Security of Wireless Networks

Rumman Ahmed - Wasfi Momen
rahmed8@gsu.edu - wmomen1@gsu.edu
Cryptography

I. Introduction

Today, wireless networks are extremely widespread. In 2012, more than 60% of households had a wireless network as their primary source of Internet access. That was the most recent study I could find, but you can imagine what that percentage has increased to. The data that is communicated over a wireless network, as opposed to a wired network, can be seen by everyone over the network. Wireless communication is done through radio waves, so the communication is visible. Therefore, it is extremely important to encrypt and secure the communication that is done over wireless networks. In our society today, so much is transmitted over the Internet. Our personal lives are shared over the Internet through social media, we can access our bank accounts, we can conduct business across the world, and countless of other things. Most of the information is something that we want to keep secure, and it can ruin our lives if it falls into the wrong hands. This is just one of the reasons learning about the security of wireless networks is important.

In this paper, I will first discuss the general evolution of Wi-Fi networks, but specifically its security protocols. Where did we start from and where are we now? Security has always been a focus from the beginning, but it has always been flawed in some way. With every network standard introduced, a security protocol was introduced with it. However, with each of these security protocols, there have been exploits discovered. I will discuss these exploits and how to gain access to various Wi-Fi networks. All of this will lead up to a recent discovery, where a researcher found an exploit to modern WPA2 networks, which up until this point has been assumed to be for the most part secure. The lone researcher has described the exploit in detail, but has not released the exploit itself, with consideration for the individuals who have not been able to properly update their devices. He has however, released a script to determine if your own device is vulnerable, which is what I have attempted to implement myself. I have looked into the details of implementing the exploit itself, but I realized I do not have enough experience with the method of exploitation used (key reinstallation attacks).

The paper will go over the different methods of exploiting wireless networks, leading up to the newly discovered hack for WPA2 networks, using a key reinstallation attack.

II. Background

A. Why Hack Wi-Fi?

There are a few (possibly obvious) reasons why someone may want to hack a Wi-Fi network in the first place:

1. You can go on the Internet anonymously. Nowadays, there are many reasons where one may want to do this. The person may be dealing in illegal activities

online, or they may want to go on a website that isn't allowed in their country.

2. You can capture and spy on the victim's traffic. A spying person may just want to know what websites you go and who you contact.
3. You can use someone else's bandwidth to download your own files.
4. You can intercept sensitive information, like email accounts and passwords to access information about an individual.

Judging from these reasons, you can see how "rewarding" it may be for someone to hack a network

B. Security Protocols

There are established security protocols when dealing with the connection to wireless networks. There are two main reasons these protocols in place. First, you would not want random users from connecting to your personal networks. Second, you want to encrypt the communication over the network. These protocols have changed over the years, mainly because major security flaws were discovered in the former. I will now go over the protocols that have been established.

C. WEP (Wired Equivalent Privacy)

This is the original standard that was developed for wireless networks. As the name suggests, the original intention was for the wireless security be the same as the one found in wired networks. But because the communication travels differently on a wireless network, the communication cannot be secured in the same manner. The reason this protocol was flawed was because its encryption was extremely weak. It allowed keys to be reused in encryption, which is possibly the easiest way to give an attacker access. Also, the initialization vector (IV), a string of bits that's used with the key to encrypt data, was only 24-bits, which is extremely small. For a busy network, the same IV may be used for two clients, which allows attackers to decipher the key. A new standard was needed immediately.

D. WPA (Wi-Fi Protected Access)

This was a partial implementation when the flaws of WEP were discovered. To fix the major issue of WEP, WPA increased the IV to 48 bits and the size of the entire key to 128 bits. It also introduced a set of new protocols called Temporal Key Integrity Protocol, in order to help with encryption. Although this was a huge improvement to WEP's security, it still wasn't the full version and there were some flaws.

E. WPA2

This is the full implementation of the WPA standard, finalized in 2004. The most important addition to WPA2 is the use of CCMP, which is a more improved version of TKIP. CCMP uses AES for key encryption, which is one of the strongest key encryptions in modern times. Mathematically, WPA2 was proven to be secure. This is also the assumption that most people even today believe. There are still ways to gain access to a restricted WPA2 network. One method would be to use ARP Spoofing, which is essentially tricking the access point (the router) to think you are the real user. You will then receive the data before the actual user receives it. In this method, you may choose to keep any data for yourself, or you may corrupt some data before sending it as well. Another method is to spy on traffic of a WPA2-PSK (Pre-Shared Key) network. These are the networks that may seem secure at first (you need a password to connect to it), but if you know what you're doing, you can gain access. All an attacker needs is an application like Wireshark and they can deduct what the secret key is. To be safe from this, you can set up a WPA2-Enterprise network, which is what large businesses and schools use. In this kind of network, every client gets a unique key instead of a pre-shared one.

