

WEP: Dead Again, Part 2

Michael Ossmann 2005-03-08

Introduction

In [part one](#) we examined the latest generation of passive WEP cracking tools that use statistical or brute force techniques to recover WEP encryption keys from captured wireless network traffic. This time, in the second and final article, we take a look at active tools that use 802.11 transmissions to attack WEP networks.

All of these active wireless attack techniques discussed in this article require the ability to inject arbitrary packets onto a wireless network. Although a variety of injection methods are available, most require Linux, are unsupported, and use hacked drivers that have support and availability problems. All of them require at least one wireless PCMCIA card based on the Prism2 chipset (such as the Senao 2511-CD-PLUS). Fortunately, the Auditor Security Collection [[ref 1](#)] live cd-rom can save you a number of headaches as it includes ready-to-use drivers for several active attack tools.

Beware of network disruptions that can be caused by active attacks. Using these tools may have unpredictable effects in various environments. In my testing, I have encountered a few systems that had to be rebooted in order to function again after being bombarded with injected packets.

Rapid traffic generation

If you've spent much time sniffing wireless networks (and, if you are reading this article, I bet you have) then you probably have noticed that the source and destination MAC addresses are plainly visible for every packet even when the packet contents are encrypted with WEP. This allows you to uniquely identify hosts on the wireless network as well as hosts on a bridged, wired LAN. If you've never tried traffic analysis of an encrypted wireless network, I highly recommend the exercise. Find a busy network, fire up Ethereal [[ref 2](#)], and try to answer as many of the following questions as you can:

- How many access points share the same ESSID?
- Does the access point bridge or route traffic?
- Is EAP used? If so, what EAP type?

- Is open system or shared key authentication in use?
- What is the MAC address of the default gateway?
- What are the NIC vendors for wireless hosts?
- What are the NIC vendors for wired hosts?
- What is the vendor of the access point?
- Can you find a DNS transaction?
- Can you find a TCP three-way handshake?
- Can you find an HTTP transaction?
- What hosts transmit/receive the most bytes/packets?
- Does any traffic occur with a distinct periodicity (like POP3 every 5 minutes)?
- Can you find any ARP traffic? (hint: frame.pkt_len==68 and wlan.da==ff:ff:ff:ff:ff:ff)

No wireless network based on WEP provides protection against replay attacks. With the right tools, you can take any captured packet and reinject it back onto the network. The packet will be correctly encrypted even though you have no idea of its contents. Then again, you may have a pretty good guess as to its contents based on traffic analysis. You might choose something that is likely to be an ARP request, hoping that it will generate a response from another host on the network. If you're right, you could replay the same packet hundreds or even thousands of times per second, forcing that host to spew an enormous stream of responses, individually encrypted with different IVs.

This method described is exactly the method used by `aireplay`, a tool that comes with `aircrack` [ref 3]. A screenshot of `aireplay` is shown below in Figure 1. As we discovered in [part one](#), both `aircrack` and `WepLab` [ref 4] are capable of cracking WEP keys after collecting just a few hundred thousand packets. With a successful `aireplay` attack, you can generate that many packets in just a few minutes. Therefore, people who say that re-keying every 10 minutes makes WEP unbreakable are dead wrong. Per-session, per-user keys also don't stand a chance against this attack. WEP is truly dead. . . again.

```

wifitest / # iwpriv wlan0 hostapd 1
wifitest / # iwpriv wlan0ap host_encrypt 1
wifitest / # iwpriv wlan0ap host_decrypt 1
wifitest / # iwconfig wlan0ap retry 1
wifitest / # iwconfig wlan0ap mode Master
wifitest / # iwconfig wlan0ap key 01:02:04:08:10
wifitest / # iwconfig wlan0ap channel 1
wifitest / # ifconfig wlan0 hw ether 00:2E:9F:8B:22:8C
wifitest / # ifconfig wlan0ap up
wifitest / # aireplay wlan0ap replay.pcap
Choosing first encrypted BSSID = 00:23:EF:3F:20:2F
Read 920 packets, got 1 potential ARP requests.
Sent 5200 packets.

