IS 2621/TEL 2813 Security Management
Spring 2014 Lab Exercise - Network Forensics

In this lab, you will:

- conduct packet analysis of various protocols and services, including wireless 802.11 traffic
- reconstruct files from packet data
- create a timeline of relevant network events
- analyze flow records
- document evidence items and use these to guide a network forensic investigation

Note, you will need to be familiar with 802.11 wireless frame types and subtypes. A brief, but good, description of wireless traffic frame types and subtypes, as well as some corresponding Wireshark filter commands, can be found at the following link. It is highly recommended that you use this resource as a reference during your analysis.
https://supportforums.cisco.com/docs/DOC-13664

What You Will Need:

You will be utilizing a number of tools for your analysis, some more easily used in a Windows environment and some more easily used in a Linux environment. However, BackTrack Linux 5 r3 already has most of the tools you will need, so if you do not already have a BackTrack Linux VM ready, I strongly suggest that you download it and get it up and running, as it is useful for many things beyond this lab as well. You may need to install and/or update a few tools in BackTrack Linux, but this is trivial and can be addressed easily if you find it is necessary. Typically, all you have to do is open a terminal and issue 'apt-get install TOOLNAME'.

→ LabData-NF.zip https://pitt.box.com/LabData-NF pw: u4aT0r@R8
→ BackTrack 5 r3 (recommended) ISO at https://pitt.box.com/BT5R3 pw: u4aT0r@R8
OR
Kali Linux with a standard user account in addition to the root account
→ SiLK Analysis Suite from CERT
-Working installation guides available at the following links if you don't already have it:
Ubuntu - https://tools.netsa.cert.org/confluence/pages/viewpage.action?pageId=23298051
Security Onion - http://www.appliednsm.com/silk-on-security-onion/
→ RSA NetWitness Investigator 9 (Windows only) – registration required
http://www.emc.com/security/rsa-netwitness.htm#!freeware

Scenario:

Dr. George Thurgood works as a contracted researcher for a bio-tech firm. He sets up a rogue hotspot/Wireless Access Point (WAP) in order to circumvent some of the security mechanisms in place because he wants to share data more easily with other researchers and access the Internet without restrictions. The problem is that Dr. GT isn't exactly technically aware when it comes to security, and he has setup his WAP with 104/128 bit WEP encryption, it is accessible through the Internet, and it is connected to the internal LAN. Yowza!

Meanwhile, the resident network team has recently been made aware of a possible insider threat and has since been conducting regular “wardriving” tests. So it was not long after Dr. GT deployed his WAP that the network team was aware of it; however, not knowing for sure who deployed it and what his or her intentions might be, they decided to capture the associated wireless traffic. The discovery of the rogue WAP and wireless traffic capture occurred on a Wednesday. The bio-tech firm is one of your clients, and on the following Thursday, you are given the packet capture to analyze. Your goal is to identify general information about the WAP, any associated devices using the WAP, any malicious activity, and identify any data transfers, conversations, the entities involved, and, if possible, reconstruct the files. Based on signal strength measurements and subsequent location triangulation, the network team already suspects that Dr. GT may be involved, but adequate evidence is needed before any disciplinary and/or legal actions may be taken. As such, the network team also gives you the following information:

Dr. GT's Laptop MAC address is 00:26:B6:AA:1A:AA

Part 1: Wireless Traffic Analysis

Some things to keep in mind while analyzing wireless traffic are that there are three types of frames, type 0 refers to management frames, type 1 refers to control frames, and type 2 refers to data frames. Management frames will not be encrypted, but data frames will be. Also, as you will see shortly, different tools use different filtering mechanisms and 802.11 wireless traffic can be particularly confusing when it comes to bit transmission order. Let's start by loading the packet capture from LabData-NF.zip in Wireshark and applying some filters. Use the frame types and subtypes link provided above.

1.1 Find the BSSID, SSID, and channel of the WAP by filtering for Beacon frames or Probe Responses.

If a WAP is advertising its presence, it will broadcast Beacon frames. Otherwise, it will send Probe Responses to probing stations. In this case, the WAP is advertising its presence, so filtering for Beacon frames should be fine. In the Packet Details pane, expand the frame and its sub-properties to view more detailed information. Select the line that says 'Type/Subtype: Beacon frame (0x08)'. Notice in the Packet Bytes pane that the highlighted byte is 80, or 0x80, which is different from the 0x08 used to filter the frame. The byte value in the Packet Bytes pane represents the real bit transmission order of the frame, but Wireshark accounts for this automatically and denotes it as the correct frame type and subtype in the Packet Details pane. Let's try getting the same information by filtering for Beacon frames using tcpdump. In BackTrack or Kali, open a terminal and issue the following command:

tcpdump -nne -r capture.cap 'wlan[0] = 0x08'

