Wireless Security Project

Andrew Schwartz

University of Michigan - Dearborn

CIS 446 - Wireless and Mobile Computing Security

Dr. Di Ma

April 19, 2024

Introduction

Problem Description and Motivation to Perform Experiments

This project investigates the WPA-PSK (WPA-Personal) Wifi security standard and its implementation using a fake testing environment. The purpose of this experiment is to show how vulnerabilities form in security standards over time. The second portion of this project was to investigate Evil Twin attacks on a wifi network. The purpose of this portion is to demonstrate how an attack would go about deploying an Evil Twin attack on wifi networks.

Summary of Results

The results of the project are alarming, showing how easy it is to mount an attack on WPA-PSK networks and how easy brute force password guessing is on modern computing hardware. The attack can be successfully made using $50 in computing gear and an open source toolkit that anyone can run, without extensive knowledge of the inner workings. The results of the Evil Twin attack is also alarming, showing how unknowingly these types of attacks can take place. An attacker can crack your wifi network in less than 3 hours and steal any data that goes through the network.

Test Environment

Clients

- Apple Macbook Air, Model A2237, M1 Chip. Running MacOS Ventura 13.6
- Apple iPhone XR, Model A1984, A12 Bionic. Running IOS 16.7

Hosts

- Raspberry Pi 3B+, Cortex-A53 (ARMv8) 64-bit SoC. Running Kali Linux ARM
    - Alfa Networks AWUS036ACM wifi usb card
- Apple Macbook Pro, Model A2442, M1 Pro Chip. Running MacOS Sonoma 14.3
- TP-Link Archer A7 AC1750 Wireless Dual Band Gigabit Router.

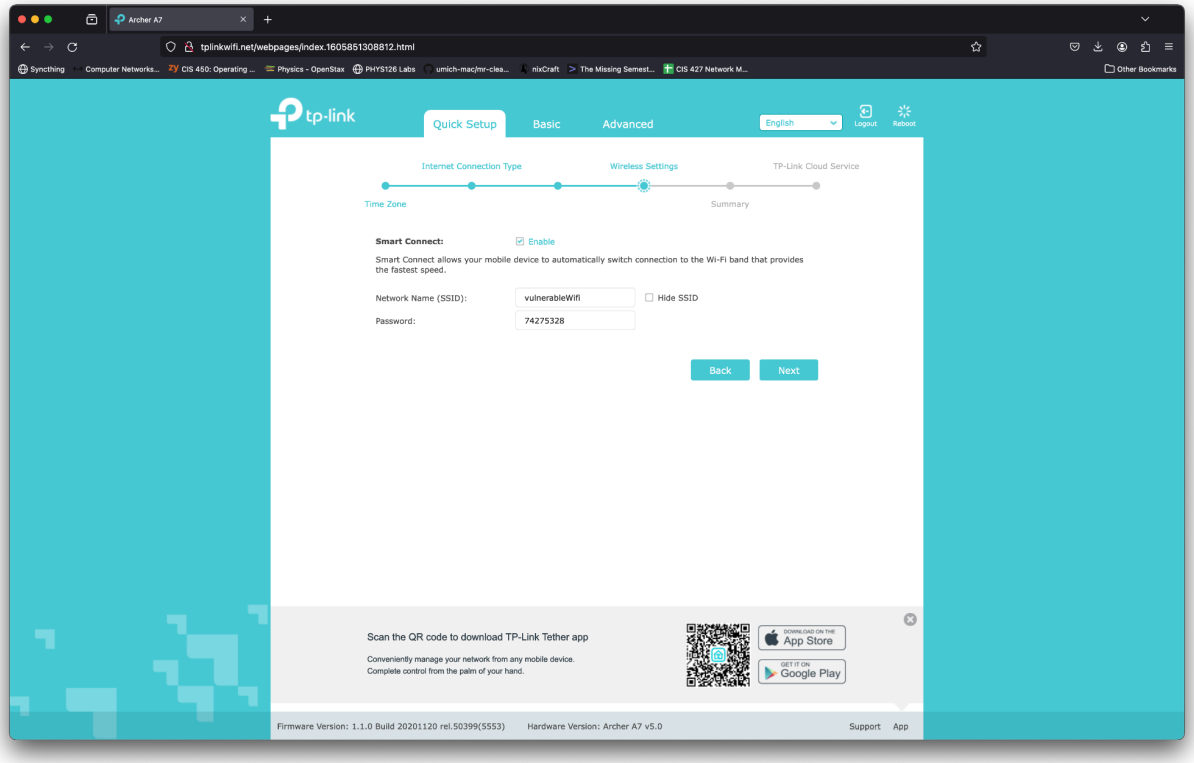
Experiment

This experiment is set up in 3 parts. The first part is setting up the testing environment. The second part is conducting the WPA-PSK experiment. The third part is conducting the Evil Twin experiment.

Part One:

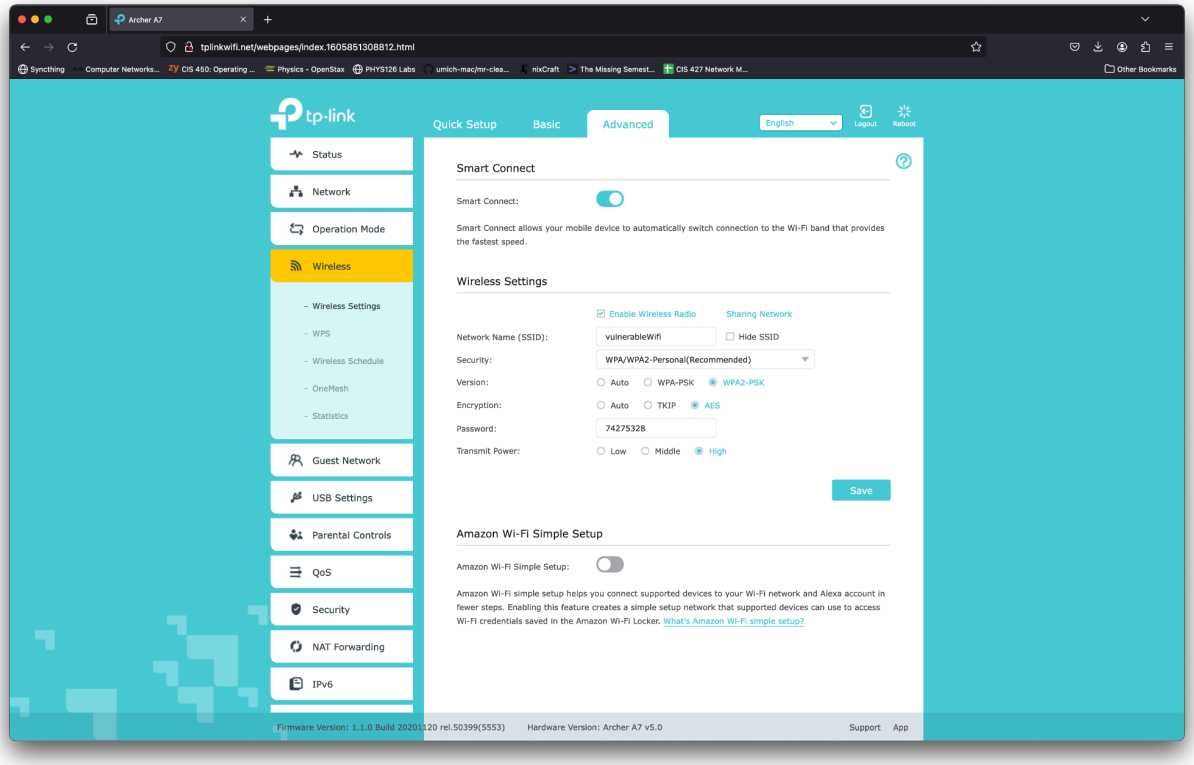
To conduct the experiment, the first thing to setup is the fake wireless network. I performed a hard reset on the TP-Link router to test its default configuration. After resetting the router and powering it on, to get to the wifi router's settings I connected my Macbook Pro with an ethernet cable to the router and opened a browser and went to https://tplinkwifi.net after getting assigned a local IP address. After walking through the initial setup, I see the wifi

settings/security settings page:



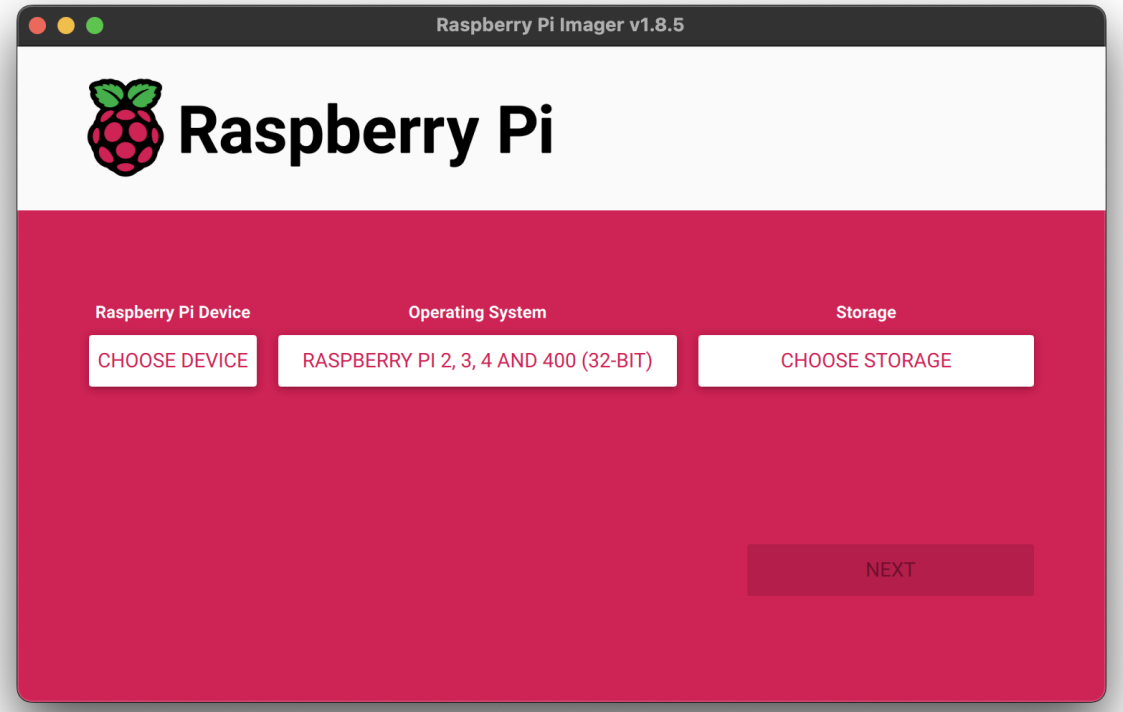
* For the sake of clarity, I changed the SSID to vulnerableWifi from the default.

After clicking through the rest of the quick setup I go to view more details about the wifi settings:

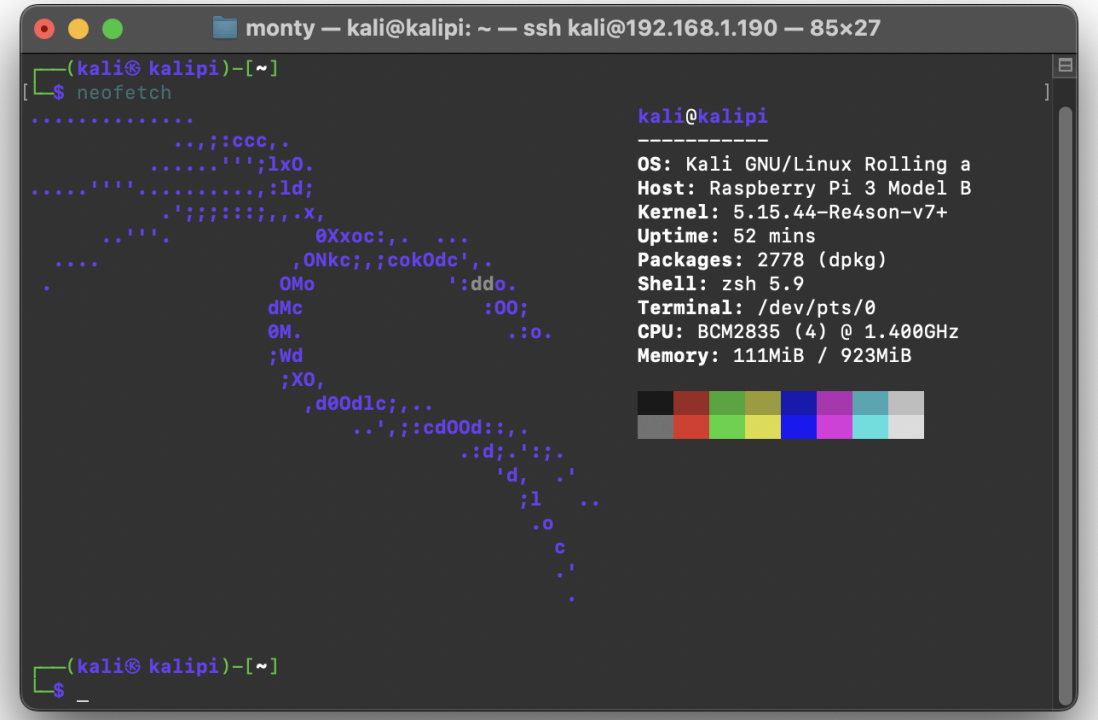


By default, the TP-Link uses WPA/WPA2-Personal (WPA2-PSK) using AES encryption and the default password is eight numeric digits. That is all the setup for the wifi network.

Next, I set up the attacker side of the environment. First, using the Raspberry Pi Imager I create a boot SD card for the Raspberry Pi of the default Kali Linux for ARM devices image:



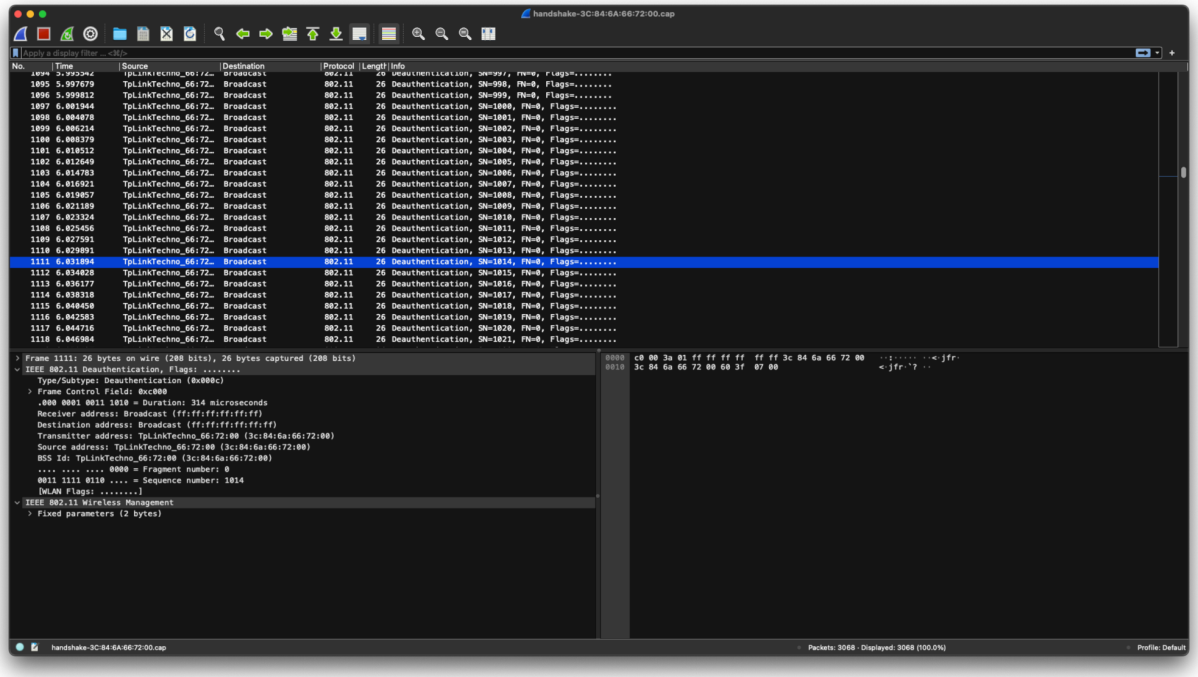
After installing the imaged SD card in the Raspberry Pi, booting up the Pi, I ssh into the Pi:



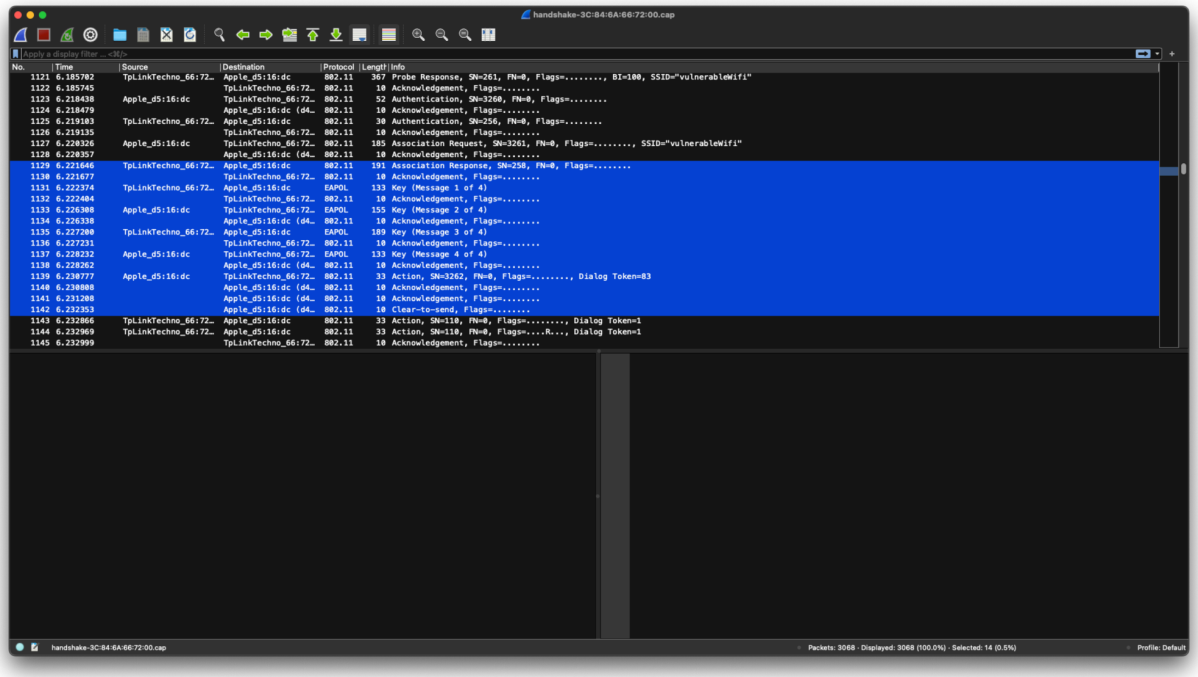
Now that the Pi is online I can install the tools needed to mount the attack. The tool I choose to use is Airgeddon (https://github.com/v1s1t0r1sh3r3/airgeddon) which explicitly states it has support for both my Raspberry Pi Model 3B+ and my Alfa Network AWUS036ACM wifi card. I clone the repository to the Pi and install any extra tools that Airgeddon uses that are not installed by default using apt.

Part Two:

Conducting the WPA-PSK attack is spilt into 2 parts, capturing the handshake and cracking the PSK. To capture the handshake the attack must wait for a client to connect to the target network, as the attacker, I can speed up the process by asking the router to send out deauthentication broadcasts:

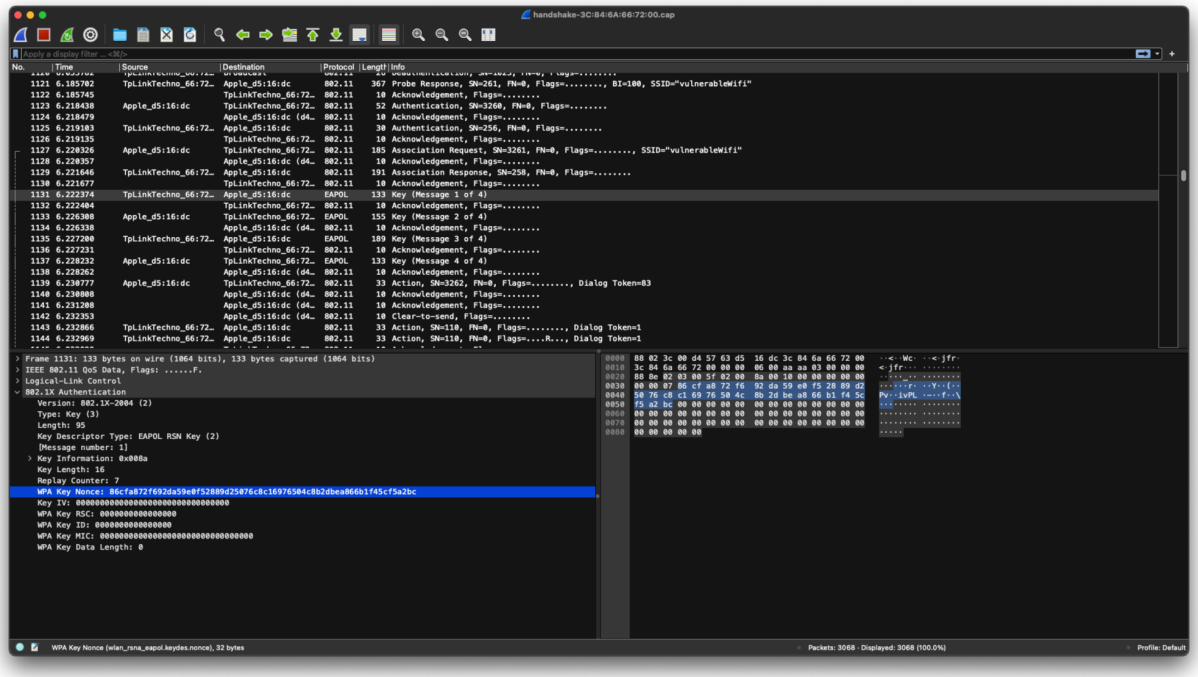


Next, I capture the handshake:



