Virginia Tech ■ ECE/CS 4570: Wireless Networks and Mobile Systems ■ Spring 2006

In-class Laboratory Exercise 12 (L14)

Part I – Objectives and Laboratory Materials

Objectives

- Observe security vulnerabilities in wireless local area networks (WLANs), particularly IEEE 802.11 WLANs
- Observe denial of service (DoS) attacks that target IEEE 802.11 WLANs
- Observe the operation of Kismet, an IEEE 802.11 WLAN detector, sniffer, and intrusion detection system (IDS)

Hardware to be used in this laboratory assignment

- Dell notebook computer, *with a fully charged battery,* and an IEEE 802.11b card
- Compaq iPAQ, *with a fully charged battery,* and a dual card sleeve with an IEEE 802.11b card
- One Intel WLAN gateway (provided by the GTA)

Software to be used in this laboratory assignment

- Kismet on the notebook under Linux
- Ethereal on the notebook under Linux
- Xircom link status meter on the iPAQ
- vxUtil on the iPAQ

Overview

The purpose of this laboratory exercise is to introduce some of the vulnerabilities of WLANs. WLANs face a number of challenges with respect to security. Among these challenges are the lack of a clear line of defense or any traffic concentration points, the broadcast nature of the transmission medium, the dynamically changing topology due to mobile hosts, the reliance on node collaboration as a key factor of network survivability, and the severely constrained computational and energy resources of some hosts. Such challenges leave such networks susceptible to a number of attacks ranging from brute-force authentication key cracking attacks to denial of service (DoS) attacks.

Pre-laboratory preparation

Before the in-class laboratory session, you should review this laboratory assignment and, also, read the following overview of the Kismet tool.

> A. Weiss, "Introduction to Kismet," March 30, 2006, Available at: http://www.wi-fiplanet.com/tutorials/article.php/3595531.

Part II – In-class Laboratory Assignment

Part A: Spoofing the Intel gateway's IP address (ARP cache poisoning)

The goal of this experiment is to observe the effect of spoofing the IP address of a WLAN gateway. Upon spoofing the IP address of the Intel gateway by an attacker, traffic destined for the wide area network (WAN) interface will be maliciously forwarded to the attacker, leading to failure of communication attempts with nodes on the WAN.

Use the following procedure for this experiment (see Figure 1).

- ❑ Boot your notebook in Linux with the IEEE 802.11b card in a PC Card slot. NVC groups should use *"WNMS"* as the ESSID. Groups in Blacksburg will be divided evenly into two networks. Groups in Network A should use *"WNMSA"* as the ESSID and groups in Network B should use *"WNMSB"* as the ESSID. Verify your ESSID with the GTA.

- ❑ Manually assign IP address *10.10.1.{group number}* to your notebook computer. Make sure the subnet mask is *255.255.255.0* and the broadcast address is *10.10.1.255* using the following command.

  ```
  > ifconfig eth1 netmask 255.255.255.0 broadcast 10.10.1.255
  ```

  Verify your connectivity to the network by pinging the Intel gateway at IP address *10.10.1.100*.

- ❑ Make the access point (acting as a gateway) the default router by executing the following command.

  ```
  > route add default gw 10.10.1.100
  ```

  Test your connectivity to the WAN by accessing the web server at *192.168.1.101* using your web browser.

