step for the attacker to bypass.
- **Create Separate Network Segments:** Separate guest and corporate networks using **Virtual Local Area Networks (VLANs)**. This prevents attackers from accessing sensitive resources if they manage to lure a victim onto a rogue access point in a guest network segment

By isolating different parts of the network and filtering out the devices, attackers are prevented from moving laterally, thereby reducing the overall impact of an Evil Twin attack.

## 2.6 Disabling Wi-Fi Protected Setup (WPS)

Wi-Fi Protected Setup (WPS) is often used to make network connections easier, but it has significant security flaws:

- **Turn off WPS:** On the wireless router, disable Wi-Fi Protected Setup (WPS). WPS is highly vulnerable to brute-force attacks, particularly from tools included in Kali Linux, such as aircrack, reaver, etc. Disabling WPS removes this weak point.

## 2.7. Advanced Wireless Intrusion Prevention System (WIPS)

- **Deploy WIPS for Active Monitoring:** Use Wireless Intrusion Prevention Systems (WIPS) to detect and respond to potential Evil Twin attacks. WIPS can monitor the RF spectrum and automatically disconnect any rogue access points set up by Kali Linux, taking action before the Windows device can connect.
- **Automated Alerts and Actions:** Configure the WIPS to trigger automated alerts and mitigation actions, such as de-authenticating suspicious devices or notifying network administrators when a rogue access point is detected.

## 2.8. Leveraging VPN for Secure Communication

- **Mandatory VPN on Public Networks:** Use a VPN when using a public or potentially insecure wireless network. VPN encryption ensures that, even if a Windows device connects to an Evil Twin, the data remains unreadable to the attacker using tools from Kali Linux.

### 3. Evil Twin Attack Prevention Implementation (Practical)

#### 3.1 Overview of Practical Script Implementation

The Python script **prevent\_evil\_twin.py** prevents connections to rogue Wi-Fi networks by continuously monitoring the connected SSID on the victim (Windows) machine. If the victim machine connects to an SSID that is not listed as a trusted network (in this case, Trusted SSID - 'FBI Surveillance Van'), the script will immediately disconnect from the network. Furthermore, the script prevents me from connecting to untrusted networks in the first place, eliminating the risk of connecting to a rouge access point and preventing the evil twin attack altogether. This prevention technique is useful to stop an Evil Twin attack by ensuring the victim does not remain connected to the rogue access point if connected by mistake as well. The key idea is that the attack is only fully executed once the victim enters their credentials into a fake captive portal. Simply connecting to a rogue access point does not mean the attack has succeeded. Therefore, even if the victim mistakenly connects to a malicious network, the script prevents the attack from advancing by disconnecting the machine before any sensitive data can be compromised. This approach emphasizes stopping the attack before it begins, rather than merely responding after damage has occurred.

To prevent the attack from the victim's perspective, repeat the attack from the attacker's machine.





### 3.2 Python Script for Prevention

To demonstrate an automated method of prevention, I have created a Python script called `prevent\_evil\_twin.py`. The script utilizes system commands to monitor the network environment and detect if the system connects to a suspicious or untrusted Wi-Fi network. If such an instance is detected, the script immediately blocks / disconnects from the network, also, if it attempts to connect to an untrusted wifi network, it blocks that as well, thus mitigating the risk of an Evil Twin attack.

```

prevent_evil_twin.py X
C:\Users\yashu> OneDrive > Desktop > Evil_Twin_Prevention > prevent_evil_twin.py > ...
1  import subprocess
2  import time
3  import re
4
5  # List of trusted SSIDs
6  TRUSTED_SSIDS = ['FBI Surveillance Van'] # Dynamically add the trusted SSIDs
7
8  def get_connectedSSID():
9      """Gets the currently connected SSID."""
10     try:
11         result = subprocess.check_output(['netsh', 'wlan', 'show', 'interfaces'], shell=True)
12         result = result.decode('utf-8')
13         ssid_search = re.search(r"SSID\s+:(.*)", result)
14         if ssid_search:
15             return ssid_search.group(1).strip()
16         return None
17     except subprocess.CalledProcessError as e:
18         print(f"Error retrieving connected network: {e}")
19         return None
20
21 def disconnect():
22     """Disconnects from any network."""
23     try:
24         subprocess.run(['netsh', 'wlan', 'disconnect'], check=True)
25         print("Blocked / Disconnected from the network as a preventive measure.")
26     except subprocess.CalledProcessError:
27         print("Failed to disconnect from the network.")