Note, this assumes that you have copied the packet capture file into the /root/ directory, otherwise make sure you include the correct path to the capture file. Now hit Ctrl-C to end the output. As you'll no doubt have noticed, instead of getting one frame back, using tcpdump with the same bit order in the filter outputs a scrolling mess of data frames, not what we wanted. This is because, unlike Wireshark, tcpdump requires the byte representation of the actual bit transmission order, and it uses BPF filters, where in this case, wlan[0] means the first byte of the wireless frame. Using tcpdump again, try using the highlighted byte from the Wireshark Packet Bytes pane, 0x80.

Show a screen shot of this command and its output.

Go back to Wireshark, still filtering on Beacon frames, and explore the values in the Packet Details pane.

Q1.1.1 What is the BSSID, SSID, and channel of the WAP?

Q1.1.2 What is the value of the Protected flag and what does this mean?

Q1.1.3 Under Capabilities Information, what are the values of the ESS and IBSS bits? Is the AP likely operating in Infrastructure mode or Ad-hoc mode?

Q1.1.4 What is the value of the Privacy bit and what does this indicate? What kind of encryption is supported?

Q1.1.5 What are BPF filters?

For the purposes of this lab, we will be using tcpdump only minimally. It was used here to emphasize the difference between the 802.11 protocol specification and the way in which different tools based on the libpcap library capture and receive wireless traffic. Most network protocols do not have this problem, but it is certainly good to be aware of since wireless traffic is quite pervasive these days.

Q1.1.6 Regarding a couple of concepts relative to this issue, explain the difference between Big Endian and Little Endian.

1.2 Find out if there is more than one WAP sending traffic; find whether or not the data frames are encrypted.

For this information, we will use part of Wireshark's set of accompanying tools, the command-line tool, tshark. Here, we filter on Beacon frames or Probe Responses, and in addition, make sure that the ESS and IBSS fields are set to 1 and 0, respectively, and print out the number of corresponding frames for each unique BSSID. Open a terminal and issue the following command:

tshark -nn -r capture.cap -R '((wlan.fc.type_subtype == 0x08 || wlan.fc.type_subtype == 0x05) && (wlan_mgt.fixed.capabilities.ess == 1) && (wlan_mgt.fixed.capabilities == 0))' -T fields -e wlan.bssid | uniq -c

Show a screen shot of this command and its output. Q1.2.1 Is there more than one WAP sending traffic?

Notice that tshark uses the same display filters as Wireshark. Now, we want to filter for data frames (0x20). In a terminal, issue the following command piped into 'wc' to give a count of the frames with an -l, as in Larry, argument:

tshark -r capture.cap -R '(wlan.fc.type_subtype == 0x20) && (wlan.bssid == 20:aa:4b:ae:5d:46)' | wc -l

Q1.2.2 What is the number of data frames?

Now, we want to filter for data frames (0x20) that are also encrypted (Protected flag is 1). In a terminal, issue the following command:

tshark -r capture.cap -R '((wlan.fc.type_subtype == 0x20) && (wlan.fc.protected == 1)) && (wlan.bssid == 20:aa:4b:ae:5d:46)' | wc -l

Q1.2.3 What is the number of encrypted data frames?

1.3 Find any associated stations and identify traffic patterns.

Typically, associated stations will have been sent an Association Response (0x01) with a 2 byte Status Code: Successful (0x0000) fixed parameter. Let's filter for these and print a reverse numerical sort of the number of successful associations for each unique destination address. In a terminal, issue the following command:

tshark -r capture.cap -R '(wlan.fc.type_subtype == 0x01) && (wlan_mgt.fixed.status_code == 0x0000)' -T fields -e wlan.da | sort | uniq -c | sort -nr

Q1.3.1 How many successful Association Responses were sent and to whom?

Notice that the associated station from the previous output is not Dr. George Thurgood's Laptop MAC address, and that his laptop does not even appear to be associated with the WAP. This is suspicious, but it is possible that Dr. GT's laptop was already associated with the WAP before the packet capture was initiated. Regardless, there is an unknown device associated with the WAP. Let's get the timestamps for these associations and make a note of them. In a terminal, issue the following command:

tshark -r capture.cap -R '(wlan.fc.type_subtype == 0x01) && (wlan_mgt.fixed.status_code == 0x0000)' -T fields -e wlan.da -e frame.time | awk '{print $1, $5}'

For a more complete picture, let's see who is sending encrypted data traffic via the WAP. In a terminal, issue the following command to display unique source addresses sending this type of traffic:

tshark -r capture.cap -R '(wlan.fc.type_subtype == 0x20) && (wlan.fc.protected == 1)' -T fields -e wlan.sa | sort | uniq -c | sort -nr

Show a screen shot of this command and its output.

