Apr 10 2016

Some Background

I was doing an assessment recently on a piece of hardware and I wanted to sniff the traffic leaving the device over it's wireless interface. In the office we have networking equipment available that would allow me to just capture this traffic off a local switch or router using `tcpdump`. This assessment was being done on-site in the client's office though, so that wasn't an option this time around. I decided that the quickest, easiest and least intrusive way to get to the devices wireless traffic would be to setup a wireless access point in Kali and then sniff the traffic as it came across.

Setting up Kali

I always keep an Alfa Wireless Adapter in my bag. It's a great, cheap adapter that supports promiscuous mode in Linux. I picked a couple up a while ago when I started playing around with mana (which is a great way to set up an EvilAP attack by the way). The basic setup looks like this:

- Connect the laptop to the network the device would normally connect to
- Plug in the Alfa adapter and pass the USB connection through to the Kali VM
- Make sure the Kali VM's network adapter is in bridged mode so it's

You'll need to figure out what interface the Alfa adapter is showing up as. You can do this by running `ifconfig -a`. You'll also need to make sure that the network-manager service isn't managing this interface (it'll cause issues later), you can do that by adding the adapters MAC address to `/etc/NetworkManager/NetworkManager.conf` with a line that looks like:

```
[keyfile]
unmanaged-devices=mac:xx:xx:xx:xx:xx:xx
```

Later in this setup, we'll be giving an IP address to this interface. I ended up using `10.0.0.1` which you will see referenced in the config files below.

Hostapd

Hostapd is used to create access points in Linux. On kali, it can be installed with `apt-get install hostpad`

Configuration is simple, below is what I ended up using. You'll have to update it to point to the proper interface. The nl80211 driver seems pretty universal. Save this config to `hostapd.conf`

```
interface=wlan0
driver=nl80211
ssid=Monitor-Network
channel=1
macaddr_acl=1
accept_mac_file=./allowed_macs
logger_syslog=-1
logger_syslog_level=2
```

This config will set up an open wireless access point named "Monitor-Network". While it's certainly not bullet proof, for the purposes of this assessment I just used MAC filtering to make sure no other devices joined the SSID while I had it up. MAC filtering is handled by a white list of MAC addresses stored in `allowed_macs`. The contents of the file is just one MAC address per line.

Dnsmasq

Dnsmasq is going to act as our DNS and DHCP server, it can be installed with `apt-get install dnsmasq`. This is another super simple service with an easy to understand config file. Below is what I used, it defines a DHCP range, sets the router and DNS servers as 10.0.0.1 (options 3 and 6) and sets our upstream DNS server to one of Google's public DNS servers (`server=8.8.8.8`).

Edit as needed and save the file as `dnsmasq.conf`

```
interface=wlan0
dhcp-range=10.0.0.10,10.0.0.100,8h
dhcp-option=3,10.0.0.1
dhcp-option=6,10.0.0.1
server=8.8.8.8
log-queries
log-dhcp
```

When we start dnsmasq, we're also going to tell it to use a `dns_entries` file for local DNS resolution in addition to the contents of `/etc/hosts`. This `dns_entries` file is in the format of "[dns name] [ip address]" much like `/etc/hosts`.

Putting it together.

I'm lazy, so I script out anything I'm going to do more than once. Below is the script I wrote for actually setting up the access point. Some things to note:

- `OUTPUT_DEVICE` is the device that's connected to the network you're sending traffic out to. For example, the interface that's connected to the internet or, in the case of this assessment, the interface that was connected to the corporate network.
- This file is where we set the IP address for our monitoring interface (`ifconfig $MONITOR_DEVICE..`)
- The `iptables` rules set up NATing so that traffic can come in the `MONITOR_INTERFACE` and out the `OUTPUT_INTERFACE`.
- The script starts an instance of `tshark` listening on the monitoring

interface and dumping packets out to `output.pcap`. This pcap file will be overwritten every time the script is run so make sure to save it if you need it.

```
#!/bin/bash

MONITOR_DEVICE=wlan0
OUTPUT_DEVICE=eth0

# Catch ctrl c so we can exit cleanly
trap ctrl_c INT
function ctrl_c(){
    echo Killing processes..
    killall dnsmasq
    killall hostapd
}

ifconfig $MONITOR_DEVICE 10.0.0.1/24 up
dnsmasq -C dnsmasq.conf -H dns_entries
sysctl -w net.ipv4.ip_forward=1
iptables -P FORWARD ACCEPT
iptables --table nat -A POSTROUTING -o $OUTPUT_DEVICE -j MASQUERADE
hostapd ./hostapd.conf -B
  hark -i $MONITOR_DEVICE  -w output.pcap -P
```

Once started, you should see a new SSID named "Monitor-Network" and you should see any traffic on that network come across tshark. Ending the script with `ctrl + c` will kill the dnsmasq and hostapd processes. If for some reason the script ends without `ctrl + c`, you will have to kill those processes manually.

Acknowledgements

Most of this was lifted from this great article on setting up APs in Kali. I made some tweaks to the configs and wrote a script, but that's about it.

When I was going through this, I kept running into issues with hostapd not grabbing the interface properly. This answer on AskUbuntu helped with getting the interface omitted from Network Manager.

You can follow new posts on this site through RSS or by following me on

twitter.