

SAT1: 006: Carving Files in Wireshark

Overview

According to research, 90% of malware is installed via http through a web site transaction. In this lab, we are going to learn how to extract files from a packet capture, which can save valuable time in an incident.

Time: 45 Minutes

Learning Objectives

Upon completion of this lab, you should be able to:

1. Understand what data can be obtained from a PCAP file, and how it can be extracted.
2. Use the extraction menu to extract files from a PCAP file.
3. Use the TCP Follow command to extract file from PCAP file.
4. Edit an extracted file in order to open it in an image viewer.

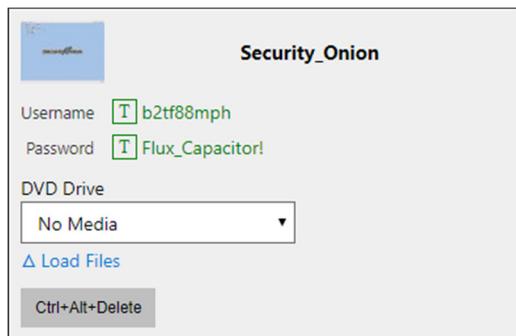
Resources

The following documents have been provided to assist you in this lab.

[Wireshark Cheat Sheet](#)

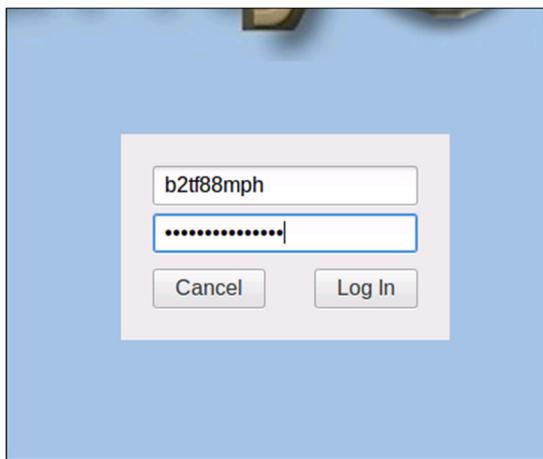
Log in to the Lab Machine

Select the **Security_Onion** machine on the Machines Tab.

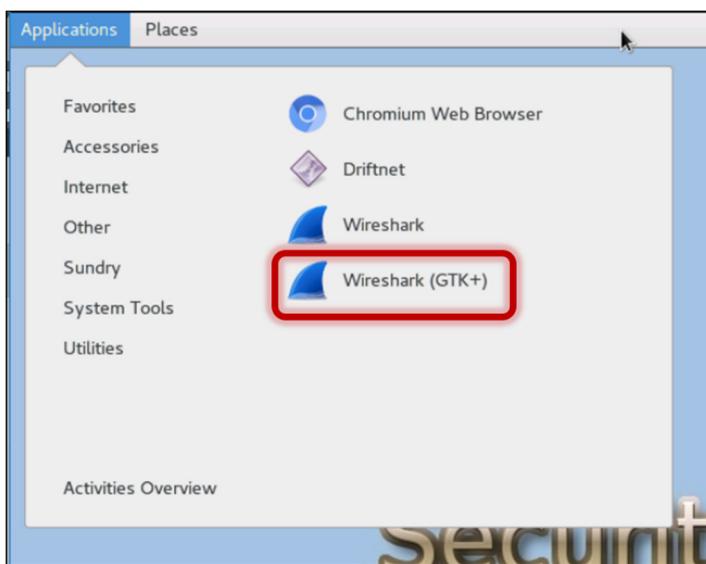




Enter the Username and Password on the **Security_Onion** machine, and click Log In.



First, you will need to open the Wireshark application. In **Security_Onion**, Wireshark can be found by clicking on **Applications**, move your mouse over **Internet**, and click on **Wireshark (GTK+)**.



1.0 Carving Files with Wireshark

1.1 On the Main Menu, click **File**, then click **Open**.

1.2 Navigate to the folder on the Desktop named **Captures**.

1.3 Open the file named **file_extract.pcap**.

In the Filter Toolbar, we are going to search for http requests using the filter `http.request`. This will filter out all traffic except HTTP GET and HTTP POST requests so we can see what information was obtained from the web server.



