PiShield: Low-Cost Shield for Public Wi-Fi

Michelle Vasconcelos[1], Michael Vasconcelos[1], Magdy Ellabidy[1], David C.S. Albanese[2] and Mira Yun[3]

[1]School of Computing and Data Science, Wentworth Institute of Technology, Boston, USA

[2]Department of History, Bentley University, Waltham, USA

[3]Department of Computer Science, Boston College, Chestnut Hill, USA

{vasconcelosm2, vasconcelosm1, ellabidym}@wit.edu, dalbanese@bentley.edu, mira.yun@bc.edu

Abstract

Most people know the danger of a public wireless network, but we cannot ignore the convenience of free Wi-Fi. As all computing and mobile devices are becoming very important in our daily lives, completely avoiding public wireless access cannot be the best option. However, failing to understand how easy it is to hack a public network can be very risky. In this paper, we demonstrate how easy it is for attackers to get user credentials from a public Wi-Fi access point. We then propose a low-cost solution, *PiShield*, which provides VPN-enabled Wi-Fi hotspot service. Learners of any level can use this method to securely connect to public Wi-Fi networks.

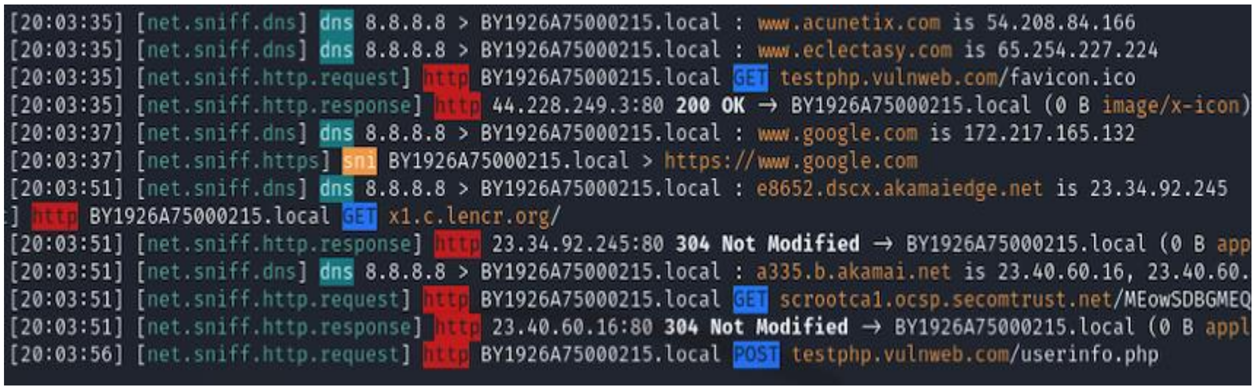
Keywords – Public Wi-Fi Security, Evil Twin Attack, VPN, Raspberry Pi.

I. Introduction

Public Wi-Fi is a huge convenience during travel. Nearly all airports and hotels have free Wi-Fi connections that the public can use. However, this convenience comes with high risks. Since anyone can connect to these networks, they are far from secure. Most people know the danger of a public wireless network, but they fail to understand how easy it is to hack such a network and to compromise all devices connected directly to it. Connecting through a virtual private network (VPN) has been a reliable way to protect mobile computing devices on public Wi-Fi. A VPN creates a secure tunnel between the user device and the public network. However, VPN services are not free for everyone. Many VPN services require a user to have an account and to pay for the service. As the scope and power of the Internet of Things (IoT) expand, users and organizations who require multiple accounts face high costs to keep their computing devices safe with VPN services.

In this paper, we present how to create a low-cost VPN hotspot solution, *PiShield*. To create *PiShield*, we started by setting up a *Raspberry Pi* device with *OpenWRT* [1], which allowed us to turn our *Raspberry Pi* into a Wi-Fi access point. For our virtual private network, we decided to use *NordVPN* as it is well known for its safety and is used by many organizations [2]. All of our configurations are described in Section III. We also present an evil twin attack in Section II to show the danger of a public wireless network. Our evil twin router is another *Raspberry Pi* that serves as a bridge between the public Wi-Fi and victim devices. Using an evil twin router, we were able to gain full control over the public network. Before we attempted our hack into our low-cost VPN system in Section III, we demonstrated how easy it is for attackers to control which websites the victim goes to and how they can view all victims' browsing information. This can include their usernames and passwords.