```

```

prevent_evil_twin.py X
C:\Users\yashu> OneDrive > Desktop > Evil_Twin_Prevention > prevent_evil_twin.py > ...
28
29 def monitor_for_untrusted_networks():
30     """Monitors and disconnects if connected to an untrusted network."""
31     while True:
32         currentSSID = get_connectedSSID()
33         if currentSSID and currentSSID not in TRUSTED_SSIDS:
34             print(f"Preventive measure activated: Potential Rogue Network on air '{currentSSID}'. Proceeding with Ca")
35             disconnect()
36             print(f"Evil Twin attack successfully prevented.")
37         else:
38             print(f"Currently connected to a trusted network: {currentSSID}")
39             time.sleep(5) # Check every 5 seconds
40
41 if __name__ == "__main__":
42     monitor_for_untrusted_networks()
43

```

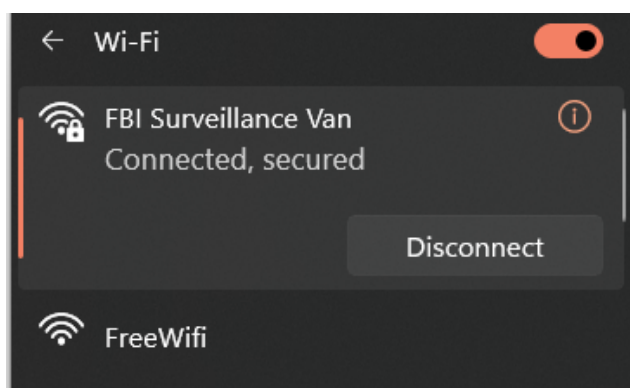
### 3.3 Script Explanation and Functionality

The script performs the following key functions:

- **Retrieve Connected SSID:** Using the netsh command, it fetches the currently connected SSID on the Windows machine.
- **Compare with Trusted Networks:** It compares the connected SSID to a predefined list of trusted SSIDs (TRUSTED\_SSIDS).
- **Block / Disconnect from Untrusted Networks:** If the connected SSID does not match any trusted network, the script disconnects from the current Wi-Fi connection as a preventive measure against an Evil Twin attack. It does not connect to an untrustworthy Rogue access point in the first place either.
- **Continuous Monitoring:** The script runs continuously, checking the connection every 5 seconds to ensure that the Windows machine doesn't remain connected to an untrusted or rogue access point.

### 3.4 Testing and Results

The victim machine is connected to “FBI Surveillance Van” which is a trusted SSID.

