

MP3: Checkpoint 2



CS 461/ECE 422

Goals

3.2.1: Network Attacks

- Crack the wireless network password
- Analyze the network
- Obtain server and client's information

3.2.2: Anomaly (port scanning) detection

- Write a Python program that detects port scanning in a given pcap

Required Tools

- Wireshark (**32 bit**)
- Aircrack-ng Suite
 - Mac users can install aircrack-ng through MacPorts or Homebrew (check Piazza)
- Nmap
- Python 2.7
- dpkt Python library

Objective

Wireless Network Attack

- WEP
- General approach to WEP key cracking
- Wireless network terms
- Aircrack-ng suite
- Network analysis

Anomaly Detection

Wireless Network Attacks



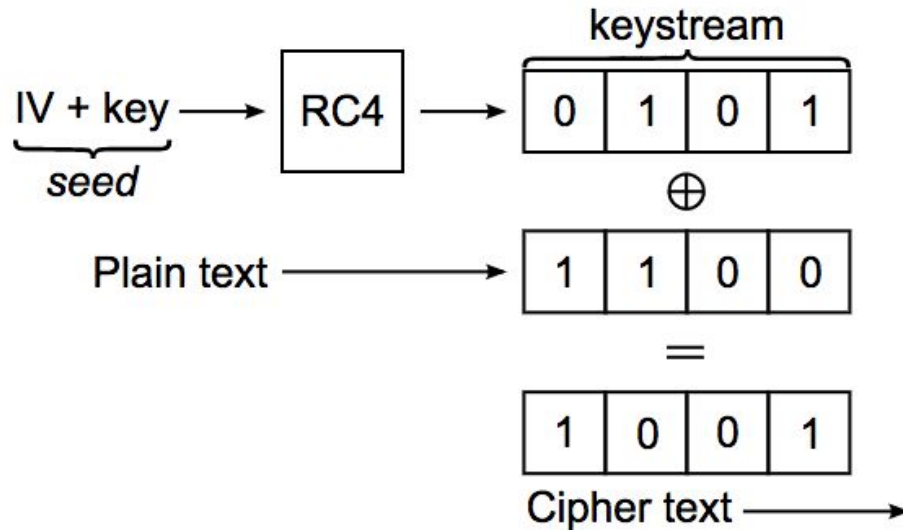
Wired Equivalent Privacy (WEP)

A security protocol designed for IEEE 802.11 wireless network

What is 802.11? Essentially the standard for how WiFi devices interact with each other

WEP Key

- Used to authenticate with access point and encrypt the traffic
- Encryption relies on RC4 stream cipher and 24-bit IVs
- IVs are sent in plain-text



WEP Security Issues

Purpose of an IV, which is transmitted as plain text, is to prevent any repetition

24-bit IV is not long enough to ensure this on a busy network

For a 24-bit IV, there is a 50% probability the same IV will repeat after 5000 packets

General Approach to WEP Key Cracking

Goal: collect a lot of IVs to derive the WEP key

1. Capture wireless frames from the target access point
2. **(DO NOT DO THIS FOR THE MP)** Inject or replay frames to collect the IVs faster
3. Derive the key
 - a. Using all the captured packets
 - b. Statistical analysis

Wireless Network Terms

BSSID – the MAC address of the wireless access point

ESSID – the wireless network name

Channel # – the number of the channel that a wireless network is configured to use for communication

IVs (data/packet) – the number of initialization vectors gathered

Managed mode – normal client, station mode

Monitor mode – allows capturing frames without connecting to the access point

Aircrack-ng Suite

Airmon-ng – Enable and disable monitor mode on wireless interfaces

Aireplay-ng – Inject and replay wireless frames

Airodump-ng – Capture raw 802.11 frames

Aircrack-ng – 802.11 WEP and WPA/WPA2-PSK key cracking program

More details: <http://www.aircrack-ng.org/documentation.html>

WEP key cracking tutorial: http://www.aircrack-ng.org/doku.php?id=simple_wep_crack

https://www.aircrack-ng.org/doku.php?id=aircrack-ng#general_approach_to_cracking_wep_keys

Aircrack-ng Suite

DO NOT USE



Airmon-ng – Enable and disable monitor mode on wireless interfaces

~~Aireplay-ng – Inject and replay wireless frames~~

Airodump-ng – Capture raw 802.11 frames

Aircrack-ng – 802.11 WEP and WPA/WPA2-PSK key cracking program

More details: <http://www.aircrack-ng.org/documentation.html>

WEP key cracking tutorial: http://www.aircrack-ng.org/doku.php?id=simple_wep_crack

https://www.aircrack-ng.org/doku.php?id=aircrack-ng#general_approach_to_cracking_wep_keys