Now we see the MAC address of Dr. GT's laptop, the unknown device, and the station address (STA), 20:aa:4b:ae:5d:44, of the WAP, with the overwhelming majority of data frames being sent from the unknown device. Make a note of these MAC addresses.

Let's see where these stations are sending encrypted data traffic. We'll use the previous tshark command, but in addition, we will print out the destination field (wlan.da) and use 'awk' to limit the output to those addresses who exchanged more than 10 frames. In a terminal, issue the following command:

tshark -r capture.cap -R '(wlan.fc.type_subtype == 0x20) && (wlan.fc.protected == 1)' -T fields -e wlan.sa -e wlan.da | sort | uniq -c | sort -nr | awk '$1 > 10'

You can see that most of these data frames were sent to the broadcast address, ff:ff:ff:ff:ff:ff, which is unusual, and typically happens when stations send out ARP requests. The incredibly high number of encrypted data frames sent to the broadcast address is indicative of a WEP-cracking attack, where, as it seems in this case, ARP requests are captured and replayed, generating responses with new Initialization Vectors (IVs) to be captured and used to obtain the encryption key.

1.3.1 Context

In a terminal, issue the following command to get the beginning timestamp of the packet capture, note that the argument after 'head' is a one, not an L: **tcpdump -nnr capture.cap | head -1**

Q1.3.1.1 What is the beginning timestamp of the packet capture?

Q1.3.1.2 What is the ending timestamp? Hint: substitute 'tail -1' for 'head -1' in the previous command.

Since we know that Dr. GT was indeed communicating via the WAP, even though there were no corresponding Association Responses, it is reasonable to assume that he was already associated prior to beginning the packet capture. If he was using the WAP for any upper layer services, we should expect his laptop to be exchanging data frames with the WAP STA interface at or near the beginning of the capture and likely throughout the whole capture, which makes sense since this is also within the time frame of normal work day hours, i.e. 9am – 5pm, or 0900 – 1700 hours. Let's confirm this. In a terminal, issue the following command to get the distribution of data frames sent from the WAP's STA interface:

tshark -r capture.cap -R '(wlan.fc.type == 0x2) && (wlan.sa==20:aa:4b:ae:5d:44)' -T fields -e wlan.da | sort | uniq -c | sort -nr

Show a screen shot of this command and its output.

Now, use the following command to get the timestamps for both the initial and final recorded exchange of data frames between the WAP's STA interface and Dr. GT's laptop:

tshark -r capture.cap -R '(wlan.fc.type == 0x2) && (wlan.sa==20:aa:4b:ae:5d:44) && (wlan.da==00:26:b6:aa:1a:aa)' -T fields -e frame.time | awk '{print $4}' | head -1

Q1.3.1.3 Do these timestamps reflect our expectations regarding the participation of Dr. GT's laptop on the wireless network? Explain.

Perhaps we can expect the same thing from our mystery device. Let's see. In a terminal, issue the following command to get the distribution of data frames sent from the unknown device, limiting the output to those exchanges with more than 10 frames:

tshark -r capture.cap -R '(wlan.fc.type == 0x2) && (wlan.sa==00:25:9c:9b:a8:2a)' -T fields -e wlan.da | sort | uniq -c | sort -nr | awk '$1 > 10'

Show a screen shot of this command and its output.

Now, use your knowledge of the above commands to get the initial and final recorded exchange of data frames between the WAP's STA interface and the unknown device. **Show a screen shot of these commands and their outputs.**

Q1.3.1.4 Consider the timestamps of the successful Association Responses collected earlier. What conclusions may be drawn between those timestamps and the timestamps collected here regarding the data frame exchanges of the unknown device with the WAP's STA interface? Explain why might this be the case.

Since we suspect a WEP-cracking attack, let's take a look at when the unknown device begins broadcasting data frames. In a terminal, issue the following command:

tshark -r capture.cap -R '(wlan.fc.type == 0x2) && (wlan.sa==00:25:9c:9b:a8:2a) && (wlan.da==ff:ff:ff:ff:ff:ff)' -T fields -e frame.time | awk '{print $4}' | head -1

Q1.3.1.5 How much time elapses between the beginning of the packet capture and the time that the unknown device begins broadcasting data frames?

