מבוא לעולם הסייבר ב'

מטלה מס' 2 – פייטון

מרצה: דביר עמית, בועז בן משה.

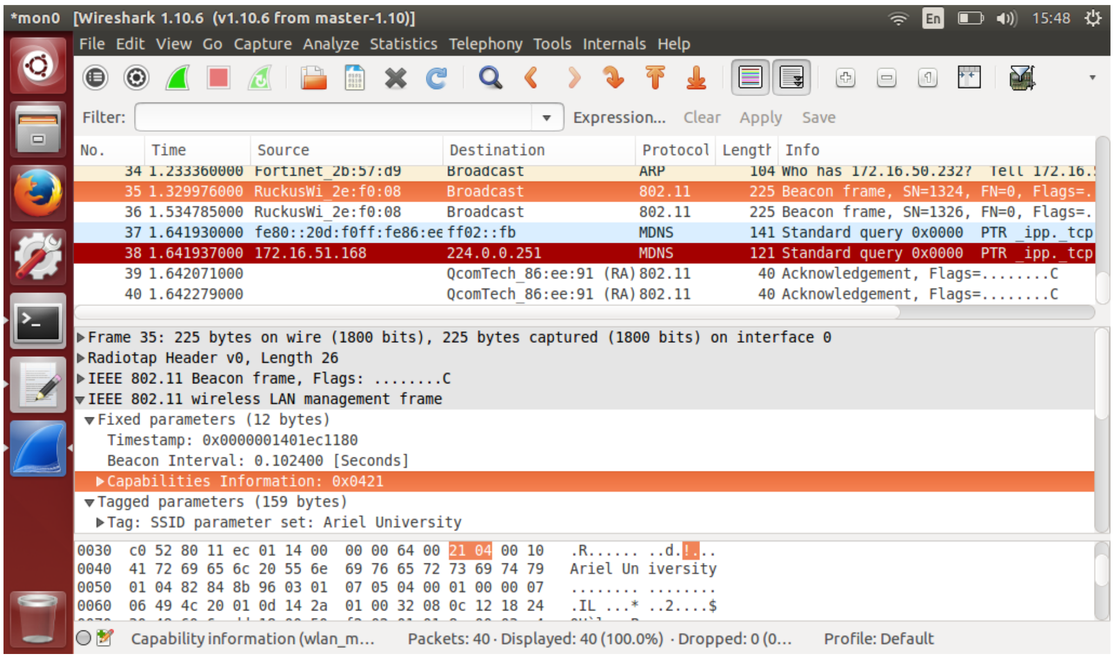
מגישים: ניתאי חסון 305691347,
שמוליק גיחון 305205932,
עוז מעתוק 305181158.

מצורפים: קטעי הקוד לכלל השאלות בקבצי py.