II. EVIL TWIN ATTACK



```
[20:03:35] [net.sniff.dns] dns 8.8.8.8 > BY1926A75000215.local : www.acunetix.com is 54.208.84.166
[20:03:35] [net.sniff.dns] dns 8.8.8.8 > BY1926A75000215.local : www.eclectasy.com is 65.254.227.224
[20:03:35] [net.sniff.http.request] http BY1926A75000215.local GET testphp.vulnweb.com/favicon.ico
[20:03:35] [net.sniff.http.response] http 44.228.249.3:80 200 OK → BY1926A75000215.local (0 B image/x-icon)
[20:03:37] [net.sniff.dns] dns 8.8.8.8 > BY1926A75000215.local : www.google.com is 172.217.165.132
[20:03:37] [net.sniff.https] sni BY1926A75000215.local > https://www.google.com
[20:03:51] [net.sniff.dns] dns 8.8.8.8 > BY1926A75000215.local : e8652.dscx.akamaiedge.net is 23.34.92.245
] http BY1926A75000215.local GET x1.c.lencr.org/
[20:03:51] [net.sniff.http.response] http 23.34.92.245:80 304 Not Modified → BY1926A75000215.local (0 B app
[20:03:51] [net.sniff.dns] dns 8.8.8.8 > BY1926A75000215.local : a335.b.akamai.net is 23.40.60.16, 23.40.60.
[20:03:51] [net.sniff.http.request] http BY1926A75000215.local GET scrootca1.ocsp.secomtrust.net/MEowSDBGMEQ
[20:03:51] [net.sniff.http.response] http 23.40.60.16:80 304 Not Modified → BY1926A75000215.local (0 B appl
[20:03:56] [net.sniff.http.request] http BY1926A75000215.local POST testphp.vulnweb.com/userinfo.php
```

Fig. 1. Package Sniffing Example

The Man-In-The-Middle (MITM) attack is one of the most well-known and powerful attack types to steal information [3]. MITM targets the actual data that flows between victims by placing the attacker between them. The attacker, the eponymous "man in the middle," is intercepting and reading the victim's transmitted data. The attacker can then deploy additional tools between the victims and access all important information such as log-in credentials, banking information, and other personal information [4]. We present in this section an evil twin attack, a type of MITM, which we used as a test. In an evil twin attack, an attacker sets up a fake Wi-Fi network that looks like a legitimate access point, in order to steal victims' sensitive details [5-7].

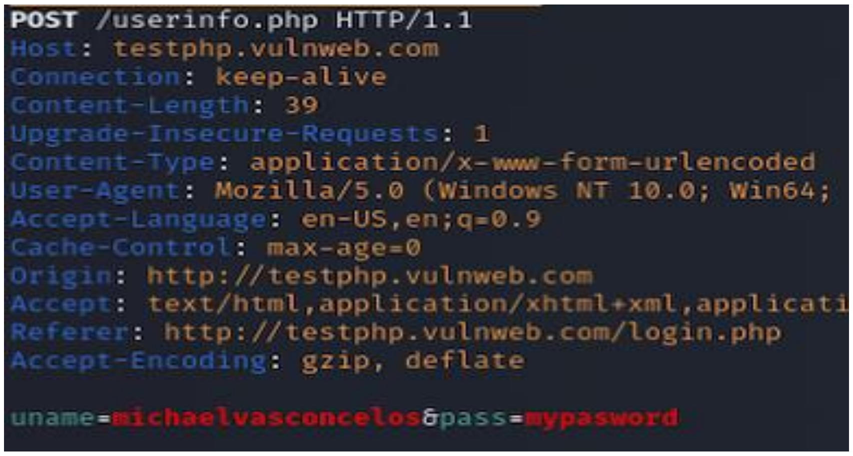
We created an access point that mimics the public Wi-Fi network to lure targets into connecting to our network. Once the target was connected to our evil twin, we begin to perform package sniffing attacks by using *bettercap* [8]. After we showed how to get login and password information from package sniffing, we proceeded with a domain name system (DNS) spoofing attack, which allowed us to control which websites the target could access.