1.4 Enter http.request in the Filter Toolbar.

The screenshot shows the Wireshark interface with a packet list titled "file_extract.pcap". The "Filter" toolbar at the top has "http.request" entered. A red box highlights this filter entry. The packet list displays several network frames, with the 50th frame selected. The details pane below shows a GET request for "Websidan/images/sydney.jpg" over HTTP. A red box highlights the "GET /Websidan/images/sydney.jpg HTTP/1.1\r\n" line in the request details.

Time	Src Port	Dst Port	Source	Destination
22:29:14.172938	3177	80	10.1.1.101	10.1.1.1
22:29:14.809218	3179	80	10.1.1.101	209.225.11.237
22:29:15.443254	3188	80	10.1.1.101	10.1.1.1
22:29:15.562952	3189	80	10.1.1.101	10.1.1.1
22:29:15.564207	3190	80	10.1.1.101	10.1.1.1
22:29:15.661054	3183	80	10.1.1.101	209.225.0.6
22:29:15.661402	3184	80	10.1.1.101	209.225.0.6
22:29:15.661676	3185	80	10.1.1.101	209.225.0.6
22:29:15.669239	3187	80	10.1.1.101	209.225.0.6
22:29:16.739805	3191	80	10.1.1.101	209.225.0.6
22:29:16.743422	3192	80	10.1.1.101	209.225.0.6
22:29:17.004513	3193	80	10.1.1.101	209.225.0.6
22:29:17.422341	3195	80	10.1.1.101	10.1.1.1

Frame 50: 654 bytes on wire (5232 bits), 654 bytes captured (5232 bits)
Ethernet II, Src: SmcNetwo_22:5a:03 (00:04:e2:22:5a:03), Dst: Kye_20:6c:df (00:c0:df:20:6c:df)
Internet Protocol Version 4, Src: 10.1.1.101, Dst: 10.1.1.1
Transmission Control Protocol, Src Port: 3190, Dst Port: 80, Seq: 1, Ack: 1, Len: 600

Hypertext Transfer Protocol
GET /Websidan/images/sydney.jpg HTTP/1.1\r\nUser-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0) Opera 7.11 [en]\r\nHost: 10.1.1.1\r\nAccept: application/x-shockwave-flash,text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,video/quicktime\r\nAccept-Language: en\r\nAccept-Charset: windows-1252, utf-8, utf-16, iso-8859-1;q=0.6, *;q=0.1\r\nAccept-Encoding: deflate, gzip, x-gzip, identity, *;q=0\r\nReferer: http://10.1.1.1/Websidan/index.html\r\n

If we look in the packet list, we can see that several jpg files were downloaded. Now that we know we are looking for image files, there are a couple of methods for extracting these files.

This screenshot is identical to the one above, showing the Wireshark interface with the "http.request" filter applied. The 50th frame is selected, and the details pane shows the same GET request for "sydney.jpg". However, the entire "GET /Websidan/images/sydney.jpg HTTP/1.1\r\n" line in the request details is now highlighted with a red box, indicating it is the target for extraction.

Time	Src Port	Dst Port	Source	Destination
22:29:14.172938	3177	80	10.1.1.101	10.1.1.1
22:29:14.809218	3179	80	10.1.1.101	209.225.11.237
22:29:15.443254	3188	80	10.1.1.101	10.1.1.1
22:29:15.562952	3189	80	10.1.1.101	10.1.1.1
22:29:15.564207	3190	80	10.1.1.101	10.1.1.1
22:29:15.661054	3183	80	10.1.1.101	209.225.0.6
22:29:15.661402	3184	80	10.1.1.101	209.225.0.6
22:29:15.661676	3185	80	10.1.1.101	209.225.0.6
22:29:15.669239	3187	80	10.1.1.101	209.225.0.6
22:29:16.739805	3191	80	10.1.1.101	209.225.0.6
22:29:16.743422	3192	80	10.1.1.101	209.225.0.6
22:29:17.004513	3193	80	10.1.1.101	209.225.0.6
22:29:17.422341	3195	80	10.1.1.101	10.1.1.1