Although attacks are definitely possible as you can see listed earlier, these techniques require you to actually connect to the network. You are protected if your traffic is encrypted with HTTPS. Up until earlier this year, these techniques were assumed the only ways to hack into wireless networks. Now, there's a way to access a protected WPA2 network even without the password

III. Key Reinstallation Attack (KRACK)

This exploit was released by Mathy Vanhoef in the middle of October 2017 and was presented at the ACM Security conference in the beginning of November. The hack compromises all modern WPA2 networks, with Linux and Android devices being the most vulnerable. Attackers, who are within range of the of the device or access point can intercept data, including passwords, emails, and all other encrypted data.

A. 4-Way Handshake

KRACK targets the 4-way handshake that occurs when a client connects to a WPA2 network. During the handshake, the protocol verifies the client has the correct credentials and generates an encryption key for the communication. After message 3, the key is sent from the access to the client and installed. However, if messages are lost (if message 3 is never received), the access point will resend the key. Therefore, they key can be sent multiple times, and the client will reinstall the same encryption key.

B. Nonce Reuse

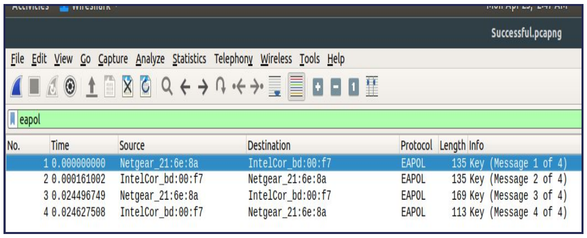
A nonce is an arbitrary number used only once along with the key to encrypt packets. In this algorithm, the initialization vector is the nonce. Each time the client receives message 3 (the key), the incremental transmit packet number (nonce) is reset. The KRACK can now force the nonce to reset and send and install the same encryption key over and over again, which the attacker uses in the replay attack.

Below: A representation of the authentication process

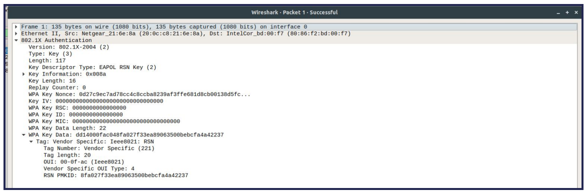
IV. Methodology

For this project, we present a model on how to derive the keys for the WPA2 4-way handshake so that an attacker would be able to sniff packets and decrypt them. The setup for the scenario is that we, the attackers, are in a coffee shop with WPA2 protected Wi-Fi, but all patrons know the WPA2 passphrase to the network. This way, we do not have to issue a dictionary attack to guess the key passphrase.

Our tools that we used in this scenario were Wireshark, a packet sniffer tool, and Python, which had the appropriate algorithms to calculate the keys. By capturing all steps of the 4-way handshake and knowing the passphrase, all traffic on the network becomes decrypted as long as no other encryption is used such as HTTPS or TLS. In the picture below, Wireshark successfully captured a 4-way handshake as can be seen by applying the "EAPOL" filter where EAPOL is the transport protocol used to transfer the keys between the router and client.


Activities
Wireshark
Successful.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
eapol

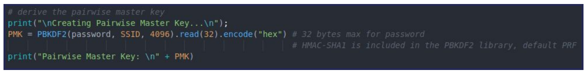
```
No.   Time         Source             Destination        Protocol Length Info
    1 0.000000000  Netgear_21:6e:8a   IntelCor_bd:00:f7  EAPOL       135 Key (Message 1 of 4)
    2 0.000161002  IntelCor_bd:00:f7  Netgear_21:6e:8a   EAPOL       135 Key (Message 2 of 4)
    3 0.024496749  Netgear_21:6e:8a   IntelCor_bd:00:f7  EAPOL       169 Key (Message 3 of 4)
    4 0.024627508  IntelCor_bd:00:f7  Netgear_21:6e:8a   EAPOL       113 Key (Message 4 of 4)
```



```
▸ Frame 1: 135 bytes on wire (1080 bits), 135 bytes captured (1080 bits) on interface 0
▸ Ethernet II, Src: Netgear_21:6e:8a (20:0c:c8:21:6e:8a), Dst: IntelCor_bd:00:f7 (80:86:f2:bd:00:f7)
▾ 802.1X Authentication
    Version: 802.1X-2004 (2)
    Type: Key (3)
    Length: 117
    Key Descriptor Type: EAPOL RSN Key (2)
  ▸ Key Information: 0x008a
    Key Length: 16
    Replay Counter: 0
    WPA Key Nonce: 0d27c9ec7ad78cc4c8ccba8239af3ffe681d8cb00138d5fc...
    Key IV: 00000000000000000000000000000000
    WPA Key RSC: 0000000000000000
    WPA Key ID: 0000000000000000
    WPA Key MIC: 00000000000000000000000000000000
    WPA Key Data Length: 22
  ▾ WPA Key Data: dd14000fac048fa027f33ea89063500bebcfa4a42237
    ▾ Tag: Vendor Specific: Ieee8021: RSN
        Tag Number: Vendor Specific (221)
        Tag length: 20
        OUI: 00-0f-ac (Ieee8021)
        Vendor Specific OUI Type: 4
        RSN PMKID: 8fa027f33ea89063500bebcfa4a42237
```

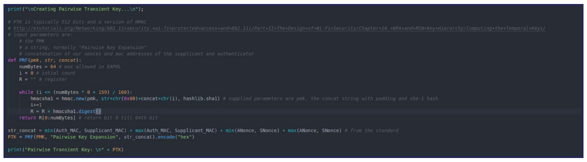
As we can see, the data contained within the packet is unencrypted from the router to the client. The first step in the handshake sends over a nonce value that allows for clients to compute the keys required for the next steps.