```

Figure 1. Aireplay at work.

The Auditor Security Collection live cd-rom makes it relatively easy to try aireplay because it includes aircrack's patched hostap driver by default, but you will need two wireless cards with at least several inches distance between their antennas. You may find it easier to use two laptops, one with a Prism2 card to replay captured packets, and a second to capture all the new traffic that is generated. Be prepared to spend some time finding an appropriate packet to replay; you may need to save individual packets with Ethereal and feed them to aireplay.

Another tool that implements a similar attack has been around for much longer in the BSD world. Part of OpenBSD's Wnet, reinj performs the same attack as aireplay and does it all with just one Prism2 card (as does the latest beta of aireplay). Whichever tool you use to generate traffic, I recommend WepLab or aircrack for cracking the WEP key.

Encrypted packet injection

Most of the WEP attack tools on the scene today focus on cracking WEP keys, but there are also other WEP vulnerabilities that can be exploited. WEPWedgie [ref 5], a tool released in 2003 by Anton Rager, allows an attacker to craft an arbitrary plaintext packet and inject it into the wireless network without knowledge of the WEP key. The receiving stations accept the packet as if the sender used the correct key to encrypt the packet. The way WEPWedgie is able to accomplish this is by reconstructing the keystream that was used to encrypt a particular plaintext. With knowledge of some plaintext and the resulting ciphertext, a simple XOR operation yields the keystream that results from a particular IV. And because WEP allows the same IV to be used over and over again, WEPWedgie can use the keystream to correctly encrypt and inject any number of packets whose contents are limited only by the length of the known keystream.

There are a number of ways that an attacker can discover the ciphertext for a known plaintext, but the method used by WEPWedgie's prgasnarf is to listen for shared key authentication. The 802.11 standard defines two types of authentication, "open system authentication" (which you can think of as "no authentication") and "shared key authentication" (which you can think of as "the most misguided authentication mechanism ever devised"). In shared key authentication, the AP transmits 128 bytes of plaintext, and then the station encrypts the plaintext and transmits the resulting ciphertext using the same key and cipher that are used by WEP to encrypt subsequent network traffic. Believe it or not, this horrifying scheme is still being recommended by certain vendors [ref 6] as a security enhancement, but it is less common in practice than open system authentication.

Once a keystream has been captured (hint: spoofed deauthentication), WEPWedgie provides a number of interesting packet injection attacks. A simple one sends a ping to a target of your choice. The other attacks provide a method of port scanning targets on the wireless network using a chosen source address. As long as the target network has Internet connectivity, you can use the address of a host you control on a remote network and sniff the results of your scan on that host. Interpretation of the results is up to you.

```
wifitest / # prgasnarf -c 1
Auth Frame: Auth Type: Shared-Key - 00 01:00:01:00
Auth Frame: Auth Type: Shared-Key - 01 01:00:02:00 ;seq = 02 ; Challenge Frame?
Auth Frame: [3]Encrypted Auth Response
Auth Frame: [4]responder OK with auth

BSSID: 0023ef3f202f      SourceMAC: 0060c10bb76e
Created 136byte PRGA for IV: b9:00:95
Created prgafile.dat in current directory
wifitest / # wepwedgie -h c0:a8:00:be -t c0:a8:00:01 -S 2 -c 1
Pingscanning Selected
Reading prgafile.dat
BSSID:      00:23:ef:3f:20:2f
Source MAC: 00:60:c1:0b:b7:6e
IV:         b9:00:95:00
Pingscan
Setting last byte of target IP to 0 -- scanning 192.168.0.0-192.168.0.255
Injecting Ping...192.168.0.190->192.168.0.0
Injecting Ping...192.168.0.190->192.168.0.1
Injecting Ping...192.168.0.190->192.168.0.2
Injecting Ping...192.168.0.190->192.168.0.3
Injecting Ping...192.168.0.190->192.168.0.4
Injecting Ping...192.168.0.190->192.168.0.5
```

Figure 2. Wepwedgie injecting pings.