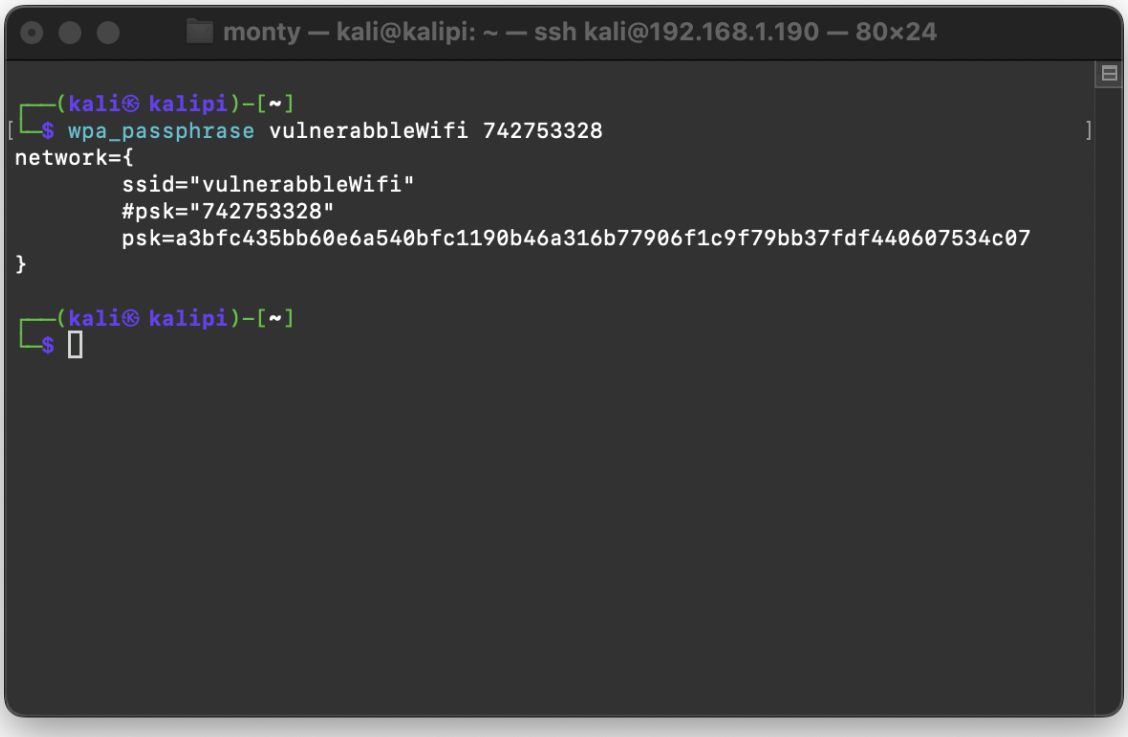
There is a captured handshake between a clent and the router host. I can see each message of the handshake. Next I will analyze each message of the handshake to understand what is happening.

Message 1:



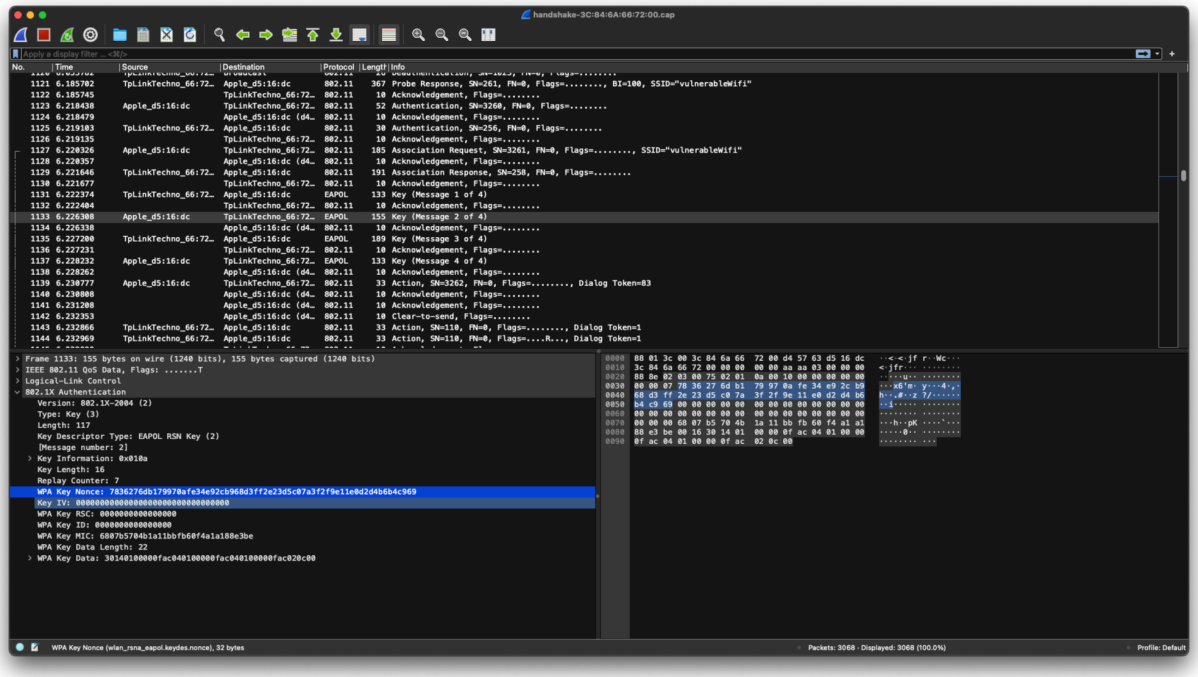
The router sends it's nonce (random string of characters) to the client. After the message is received on the client side, the client will use five pieces of information to derive the Pairwise Transient Key, PTK, for encrypted communication with the router. That information is; router's nonce, client's nonce, router's mac address, client's mac address, and the Pre-Shared Key (PSK).

A side note about the PSK, it is derived from the SSID and wifi password:



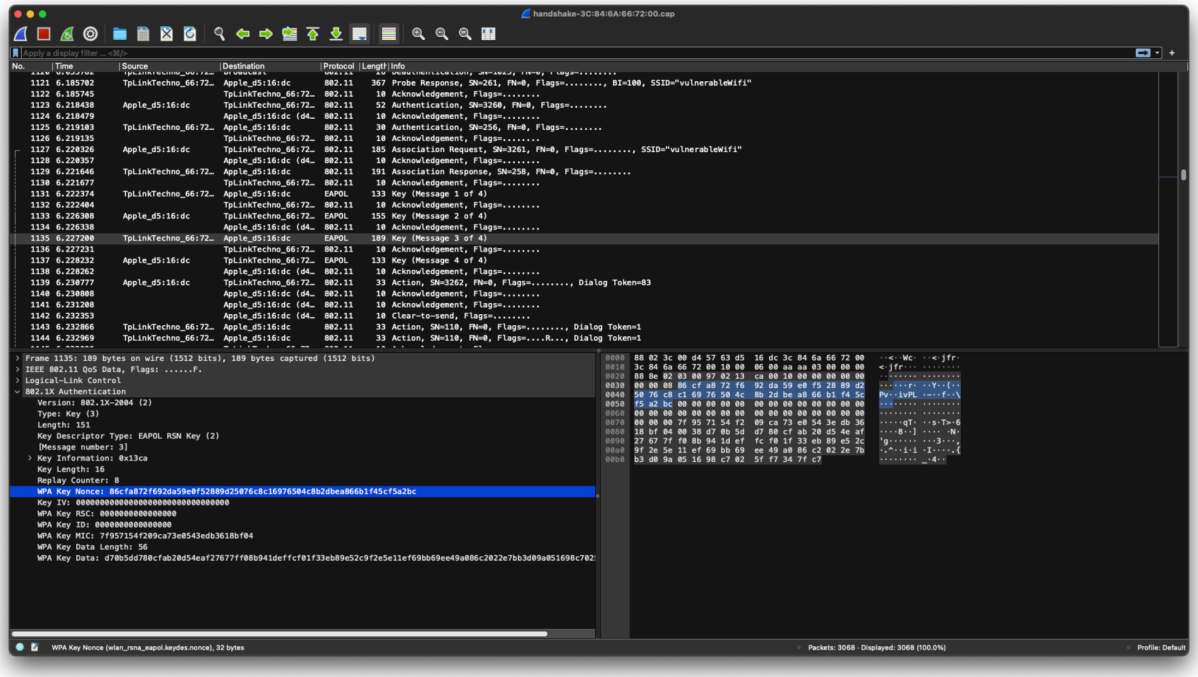
Back to the handshake.