שאלה 1

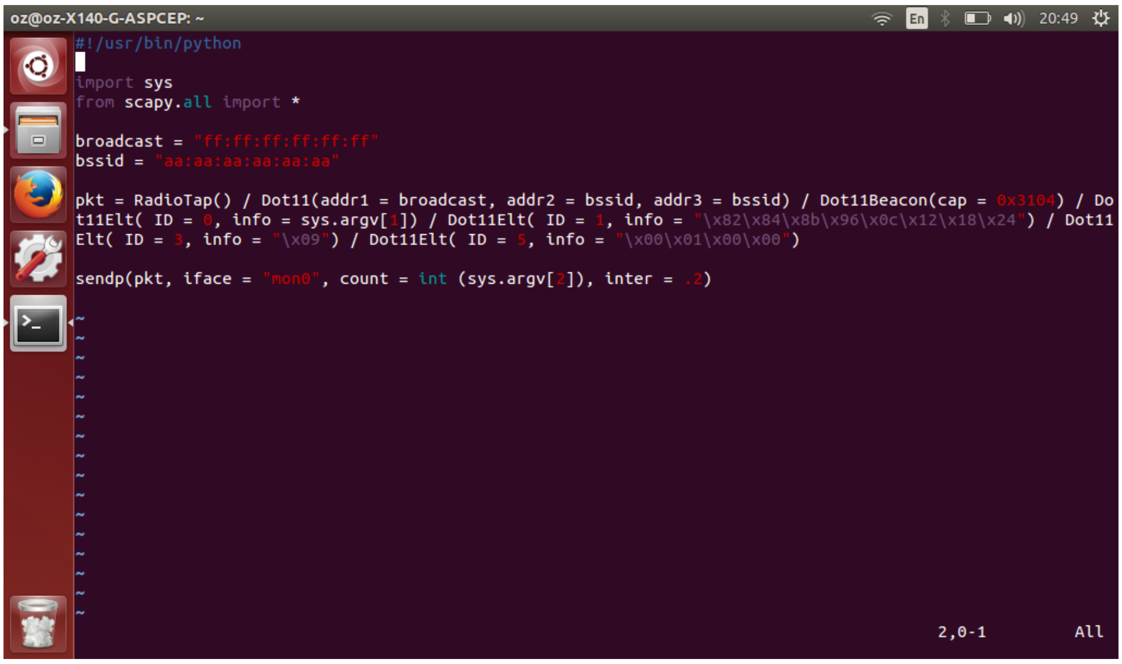
משלוח beacons ע"פ Vivek Ramachandran

מציאת beacon מתוך רשימת התעבורה ב wireshark ואימות פרטים נדרשים מול ה beacon שניצור:



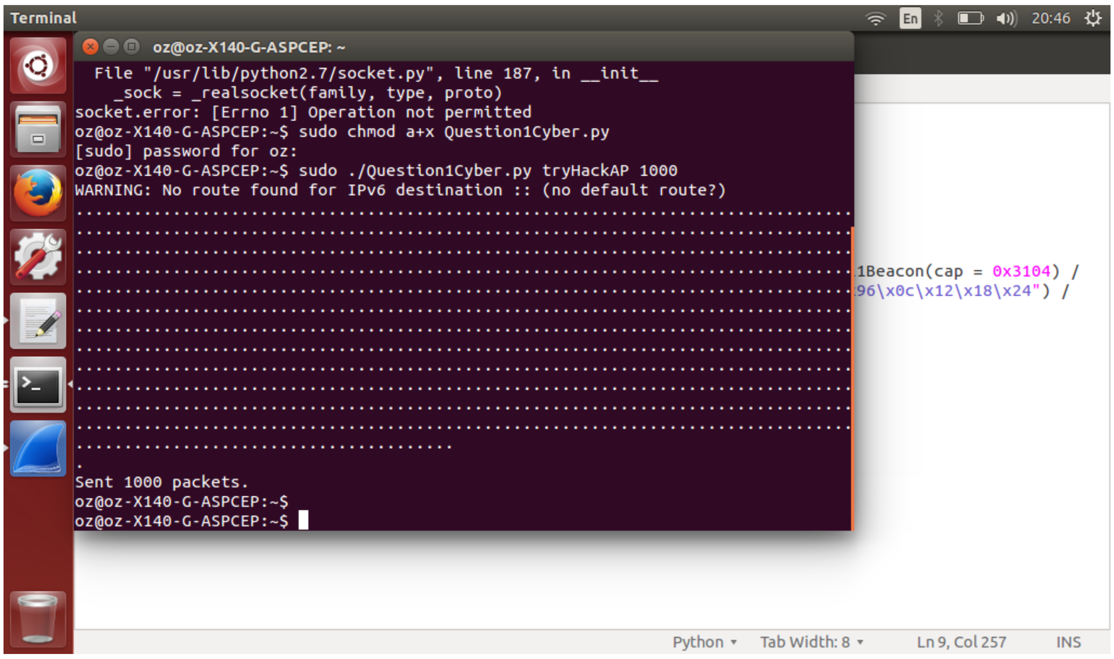
```
*mon0  [Wireshark 1.10.6 (v1.10.6 from master-1.10)]                                   15:48
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:                                        Expression...  Clear  Apply  Save

No.     Time         Source                  Destination          Protocol Length Info
     34 1.233360000  Fortinet_2b:57:d9       Broadcast            ARP       104 Who has 172.16.50.232?  Tell 172.16.
     35 1.329976000  RuckusWi_2e:f0:08       Broadcast            802.11    225 Beacon frame, SN=1324, FN=0, Flags=.
     36 1.534785000  RuckusWi_2e:f0:08       Broadcast            802.11    225 Beacon frame, SN=1326, FN=0, Flags=.
     37 1.641930000  fe80::20d:f0ff:fe86:ee ff02::fb              MDNS      141 Standard query 0x0000  PTR _ipp._tcp
     38 1.641937000  172.16.51.168           224.0.0.251          MDNS      121 Standard query 0x0000  PTR _ipp._tcp
     39 1.642071000                          QcomTech_86:ee:91 (RA) 802.11   40 Acknowledgement, Flags=........C
     40 1.642279000                          QcomTech_86:ee:91 (RA) 802.11   40 Acknowledgement, Flags=........C

▶Frame 35: 225 bytes on wire (1800 bits), 225 bytes captured (1800 bits) on interface 0
▶Radiotap Header v0, Length 26
▶IEEE 802.11 Beacon frame, Flags: ........C
▼IEEE 802.11 wireless LAN management frame
 ▼Fixed parameters (12 bytes)
    Timestamp: 0x0000001401ec1180
    Beacon Interval: 0.102400 [Seconds]
  ▶Capabilities Information: 0x0421
 ▼Tagged parameters (159 bytes)
  ▶Tag: SSID parameter set: Ariel University

0030  c0 52 80 11 ec 01 14 00  00 00 64 00 21 04 00 10   .R...... ..d.!...
0040  41 72 69 65 6c 20 55 6e  69 76 65 72 73 69 74 79   Ariel Un iversity
0050  01 04 82 84 8b 96 03 01  07 05 04 00 01 00 00 07   ........ ........
0060  06 49 4c 20 01 0d 14 2a  01 00 32 08 0c 12 18 24   .IL ...* ..2....$

Capability information (wlan_m...   Packets: 40 · Displayed: 40 (100.0%) · Dropped: 0 (0...   Profile: Default
```