2.1 Evil Twin Setup

We decided to use a *Raspberry Pi* for the evil twin access point since it is relatively easy to configure with a single-board computer into an access point. We started by setting up our *Raspberry Pi 3 B+* using *Kali*, which is an open-source, Debian-based Linux distribution [9]. *Kali* provides various tools for information security such as penetration testing, security research, computer forensics, and reverse engineering. *Kali* comes with several useful wireless hacking tools that allowed us to accomplish our attacks on the open wireless network. We created a Wi-Fi host using *hostapd* - host access point daemon. *Hostapd* is a user-space daemon software that enables a network interface card to act as an access point and authentication server. *Udhcpd* is used to set up a static internet protocol (IP) address, netmask, domain name system (DNS), and gateway address to the *Raspberry Pi*.

After the device setup, our *Raspberry Pi* evil twin now acted like a Wi-Fi access point and a bridge that connected to the internet and routed traffic to target devices. For this, we used a network belonging to a *Dunkin Donuts* coffee shop, with the authorization of the franchise owner. This is representative of the sort of public network many people might use in their everyday lives. We started by connecting our evil twin device to the Dunkin' public network, then turned the Wi-Fi on the evil twin. Users who searched for free Wi-Fi would detect two Wi-Fi service set identifiers (SSIDs) with similar names. The original *Dunkin Donuts* public network had the name "Dunkin" while the evil twin had the name "Dunkin Guest". Since the evil twin simply works as a bridge, there is no way for the target device to recognize an unsecured Wi-Fi access point. The connection looks identical to the original Dunkin Wi-Fi network.

2.2 Package Sniffing



```
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
Connection: keep-alive
Content-Length: 39
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
Accept-Language: en-US,en;q=0.9
Cache-Control: max-age=0
Origin: http://testphp.vulnweb.com
Accept: text/html,application/xhtml+xml,applicati
Referer: http://testphp.vulnweb.com/login.php
Accept-Encoding: gzip, deflate

uname=michaelvasconcelos&pass=mypasword
```

Fig. 2. Retrieved Login Credentials

A victim device, such as a laptop, connected to the "Dunkin Guest" evil twin network. Since the victim device was connected to our evil twin, we could collect all the network traffic between the victim and the original *Dunkin Donuts* public network. This was done using a tool called *bettercap* [8], which is a powerful, easily extensible, and portable framework that can be used to perform reconnaissance and to attack Wi-Fi networks. As shown in Fig. 1, we performed package sniffing attacks which captured network traffic at the Ethernet frame level. After sniffing, the collected data could be analyzed, and sensitive information could be retrieved. Fig. 2 shows retrieved network traffic information including username and password information.

2.3 DNS Spoofing

DNS spoofing, also known as DNS cache poisoning, is an attack in which altered DNS records are used to redirect online traffic to a fraudulent website that resembles its intended destination. This can be highly effective, and it is the most often used evil twin attack. A successful DNS attack can control which websites