Message 2:



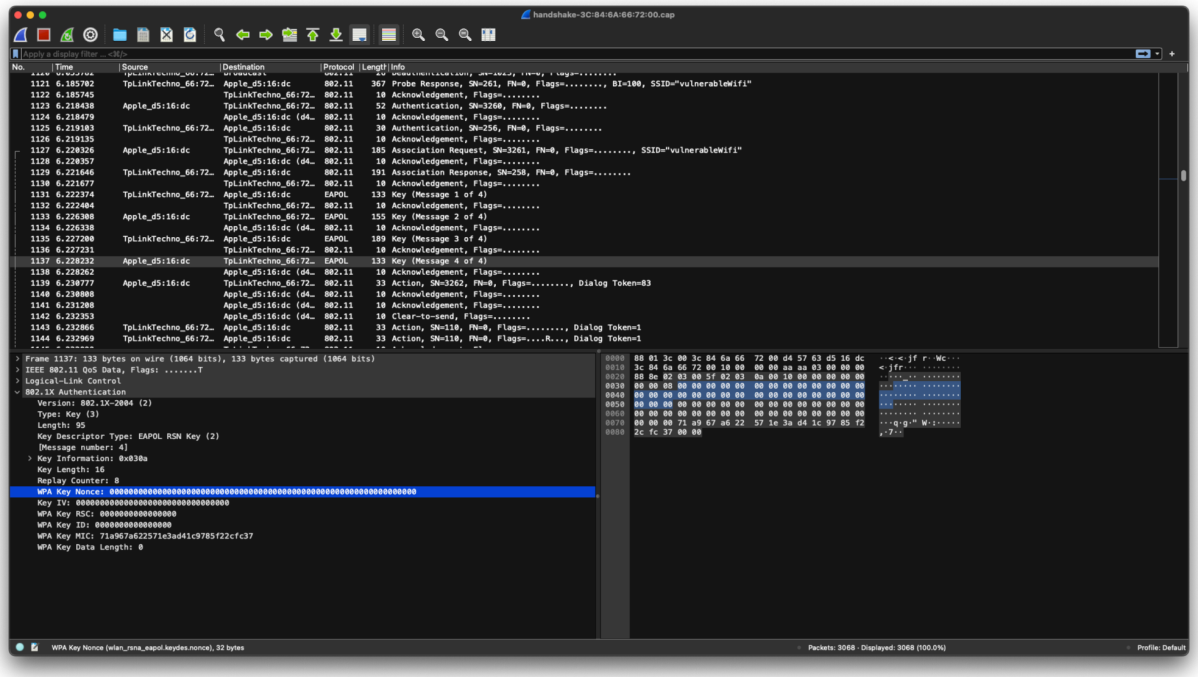
Now, the client replies back to the router with it's own nonce, what it derived for the PTK and a Message Intergity Check (MIC). The MIC will be used later to check our brute force password guessing.

Message 3:



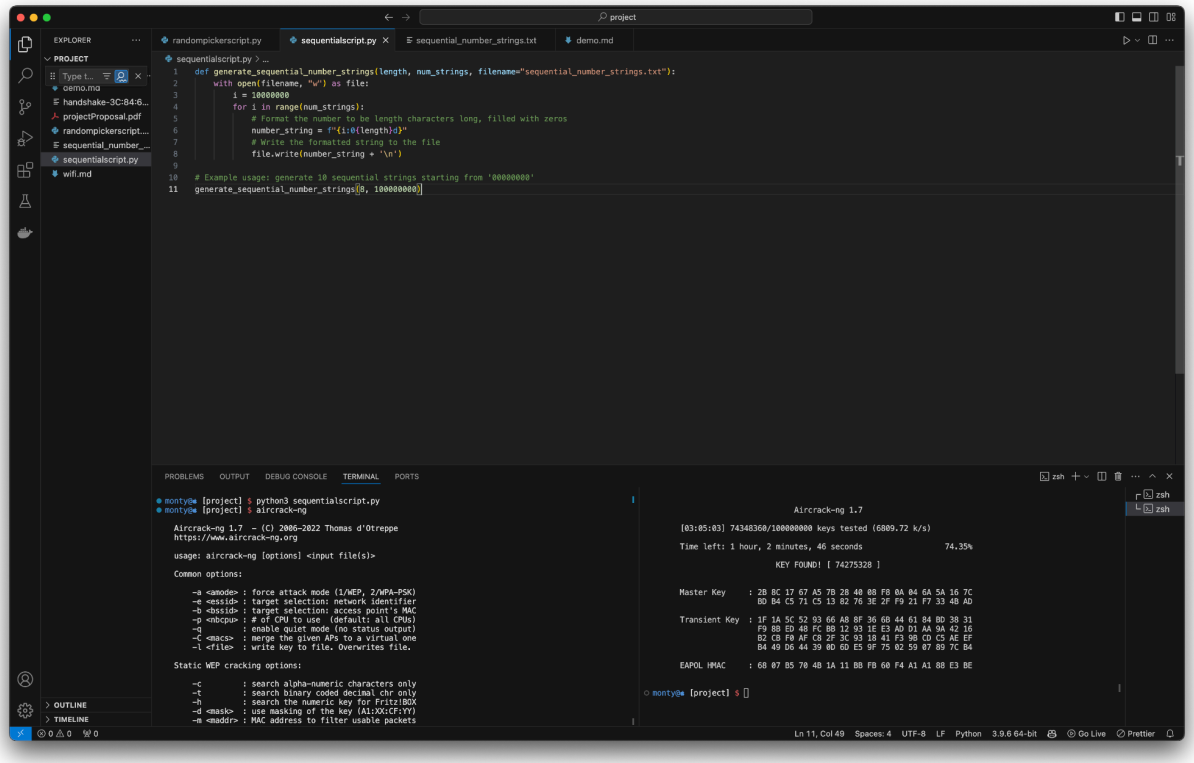
In message 3, the router replies with one of two messages. If the wifi password is correct and the the derived PSK and derived PTK is correct then the router will send a message to install the PTK and Group Temporal Key, GTK, so communication can be encrypted. If the wifi password is wrong, then the derived PSK and derived PTK would be wrong and the router would send a message saying the PTK do not match and try to restart the handshake. The router will always resend its nonce incase the latter message was sent. This message also has an MIC for the client.

Message 4:



Lastly, the client sends an acknowledgement saying signalling the encryption keys were installed. When the router receives this acknowledgement, it will install the PTK and GTK as well.

The second part of the attack is cracking the PTK that was derived. Here, I will use a tool called Aircrack-ng to brute force the wifi password. As a smart attack, I know to limit my keyspace, the number of possible wifi passwords to guess, so I will know that I am attacking a TP-link router. After doing some open source intelligence, OSINT, I have noticed that TP-link routers use eight numeric digits default passwords like 01234567. That would give me 10^8 possible passwords. That is 100 million passwords, may seem like a lot to a human but to a modern computer this is very manageable. I wrote a script to generate a text file with all 100 million passwords in it. Then I can run Aircrack-ng against that text file and the handshake capture. Here is my output after letting Aircrack run on my laptop:



Aircrack-ng found the key in about 3 hours! Looking closer at the Aircrack-ng output, it shows some information that shows up in the handshake capture.