Frame 50: 654 bytes on wire (5232 bits), 654 bytes captured (5232 bits)
Ethernet II, Src: SmcNetwo_22:5a:03 (00:04:e2:22:5a:03), Dst: Kye_20:6c:df (00:c0:df:20:6c:df)
Internet Protocol Version 4, Src: 10.1.1.101, Dst: 10.1.1.1
Transmission Control Protocol, Src Port: 3190, Dst Port: 80, Seq: 1, Ack: 1, Len: 600

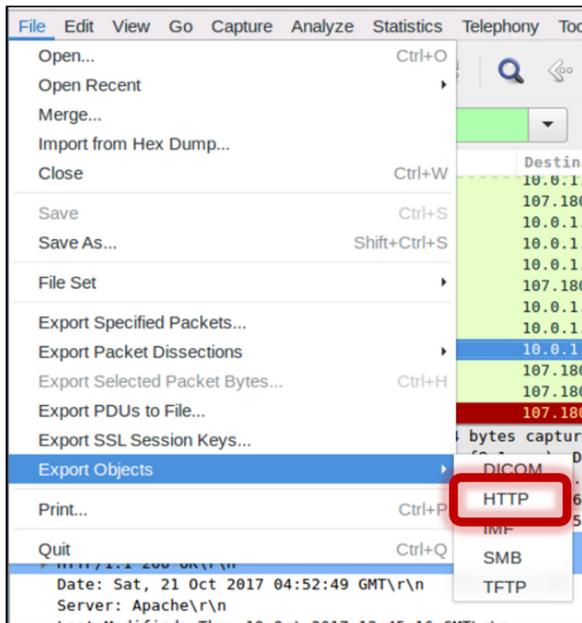
Hypertext Transfer Protocol
GET /Websidan/images/sydney.jpg HTTP/1.1\r\nUser-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0) Opera 7.11 [en]\r\nHost: 10.1.1.1\r\nAccept: application/x-shockwave-flash,text/xml,application/xml,application/xhtml+xml;text/html;q=0.9;text/plain;q=0.8,video/quicktime\r\nAccept-Language: en\r\nAccept-Charset: windows-1252, utf-8, utf-16, iso-8859-1;q=0.6, *;q=0.1\r\nAccept-Encoding: deflate, gzip, x-gzip, identity, *;q=0\r\nReferer: http://10.1.1.1/Websidan/index.html\r\n



Extracting Files from Wireshark

2.0 Using the Extraction Menu

2.1. Using Wireshark, we can extract the file from the TCP stream. In Wireshark, click **File**, then move your mouse over **Export Objects**, and click **HTTP**.



2.2 This will produce an **HTTP Object List**.

Wireshark: HTTP object list				
Packet num	Hostname	Content Type	Size	Filename
6	10.1.1.1	text/html	160 bytes	/
16	ins1.opera.com	application/vnd.xacp	433 bytes	xcms.asp
19	ins1.opera.com		5 bytes	xcms.asp
38	10.1.1.1	text/html	4,323 bytes	index.html
61	10.1.1.1	image/jpeg	8,281 bytes	bg2.jpg
72	10.1.1.1	image/jpeg	9,045 bytes	sydney.jpg
100	opera1-servedby.advertising.com		1,247 bytes	dst=Win_700
109	opera2-servedby.advertising.com		1,247 bytes	dst=Win_700
120	opera4-servedby.advertising.com		1,247 bytes	dst=Win_700
137	opera3-servedby.advertising.com		1,247 bytes	dst=Win_700
159	10.1.1.1	text/html	416 bytes	dagbok.html
207	opera4-servedby.advertising.com		1,136 bytes	bins=1
218	10.1.1.1	text/html	1,263 bytes	dagbok.html
230	10.1.1.1	text/html	2,232 bytes	dagbok.html
259	10.1.1.1	image/jpeg	8,963 bytes	DSC07858.JPG
269	10.1.1.1	image/jpeg	10 kB	DSC07859.JPG
479	10.1.1.1	image/jpeg	191 kB	DSC07858.JPG



