Assignment no. 1

Q.1) Case study on security tools (Kismet and NetStumbler)

Kismet

i) Kismet is a monitoring tool for wireless - originally only supporting 802.11 Wi-Fi, with the right hardware Kismet can now capture Bluetooth advertisements, BTLE, nRF-based wireless mice and keyboards, weather stations, wireless thermometers, switches, smoke detectors, 802.15.4 / Zigbee, ADSB airplane transponders, AMR wireless power, water, and gas meters, and more.

ii) Kismet works with Wi-Fi interfaces, Bluetooth interfaces, some SDR (software defined radio) hardware like the RTLSDR, and other specialized capture hardware.

iii) Kismet works on Linux, OSX, and, to a degree, Windows 10 under the WSL framework. On Linux it works with most Wi-Fi cards, Bluetooth interfaces, and other hardware devices. On OSX it works with the built-in Wi-Fi interfaces, and on Windows 10 it will work with remote captures.

iv) **Passive monitoring:** Kismet is largely focused on collecting, collating, and sorting wireless data. The logs generated by Kismet can be fed into other tools (the pcap, handshakes, and other data) like hashcat, aircrack, and more.

v) **Device-oriented display:** a) Kismet is different from Wireshark; Kismet primarily focuses on representing devices - access points, clients, bridged wired devices, sensors, Bluetooth entities, and so on, while Wireshark focuses on displaying a deep dive of specific packets and all the content.

b) Kismet and Wireshark work best when used *together* - Kismet collects packets and logs them to standard formats (pcap and pcapng) or the kismetdb format which can be converted directly to pcap and pcapng, and collects location, changes over time, etc, while Wireshark can open the pcap logs and give extensive detailed information about specific packets. Each tool is designed for a different job, but operate together.

vi) **Features:** a) Kismet differs from other wireless network detectors in working passively. Namely, without sending any loggable packets, it is able to detect the presence of both wireless access points and wireless clients, and to associate them with each other. It is also the most widely used and up to date open source wireless monitoring tool.

b) Kismet also includes basic wireless IDS features such as detecting active wireless sniffing programs including NetStumbler, as well as a number of wireless network attacks.

c) Kismet features the ability to log all sniffed packets and save them in a tcpdump/Wireshark or Airsnort compatible file format. Kismet can also capture "Per-Packet Information" headers.

d) Kismet also features the ability to detect default or "not configured" networks, probe requests, and determine what level of wireless encryption is used on a given access point. In order to find as many networks as possible, Kismet supports channel hopping. This means that it constantly changes from channel to channel non-sequentially, in a user-defined sequence with a default value that leaves big holes between channels (for example, 1-6-11-2-7-12-3-8-13-4-9-14-5-10). The advantage with this method is that it will capture more packets because adjacent channels overlap.

e) Kismet also supports logging of the geographical coordinates of the network if the input from a GPS receiver is additionally available.

vii) **Disadvantages of Kismet:** a) It takes long time to search networks.

b) It can only identify the wireless network (Wi-Fi) in a small area, if the range is more it cannot work properly.

NetStumbler

i) NetStumbler detects wireless local area networks (WLANs) that are based on the 802.11b and 802.11g data formats in the Industrial Scientific and Medical (ISM) radio band and Unlicensed National Information Infrastructure (U-NII), a band using 802.11a data formats.

ii) It provides radio frequency (RF) signal information and other data related to the peculiarities of combining computers and radios. MiniStumbler is the PocketPC version of NetStumbler. Both NetStumbler and MiniStumbler are active wireless network detection applications.

iii) **Uses:** a) Verify that your network is set up the way you intended.

b) Find locations with poor coverage in your WLAN.

c) Detect other networks that may be causing interference on your network.

d) Detect unauthorized "rogue" access points in your workplace.

e) Help aim directional antennas for long-haul WLAN links.

f) Use it recreationally for WarDriving.

iv) NetStumbler is an “active” wireless network detection application. This means the program takes a specific action to accomplish the WLAN detection. The action is to send out a specific data probe called a Probe Request. The Probe Request frame and the associated Probe Response frame are part of the 802.11 standard.

v) **Wireless Ethernet Cards that Work with NetStumbler and MiniStumbler:** To use NetStumbler or MiniStumbler, you need a wireless Ethernet card. There are a wide variety of makes and models available, and every day new models are released, so the question becomes:

Which ones work with NetStumbler? Generally, the best cards are those that use the Hermes chipset. Primarily, this refers to the ORiNOCO Gold or Silver "Classic" cards or "re-badged" versions of those cards. The big disadvantage to these cards, however, is that they only work with 802.11b data. "Re-badges", are made by manufacturers such as ORiNOCO, but sold under another brand name, such as Dell. The marking decals or "badge" is changed to reflect the new brand, hence the term "re-badge."

vi) **Damage & Defense… Disabling the Beacon:** a) NetStumbler transmits a "Broadcast Request" probe to discover the WLAN. Most access points will respond to a Broadcast Request by default. When it responds, the AP (access point) transmits its SSID, MAC number, and other information. However, many brands and models of AP allow this feature to be disabled. Once an AP ceases to respond to the request, NetStumbler can no longer detect it. If you don't want your wireless LAN to show up on the screen of another NetStumbler user, disable the SSID broadcast on your access point. Check your AP manual for "Disable SSID Broadcast", "Closed SSID," or similar features.

b) The one caveat to this is if the SSID that the WarDriver enters for NetStumbler happens to have the same SSID as your network, then your AP will still respond to the probe. This is another good reason to change the default SSID.

Q2. Explain and implement Hidden Terminal/Exposed terminal problem.

Hidden and exposed terminals

Consider the scenario with three mobile phones as shown in Figure 1. The transmission range of A reaches B, but not C (the detection range does not reach C either). The transmission range of C reaches B, but not A. Finally, the transmission range of B reaches A and C, i.e., A cannot detect C and vice versa. A starts sending to B, C does not receive this transmission. C also wants to send something to B and senses the medium. The medium appears to be free, the carrier sense fails. C also starts sending causing a collision at B. But A cannot detect this collision at B and continues with its transmission. A is hidden for C and vice versa.

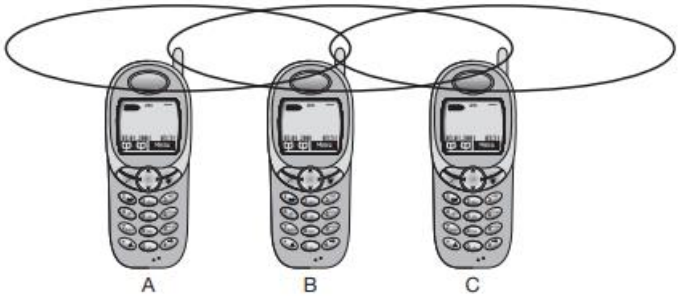

Figure 1

While hidden terminals may cause collisions, the next effect only causes unnecessary delay. Now consider the situation that B sends something to A and C wants to transmit data to some other mobile phone outside the interference ranges of A and B. C senses the carrier and detects that the carrier is busy (B's signal). C postpones its transmission until it detects the medium as being idle again. But as A is outside the interference range of C, waiting is not necessary. Causing a 'collision' at B does not matter because the collision is too weak to propagate to A. In this situation, C is exposed to B.

Multiple access with collision avoidance

Solution to hidden terminal

Multiple access with collision avoidance (MACA) presents a simple scheme that solves the hidden terminal problem, does not need a base station, and is still a random access Aloha scheme – but with dynamic reservation. Figure 2 shows the same scenario as Figure 1 with the hidden terminals. Remember, A and C both want to send to B. A has already started the transmission, but is hidden for C, C also starts with its transmission, thereby causing a collision at B.

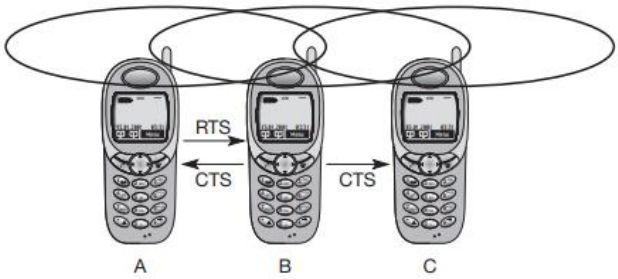

Figure 2

With MACA, A does not start its transmission at once, but sends a request to send (RTS) first. B receives the RTS that contains the name of sender and receiver, as well as the length of the future transmission. This RTS is not heard by C, but triggers an acknowledgement from B, called clear to send (CTS). The CTS again contains the names of sender (A) and receiver (B) of the user data, and the length of the future transmission. This CTS is now heard by C and the medium for future use by A is now reserved for the duration of the transmission. After receiving a CTS, C is not allowed to send anything for the duration indicated in the CTS toward B. A collision cannot occur at B during data transmission, and the hidden terminal problem is solved – provided that the transmission conditions remain the same. (Another station could move into the transmission range of B after the transmission of CTS.) Still, collisions can occur during

the sending of an RTS. Both A and C could send an RTS that collides at B. RTS is very small compared to the data transmission, so the probability of a collision is much lower. B resolves this contention and acknowledges only one station in the CTS (if it was able to recover the RTS at all). No transmission is allowed without an appropriate CTS. This is one of the medium access schemes that is optionally used in the standard IEEE 802.11.