Let's use the latter timestamp to see how many IVs were generated during the time before the unknown device began broadcasting data frames. In a terminal, issue the following command:

tshark -r capture.cap -R 'frame.time < "Feb 19, 2014 16:06:35.197696000"' -T fields -e wlan.wep.iv | sort -u | wc -l

Q1.3.1.6 How many unique IVs were generated during this time? Average number of unique IVs generated per second?

Now, use your knowledge of the above commands to find the number of unique IVs generated between the time the unknown device began broadcasting data frames and the time of the unknown device's second successful association with the WAP. **Show a screen shot of this command and its output.**

Q1.3.1.7 How many unique IVs were generated during this time? Average number of unique IVs generated per second?

1.4 Timeline

Given all the data we have collected so far, it seems that our earlier suspicions of a WEP-cracking attack are valid. Moreover, if this is correct and the unknown device successfully obtained the WEP key via the forced generation of IVs, then we should be able to obtain the key from this packet capture as well. Let's try it. In a terminal, issue the following command: **aircrack-ng -b 20:aa:4b:ae:5d:46 capture.cap**

Q1.4.1 What is the WEP key?

Q1.4.2 Obviously, we can now decrypt the packet capture and do further analysis. Before we move on, however, it would be wise to construct a timeline of network events with the information we have now so that we may correlate with other events we might potentially discover in a Layer 3+ analysis. Below, construct a list of 7-9 network events that you believe are important for making a case. You should have a simple one or two line statement describing each event and be sure to include relevant timestamps. Also, make sure your timeline is in chronological order, i.e. earlier events listed first and so on. If you want to include other information not exposed by any of the above commands, you may, but be sure to show how you got the information. Hint: the beginning and end times of the capture are relevant network events.

Part 2: Layer 3+ Analysis

In the wireless traffic analysis, we left off having obtained the WEP key. Let's decrypt the capture file so that we can analyze the Layer 3+ traffic. For this, you will need to input the WEP key in hex form just like the BSSID. The following command will print some general stats about the capture file, but it will also create a "capture-dec.cap" file in the current working directory. Don't worry about any corrupted packets. In a terminal, issue the following command:

airdecap-ng -b BSSID -w WEPKEY capture.cap

You should replace BSSID and WEPKEY with the appropriate values.

The goal here is to find and extract any information about communications to and from Dr. GT, including aliases, addresses, credentials, conversations, and transferred files.

2.1 Narrow the search by identifying relevant protocols and IP and port information.

Open up the decrypted capture file in Wireshark. Go to the Statistics menu and click Protocol Hierarchy. This will give you a statistical summary of protocols in use over the course of the capture. The high percentage of ARP packets is no surprise and confirms our suspicions from Part 1. Note other Layer 3+ protocols of significance, including POP and SSL/TLS, the two TCP protocols which account for about 25% of packet bytes. There is even a Dropbox LAN sync Discovery Protocol in use here. Interesting. Go back to the Packet List pane and explore the first 15-20 packets.

Q2.1.1 What are the IP addresses associated with Dr. GT's laptop MAC address and WAP STA interface?

Q2.1.2 What is the IP address of the Domain Name Server being used here? What is it's Hostname? (may need external tool)

Q2.1.3 You should know that POP is used to retrieve email, and it was used here, but there is not anything indicating that emails were sent out, such as the use of SMTP. Type 'smtp' in the display filter and press Apply or hit Enter. Do you see any packets returned?

We know that packet bytes were transmitted via SSL/TLS, which can run on top of various services, including SMTP, and the Dropbox protocol is definitely suspicious. Before we explicitly check out either of these protocols, let's take a look at any DNS information that may have been captured. Filter for 'dns' and examine the results.

Q2.1.4 Are all of the DNS queries shown from Dr. GT's laptop IP address?

Q2.1.5 Locate the two queries with 'smtp' in the hostname, and note the associated IP address(es) in the responses. List any successful connections to these addresses. Include the hostname, IP address, and port number of the remote host, the time the connection was established, the time the connection was terminated, and whether or not it was using SSL/TLS.

Q2.1.6 Were there successful connections to the Dropbox IP addresses? Did each of these use SSL/TLS? If not, which ones did not use SSL/TLS? Which organization was being used to host Dropbox services?

Q2.1.7 Is it reasonable to conclude that outbound email and cloud data storage services were indeed utilized to transfer data, potentially leaking sensitive information?

Q2.1.8 Is it reasonable to conclude that Dr. GT is leaking sensitive information at this point?