To try out WEPWedgie, you'll need a system running a Linux 2.4 kernel, a Prism2 card, and Abaddon's AirJack [ref 7] driver. Unfortunately the Auditor CD's 2.6 kernel isn't supported by

AirJack, so you'll have to prepare a system on your own. You might find the Wi-Fi Dog of War [ref 8] instructions helpful to get AirJack working.

Single packet decryption

KoreK, the individual who brought us the improved algorithms used in aircrack and WepLab, released a tool a few months ago on the NetStumbler forums that enables an attacker to decrypt individual packets without knowledge of the WEP key. Called chopchop [ref 9], this tool replays a single encrypted packet, modifying one byte at a time. By monitoring the access point to find out if it accepts the modified packet, chopchop is able to determine the plaintext value of that particular byte and move on to the next. Within several seconds (and thousands of replayed packets), chopchop can decrypt an entire packet. It doesn't matter what encryption key was used, or if a separate key is used for each user, or if the key changes every hour or minute; any packet can be decrypted.

```
wifitest / # tethereal -nr test.pcap
 1  0.000000 00:60:c1:0b:b7:6e -> 00:10:5a:35:8e:c1 IEEE 802.11 Data
wifitest / # switch-to-wlanng
wifitest / # monitor,wlan wlan0 1
wifitest / # time -p chopchop -burst 40 -m 00:60:c1:0b:b7:6e \
-b 00:23:ef:3f:20:2f -p test.pcap
00:60:c1:0b:b7:6e 6
00:23:ef:3f:20:2f 6
0
first pass
-----
packet number 001
OK

second pass
real 34.35
user 0.01
sys 9.30
wifitest / # tethereal -nr test.pcap.dec
 1  0.000000 192.168.0.192 -> 192.168.0.2 ICMP Echo (ping) request
```

Figure 3. Chopchop decrypting a single packet.

You can use the Auditor CD and a single Prism2 card to try chopchop. Use the switch-to-wlanng script that Auditor provides, pop the card out and then back in again, and the linux-wlan-ng driver will be working, complete with KoreK's injection modifications.

The next generation

Since the release of chopchop, the task of acquiring a valid keystream for encrypted packet

injection has become trivial for all WEP encrypted networks. Joshua Wright is working on a new version of WEPWedgie that incorporates the chopchop attack and works with newer drivers. Christophe Devine's upcoming version of aireplay, already released as a beta, uses the same technique to allow the forgery of any ARP request. Various people are working to improve wireless drivers, including implementation of packet injection with a wider variety of hardware (prism54 is reported to work already), and construction of an abstraction layer for packet injection.

Conclusion

Some vendors continue to sell products that completely lack reasonable wireless security features. In just two months since the publication of part one of this article, I've encountered multiple brand new devices, including Wi-Fi VOIP phones and an access point provided by a cable Internet provider, that provide no encryption capability other than WEP. As long as this continues, white hats and black hats alike will keep improving the attack techniques that render WEP even worse than useless.

For the most part, the newer WEP attack tools exploit vulnerabilities that were described in theory four or more years ago. Perhaps people will learn from the history of WEP the lesson that theoretical vulnerabilities will become real vulnerabilities. Until they do, you can use these penetration testing tools to assess the weaknesses of your own network and maybe even convince someone that change is needed.

Tools and links

[1] Auditor Security Collection: <http://remote-exploit.org/?page=auditor>

[2] Ethereal: <http://www.ethereal.com/>

[3] aircrack: <http://www.cr0.net:8040/code/network/aircrack/>

[4] WepLab: <http://weplab.sourceforge.net/>

[5] WEPWedgie: <http://sourceforge.net/projects/wepwedgie/>

[6] Linksys recommends shared key authentication: <http://www.linksys.com/splash/>

[wirelessnotes.asp](#)

[7] AirJack: <http://sourceforge.net/projects/airjack/>

[8] Wi-Fi Dog of War Mini How-To: http://www.geekspeed.net/~beetle/download/wifi_dog.html

[9] chopchop: <http://www.netstumbler.org/showthread.php?t=12489>

About the author

[Michael Ossmann](#) is a security administrator for Exempla Healthcare.

[Privacy Statement](#)

Copyright 2006, SecurityFocus