מימוש הקוד:



```
oz@oz-X140-G-ASPCEP: ~                                                                  20:49
#!/usr/bin/python

import sys
from scapy.all import *

broadcast = "ff:ff:ff:ff:ff:ff"
bssid = "aa:aa:aa:aa:aa:aa"

pkt = RadioTap() / Dot11(addr1 = broadcast, addr2 = bssid, addr3 = bssid) / Dot11Beacon(cap = 0x3104) / Do
t11Elt( ID = 0, info = sys.argv[1]) / Dot11Elt( ID = 1, info = "\x82\x84\x8b\x96\x0c\x12\x18\x24") / Dot11
Elt( ID = 3, info = "\x09") / Dot11Elt( ID = 5, info = "\x00\x01\x00\x00")

sendp(pkt, iface = "mon0", count = int (sys.argv[2]), inter = .2)

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
                                                                              2,0-1         All
```

שליחת ה beacons, והודעת הסיום:



```
  File "/usr/lib/python2.7/socket.py", line 187, in __init__
    _sock = _realsocket(family, type, proto)
socket.error: [Errno 1] Operation not permitted
oz@oz-X140-G-ASPCEP:~$ sudo chmod a+x Question1Cyber.py
[sudo] password for oz:
oz@oz-X140-G-ASPCEP:~$ sudo ./Question1Cyber.py tryHackAP 1000
WARNING: No route found for IPv6 destination :: (no default route?)
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
.......................................
.
Sent 1000 packets.
oz@oz-X140-G-ASPCEP:~$
oz@oz-X140-G-ASPCEP:~$
```

```
1Beacon(cap = 0x3104) /
96\x0c\x12\x18\x24") /
```

הקוד שלנו:

```
#!/usr/bin/python
# Question 1
# Receiving 3 arguments:
# 1) The SSID
# 2) How many time to send the package
# 3) interface

import sys
import random
from scapy.all import *

def randomMAC():
	mac = [ random.randint(0x00, 0x7f),
		random.randint(0x00, 0x7f),
		random.randint(0x00, 0x7f),
		random.randint(0x00, 0x7f),
		random.randint(0x00, 0xff),
		random.randint(0x00, 0xff) ]
	return ':'.join(map(lambda x: "%02x" % x, mac))

broadcast = "ff:ff:ff:ff:ff:ff"
bssid = randomMAC()
print bssid
print len(sys.argv[1])

pkt = RadioTap() / Dot11(addr1 = broadcast, addr2 = bssid, addr3 = bssid) /
Dot11Beacon(cap = 0x3114) / Dot11Elt( ID = 0, info = sys.argv[1]) / Dot11Elt( ID = 1,
info = "\x82\x84\x8b\x96\x0c\x12\x18\x24") / Dot11Elt( ID = 3, info = "\x01") /
Dot11Elt( ID = 5, info = "\x01\x03\x00\x08")

sendp(pkt, iface = sys.argv[3], count = int(sys.argv[2]), inter = .2)
```

שאלה 2

סיכום השיטות לביצוע beacon stuffing:

The beacons stuffing is based on the “push model” of information delivery. The key idea is to overload IEEE 802.11 beacons to carry additional information. Beacon frames are a local (100-200 meters if it is not changed by transmitting power or encoding scheme) broadcast that are used to announce the presence of a Wi-Fi network.