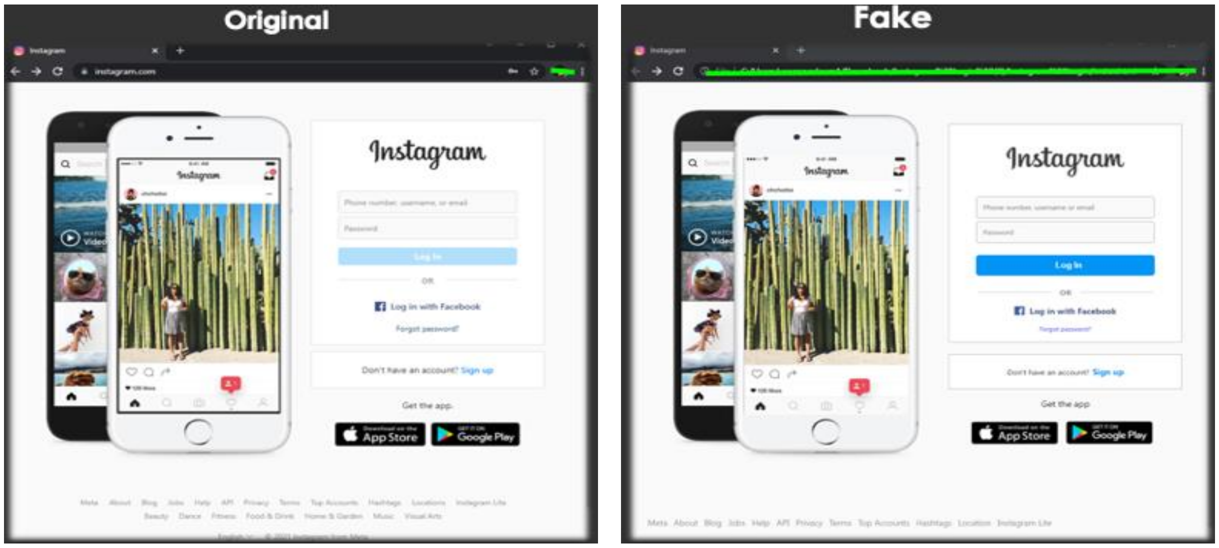
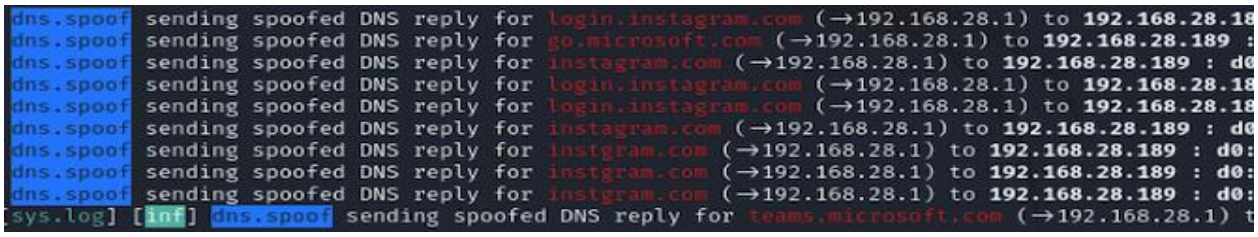


Fig. 3. Instagram Clone

the victim device can access. *Bettercap* allowed us to easily alter the DNS records for popular websites. In our setup, we redirected a victim trying to access their *Instagram* to our homemade website. For this, we created an *Instagram* clone that looked similar to the *Instagram* login page as shown in Fig. 3.