Solution to exposed terminal

Can MACA also help to solve the 'exposed terminal' problem? Remember, B wants to send data to A, C to someone else. But C is polite enough to sense the medium before transmitting, sensing a busy medium caused by the transmission from B. C defers, although C could never cause a collision at A. With MACA, B has to transmit an RTS first (as shown in Figure 3) containing the name of the receiver (A) and the sender (B). C does not react to this message as it is not the receiver, but A acknowledges using a CTS which identifies B as the sender and A as the receiver of the following data transmission. C does not receive this CTS and concludes that A is outside the detection range. C can start its transmission assuming it will not cause a collision at A. The problem with exposed terminals is solved without fixed access patterns or a base station. One problem of MACA is clearly the overheads associated with the RTS and CTS transmissions – for short and time-critical data packets, this is not negligible. MACA also assumes symmetrical transmission and reception conditions. Otherwise, a strong sender, directed antennas etc. could counteract the above scheme

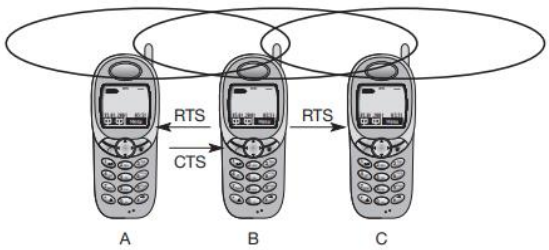

Figure 3

Figure 4 shows simplified state machines for a sender and receiver. The sender is idle until a user requests the transmission of a data packet. The sender then issues an RTS and waits for the right to send. If the receiver gets an RTS and is in an idle state, it sends back a CTS and waits for data. The sender receives the CTS and sends the data. Otherwise, the sender would send an RTS again after a time-out (e.g., the RTS could be lost or collided). After transmission of the data, the sender waits for a positive acknowledgement to return into an idle state. The receiver sends back a positive acknowledgement if the received data was correct. If not, or if the waiting time for data is too long, the receiver returns into idle state. If the sender does not receive any acknowledgement or a negative acknowledgement, it sends an RTS and again waits for the right to send. Alternatively, a receiver could indicate that it is currently busy via a separate RxBusy. Real implementations have to add more states and transitions, e.g., to limit the number of retries.

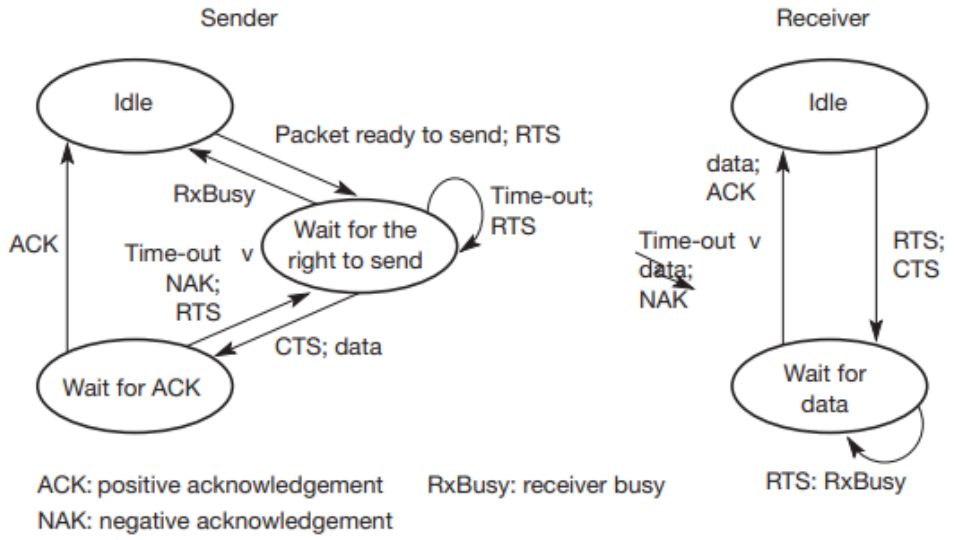

Figure 4