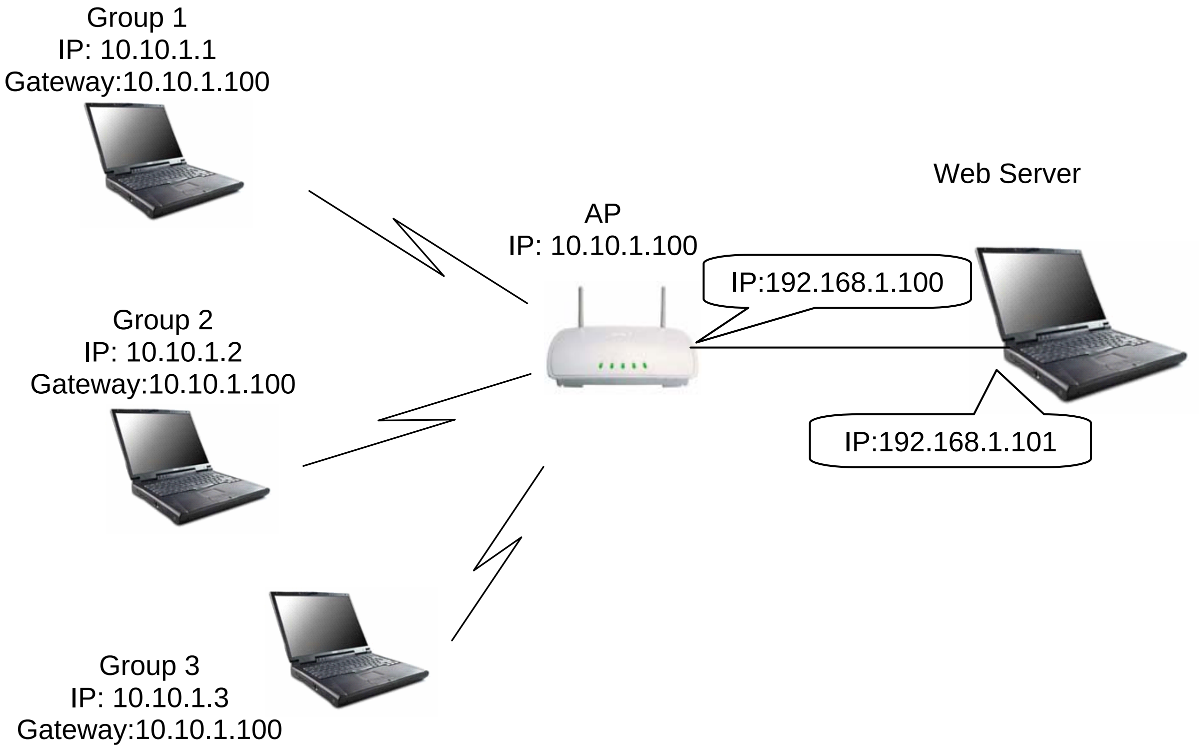


Figure 1. Network setup for Part A (spoofing the gateway's IP address).

- ❑ All groups should start Ethereal and use menu "Capture->Start" to open the "Capture Options" dialog box. Select the IEEE 802.11b wireless interface (eth1). Disable the "capture packets in promiscuous mode," "MAC name resolution," and "transport name resolution" options. Enable the "update list of packets in realtime" option. Click "OK" to start tracing packets.

- ❑ Check the entries in your routing table by running the following command.

  ```
  > route –n
  ```

  Capture a screenshot of the table and notice the entry for the default route.

- ❑ Check the entries in your ARP table by running the following command:

  ```
  > arp –n –a
  ```

  Capture a screenshot of the table. Observe the MAC address of the default route noted above.

- ❑ The group with the lowest group id from each network (group 1 in *WNMS*, group 1 in *WNMSA*, and group 8 in *WNMSB*) will change their IP address to the IP address of the Intel gateway noted above. This can be done using the following command

  ```
  > ifconfig eth1 10.10.1.100
  ```

  This group will then ping the group with the next lowest id from the same network (group 2 in *WNMS*, group 2 in *WNMSA*, and group 9 in *WNMSB*).

- ❑ Check the entries in your ARP table again and capture a screenshot of the table. Observe the MAC address of the default route.

- ❑ Observe that the web server is not reachable and that any attempts to ping it or access it will timeout.

- ❑ Stop the Ethereal capture and save the captured trace for analysis in the associated at-home exercise.

Part B: Network sniffing, detection, and intrusion detection

Kismet is an IEEE 802.11 Layer 2 wireless network detector, sniffer, and intrusion detection system. Kismet will work with any wireless card that supports raw monitoring (rfmon) mode, and can sniff IEEE 802.11b, 802.11a, and 802.11g traffic. Kismet identifies networks by passively collecting packets and detecting standard named networks, detecting hidden networks, and inferring the presence of networks that do not send a beacon via data traffic. We will explore how Kismet detects a number of intrusion attempts.

Use the following procedure for this experiment.

- ❑ Update the following parameters in the kismet config files.

  a. *kismet.conf* located in /usr/local/etc:

     Open the file *kismet.conf* in a text editor, like vi or kedit, and make sure the value of the active *source* parameter reflects the driver used. In our case, there should be a line that says something like the following.

     ```
     source=cisco,eth1,ciscosource
     ```

     This statement identifies the driver, the interface, and the name of the source.

  b. *kismet_ui.conf* located in */usr/local/etc*:

     Open the file *kismet_ui.conf* in a text editor, like vi or kedit, and change the value of *host* to the IP address and port number given to you by the GTA.