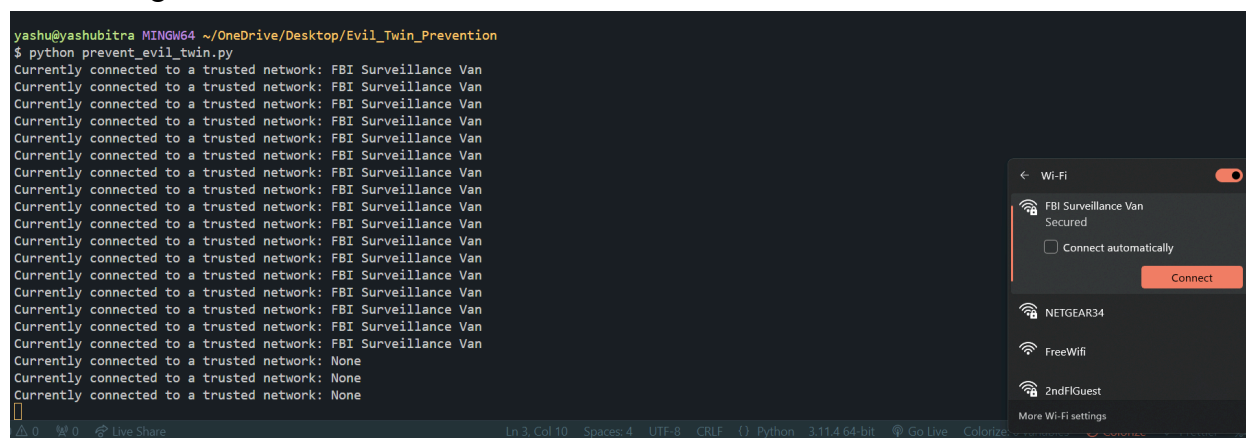


In order to safeguard the Windows machine, I ran the Python script prevent\_evil\_twin.py beforehand. Currently, it is connected to the FBI Surveillance Van.

```
yashu@yashubitra MINGW64 ~/OneDrive/Desktop/Evil_Twin_Prevention
$ python prevent_evil_twin.py
Currently connected to a trusted network: FBI Surveillance Van
Currently connected to a trusted network: FBI Surveillance Van
```

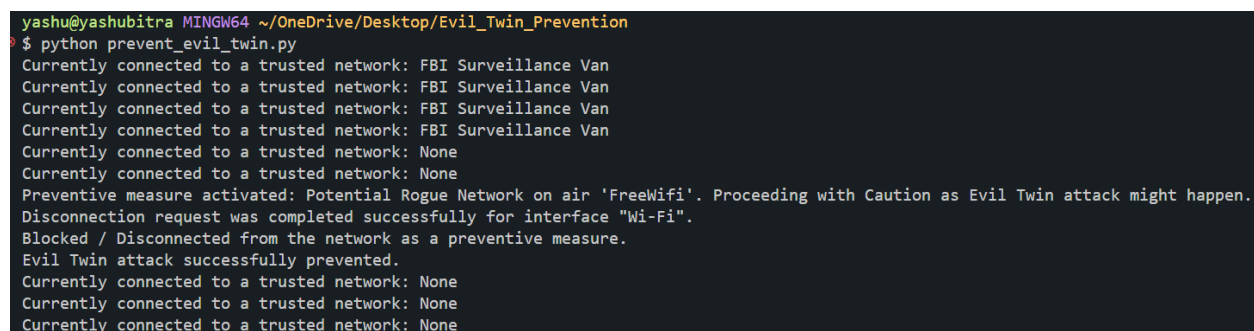
Ln 3, Col

Following the de-authentication attack on “FBI surveillance van”, the attacker forces the victim to connect to the next strongest signal, FreeWifi in this case, basically the rogue access point. Immediately after the de-authentication attack, the connected network is none for a brief period before being forced to connect to FreeWifi.



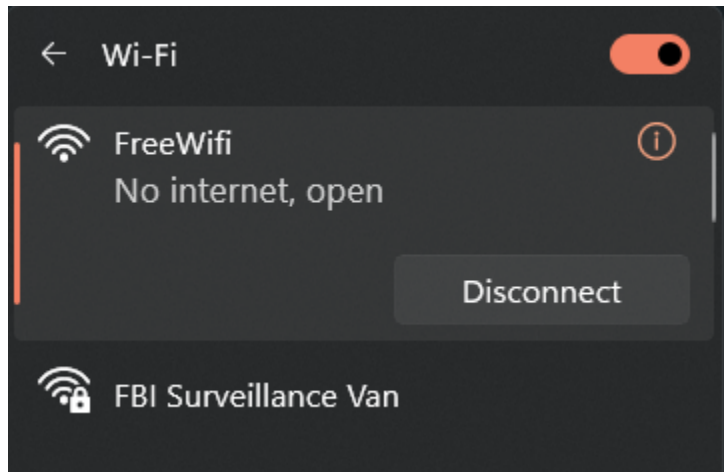
```
yashu@yashubitra MINGW64 ~/OneDrive/Desktop/Evil_Twin_Prevention
$ python prevent_evil_twin.py
Currently connected to a trusted network: FBI Surveillance Van
Currently connected to a trusted network: FBI Surveillance Van
Currently connected to a trusted network: FBI Surveillance Van
Currently connected to a trusted network: FBI Surveillance Van
Currently connected to a trusted network: FBI Surveillance Van
Currently connected to a trusted network: FBI Surveillance Van
Currently connected to a trusted network: FBI Surveillance Van
Currently connected to a trusted network: FBI Surveillance Van
Currently connected to a trusted network: FBI Surveillance Van
Currently connected to a trusted network: FBI Surveillance Van
Currently connected to a trusted network: FBI Surveillance Van
Currently connected to a trusted network: FBI Surveillance Van
Currently connected to a trusted network: FBI Surveillance Van
Currently connected to a trusted network: FBI Surveillance Van
Currently connected to a trusted network: FBI Surveillance Van
Currently connected to a trusted network: FBI Surveillance Van
Currently connected to a trusted network: None
Currently connected to a trusted network: None
Currently connected to a trusted network: None
```