Unfortunately, due to the use SSL/TLS, we cannot garner specific information about the data transferred using these services without the proper key. Nevertheless, it is good to record as much information as possible about these transactions, and, as the network investigator, you may have to submit preservation letters to the relevant businesses and/or service providers to ensure later access to this information if necessary.

2.2 Analyze POP, extract relevant information, and carve files from packet data

As we saw from the previous section, Dr. GT's laptop was using outbound SMTP and cloud storage services with SSL/TLS. We know that data was also transmitted using POP. There is a secure version of POP, which uses SSL/TLS and has a default port of 995, while the regular specification typically uses port 110 with no encryption. Since POP showed up in the Protocol Hierarchy summary statistics but SMTP did not, even though both services were used, we might expect that POP in this case was used without SSL/TLS. Let's confirm this. Filter for 'pop' and examine the results. Note, you will see POP3 or POP in different places. Both reference the protocol in use.

Q2.2.1 Do you see any indication of SSL/TLS? What port is being used for the POP3 service on the remote host?

Now, let's see what we can garner from this protocol. Instead of picking through the list of individual packets, we can use the 'Follow TCP Stream' feature in Wireshark to easily view and read Layer 7 information. Simply Right-click on a packet and select Follow TCP Stream. As you can see, some of the information is readable and some not so readable. The not so readable stuff is typically some kind of encoding, usually Base-64. If you have default settings, lines in red indicate information flowing to the POP3 service, and lines in blue indicate information flowing from the POP3 service.

At the bottom of the TCP stream window, in the drop-down box where it says Entire Conversation, select the flow that goes to the POP3 service (you should see only the lines in red). Here, we can see that some mail client on Dr. GT's laptop authenticated to the POP3 service and retrieved two messages, as is indicated by the RETR # commands. Notice the line, AUTH PLAIN, and the not so readable stuff underneath it. These are the Base-64-encoded credentials used to authenticate. Copy the encoded line and paste it between the quotes in the following command. Open a terminal and issue the following command: **echo "PasteEncodedLineHere" | base64 -d**

Q2.2.2 What are the credentials used to authenticate to the POP3 service? Discern the username and password separately.

Q2.2.3 From the drop-down box, select the Entire Conversation again. Identify relevant information regarding each retrieved message, including date and time of message, name/alias and/or email address of sender and receiver, subject, the actual message, and the names of any attached files. Hint: use the 'Find' button with keyword 'RETR'.

2.2.1 Carve file attachments

We are going to approach this exercise in three different ways in order to gain exposure to different tools and to consider limitations of each approach. To reconstruct the image file, go back to the TCP stream window in Wireshark. You should see a huge section of Base-64-encoded garble starting below the last instance of the filename, which you identified above, and continuing until it reaches an end-of-section line, which is indicated by --========GMX14703139284267451935 8--. Copy and paste that entire section of Base-64-encoded garble into a file and save it to the root directory. Open a terminal and issue the following command:

base64 -d FILEYOUJUSTMADE > NAMEOFIMAGEFILE.png

where FILEYOUJUSTMADE and NAMEOFIMAGEFILE are to be replaced with the appropriate file names.

Take a hash of the image file you just reconstructed using the following command:

md5sum NAMEOFIMAGEFILE.png

Q2.2.1.1 What is the MD5 hash of the image file? What is the image? Is this suspicious?

Now, this is a fine way of reconstructing files from packet data, but this approach does not scale well when there are a bunch of file transfers and will not work so easily if you are working with some tunneled or unknown protocols and/or encoding and/or fragmented files. While that is not the case here, there will be times when you'll need to export some generic TCP flow and search for file indicators manually. Let's do this for the PDF document that you identified above. We'll use a program called, tcpflow, to export the relevant flows from the decrypted capture file. In a terminal, issue the following command: **tcpflow -v -r capture-dec.cap 'port 110'** You should see a number of lines indicating a new output file was created. These files represent unidirectional flows and are named according to the source and destination sockets involved. Remember, we want to reconstruct files retrieved from the POP3 service, so we are interested in the flows from the POP3 service to Dr. GT's laptop address. In Wireshark, check the port numbers in use during the POP3 connection establishment to know which flow you need.

Next, we will use a hex editor to open the relevant tcp flow. We will be using Bless hex editor. If you are using BackTrack 5 r3, you will need to do an 'apt-get install bless' first. Follow the prompt and install it. It is helpful to list the directory contents first, so issue the 'ls' command. Open the relevant tcp flow using the following command:

bless FILENAME where FILENAME is the name of the tcpflow output file you want.