     ```
     host=IP_ADDRESS:PORT_NUMBER
     ```

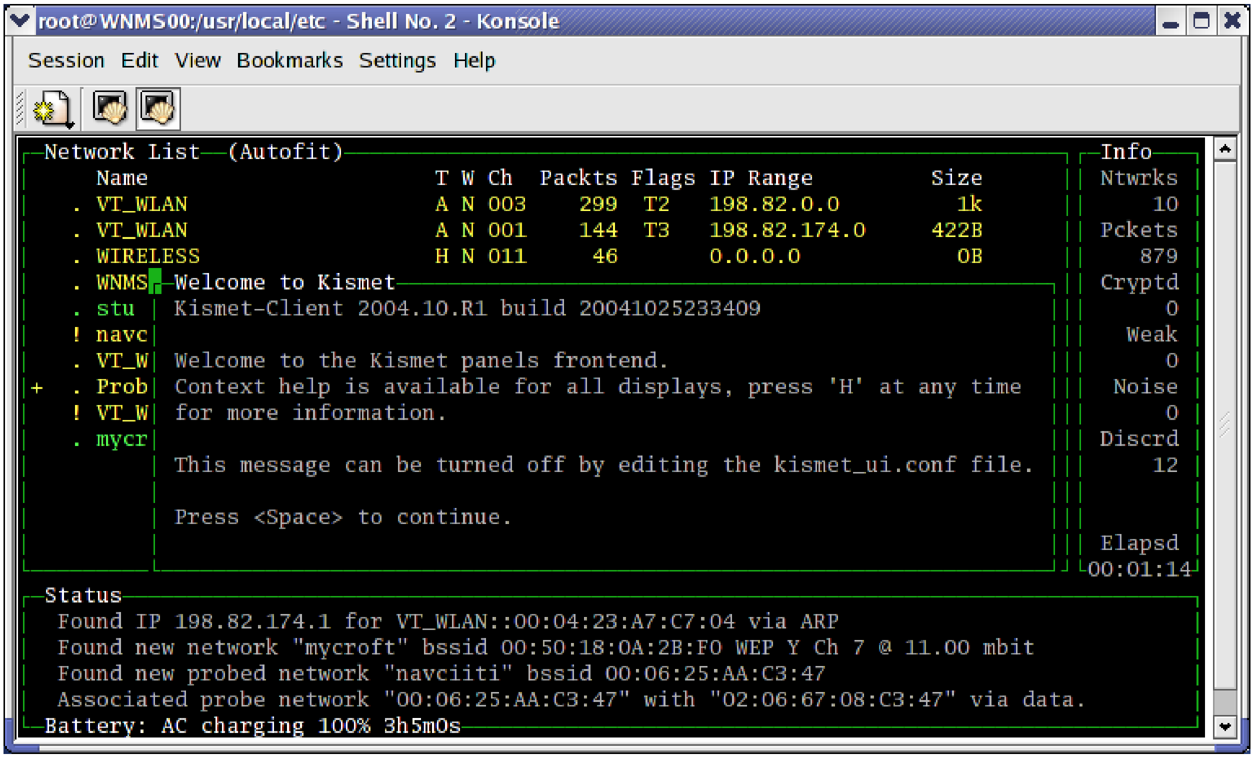
- ❑ Set up an infrastructure network consisting of student notebooks and the Intel WLAN gateway setup by the GTA. The ESSID should be *"WNMS"* at the NVC and *"WNMSA"* in Blacksburg.

Manually assign the IP address *10.10.1.{group number}* to your notebook. Make sure the subnet mask is *255.255.255.0* and the broadcast address is *10.10.1.255* using the following command.

> *> ifconfig eth1 netmask 255.255.255.0 broadcast 10.10.1.255*

Verify your connectivity to the network by pinging the Intel gateway at *10.10.1.100*.

- ❑ Groups 1 and 2 will initiate an *iperf* session. Group 1 should run *iperf* as a server and Group 2 should run *iperf* as a client. Configure and run the *iperf* client and the *iperf* server to transmit and receive UDP datagrams for 10 minutes (600 seconds).