After the handshake and first nonce are sent, we can start to compute the keys. The first key to derive is called the Pairwise Master Key, which drives the production of all other keys. In the 802.11 standard, the PMK is computed using the Password-Based Key Derivation Function (PBKDF2) which internally uses a pseudorandom function Hash Based Message Authentication (HMAC) to compute the key over 4096 iterations.



```
# derive the pairwise master key
print("\nCreating Pairwise Master Key...\n");
PMK = PBKDF2(password, SSID, 4096).read(32).encode("hex") # 32 bytes max for password
                                                          # HMAC-SHA1 is included in the PBKDF2 library, default PRF
print("Pairwise Master Key: \n" + PMK)
```

After derivation of the first key, the client generates its own nonce value that it uses with the PMK to create the Pairwise Transient Key (PTK). The PTK is actually a concatenation of keys that will be used later in the handshake to verify the client's integrity and encryption later data frames. According to the 802.11 standard, a psuedorandom function should be used compute the key. We used HMAC-SHA1 and followed the standard's parameters by using the PMK, a string (normally "Pairwise Key Expansion", and a concatenation of the min-max of the MAC address of the client and router. Once the PTK is created, the client sends its nonce value and a special Message Integrity Code (MIC) value to verify its knowledge of the private pre-shared passphrase. Once the router gets this information, it can verify that the client knows the pre-shared passphrase and start the other 2 handshakes to create the new group key. For the client, key generation is completed.



```
print("\nCreating Pairwise Transient Key...\n");

# PTK is typically 512 bits and a version of HMAC
# http://etutorials.org/Networking/802.11+security.+wi-fi+protected+access+and+802.11i/Part+II+The+Design+of+Wi-Fi+Security/Chapter+10.+WPA+and+RSN+Key+Hierarchy/Computing+the+Temporal+Keys/
# input parameters are:
    # the PMK
    # a string, normally "Pairwise Key Expansion"
    # concatenation of our nonces and mac addresses of the supplicant and authenticator
def PRF(pmk, str, concat):
    numBytes = 64 # max allowed in EAPOL
    i = 0 # intial count
    R = '' # register

    while (i <= (numBytes * 8 + 159) / 160):
        hmacsha1 = hmac.new(pmk, str+chr(0x00)+concat+chr(i), hashlib.sha1) # supplied parameters are pmk, the concat string with padding and sha-1 hash
        i+=1
        R = R + hmacsha1.digest()
    return R[0:numBytes] # return bit 0 till 64th bit

str_concat = min(Auth_MAC, Supplicant_MAC) + max(Auth_MAC, Supplicant_MAC) + min(ANonce, SNonce) + max(ANonce, SNonce) # from the standard
PTK = PRF(PMK, "Pairwise Key Expansion", str_concat).encode("hex")

print("Pairwise Transient Key: \n" + PTK)
```

By using Python code to generate the keys, we can understand how the 4-way handshake works in detail and how some tools like aircrack can compute the keys and decrypt the data with a known passphrase.

V. Conclusion

Although the script works successfully, I was not able to modify it in the way that I had intended. The code was written in Python, which is already a portable language. The modules were written in a way where it did not need to be modified for use. Also, I am a beginner in Python. I had some experience with it in a data analytics class, which is why I wanted to take on this challenge. However, this code was beyond my level, and I could not do much work with it. In the future, I would have more experience with the Python language itself, and I would set my sights on a smaller goal. The researcher has spent years on this topic, so it wasn't extremely wise of me to think I would be able to tackle this in only a couple of months.

Among the other valuable lessons I learned from this research is the importance of security in our day-to-day lives, especially considering how many of us have no clue about it. I am sure you would be able to go around and ask about the KRACK exploit that was released, and how it affects “secure” WPA2 networks, and even many students of Computer Science would not know about it. The study of security, especially for networks, in which we share all aspects of our lives today, is something that has gained my interest greatly in the past few months, and even more after researching this topic. There is still a lot of work to be done in this area of wireless network security. Although a method may seem secure now, there will be a day where that may not be said again.

References

Jon Edney and William A. Arbaugh. 2003. Derive of Key Derivation for WPA. In "Real 802.11 Security: Wi-Fi Protected Access and 802.11i". 1st Edition.

Mathy Vanhoef and Frank Piessens. 2017. Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2. In Proceedings of the 24th ACM Conference on Computer and Communications Security (CCS). ACM.

http://searchsecurity.techtarget.com

https://www.veracode.com/security/arp-spoofing

https://www.howtogeek.com/204335/warning-encrypted-wpa2-wi-fi-networks-are-still-vulnerable-to-snooping/