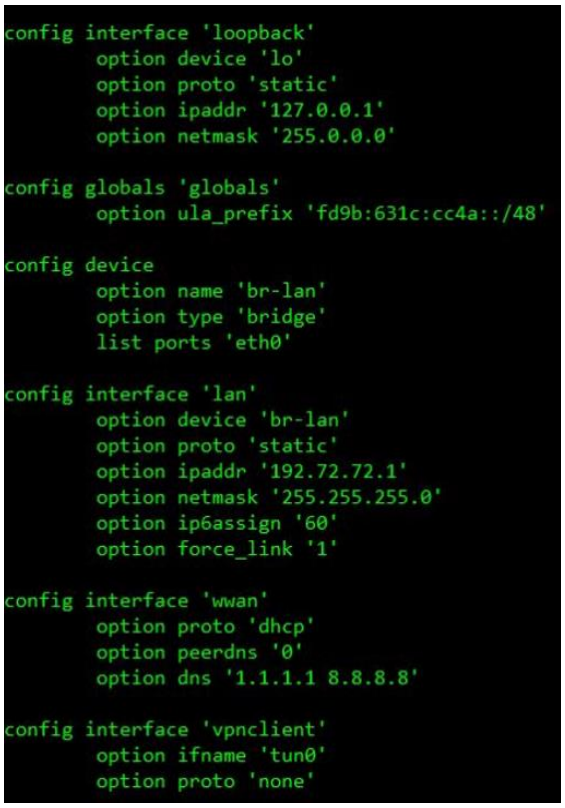
```
dns.spoof sending spoofed DNS reply for login.instagram.com (→192.168.28.1) to 192.168.28.18
dns.spoof sending spoofed DNS reply for go.microsoft.com (→192.168.28.1) to 192.168.28.189 :
dns.spoof sending spoofed DNS reply for instagram.com (→192.168.28.1) to 192.168.28.189 : d0
dns.spoof sending spoofed DNS reply for login.instagram.com (→192.168.28.1) to 192.168.28.18
dns.spoof sending spoofed DNS reply for login.instagram.com (→192.168.28.1) to 192.168.28.18
dns.spoof sending spoofed DNS reply for instagram.com (→192.168.28.1) to 192.168.28.189 : d0
dns.spoof sending spoofed DNS reply for instgram.com (→192.168.28.1) to 192.168.28.189 : d0:
dns.spoof sending spoofed DNS reply for instgram.com (→192.168.28.1) to 192.168.28.189 : d0:
dns.spoof sending spoofed DNS reply for instgram.com (→192.168.28.1) to 192.168.28.189 : d0:
[sys.log] [inf] dns.spoof sending spoofed DNS reply for teams.microsoft.com (→192.168.28.1) t
```

Fig. 4. DNS Spoofing Sample

Once we turned our DNS spoof on using *bettercap*, we modified the *Instagram* DNS record to point to 192.168.28.189 which is our local website. If the victim tried to connect to *Instagram*, the traffic would automatically redirect to our fake website (192.168.28.189). If the victim entered any information such as login and password, the attacker could capture all this information as shown in Fig. 4.

III. PISHIELD : PREVENTION OF ATTACKS

A VPN is an enterprise network based on a shared public network infrastructure that provides the same security, management, and throughput policies as applied in a private network. A VPN is one of the most effective ways to protect user data when it comes to open access wireless networks. Since a VPN can encrypt all traffic before it leaves the mobile device, it can ensure that no one can sniff the traffic and analyze browsing behaviors [5]. However, existing VPN services are paid subscriptions created for single account holders. This means only a single account has control over the VPN service, and only a limited number of



```
config interface 'loopback'
        option device 'lo'
        option proto 'static'
        option ipaddr '127.0.0.1'
        option netmask '255.0.0.0'

config globals 'globals'
        option ula_prefix 'fd9b:631c:cc4a::/48'

config device
        option name 'br-lan'
        option type 'bridge'
        list ports 'eth0'

config interface 'lan'
        option device 'br-lan'
        option proto 'static'
        option ipaddr '192.72.72.1'
        option netmask '255.255.255.0'
        option ip6assign '60'
        option force_link '1'

config interface 'wwan'
        option proto 'dhcp'
        option peerdns '0'
        option dns '1.1.1.1 8.8.8.8'

config interface 'vpnclient'
        option ifname 'tun0'
        option proto 'none'
```