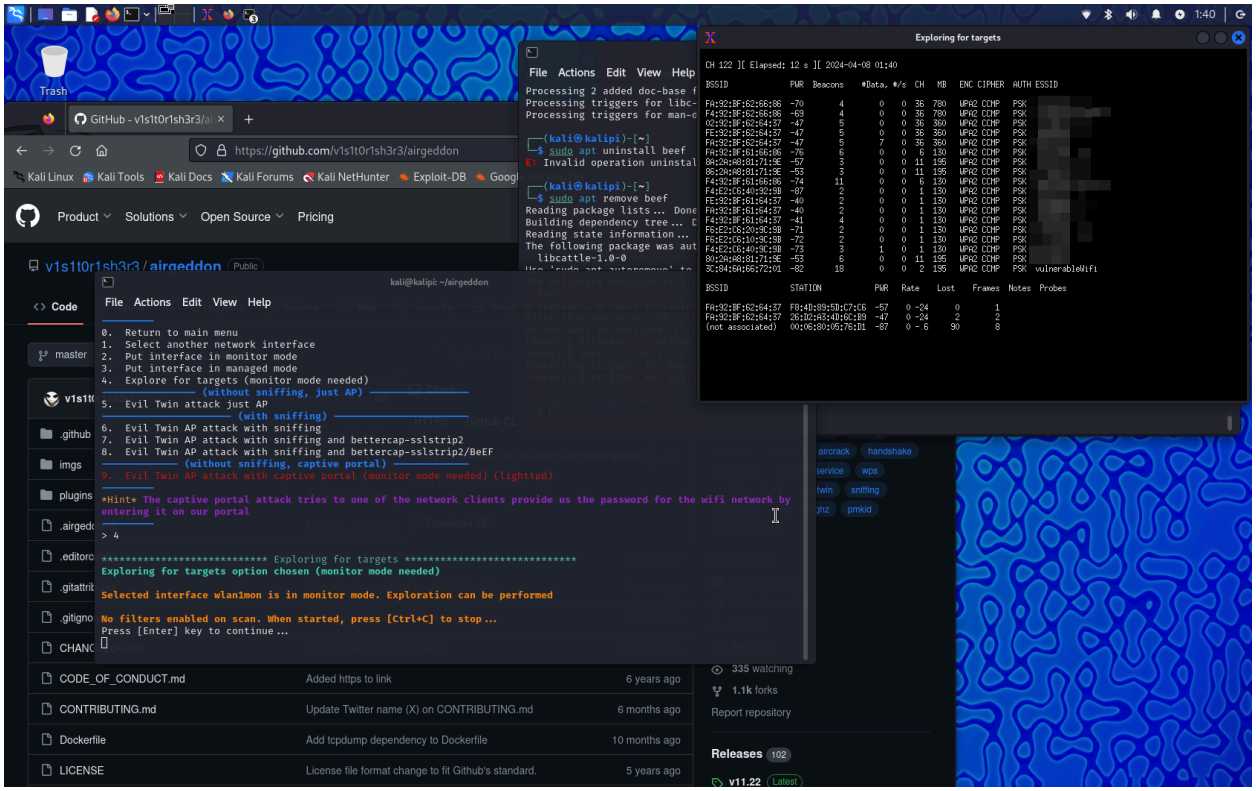
```
                              Aircrack-ng 1.7

      [03:05:03] 74348360/100000000 keys tested (6809.72 k/s)

      Time left: 1 hour, 2 minutes, 46 seconds                      74.35%

                              KEY FOUND! [ 74275328 ]


      Master Key     : 2B 8C 17 67 A5 7B 28 40 08 F8 0A 04 6A 5A 16 7C
                       BD B4 C5 71 C5 13 82 76 3E 2F F9 21 F7 33 4B AD

      Transient Key  : 1F 1A 5C 52 93 66 A8 8F 36 6B 44 61 84 BD 38 31
                       F9 8B ED 48 FC BB 12 93 1E E3 AD D1 AA 9A 42 16
                       B2 CB F0 AF C8 2F 3C 93 18 41 F3 9B CD C5 AE EF
                       B4 49 D6 44 39 0D 6D E5 9F 75 02 59 07 89 7C B4

      EAPOL HMAC     : 68 07 B5 70 4B 1A 11 BB FB 60 F4 A1 A1 88 E3 BE


○ monty@ [project] $
```

That EAPOL HMAC that is generated is the same as the MIC that the client sent to the router in message 2 of the handshake. This is how Aircrack-ng knows that the 74275328 is the correct key. Aircrack-ng uses the same method to derive the PTK with the same information the client did and also generates the MIC and compares it with the one in the handshake.

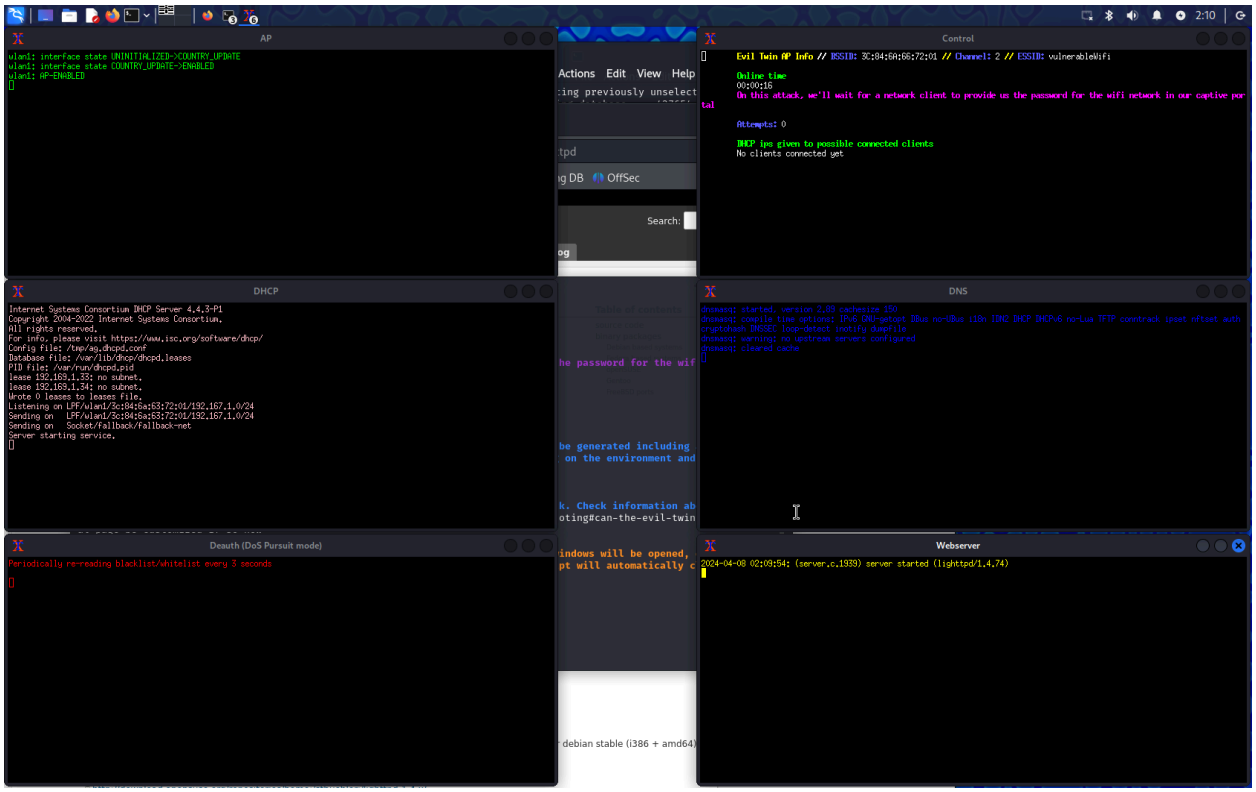
Part Three:

An Evil Twin attack is an attack where the attacker, in this experiment me, poses as an access point or router that matches the network that it is targeting to get clients to connect to itself instead. The attack here is preying on a standard function that is used in IEEE802.11 standards, that is clients are configured to disconnect and join the access point that is giving off the strongest signal. Which is good so clients don't stay connected to an access point on the opposite side of the building, but can be taken advantage of by attackers.