2.3 We can see that there are multiple jpg files in the traffic.

Packet num	Hostname	Content Type	Size	Filename
6	10.1.1.1	text/html	160 bytes	/
16	ins1.opera.com	application/vnd.xacp	433 bytes	xcms.asp
19	ins1.opera.com		5 bytes	xcms.asp
38	10.1.1.1	text/html	4,323 bytes	index.html
61	10.1.1.1	image/jpeg	8,281 bytes	bg2.jpg
72	10.1.1.1	image/jpeg	9,045 bytes	sydney.jpg
100	opera1-servedby.advertising.com		1,247 bytes	dst=Win_700
109	opera2-servedby.advertising.com		1,247 bytes	dst=Win_700
120	opera4-servedby.advertising.com		1,247 bytes	dst=Win_700
137	opera3-servedby.advertising.com		1,247 bytes	dst=Win_700
159	10.1.1.1	text/html	416 bytes	dagbok.html
207	opera4-servedby.advertising.com		1,136 bytes	bins=1
218	10.1.1.1	text/html	1,263 bytes	dagbok.html
230	10.1.1.1	text/html	2,232 bytes	dagbok.html
259	10.1.1.1	image/jpeg	8,963 bytes	DSC07858.JPG
269	10.1.1.1	image/jpeg	10 kB	DSC07859.JPG
479	10.1.1.1	image/jpeg	191 kB	DSC07858.JPG

2.4 Because we cannot choose to save a specific file type, we will use the **Save All** command to capture all the files, and place them in **Pictures** folder.

Packet num	Hostname	Content Type	Size	Filename
6	10.1.1.1	text/html	160 bytes	/
16	ins1.opera.com	application/vnd.xacp	433 bytes	xcms.asp
19	ins1.opera.com		5 bytes	xcms.asp
38	10.1.1.1	text/html	4,323 bytes	index.html
61	10.1.1.1	image/jpeg	8,281 bytes	bg2.jpg
72	10.1.1.1	image/jpeg	9,045 bytes	sydney.jpg
100	opera1-servedby.advertising.com		1,247 bytes	dst=Win_700
109	opera2-servedby.advertising.com		1,247 bytes	dst=Win_700
120	opera4-servedby.advertising.com		1,247 bytes	dst=Win_700
137	opera3-servedby.advertising.com		1,247 bytes	dst=Win_700
159	10.1.1.1	text/html	416 bytes	dagbok.html
207	opera4-servedby.advertising.com		1,136 bytes	bins=1
218	10.1.1.1	text/html	1,263 bytes	dagbok.html
230	10.1.1.1	text/html	2,232 bytes	dagbok.html
259	10.1.1.1	image/jpeg	8,963 bytes	DSC07858.JPG
269	10.1.1.1	image/jpeg	10 kB	DSC07859.JPG
479	10.1.1.1	image/jpeg	191 kB	DSC07858.JPG



2.5 Wireshark will give you the chance to create a folder, which we named Images here. (We will analyze these files later).

3.0 Following the TCP Stream

NOTE: For this method we are only going to extract the sydney.jpg file.

3.1 In the Filter Toolbar, type http.request again.

3.2 Look for the GET request for the sydney.jpg file and click on it.

The screenshot shows the Wireshark interface with a list of network frames. A filter bar at the top displays "http.request". The main pane lists several HTTP requests from various sources (e.g., 10.1.1.101) to destination port 80. One specific frame is highlighted in blue, corresponding to the 50th frame. Below this frame, a detailed packet information pane is visible, showing the following details:

- Frame 50: 654 bytes on wire (5232 bits), 654 bytes captured (5232 bits)
- Ethernet II, Src: SmcNetwo_22:5a:03 (00:04:e2:22:5a:03), Dst: Kye_20:6c:df (00:c0:df:20:6c:df)
- Internet Protocol Version 4, Src: 10.1.1.101, Dst: 10.1.1.1
- Transmission Control Protocol, Src Port: 3190, Dst Port: 80, Seq: 1, Ack: 1, Len: 600
- Hypertext Transfer Protocol
 - GET /Websidan/images/sydney.jpg HTTP/1.1\r\n

The "GET /Websidan/images/sydney.jpg HTTP/1.1\r\n" line is highlighted with a red rectangle.