Fig. 5. Network Configuration

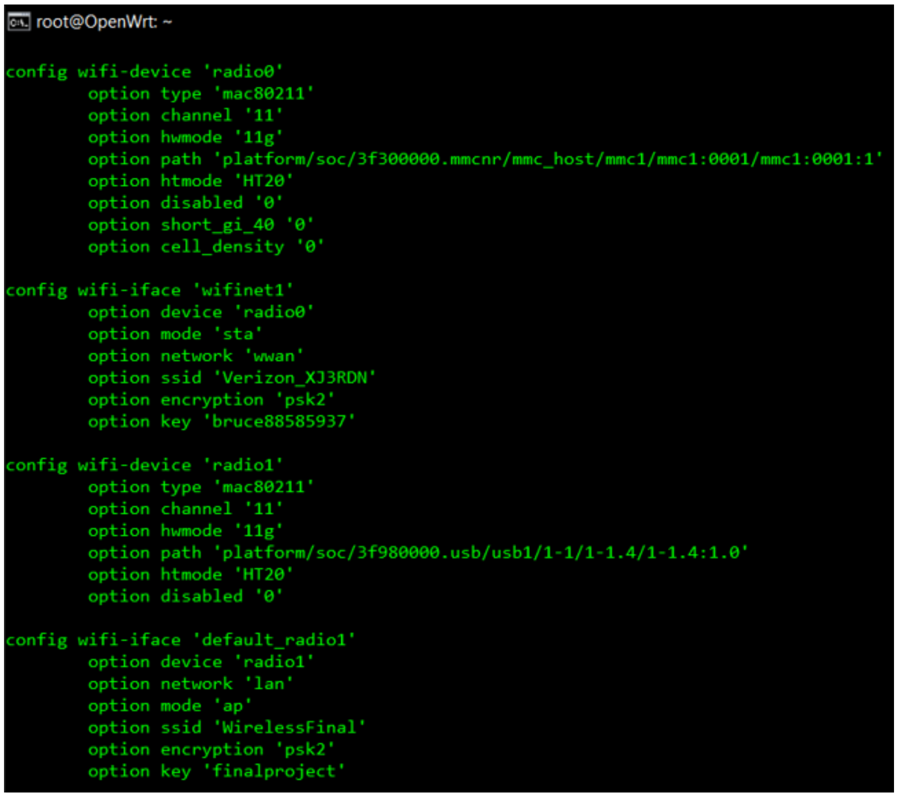
devices can be associated with the account. As the scope and power of the Internet of Things (IoT) grow, a larger group of computing devices need VPN services, and that can be very costly. In this section, we present how to create a low-cost Wi-Fi access point which has a VPN installed with *Raspberry PI*.

3.1 OpenWRT & NordVPN

We started by setting up our *Raspberry Pi* with *OpenWRT* [1], an open-source Linux router for embedded devices. *OpenWRT* turned our *Raspberry Pi* into a Wi-Fi access point. It also gave us full control over our file systems package manager and wireless configurations in our *Raspberry Pi*.

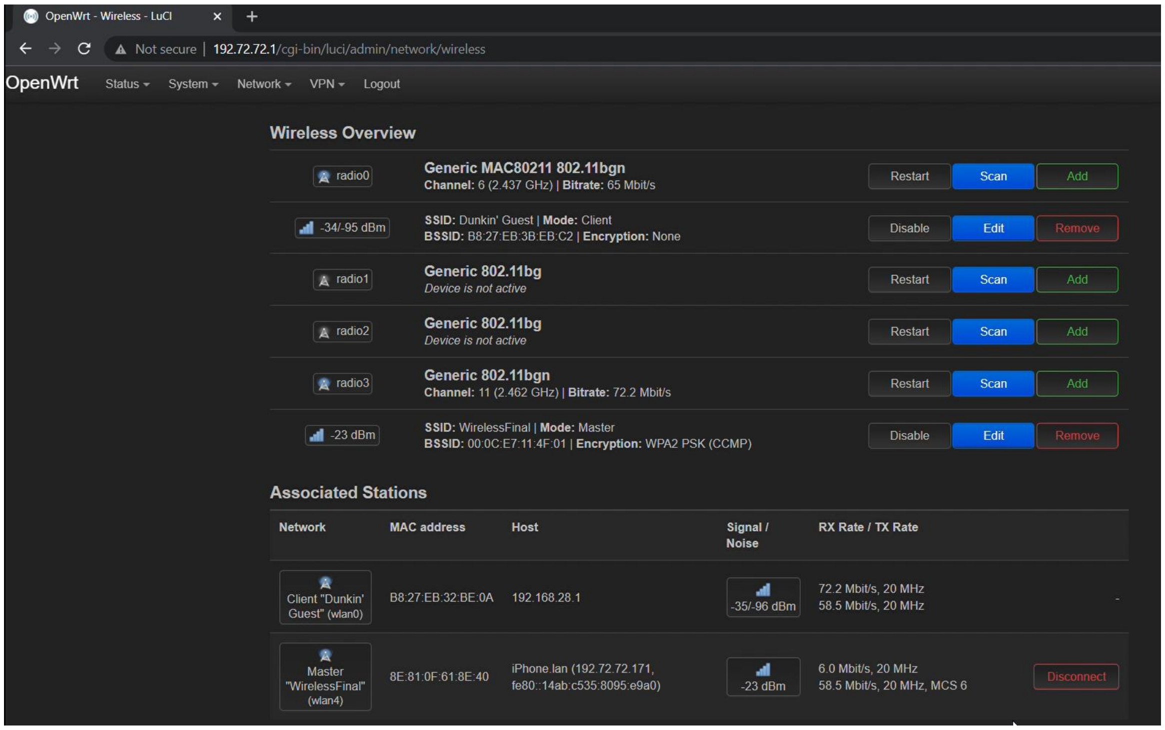
By default, *OpenWRT* has a static IP of 192.168.1.1 which allowed us to Secure Shell Protocol (SSH) into our *Raspberry Pi* and configure it from a laptop. First, we set up a password for security, then proceeded to configure our network and firewall settings. As shown in Fig. 5, we edited the *Network* file to have a private IP of our choosing. We then proceeded with the configuration by adding a *WWAN* interface and configuring the Dynamic Host Configuration Protocol (DHCP) to match the public network connection. After the network file was completed, we created a DNS server. We also created the interface for the client VPN as shown in Fig. 5. Once we saved the firewall and network file, we rebooted the *Raspberry Pi* and logged back in with our chosen IP address '192.72.72.1'. Fig. 6 shows our Wireless configuration. We used two wireless adapters to server as a bridge between the public network and the connected device.