root@WNMS00:/usr/local/etc - Shell No. 2 - Konsole
Session Edit View Bookmarks Settings Help

```
Network List—(Autofit)                                              Info
      Name                       T W Ch  Packts Flags IP Range      Size   Ntwrks
    . VT_WLAN                    A N 003    299  T2   198.82.0.0      1k      10
    . VT_WLAN                    A N 001    144  T3   198.82.174.0  422B   Pckets
    . WIRELESS                   H N 011     46       0.0.0.0         0B     879
    . WNMS  Welcome to Kismet                                              Cryptd
    . stu  Kismet-Client 2004.10.R1 build 20041025233409                       0
    ! navc                                                                   Weak
    . VT_W Welcome to the Kismet panels frontend.                              0
 +  . Prob Context help is available for all displays, press 'H' at any time  Noise
    ! VT_W for more information.                                               0
    . mycr                                                                 Discrd
           This message can be turned off by editing the kismet_ui.conf file.  12

           Press <Space> to continue.

                                                                           Elapsd
                                                                         00:01:14
Status
 Found IP 198.82.174.1 for VT_WLAN::00:04:23:A7:C7:04 via ARP
 Found new network "mycroft" bssid 00:50:18:0A:2B:F0 WEP Y Ch 7 @ 11.00 mbit
 Found new probed network "navciiti" bssid 00:06:25:AA:C3:47
 Associated probe network "00:06:25:AA:C3:47" with "02:06:67:08:C3:47" via data.
Battery: AC charging 100% 3h5m0s
```

Figure 2. Kismet_client running for Part B experiment.

- ❑ All other groups should run *kismet_client* (see Figures 2 and 3) using the following command.

  > *> kismet_client*

- ❑ The following is a brief list of commands supported by kismet.

  “h” pops up the help menu

  “q” closes the current pop up menu

  “Q” quits kismet_client

  “s” pops up sort menu (choose anything other than Auto-fit)

  “w” pops up the alert menu (IDS)

  “L” locks the sniffer to the current network channel

  “H” hops the sniffer to different channels

- ❑ Change the sort value by using the following procedure.

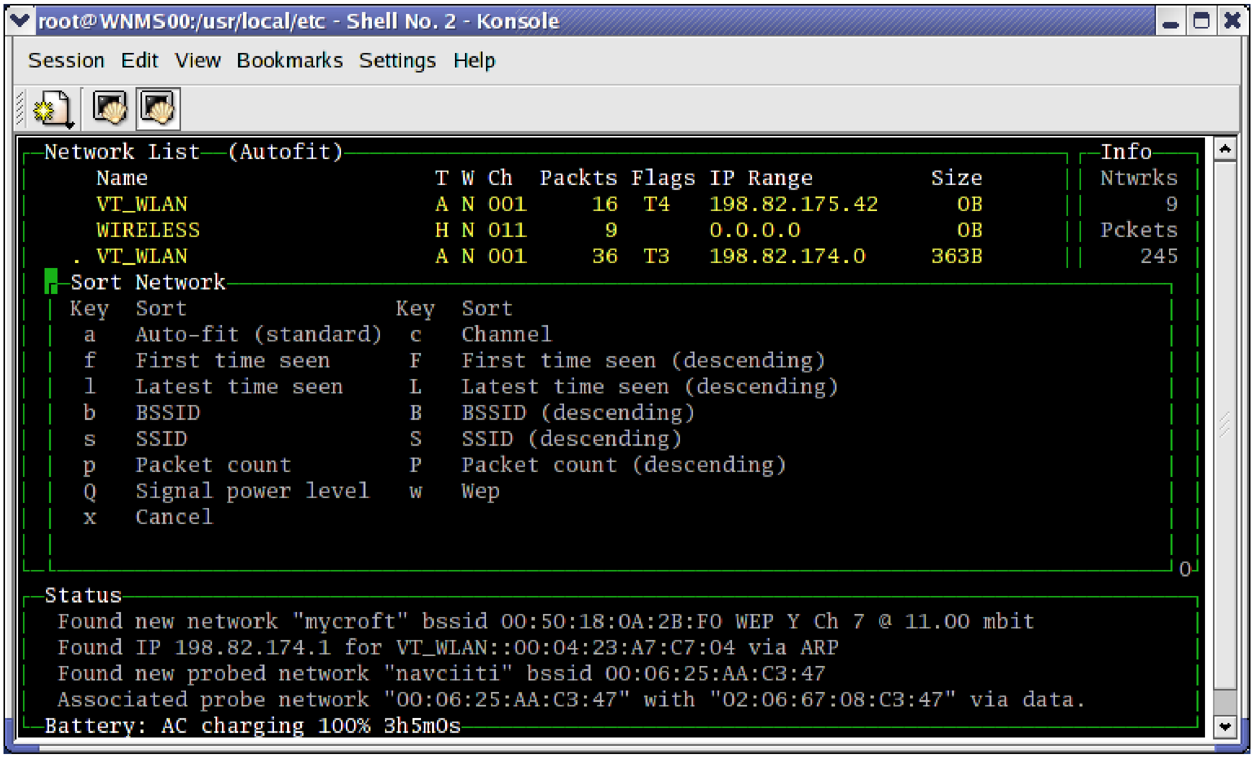
  a. Press the “s” key to pop up the sort menu

  b. Press the “s” key again to sort by SSID

  c. Press the “q” key to close the pop up menu

- ❑ Kismet will detect the network set up by the GTA and any other WLANs within range. Packets captured by kismet are saved periodically to the directory where kismet_client was executed. Examine these files (*.*dump*) by opening one or two using Ethereal.