In the hex editor, hit Ctrl-F or click the menu button with the magnifying glass on it. Then, in the 'Search for:' text box, type 70 64 66, the hexadecimal representation of 'pdf'. This should bring you to the part of the original TCP stream where you identified the file name before. Just below this starts another huge section of Base-64-encoded garble, i.e. it begins immediately after the two CRLFs, indicated by '**….**'. The encoded section ends very near the end of the flow, immediately before a single CRLF, and this, followed by another end-of-section line. Including the single CRLF, the encoded section ends immediately before '**..**--047d7bdc8a0a989d3404f2c88399--'. To reconstruct the PDF, you need to snip off everything before and after this Base-64-encoded section. After you have done that, save the file to your root directory.

Since the carved attachment still has some line breaks in it, we need to get rid of these before it can be decoded. For this, we will be using a program called, 'fromdos', from the Debian package, 'tofrodos'. If it's not installed, do an 'apt-get install tofrodos', follow the prompt and install it. In a terminal, issue the following command:

fromdos -b CARVEDFILE where CARVEDFILE is the name of the file you just saved from Bless.

Now, issue the following command to decode and reconstruct the PDF file:

base64 -d CARVEDFILE > NAMEOFPDFFILE.pdf

Q2.2.1.2 What is the MD5 hash of the PDF file? What is the title of the paper in this document? Is this suspicious?

This is also a fine way to reconstruct files from packet data, but it does not scale when you have a large number of file transfers. Moreover, if you are not working with tunneled or unknown protocols and/or encoding, fragmented files, or some specific problem, which requires manual dissection, then you will likely want to be able to take care of file reconstruction automatically. RSA's NetWitness Investigator freeware is an excellent tool to analyze network data. Download and install this using the link provided. You must go through the registration and provide a valid email. Once its installed, run it, and create a New Local Collection. Right-click on the collection you just created and connect to it. Right-click on the collection again and select Import Packets. Locate and import the decrypted capture file (you should be able to copy and paste files between your VMs and host OS). Now, Double-click your collection to view it. You should see a row of buttons under your collection's tab. Find and click the one that says File Extract when you hover your mouse over it. Select a location to store extracted files and uncheck everything except Documents and Images. Press OK to extract the files. Take hashes of these files.

Q2.2.1.3 Do these hashes match the hashes of the files you extracted previously?

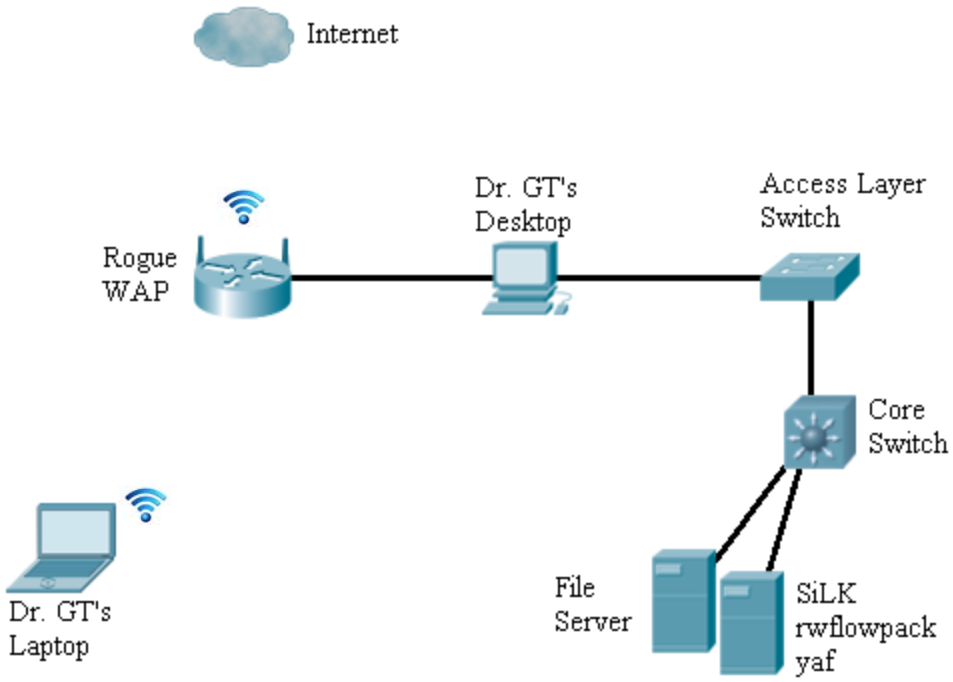
RSA NetWitness Investigator extracted these files easily and with little effort. This freeware has a lot of functionality and displays much good and concise information. You are encouraged to explore it and use it.