In order to finalize *OpenWRT* setup, we had to enter the *OpenWRT* Graphical User Interface (GUI) in a browser by entering our router IP address. Similar to any other router, this provided us with an admin GUI for the router connections. From the admin webpage, users could connect to the public Wi-Fi. Our built-in wireless adapter in the *Raspberry Pi* connected to the public network, while the second wireless adapter connected via Wi-Fi USB dongle accessed to any target device.



```
root@OpenWrt: ~

config wifi-device 'radio0'
        option type 'mac80211'
        option channel '11'
        option hwmode '11g'
        option path 'platform/soc/3f300000.mmcnr/mmc_host/mmc1/mmc1:0001/mmc1:0001:1'
        option htmode 'HT20'
        option disabled '0'
        option short_gi_40 '0'
        option cell_density '0'

config wifi-iface 'wifinet1'
        option device 'radio0'
        option mode 'sta'
        option network 'wwan'
        option ssid 'Verizon_XJ3RDN'
        option encryption 'psk2'
        option key 'bruce88585937'

config wifi-device 'radio1'
        option type 'mac80211'
        option channel '11'
        option hwmode '11g'
        option path 'platform/soc/3f980000.usb/usb1/1-1/1-1.4/1-1.4:1.0'
        option htmode 'HT20'
        option disabled '0'

config wifi-iface 'default_radio1'
        option device 'radio1'
        option network 'lan'
        option mode 'ap'
        option ssid 'WirelessFinal'
        option encryption 'psk2'
        option key 'finalproject'
```

Fig. 6. Wireless Configuration


OpenWrt - Wireless - LuCI
Not secure | 192.72.72.1/cgi-bin/luci/admin/network/wireless
OpenWrt
Status
System
Network
VPN
Logout
Restart
Scan
Add
Disable
Edit
Remove
Disconnect

Wireless Overview

radio0	**Generic MAC80211 802.11bgn** **Channel:** 6 (2.437 GHz) | **Bitrate:** 65 Mbit/s	Restart Scan Add
-34/-95 dBm	**SSID:** Dunkin' Guest | **Mode:** Client **BSSID:** B8:27:EB:3B:EB:C2 | **Encryption:** None	Disable Edit Remove
radio1	**Generic 802.11bg** *Device is not active*	Restart Scan Add
radio2	**Generic 802.11bg** *Device is not active*	Restart Scan Add
radio3	**Generic 802.11bgn** **Channel:** 11 (2.462 GHz) | **Bitrate:** 72.2 Mbit/s	Restart Scan Add
-23 dBm	**SSID:** WirelessFinal | **Mode:** Master **BSSID:** 00:0C:E7:11:4F:01 | **Encryption:** WPA2 PSK (CCMP)	Disable Edit Remove