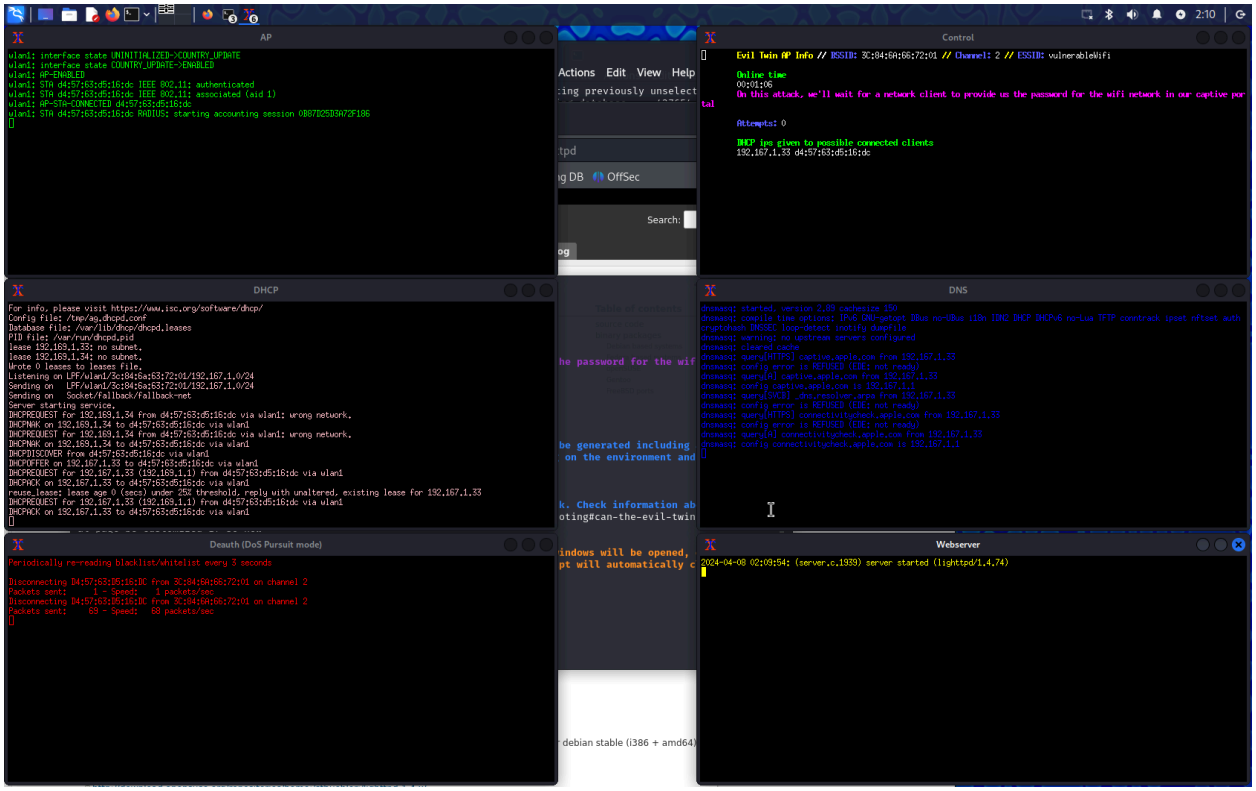
First, using Airgeddon still, I set up the Evil Twin network, by searching for available networks and then selecting which one I want to mimic.



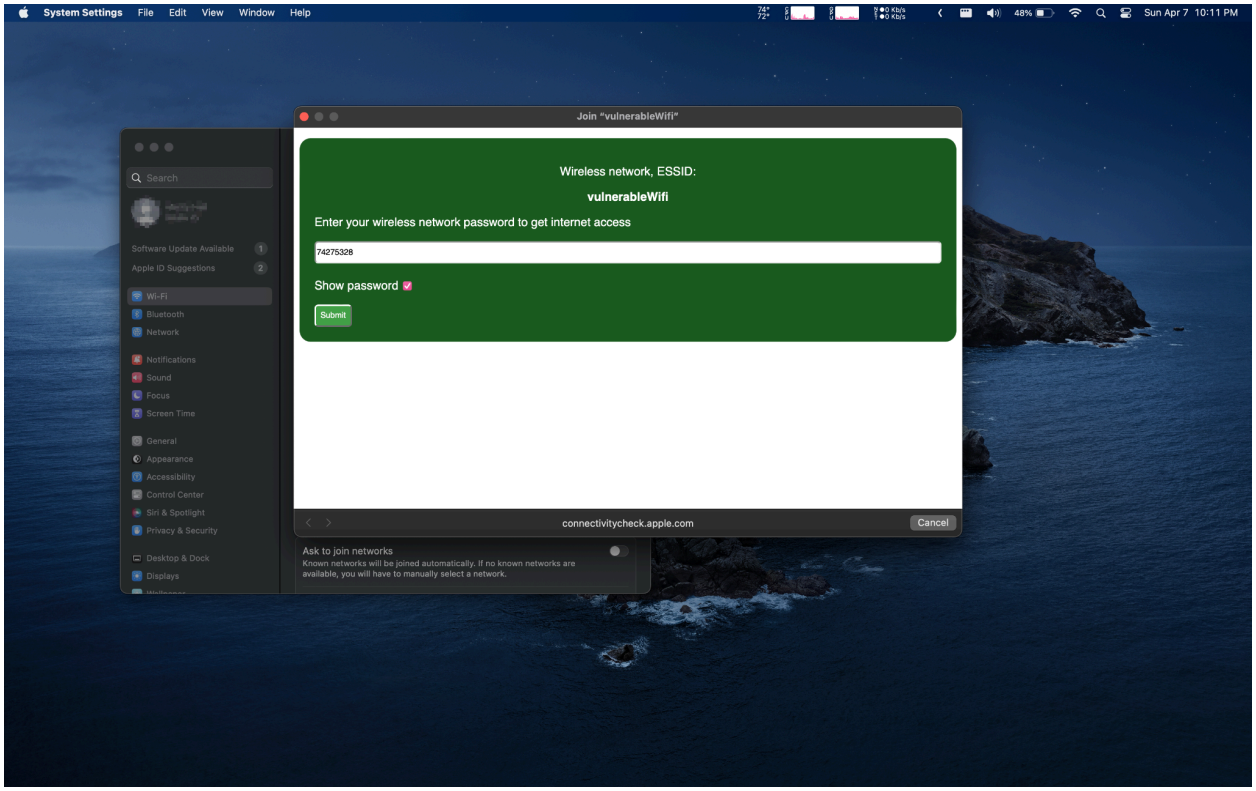
Next, I standup all the services that the network needs like DNS, DHCP and the webserver for the captive portal, and all the services I need as the attacker like Deauth.