We treat the information to be broadcast as a string of bytes. In most cases, we expect the information to be a short text message. However, with this techniques we can also send files.

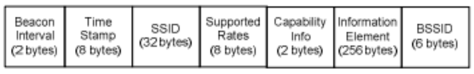
The AP splits the message into smaller fragments, and transmits each fragment in a separate beacon. The size of the fragment depends on the mechanism being used. The fragment sent in each beacon has the following format:

UniqueID : SequenceNumber : MoreFlag : InfoChunk

UniqueID identifies the message being broadcast, the SequenceNumber is the fragment number, and MoreFlag informs the client if it should expect more fragments. Finally, the InfoChunk has the contents of the message. Clients reassemble the message after receiving all fragments.

We now discuss details of three specific techniques to carry the messages in a beacon – SSID, BSSID and Beacon Information Element.

The main difference between the techniques is the field in the beacon packet that is used to carry the messages. None of the three techniques require any modifications to the hardware or firmware of the client device to receive the messages. For the SSID and BSSID based techniques, a simple user-level application is sufficient to reassemble the fragmented messages. The third technique, which uses Information Element, requires changes to the Wi-Fi driver on the client devices.



Beacon Interval (2 bytes)	Time Stamp (8 bytes)	SSID (32 bytes)	Supported Rates (8 bytes)	Capability Info (2 bytes)	Information Element (256 bytes)	BSSID (6 bytes)

Figure 1: *Some fields in the IEEE 802.11 beacon packet.*

SSID Concatenation: The SSID field in the Beacon carries the name of the wireless network. The maximum length is 32 bytes. Assuming the UniqueID is 1 byte and SequenceNumber and MoreFlag can fit in 1 byte, we are left with 29 bytes for the InfoChunk. Fragments are transmitted in successive beacons. The maximum length of each unique message is 3712 bytes.

This approach is easy to implement with the user interface and to set the SSID. A simple user-level program is sufficient reassemble the fragments and display the reassembled message.

This approach enables to send messages at the rate of 23Kbps with a 10ms AP Beacon Interval. However, this approach has a significant limitation. Most client devices include an application that displays the SSIDs of networks within the range of the device. Unless this application is modified, the user interface of this application will get swamped with the large number of SSIDs, which might obscure legitimate SSIDs.

BSSID Concatenation: BSSIDs are 6 byte unique identifiers of an AP. Generally, they are set to be equal to the MAC address of the AP. However, these can generally be set to any value. Assuming once again that the UniqueID is 1 byte and SequenceNumber and MoreFlag can fit in 1 byte, we can transmit 4 bytes of the message in a beacon. This gives us a maximum length of 512 bytes for each unique message.

All beacons have their SSID set to a fixed, well-known SSID. When a client receives a beacon with this SSID, it queries the BSSID field from the user-level. It assembles the message after receiving all fragments.

This approach overcomes the primary limitation of the SSID concatenation. The user is only presented with a list of unique SSIDs, and when the user selects a particular SSID, the driver determines which BSSID to connect to. The only limitation of this approach is the bandwidth. Assuming that beacons are sent every 10ms, the transmission rate is 3.2Kbps.

Beacon Information Element: The IEEE 802.11 standard allows AP vendors to add 253 bytes of vendor-specific Beacon Information Elements in its beacon. We use this feature to define a special BIE for broadcasting information. Each message is fragmented into 251 byte chunks and sent in successive beacons. All beacons have their SSID set to a fixed, well-known SSID. The Wi-Fi driver at client devices are modified to recognize this BIE and pass it to the user level. This approach provides a bandwidth of approximately 200Kbps, which is higher than the other two approaches. It also does not spam the wireless UI of the client device. The main drawback of this approach is that it requires driver modification for client devices, making it relatively difficult to deploy.