3.3 Right-click the **GET** request and select **Follow TCP Stream**.



The screenshot shows the Wireshark interface with a packet capture named "file_extract.pcap". The packet details pane shows several HTTP requests, with the 50th frame selected. The bytes pane shows the raw hex and ASCII data for this frame. A context menu is open over the selected packet, with the "Follow TCP Stream" option highlighted.

NOTE: The stream will be in red and blue text. The red text is from the source machine and the blue text is from the destination machine.



3.4 In the drop down menu at the bottom, change the conversation from 'Entire Conversation (9930 bytes)' to '10.1.1.1:80 to 10.1.1.101:3190 (9330 bytes)'.

3.5 This will allow us to see the data only from the web server to the host.

2.....13.....14.....5..v...cz#d.x...
6.....10.....0#d..v.\..d.{CB..<....YJ...;<<....(....
\$.....w.....|w.....;..k...z...w...#.S1.l...
%.N.Qo,..0`...fx>....1>..36.....p.33.....z.....#.fa...
Y.....|.....m...VA.I.>.....]
#.....|.....

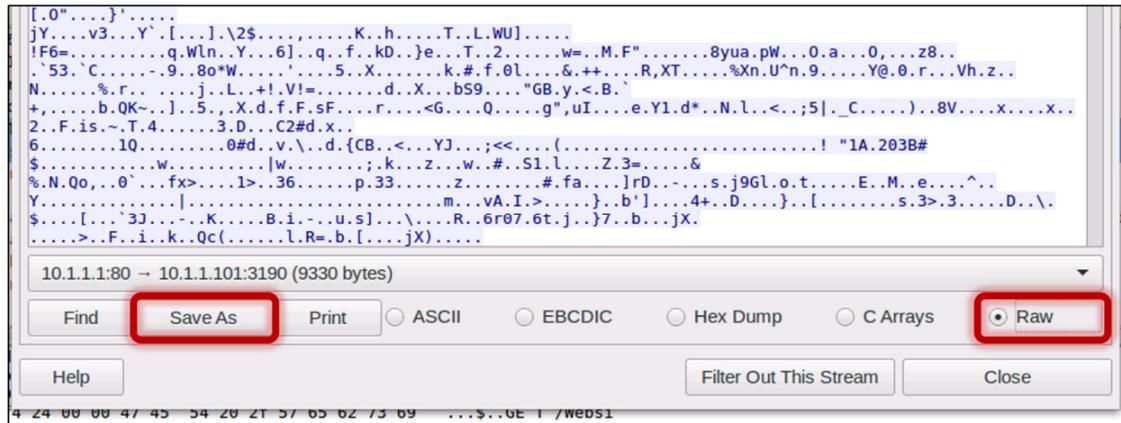
Entire conversation (9930 bytes)
10.1.1.1:80 → 10.1.1.101:3190 (9330 bytes)
10.1.1.101:3190 → 10.1.1.1:80 (600 bytes)

Note: After the header information, you will see a line that contains the file information. The 'JFIF' is the file type and we need that to search for the magic number.



3.6 In the menu at the bottom of the TCP Stream window, change the data from 'ASCII' to 'Raw'. This will allow us to view the raw data.

3.7 Click **Save As** and save the file in the **Captures** folder, as Sydney.jpg.



3.8 Close the TCP stream window.

3.9 Open Chromium Web Browser (click **Applications**, move your mouse over **Internet**, and click **Chromium Web Browser**). Then, click on the **File Signature Database** bookmark in the toolbar. This is a well maintained list of HEX file signatures (aka "magic numbers").