What we've discovered so far:

From the wireless traffic analysis, we know that Dr. GT's laptop was associated with the rogue WAP. We also found that an unknown device was active on the same WAP and probably obtained the WEP key via an ARP-replay attack. From the Layer 3+ analysis, we know that Dr. GT's laptop was used to access SMTP and cloud storage services over secure connections, potentially leaking sensitive information, but this requires further inquiry to be conclusive. We know that Dr. GT's laptop was used to retrieve mail messages from a POP3 service using plain authentication. We were able to recover these authentication credentials, and it is likely that the attacker associated with the unknown device was able to recover these as well. From the information collected about the retrieved mail messages and all file attachments, it remains to be determined whether Dr. GT was acting as an insider threat; for, aside from violating certain security policies, Dr. GT has not been shown to have done anything to deliberately disclose sensitive information.

Scenario continued:

The investigation is ongoing at this point, and during a subsequent meeting on the following Friday, you inform the network team at the bio-tech firm of all that you have discovered thus far. The network team tells you that prior to you confirming that Dr. GT actually did deploy and use the WAP, they searched around Dr. GT's desk, found the WAP, and noted its physical connections. The rogue WAP was removed on the Thursday following its discovery. They also express concern that one of their file servers may have been compromised. You are given the following topology:



You are asked to analyze the flow records and identify when, how, and from where the file server may have been probed and/or compromised. This will determine if the suspected compromise of the file server was related to the presence of the rogue WAP or if it was a separate incident, which may also possibly lead to the, as yet unidentified, insider threat. Finally, the network team also provides you with the following information:

The flow records are exactly 5 hours ahead of local time.

Dr. GT's Desktop IP address: 10.140.10.76/24

File Server IP address: 10.140.10.1/24

SiLK monitoring station IP: 10.140.10.200/24

Part 3: Flow Analysis

In this part of the lab, we will be using the SiLK analysis suite from SEI CERT to analyze flow records. If you don't already have it, you need to get an Ubuntu or Security Onion (recommended) VM up and running. Then, use one of the provided installation links to install SiLK. Since you will not be recording any flows, you do not need to have rwflowpack and yaf running at all, but you will need to be able to use SiLK rwtools. In LabData-NF.zip, there should be a folder called, silk-labdata. Copy this folder to your VM running SiLK. Make sure you know the full path to it. To point rwtools to these flow records, you need to open a terminal and issue the following command:

export SILK_DATA_ROOTDIR=PATH/silk-labdata/

where PATH should be replaced by the full path to the silk-labdata folder. You should be good to go, but you should issue this command in every new terminal you open. Flow records are stored in an hierarchical folder structure by type, year, month, day, and then, if there are multiple sensors, by sensor. If you navigate through the silk-labdata folder, you will notice that the only flow records are from February 19-20, 2014.

First, let's get some general stats to see if there is any relevant activity during the time of the flow records. We will check the 10.140.10.0/24 subnet, as this is where the file server resides. In a terminal, issue the following command:

**rwfilter --type=all --start-date=2014/02/19 --end-date=2014/02/20 --daddress=10.140.10.0/24 --pass=stdout | **
rwaddrcount --print-recs

Show a screen shot of this command and its output.

As you can see, there is definitely some activity, which we'll investigate further now. Since our goal is to determine when and how the file server may have been compromised, let's check out all flow records going to the file server. In a terminal, issue the following command to show flow records of any type, any protocol, having a destination address of the file server, and occurring any time on Feb. 19:

rwfilter --type=all --start-date=2014/02/19 --proto=0-255 --daddress=10.140.10.1 --pass=stdout | rwcut

Note: Although it may not look like it sometimes, each argument is passed with two dashes preceding it.
How many records do you see? Remember, these flow records were recorded exactly 5 hours ahead of local time. In the wireless traffic analysis, we determined that the second successful Association Response to the unknown device occurred at about 16:14:44 Feb. 19, and this device began exchanging data frames with the WAP STA interface shortly after, i.e. the device was active on the wireless LAN and able to use Layer 3+ services. If the unknown device was further engaged in attacking the file server, we can hypothesize that any subsequent flow records pertaining to this device would show up, at the earliest, around 16 + 5, or 2100 hours, on Feb. 19. As we just saw, not much showed up filtering for Feb. 19, but our hypothesized earliest time bound is only 3 hours away from Feb. 20. Let's see if there was any activity after these few hours. Issue the previous command, but replace 2014/02/19 with 2014/02/20 for the start date.