For each of the above encoding schemes, messages that span multiple beacons run the risk that if any fragment is lost then the entire message is lost. One can combat this problem by using forward error correction when encoding the message.

שאלה 3

תוכנית python אשר קולטת חבילות wifi beacon בתדר מסויים וכותבת אותם לקובץ טקסט:

```
#!/usr/bin/env python
# Question 3
# Receiving 1 argument:
# 1) interface

from scapy.all import *

ap_list = []
iface = sys.argv[1]

def PacketHandler(pkt) :
    if pkt.haslayer(Dot11) :
        if pkt.type == 0 and pkt.subtype == 8 :
            if pkt.addr2 not in ap_list :
                ap_list.append(pkt.addr2)
                print "AP MAC: %s with SSID: %s " %(pkt.addr2,
pkt.info)
                p = pkt
                ssid,channel,crypto = None,None,None
                while Dot11Elt in p :
                    p = p[Dot11Elt]
                    if(p.ID == 0):
                        ssid = p.info
                    elif(p.ID == 1):
                        print p.info
                    elif(p.ID == 3):
                        channel = ord(p.info)
                    elif(p.ID == 48):
                        crypto = "WPA2"
                    elif(p.ID == 221 and crypto == None):
                        crypto = "WPA"
                    p=p.payload
                if not crypto :
                    if "privacy" in  str(pkt.sprintf):
                        crypto = "WEP"
                    else:
                        crypto = "OPN"
                print "SSID = %s , Channel = %s , Crypt = %s "
%(ssid,channel,crypto)
```

```
                                        text_file = open("file_name.txt","a")
                                        text_file.write("Beacon Frame(BSSID: %s, SSID: %s,
Channel: %s, Encryption: %s )" %(pkt.addr2, ssid,channel,crypto))
                                        text_file.write("\n")
                                        text_file.close()

sniff(iface=iface, prn = PacketHandler)
```

שאלה 4

תוכנית python אשר קולטת חבילות wifi beacon, ושולחת בחזרה את החבילות המתאימות לפילטר שהוגדר:

```
#!/usr/bin/env python
# Question 4
# Receiving 3 arguments:
# 1) The filter for the SSID
# 2) How many time to send the package
# 3) interface

import random
from scapy.all import *

def randomMAC():
	mac = [ 0x00, 0x16, 0x3e,
		random.randint(0x00, 0x7f),
		random.randint(0x00, 0xff),
		random.randint(0x00, 0xff) ]
	return ':'.join(map(lambda x: "%02x" % x, mac))

ap_list = []
fltr = sys.argv[1]

print "The filter is: %s" %(fltr)

def Broadcast(p) :
	broadcast = "ff:ff:ff:ff:ff:ff"
	bssid = randomMAC()

	pkt = RadioTap() / Dot11(addr1 = broadcast, addr2 = bssid, addr3 = bssid) /
Dot11Beacon(cap = 0x3114) / Dot11Elt( ID = 0, info = p.info.replace(fltr,"[SENT]")) /
Dot11Elt( ID = 1, info = "\x82\x84\x8b\x96\x0c\x12\x18\x24") / Dot11Elt( ID = 3, info
= "\x01") / Dot11Elt( ID = 5, info = "\x01\x03\x00\x08")

	sendp(pkt, iface = sys.argv[3], count = int(sys.argv[2]), inter = .2)

def PacketHandler(pkt) :
	if pkt.haslayer(Dot11) :
		if pkt.type == 0 and pkt.subtype == 8 :
			if pkt.addr2 not in ap_list and fltr in pkt.info and "[SENT]" not
in pkt.info :
				ap_list.append(pkt.addr2)
				print pkt.info
```

```
                                        Broadcast(pkt)

sniff(iface=sys.argv[3], prn = PacketHandler)
```