3.10 We know the file is a JFIF so we can search the webpage for 'JFIF'. Enter **JFIF** in the **Search** and click **Submit** to view the results. The magic number we will use is FF D8 FF E0



File Signatures

01100110011010010101001100010010100100000011001011010010011011001000011010011001011001100101011001
 6 6 : 6 9 : 6 c : 6 5 : 2 0 : 7 3 : 6 9 : 6 7 : 6 e : 6 1 : 7 4 : 7 5 : 7 2 : 6 5 : 7 3

[Search](#)

[All Signatures](#)

[Submit Sigs](#)

[My Favorites](#)

[Control Panel](#)

Disable autocomplete

Extension Signature

2 Results Found For JFIF File Extension

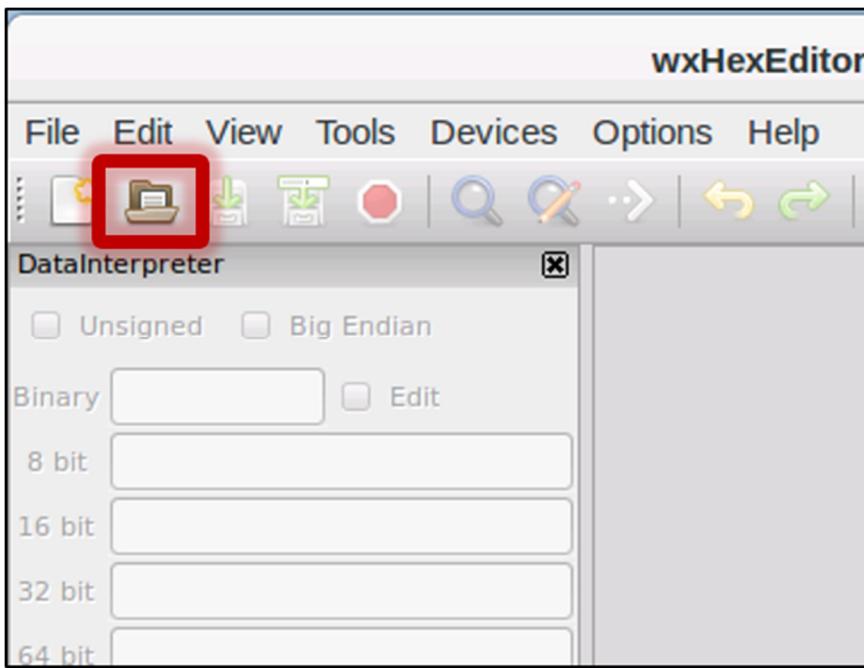
Extension	Signature	Description
 JFIF	<u>FF D8 FF E0</u> ASCII	JPEG IMAGE Sizet: 4 Bytes Offset: 0 Bytes
 JFIF	<u>FF D8 FF E0</u> ASCII	JFIF IMAGE FILE - jpeg Sizet: 4 Bytes Offset: 0 Bytes

3.11 Open the raw dump file using wxHexEditor. Click on **Applications**, then move your mouse over **Accessories**, then click **wxHexEditor**.