### Case 1: Attempting to Connect to Rogue Access Point (Not Yet Connected):

```
yashu@yashubitra MINGW64 ~/OneDrive/Desktop/Evil_Twin_Prevention
$ python prevent_evil_twin.py
Currently connected to a trusted network: FBI Surveillance Van
Currently connected to a trusted network: FBI Surveillance Van
Currently connected to a trusted network: FBI Surveillance Van
Currently connected to a trusted network: FBI Surveillance Van
Currently connected to a trusted network: None
Currently connected to a trusted network: None
Preventive measure activated: Potential Rogue Network on air 'FreeWifi'. Proceeding with Caution as Evil Twin attack might happen.
Disconnection request was completed successfully for interface "Wi-Fi".
Blocked / Disconnected from the network as a preventive measure.
Evil Twin attack successfully prevented.
Currently connected to a trusted network: None
Currently connected to a trusted network: None
Currently connected to a trusted network: None
```

## Case 2: Already Connected to the Rogue Access Point



```
yashu@yashubitra MINGW64 ~/OneDrive/Desktop/Evil_Twin_Prevention
$ python prevent_evil_twin.py
Preventive measure activated: Potential Rogue Network on air 'FreeWifi'. Proceeding with Caution as Evil Twin attack might happen.
Disconnection request was completed successfully for interface "Wi-Fi".
Blocked / Disconnected from the network as a preventive measure.
Evil Twin attack successfully prevented.
Currently connected to a trusted network: None
```

The tests showed that the `prevent_evil_twin.py` script works effectively. Whether the machine was already connected to a rogue network or trying to connect, the script blocked or disconnected it immediately. This prevents an Evil Twin attack before any data can be compromised.

## 4. Conclusion and Future Work

In this project, we explored both theoretical and practical approaches to prevent Evil Twin attacks on wireless networks. Users are especially at risk from the Evil Twin attack, which exploits the vulnerability of devices by creating rogue access points (APs) with the same SSID as legitimate networks or appearing free. In the first phase, we successfully conducted this attack, demonstrating how easily sensitive information such as login credentials can be intercepted.

The second phase, which is the focus of this project, involves implementing preventive measures. Theoretically, We began by hardening the network and device configurations, utilizing stronger authentication and encryption protocols, and employing tools for rogue AP detection. Key theoretical solutions included using WPA3 encryption, MAC address filtering, disabling WPS, deploying Wireless Intrusion Prevention Systems (WIPS), and promoting user education. These measures work in tandem to enhance network security and reduce the chance of successful attacks.

From a practical standpoint, the Python script **prevent\_evil\_twin.py** effectively demonstrated an automated method for preventing a Windows machine from connecting to untrusted or rogue Wi-Fi networks, hence preventing an evil twin attack.

### Future Work:

- Develop a custom firewall to detect and block rogue access points and de-authentication attempts.
- Implement a custom VPN that encrypts all traffic when connecting to public or unknown networks.
- Build software that abstracts complex security settings, providing easy-to-use Evil Twin attack prevention.
- Extend prevention tools for cross-platform support on Windows, macOS, Linux, Android, and iOS.
- Create lightweight security mechanisms for IoT devices to prevent connections to rogue access points.

## 5. Bibliography

1. Atheros AR9271 Chipset - [https://techinfodepot.shoutwiki.com/wiki/Atheros\\_AR9271](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](https://en.wikipedia.org/wiki/Wi-Fi_deauthentication_attack)
9. Rogue Access Point - [https://en.wikipedia.org/wiki/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>