- ❑ Note that the GTA’s notebook, which is running *kismet_server*, is able to capture traffic even though it is not connected to any WLAN.



```
┌Network List─(Autofit)───────────────────────────────────────┐┌Info─┐
│    Name                      T W Ch  Packts Flags IP Range        Size ││ Ntwrks│
│    VT_WLAN                   A N 001     16  T4   198.82.175.42     0B ││     9 │
│    WIRELESS                  H N 011      9       0.0.0.0           0B ││ Pckets│
│  . VT_WLAN                   A N 001     36  T3   198.82.174.0    363B ││   245 │
│ ┌Sort Network──────────────────────────────────────────────────────┐       │
│ │Key  Sort                  Key  Sort                               │       │
│ │ a   Auto-fit (standard)    c   Channel                            │       │
│ │ f   First time seen        F   First time seen (descending)       │       │
│ │ l   Latest time seen       L   Latest time seen (descending)      │       │
│ │ b   BSSID                  B   BSSID (descending)                 │       │
│ │ s   SSID                   S   SSID (descending)                  │       │
│ │ p   Packet count           P   Packet count (descending)          │       │
│ │ Q   Signal power level     w   Wep                                │       │
│ │ x   Cancel                                                        │       │
│ │                                                                   │       │
└─┴───────────────────────────────────────────────────────────────────┘0┘
┌Status─────────────────────────────────────────────────────────────────────┐
│ Found new network "mycroft" bssid 00:50:18:0A:2B:F0 WEP Y Ch 7 @ 11.00 mbit
│ Found IP 198.82.174.1 for VT_WLAN::00:04:23:A7:C7:04 via ARP
│ Found new probed network "navciiti" bssid 00:06:25:AA:C3:47
│ Associated probe network "00:06:25:AA:C3:47" with "02:06:67:08:C3:47" via data.
└Battery: AC charging 100% 3h5m0s──────────────────────────────────────────┘
```

Figure 3. Sorting the networks display on *kismet_client* for Part B experiment.

Part C: Impersonating an access point

The GTA will run *hostAP* on his or her notebook, which is a driver that enables the notebook to run as an access point. The GTA will impersonate the Intel WLAN gateway by assigning the notebook the same BSSID (MAC Address), ESSID, IP address, and the channel of operation as the gateway. The attack takes advantage of the vulnerability due to unauthenticated management frames in IEEE 802.11 by sending disassociation frames to nodes in the network. To these nodes, the GTA’s notebook acting as a rogue access point looks no different from the valid access point. Thus, nodes believe they were disassociated from the network and will then attempt to reassociate.

Use the following procedure for this experiment.

- ❑ Connect iPAQs to the infrastructure network that was setup in the previous experiment. You are to configure your iPAQ to associate with the Intel WLAN gateway (with ESSID *"WNMS"* at the NVC and *"WNMSA"* in Blacksburg). Manually assign an IP address 10.10.1.{*groupnumber + 20}* to the iPAQ. On the iPAQ, verify that the correct IP address is used by executing the vxUtil application. Using vxUtil, ping the Intel WLAN gateway, with IP address *10.10.1.100*, to make sure you are connected. Keep kismet_client running on your notebook.

- ❑ Observe the signal strength and the status of the iPAQ through the Xircom Link Status meter in Start>Programs>Xircom.

- ❑ The GTA will perform a DoS attack on all nodes by impersonating the access point with a notebook and sending disassociation frames. Observe the link status meter falling to zero and the status of a node changing to "unassociated."

- ❑ The IDS of kismet should raise an alert about "Broadcast Deauthentication" and "Broadcast Deauthentication/Disassociation flooding" originating from the rogue access point.

- ❑ Capture a screenshot of the Kismet client interface running on your notebook showing the alerts. Also, save the *.dump files. You will use the screenshot and the dump files for the associated at-home exercise.

Summary of Items Needed for this Week's Report

Part A: ARP cache poisoning

- □ Screenshot of routing table
- □ Screenshot of original ARP table
- □ Screenshot of updated ARP table
- □ Captured trace file from Ethereal

Part C: Impersonating an access point

- □ Screenshot of the Kismet client interface showing the alerts
- □ *.dump files from the Kismet client