שאלה 5

תוכנית python אשר שולחת חבילת beacon מלאה ב – 256Kb:

```
#!/usr/bin/python
# Question 5
# Receiving 4 arguments:
# 1) SSID - the title
# 2) Rates - the content
# 3) How many time to send the package
# 4) interface

import sys
import random
from scapy.all import *

def randomMAC():
        mac = [ random.randint(0x00, 0x7f),
                random.randint(0x00, 0x7f),
                random.randint(0x00, 0x7f),
                random.randint(0x00, 0x7f),
                random.randint(0x00, 0xff),
                random.randint(0x00, 0xff) ]
        return ':'.join(map(lambda x: "%02x" % x, mac))

broadcast = "ff:ff:ff:ff:ff:ff"
bssid = randomMAC()
print bssid

ssid = sys.argv[1]
#ssid = [:29]
rates = sys.argv[2]
rates = rates[:255]

pkt = RadioTap() / Dot11(addr1 = broadcast, addr2 = bssid, addr3 = bssid) /
Dot11Beacon(cap = 0x3114) / Dot11Elt( ID = 0, info = ssid) / Dot11Elt( ID = 1, info =
rates) / Dot11Elt( ID = 3, info = "\x01") / Dot11Elt( ID = 5, info = "\x01\x03\x00\x08")

sendp(pkt, iface = sys.argv[4], count = int(sys.argv[3]), inter = .2)
```

שאלה 6

תוכנית python ראשונה אשר קוראת קובץ טקסט נבחר, ושולחת אותו ע"י מס' חבילות beacon לאחר חילוקו לחבילות נפרדות.

```
#!/usr/bin/python
# Question 6 - sendfiles
# Receiving 3 arguments:
# 1) Path to the file to send
# 2) How many time to send the package
# 3) interface

import sys
import random
from scapy.all import *

def randomMAC():
	mac = [ random.randint(0x00, 0x7f),
		random.randint(0x00, 0x7f),
		random.randint(0x00, 0x7f),
		random.randint(0x00, 0x7f),
		random.randint(0x00, 0xff),
		random.randint(0x00, 0xff) ]
	return ':'.join(map(lambda x: "%02x" % x, mac))

first = random.randint(0x00, 0x7f) + random.randint(0x00, 0x7f)
broadcast = "ffffffff" + `first`
bssid = "ffffffffffff"

filepath = sys.argv[1]
ssid = os.path.basename(filepath)
filesize = int(os.path.getsize(filepath))
pieces = filesize/255 + (filesize%255>0)
count = 1
print ssid

bssid = `pieces` + bssid[len(str(pieces)):12]
bssid = ":".join(bssid[i:i+2] for i in range(0,len(bssid),2))
print bssid

with open(filepath,"rb")as in_file:
	while count <= pieces:
		piece = in_file.read(255)
		print count
		broadcast = `count` + broadcast[len(str(count)):12]
```

```
				pkt = RadioTap() / Dot11(addr1 = bssid, addr2 = 
":".join(broadcast[i:i+2] for i in range(0,len(broadcast),2)), addr3 = bssid) / 
Dot11Beacon(cap = 0x3114) / Dot11Elt( ID = 0, info = "[F]"+ssid) / Dot11Elt( ID = 1, 
info = piece) / Dot11Elt( ID = 3, info = "\x01") / Dot11Elt( ID = 5, info = 
"\x01\x03\x00\x08")
				sendp(pkt, iface = sys.argv[3], count = int(sys.argv[2]), inter = .2)
				count=count+1
```

תוכנית python שנייה אשר מאזינה לחבילות beacon הנשלחות ברשת עד שמזהה את החבילות מהתוכנית הראשונה (ע"י התאמת כתובת). התוכנית כותבת את המידע שהתקבל בחבילות ה beacon לתוך קובץ טקסט לפי סדרם המקורי.

```
#!/usr/bin/env python
# Question 6 - receivefiles
# Receiving 1 argument:
# 1) interface

from scapy.all import *

ap_list = []
iface = sys.argv[1]
os.chdir(os.path.dirname(__file__))
directory = os.getcwd()+"/Downloads/"

if not os.path.exists(directory) :
	os.makedirs(directory)

def PacketHandler(pkt) :
	if pkt.haslayer(Dot11) :
		if pkt.type == 0 and pkt.subtype == 8 :
			if pkt.addr2 not in ap_list and "[F]" in pkt.info :
				ap_list.append(pkt.addr2)
				p = pkt
				filename = p.info.replace("[F]","")
				content = None
				while Dot11Elt in p :
					p = p[Dot11Elt]
					if(p.ID == 0):
						ssid = p.info
					elif(p.ID == 1):
						content = p.info
					p=p.payload
				text_file = open(directory+filename,"a")
				text_file.write(content)
				text_file.close()
```

```
sniff(iface=iface, prn = PacketHandler)
print "end"
```