Skip any step that use aireplay-ng (i.e. step 2, 4, 5)

Airmon-ng

```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# airmon-ng start wlan0  
No interfering processes found  
PHY      Interface      Driver      Chipset  
phy0      wlan0              ath9k_htc    Atheros Communications, Inc. AR9271 802.11n  
              (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)  
              (mac80211 station mode vif disabled for [phy0]wlan0)  
  
root@kali:~# iwconfig  
eth0      no wireless extensions.  
  
wlan0mon  IEEE 802.11bgn  Mode:Monitor  Frequency:2.457 GHz  Tx-Power=20 dBm  
          Retry short limit:7   RTS thr:off   Fragment thr:off  
          Power Management:off  
  
lo        no wireless extensions.  
root@kali:~#
```

Airodump-ng

CH 11][Elapsed: 24 s][2014-11-21 09:39

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
00:25:9C:97:4F:48	-33	20	14 0	9	54e.	WPA2	CCMP	PSK	Mande
78:CD:8E:3B:B7:0B	-47	14	0 0	1	54e	WPA2	CCMP	PSK	<leng
78:CD:8E:3B:B7:09	-42	13	0 0	1	54e	WPA2	CCMP	PSK	<leng
78:CD:8E:3B:B7:0A	-42	12	0 0	1	54e	WPA2	CCMP	PSK	<leng
78:CD:8E:3B:B7:08	-44	14	2 0	1	54e	WPA2	CCMP	PSK	TheDr
B0:C7:45:75:13:9E	-58	4	0 0	9	54e.	WPA2	CCMP	PSK	tedpe

BSSID	STATION	PWR	Rate	Lost	Frames	Probe
(not associated)	00:C0:CA:3F:EE:02	0	0 - 1	0	11	
00:25:9C:97:4F:48	00:1E:8F:8D:18:25	-17	0 - 36	0	10	
B0:C7:45:75:13:9E	10:A5:D0:F5:31:19	-51	0 - 6	0	1	

Aircrack-ng

```
root@kali: ~  
File Edit View Search Terminal Help  
Aircrack-ng 1.2 rcl  
[00:04:26] Tested 802 keys (got 32515 IVs)  
KB    depth  byte(vote)  
0     3/ 5    22(38144) FA(37888) 57(37632) 1F(37376) D2(37376)  
1     8/ 1    48(38400) 02(38144) 3C(38144) C8(37888) AC(37632)  
2     0/ 2    4C(45568) 85(48704) 0F(39424) 19(39168) 31(38912)  
3    31/ 3    CD(36896) 88(35840) D5(35840) DA(35840) 21(35584)  
4     0/ 4    38(45056) 36(40192) 6F(40192) 57(39680) 4C(38912)  
  
KEY FOUND! [ 74:65:73:74:70:61:73:73:64:61:79:6F:6E ] (ASCII: testpassdayon  
)  
Decrypted correctly: 100%  
root@kali:~#
```

Airport (Macs)

Add to path [@640](#)

sudo airport -s – captures packets and provides information such as ESSID, BSSID, Channel #, Security type, etc.

- Equivalent to airodump-ng

sudo airport [interface] sniff [channel #] – sniff this channel in monitor mode

- Stores a resulting .cap file in /var/tmp/

Tips

Capturing and cracking process can be run simultaneously. Aircrack-ng will automatically reload new captured data.

Keep the captured packets.

Capturing and cracking the key shouldn't take longer than 40 minutes.

Network Analysis

How many hosts are there?

Which one is the server?

What services are present on the network?

Use Wireshark to identify hosts and analyze live traffic

Use nmap (network mapper) to obtain more details about hosts


Your own interpretation of network behavior

Don't rely on one result. See if other findings agree with what you observe.

Nmap

```
Starting Nmap 5.00 ( http://nmap.org ) at 2012-11-19 16:44 IST
Interesting ports on 192.168.1.1:
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address: BC:AE:C5:C3:16:93 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.45 seconds
```



Open/closed/filtered ports – <https://nmap.org/book/man-port-scanning-basics.html>

Port scanning techniques – <https://nmap.org/book/man-port-scanning-techniques.html>

Don't forget about UDP

No services running above port 4096

Use it with caution – practice on scanme.nmap.org

Capturing Network Traffic

How would you do it?

