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.

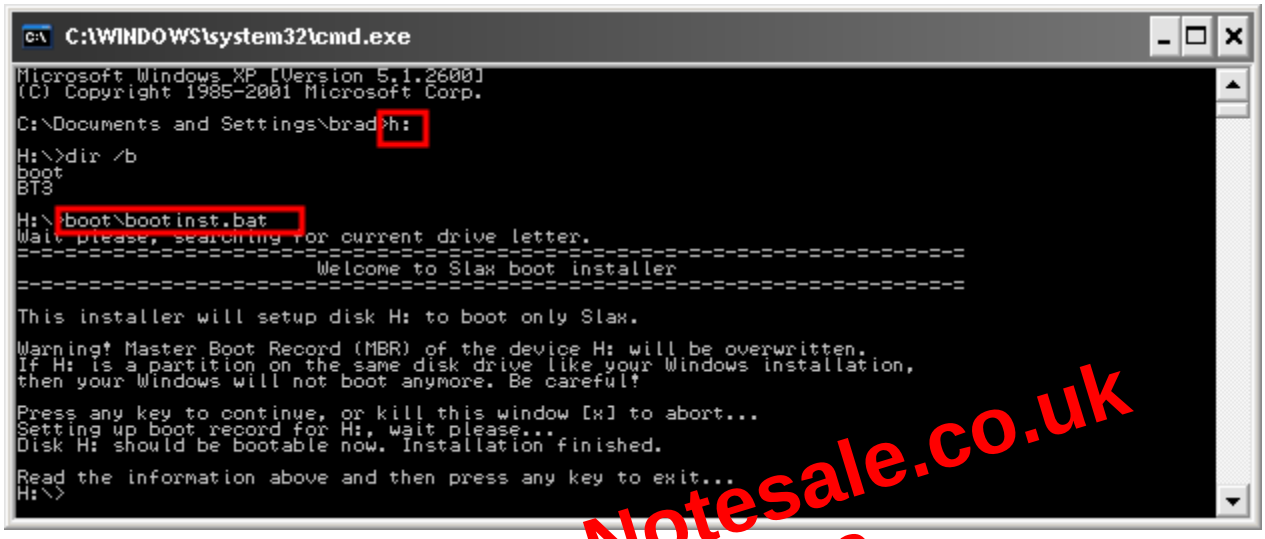

Figure 1: Reformating the MBR on your USB stick

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.

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 "`macchanger.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 ------------------------
```

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

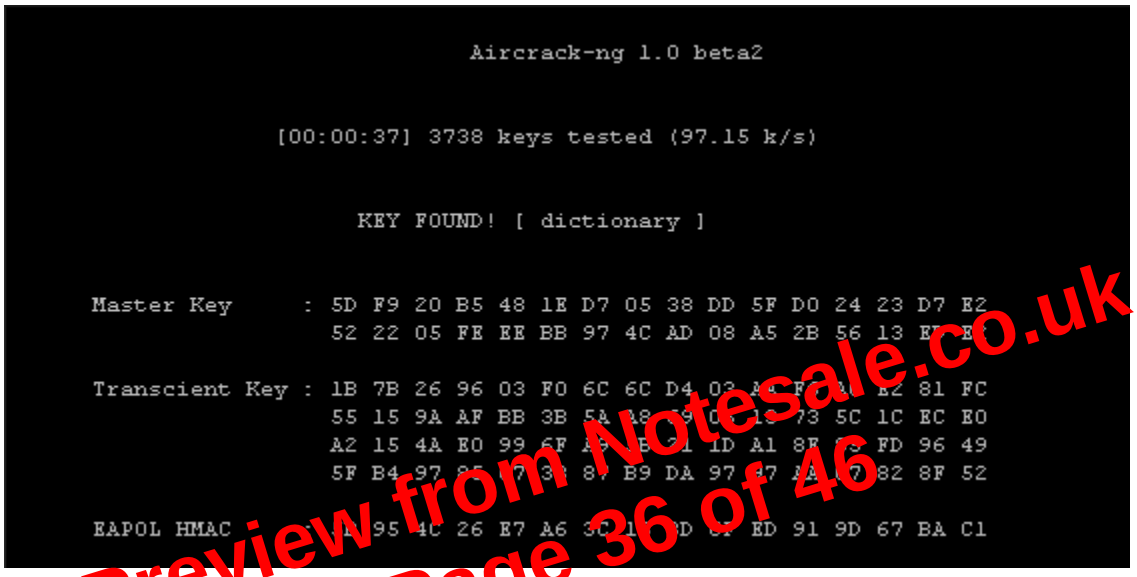

Figure 17: Using aircrack-ng to brute force WPA-PSK

```
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.

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 victim 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

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

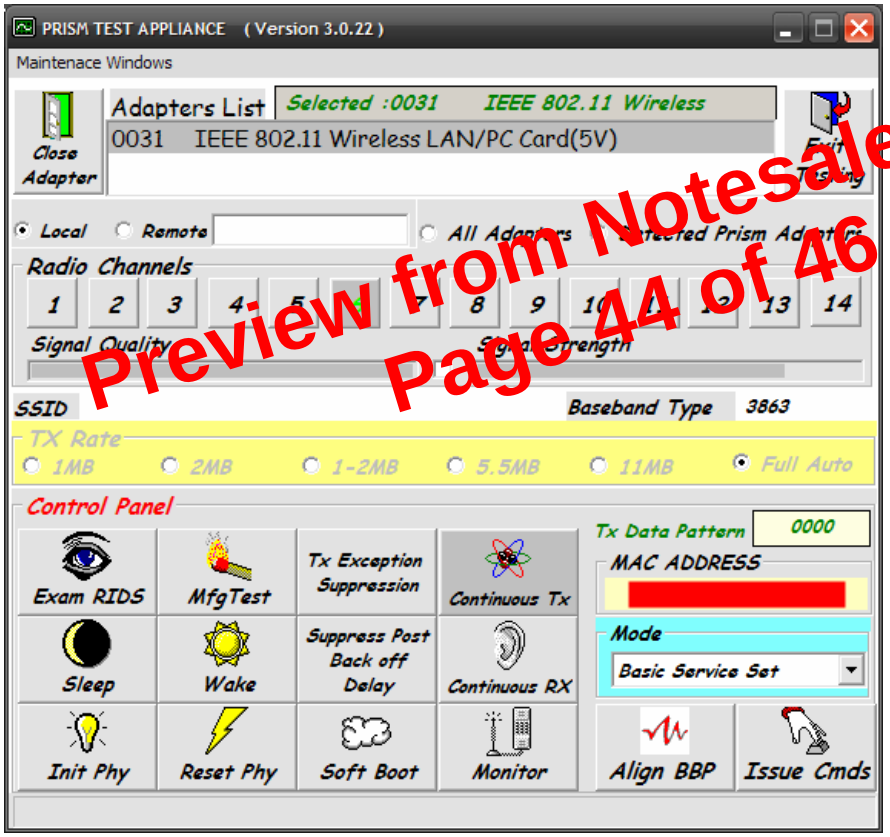

Figure 20: Prism Test Utility