Associated Stations

Network	MAC address	Host	Signal / Noise	RX Rate / TX Rate	
Client "Dunkin' Guest" (wlan0)	B8:27:EB:32:BE:0A	192.168.28.1	-35/-96 dBm	72.2 Mbit/s, 20 MHz 58.5 Mbit/s, 20 MHz	-
Master "WirelessFinal" (wlan4)	8E:81:0F:61:8E:40	iPhone.lan (192.72.72.171, fe80::14ab:c535:8095:e9a0)	-23 dBm	6.0 Mbit/s, 20 MHz 58.5 Mbit/s, 20 MHz, MCS 6	Disconnect

Fig. 7. OpenWRT Wireless Page

After setting up the Wi-Fi access point, we set up our VPN host with *NordVPN*. Once we setup an account with *NordVPN* we could access the *OpenVPN* configuration file that we used on our *Raspberry Pi* router. Fig. 7 shows our *OpenWRT* GUI. It shows that our *Raspberry Pi* router connected to the evil twin Wi-Fi hotspot “Dunkin Guest” while accessing “WirelessFinal” VPN-enabled network.

3.2 Testing with Evil Twin

We tested our *PiShield* by re-trying the network attacks using the evil twin. As shown in Fig 7, our VPN router connected to the evil twin “Dunkin Guest.” We attempted to perform the same attacks as done before using *Bettercap*. Instantly we noticed a significant change in the package sniffing attempts. We could no longer pinpoint a package to a device. We also noticed that when the target was connected to a HTTP website the evil twin could no longer read any of the package information such as login and password, since the VPN access point encrypted this data. Lastly, we attempted the DNS spoofing attack in the attempt of redirecting the target from accessing *Instagram* and redirecting into our fake website. While the DNS records on our evil twin were modified to redirect *Instagram* into 192.183.28.1, it did not redirect the target device into our fake website. The target device connected to the authentic *Instagram* website.

IV. CONCLUSION

In this paper, we created a secure access point, *PiShield*, that would protect our public Wi-Fi access from various network attacks. We accomplished this by creating a VPN-enabled Wi-Fi access point using a *Raspberry Pi*. Our solution serves as a secure bridge between the public wireless network and the personal mobile devices. We tested our solution by connecting to an evil twin router which launched a series of network attacks such as packet sniffing and DNS spoofing. Since we presented how to create a secure VPN solution using a *Raspberry Pi* step by step, learners of all levels can build their own low-cost solution to have access to a secure network on a public Wi-Fi.

References

[1] OpenWrt, https://openwrt.org/

[2] NordVPN, https://nordvpn.com/

[3] M. Conti, N. Dragoni and V. Lesyk, "A Survey of Man In The Middle Attacks," in IEEE Communications Surveys & Tutorials, vol. 18, no. 3, pp. 2027-2051, thirdquarter 2016.

[4] F. Callegati, W. Cerroni and M. Ramilli, "Man-in-the-Middle Attack to the HTTPS Protocol," in IEEE Security & Privacy, vol. 7, no. 1, pp. 78-81, Jan.-Feb. 2009.

[5] Y. Song, C. Yang and G. Gu, "Who is peeping at your passwords at Starbucks? — To catch an evil twin access point," 2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN), 2010, pp. 323-332.

[6] P. Shrivastava, M. S. Jamal and K. Kataoka, "EvilScout: Detection and Mitigation of Evil Twin Attack in SDN Enabled WiFi," in IEEE Transactions on Network and Service Management, vol. 17, no. 1, pp. 89-102, March 2020.

[7] H. Mustafa and W. Xu, "CETAD: Detecting evil twin access point attacks in wireless hotspots," 2014 IEEE Conference on Communications and Network Security, 2014, pp. 238-246.

[8] Bettercap, https://www.bettercap.org/

[9] Kali, https://www.kali.org/