- Whose conversation are you capturing? Are you involved?

Promiscuous mode?

- Promiscuous mode – capturing after connecting to the access point (as part of the network); processes layer 2 (e.g. Ethernet, wireless)
- Works on both wired and wireless
- Works on hubs but not on switches. Why?

Wireless adapters should be able to receive all packets; shouldn't promiscuous mode work?

- Yes, for unprotected network
- Why not for protected network?

More details on Promiscuous vs. Monitor

https://wiki.wireshark.org/CaptureSetup/WLAN#Promiscuous_mode

<http://networkengineering.stackexchange.com/questions/3774/using-wireless-cards-in-promiscuous-mode>

<http://lazysolutions.blogspot.ca/2008/10/difference-promiscuous-vs-monitor-mode.html>

Information Gathering

Go after the low-hanging fruit

Question your findings

What would you do if you find

- Web server?
- SSH server?

Anomaly Detection



Anomaly Detection

Assumption: If $\#SYN > 3 * (\# SYN-ACK)$, then consider activity as attack (SYN scanning)

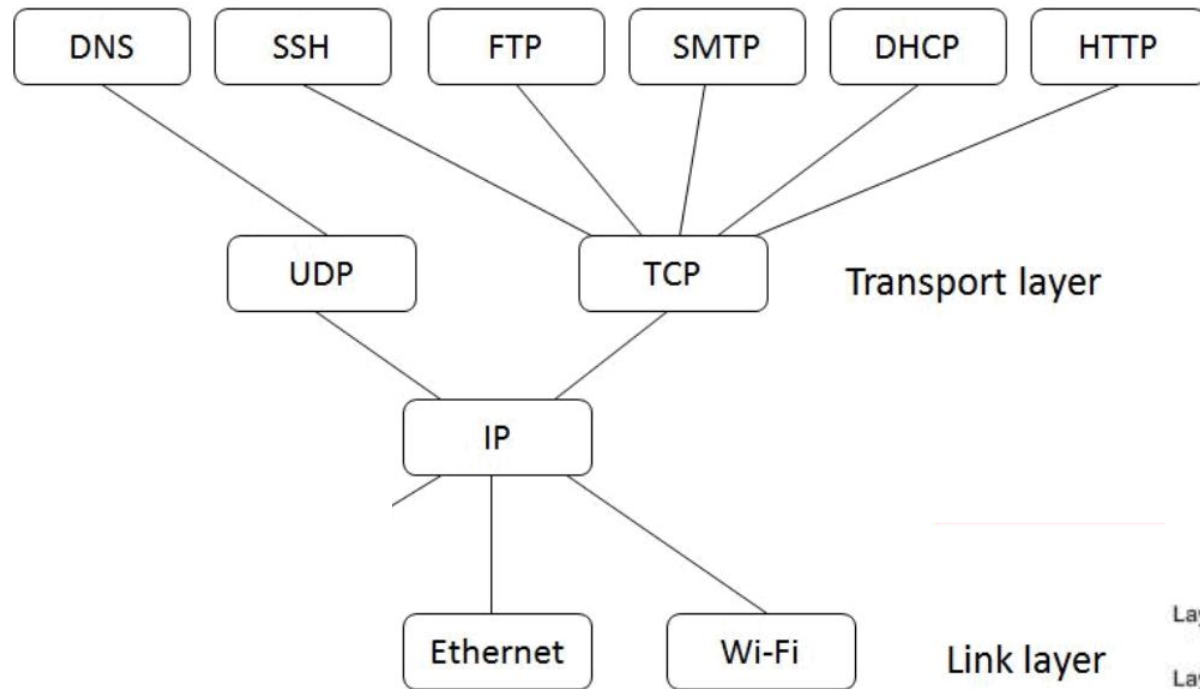
- SYN packet – only SYN flag set
- SYN-ACK packet – only SYN and ACK flags set

You must use dpkt; we will not grade code using scrapy

Know

- Which protocols are involved
- Which protocol encapsulates which
- What fields exist within protocol headers