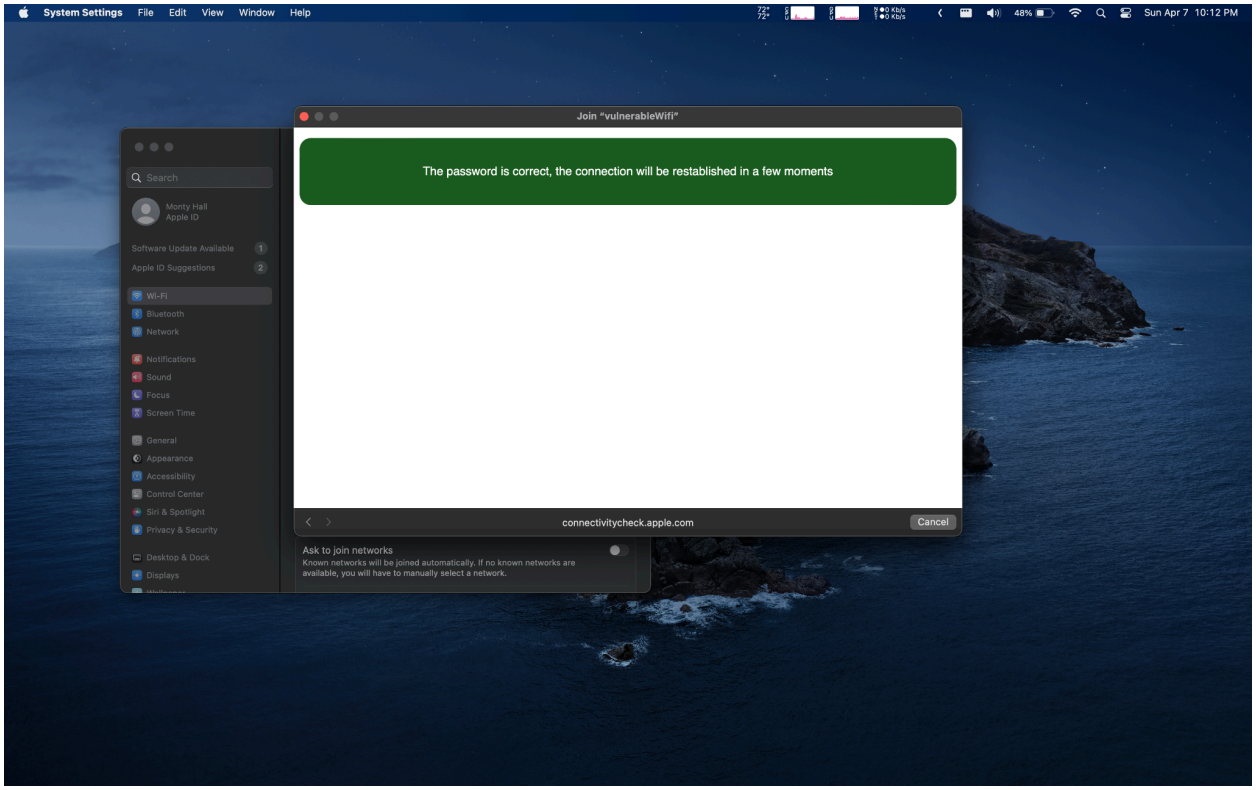

AP
wlan1: interface state UNINITIALIZED->COUNTRY_UPDATE
wlan1: interface state COUNTRY_UPDATE->ENABLED
wlan1: AP-ENABLED
Control
Evil Twin AP Info // BSSID: 3C:84:6A:66:72:01 // Channel: 2 // ESSID: vulnerableWifi
Online time
00:00:16
On this attack, we'll wait for a network client to provide us the password for the wifi network in our captive por
tal
Attempts: 0
DHCP ips given to possible connected clients
No clients connected yet
DHCP
Internet Systems Consortium DHCP Server 4.4.3-P1
Copyright 2004-2022 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/
Config file: /tmp/ag_dhcpd.conf
Database file: /var/lib/dhcp/dhcpd.leases
PID file: /var/run/dhcpd.pid
lease 192.168.1.33: no subnet.
lease 192.168.1.34: no subnet.
Wrote 0 leases to leases file.
Listening on LPF/wlan1/3c:84:6a:63:72:01/192.167.1.0/24
Sending on LPF/wlan1/3c:84:6a:63:72:01/192.167.1.0/24
Sending on Socket/fallback/fallback-net
Server starting service.
DNS
dnsmasq: started, version 2.89 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt DBus no-UBus i18n IDN2 DHCP DHCPv6 no-Lua TFTP conntrack ipset nftset auth
cryptohash DNSSEC loop-detect inotify dumpfile
dnsmasq: warning: no upstream servers configured
dnsmasq: cleared cache
Deauth (DoS Pursuit mode)
Periodically re-reading blacklist/whitelist every 3 seconds
Webserver
2024-04-08 02:09:54: (server.c.1939) server started (lighttpd/1.4.74)
Actions Edit View Help
:ing previously unselect
tpd
ig DB OffSec
Search:
og
he password for the wif
be generated including
on the environment and
k. Check information ab
oting#can-the-evil-twin
indows will be opened,
pt will automatically c
debian stable (i386 + amd64)

Once, I get a successful deauth or a new client trys to connect to my Evil Twin, they will show up as connected.

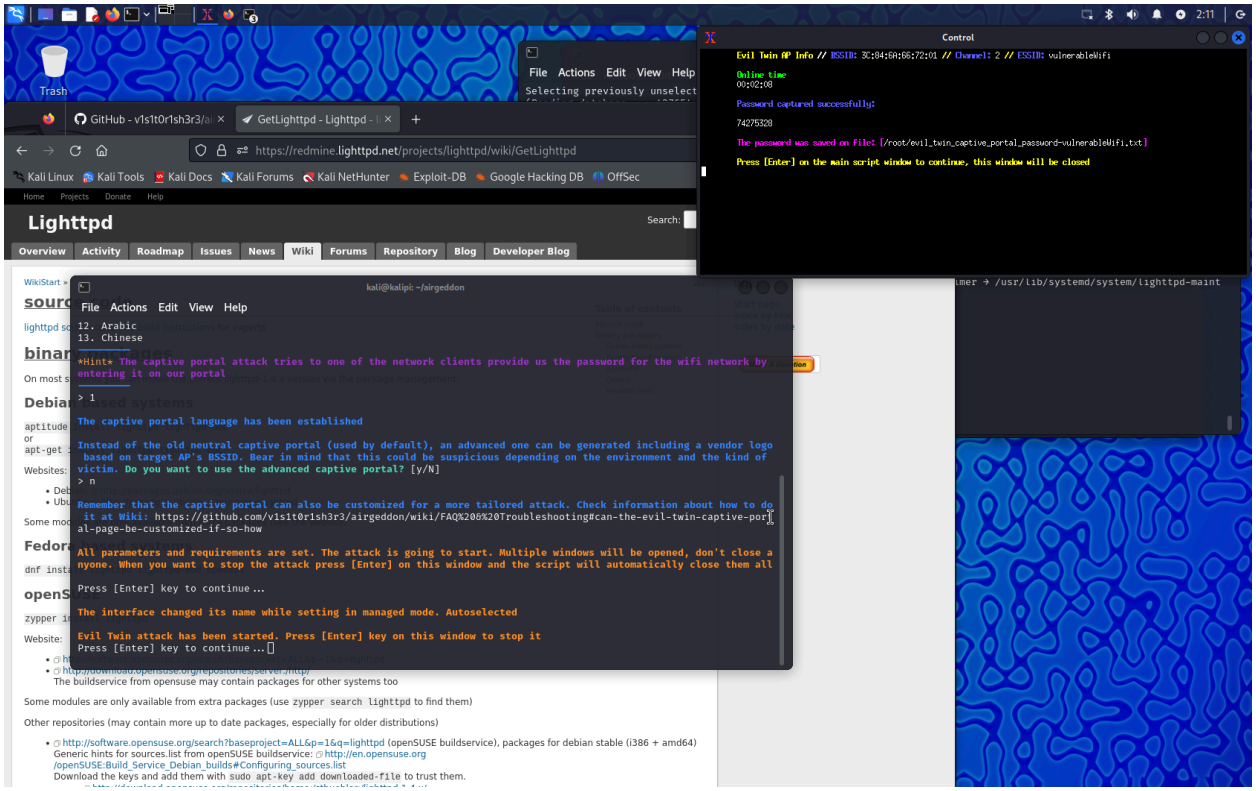


The client will go to enter the wifi password on their device. For this example I choose to use a captive portal but the attack can go without it.





Then I will see that success on the backend



Now, I have taken the password without needing to crack it.

Results

In this experiment, I successfully demonstrated two different types of Wifi attacks to gain unauthorized access to a network. First, I showed a WPA-PSK attack that shows a weakness is comsumer wifi products that many end users will unknowningly plug in and use in their homes. Stronger default passwords and password requirements are should be required to be able to thoroughly protect data and computing devices. In the second part of the experiment, I showed the another attack that can happen on a wifi network. Users should be aware of the networks they are connecting to before sending data through them. Both of these attacks can be mounted in minutes and gain access in a few hours or a few seconds depending on the hardware and sophistication of the attacker and their methods. I believe this shows that even with secure encryption algorithms and direct encryption keys, wifi is naturally insecure because of the physical openness in which data is transmitted. I took all the open information available to anyone to “reverse engineer” the password that was never directly sent in the open air.

Conclusions

In conclusion, although Wifi is a wonderful technology that has allowed millions of people get access to information that they might not the chance to, it has draw backs that can leave unsuspecting people vulnerable cyber crime. Comsumers should be aware of these drawbacks and do everything in their power to mitigate them through the use of strong wifi

passwords and Virtual Private Networks. Additionally, manufacturers should know better than to set a default password to the lowest possible security level.

References

1. Airgeddon, https://github.com/v1s1t0r1sh3r3/airgeddon
2. Kali Linux, https://www.kali.org/
3. Raspberry Pi, https://www.raspberrypi.com/
4. Aircrack-ng, https://www.aircrack-ng.org/
5. Wifi Wikipedia, https://en.wikipedia.org/wiki/Wi-Fi
6. WPA Wikipedia, https://en.wikipedia.org/wiki/Wi-Fi_Protected_Access
7. Alfa Networks inc., https://www.alfa.com.tw/
8. Wireshark, https://www.wireshark.org/