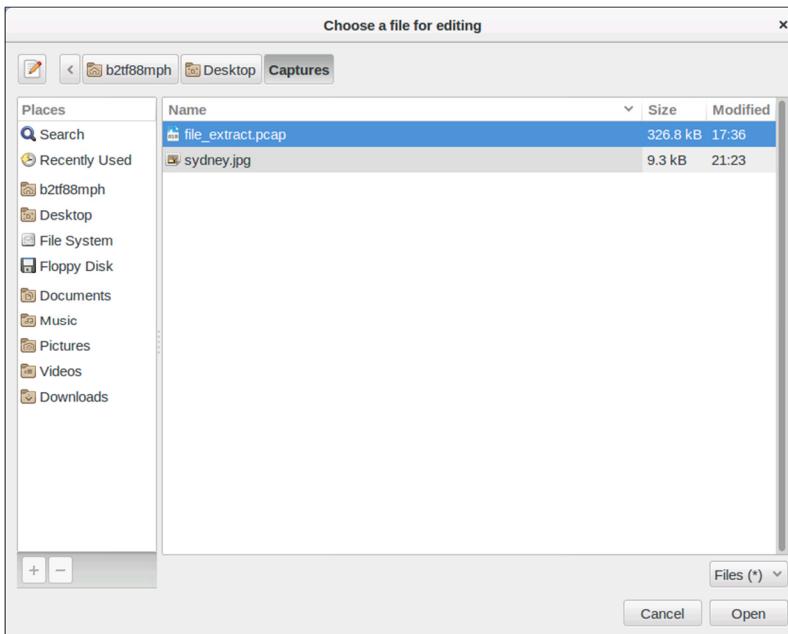




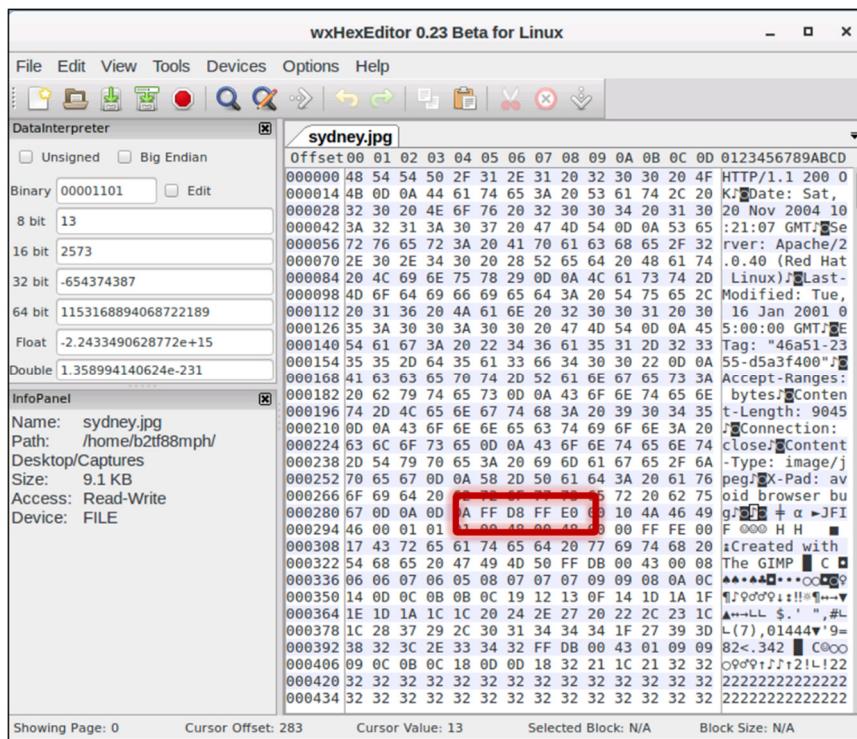
3.12 Click the folder icon.