If you don't know what some of the output fields indicate, look it up online at http://tools.netsa.cert.org/silk/docs.html
Q3.1 How would you characterize this activity? Why? Include references to at least five of the output fields.

Q3.2 On which ports of the file server were connections established?

Of all the activity, four records in particular stand out. The first is a connection made on port 22 of the file server, which was sent 17 packets and 2355 bytes in about 30 seconds. The second is also a connection made on port 22 of the file server, which was sent 288 packets and 26424 bytes in a little over three and a half minutes. The other two records of interest are the last two connections made on port 443 of the file server. Let's dig into this more and take a look at all conversations involving port 22 and 443. Use your knowledge of the above commands and the SiLK docs to issue a command that returns all but only those flow records from Feb. 19 to Feb. 20 involving TCP port 22 and/or 443 in any direction on the 10.140.10.0/24 subnet.

Show a screen shot of this command and its output.

Notice the huge amount of bytes transferred back to Dr. GT's desktop machine, 10.140.10.76:49206, during the three and a half minute conversation noted earlier. This is definitely a red flag and is indicative of a bulk file transfer.

Q3.3 What would be the **local time** that this bulk file transfer took place?

Q3.4 Did this occur during normal business hours?

Q3.5 Was the rogue WAP still operating at this time?

Note: Here, our use of these tools is minimal; however, the SiLK analysis suite offers a ton of tools to analyze flow records. Of course, you can peruse and manipulate flow records with rwfilter, rwcut, and others. But one very cool tool is rwidsquery, which enables you to match flow records against Snort rules and alert files. Another cool tool is rwscan, which enables you to examine a given block of flow records to determine if any IP addresses in a specified set exhibit scanning behavior. You are encouraged to explore and use these tools.

Conclusions:

You bring your findings to the network team at the bio-tech firm and explain that, in light of the flow analysis, there was in fact suspicious activity found between Feb. 19-20, 2014, regarding the file server in question. Specifically, it appears that there was a bulk file transfer made on the evening of Feb. 19 back to Dr. GT's desktop from port 22 of the file server. You confirm with the network team and building security personnel that no one was in the building at the time of this network event. At this point, the evidence has us looking at three logical conclusions: either (1) Dr. GT is deliberately participating in the performing and/or masking of these network events, (2) Dr. GT's desktop has been compromised and has subsequently been used to perpetrate these network events, or (3) there is another threat that has not been considered and/or discovered.

Given Dr. GT's technical know-how, or lack thereof, it is rather unlikely that he configured any automated scripts, which might explain such network activity. So far, Dr. GT has only grossly violated security policy by introducing a rogue WAP at the firm, but based on what we've been able to determine, we cannot demonstrate that he purposely disclosed or stole sensitive information. Conclusion (1) remains to be determined, and the bio-tech firm plans to question Dr. GT for further information at the next available opportunity. Also, while the network team has recently been alert to the possibility of an insider threat, there simply is no other evidence pointing to sources that we have not considered; thus, conclusion (3) is unlikely. The best angle we have is conclusion (2), as most of the evidence points in this direction.

Final task and next steps:

In order to confirm some suspicions and to get a more granular look at the activity captured by the flow records, you ask the network team to provide you with the server logs for the day of the suspected intrusion. You will find these under 'ServerLogs' in LabData-NF.zip. Examine each of these logs, and take note of access attempts, user accounts, times, etc.

Q3.6 Correlating these logs with other evidence of network events, determine which services were provided on which ports on the file server.

Q3.7 In a paragraph or two, recount how you think the intrusion took place based on **all** evidence gathered. Make sure to include aspects of each of the previous analysis, i.e. Parts 1-3. Your recounting should be chronological. Think of your intended audience in a legal setting and as having limited technical capacities.

Q3.8 Think of any gaps in your investigation. List and briefly explain **two additional sources** of network and/or host-based evidence that you would like to acquire and further analyze in order to make your case stronger and perhaps more conclusive.

Bonus: Find a tool that allows you to search for one or more regular expressions in packet data payloads. Provide the name of the tool below. Does it support BPF filters as well?

*This lab was inspired by case studies and exercises from the following sources:

[1] Sherri Davidoff and Jonathan Ham. *Network Forensics: Tracking Hackers through Cyberspace*. Prentice Hall, 2012.
[2] Steven Anson et al. *Mastering Windows Network Forensics and Investigation*. 2nd Ed. Sybex, 2012.