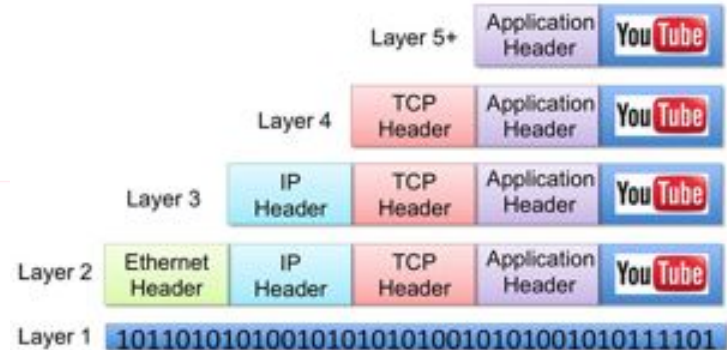
Layering of Protocols



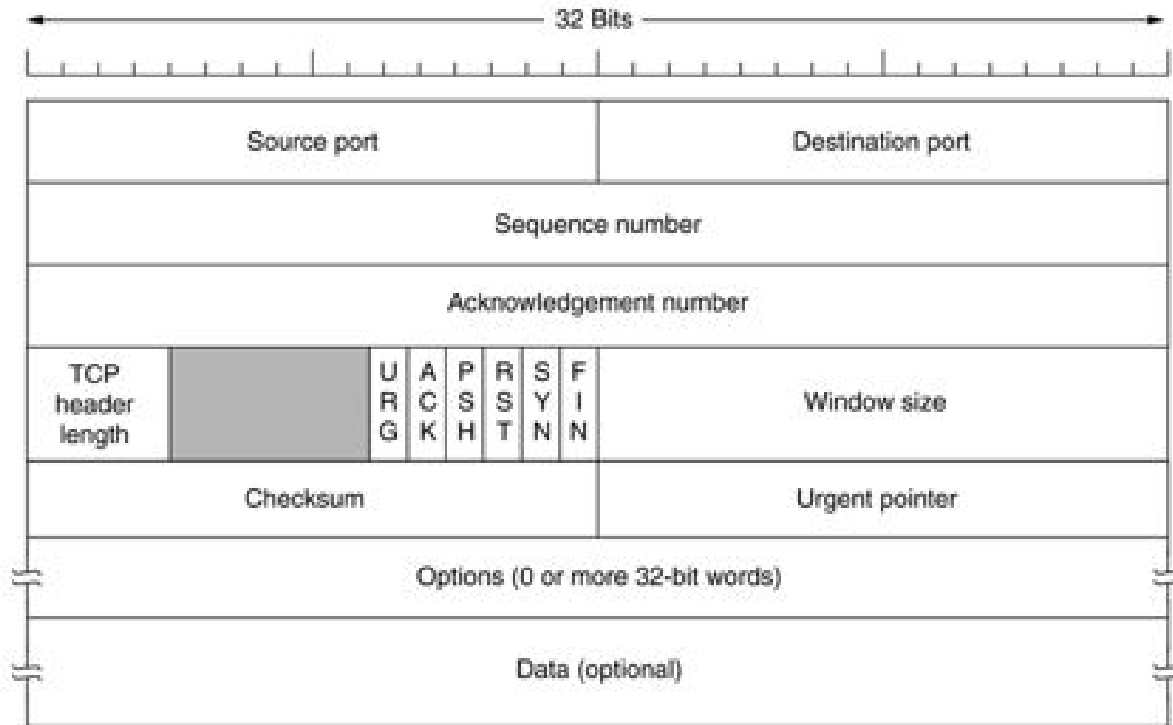
Application layer

Transport layer

Link layer



TCP Header



dpkt Example

```
#!/usr/bin/python2.7
import dpkt

f = open('test.pcap')
pcap = dpkt.pcap.Reader(f)
for ts, buf in pcap:
    eth = dpkt.ethernet.Ethernet(buf)
    ip = eth.data
    tcp = ip.data

    if tcp.dport == 80 and len(tcp.data) > 0:
        http = dpkt.http.Request(tcp.data)
        print http.uri
f.close()
```

<https://jon.oberheide.org/blog/2008/10/15/dpkt-tutorial-2-parsing-a-pcap-file/>

Sample PCAPs

Sample pcap at https://subversion.engr.illinois.edu/svn/fa17-cs461/_shared/mp3/

- Will upload more soon

Tips

Use try-except to handle/ignore malformed packets

Don't just dissect every packet

- <http://stackoverflow.com/questions/8849635/python-dpkt-find-out-if-packet-is-a-tcp-packet-or-a-udp-packet>

dpkt cheatsheet – <http://engineering-notebook.readthedocs.org/en/latest/engineering/dpkt.html>

Questions?