3.13 Navigate to the file you extracted from the PCAP in the Captures folder on the Desktop.

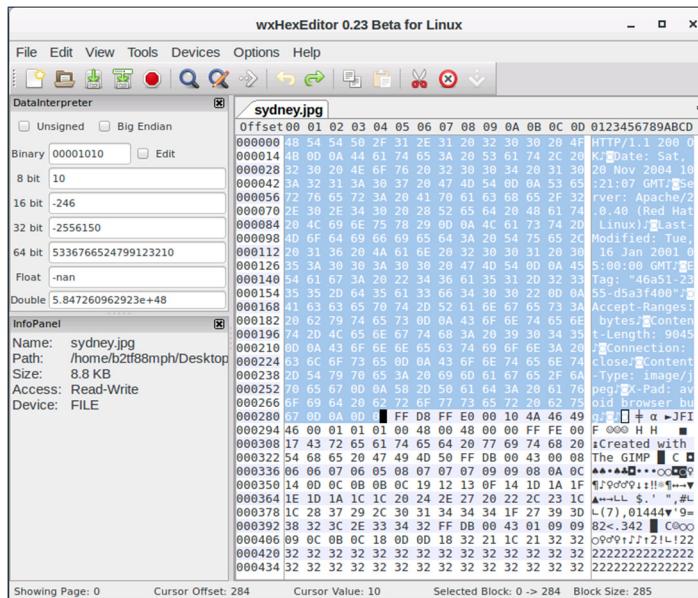


3.14 Now we look for the magic number FF D8 FF E0



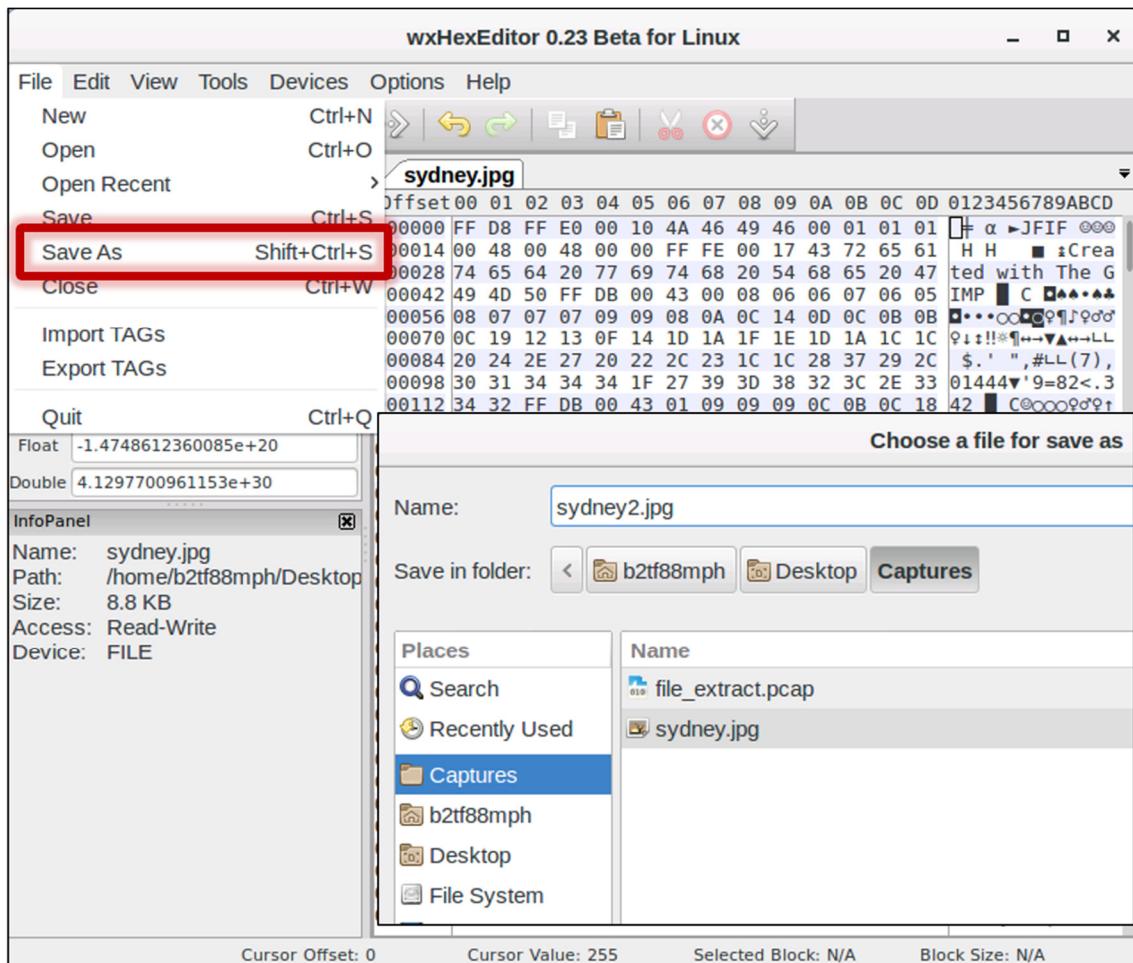
3.15 Delete all HEX prior to the magic number.

You can do this by selecting the first characters at **00 x 000000** and dragging it to the first two digits of the magic number.

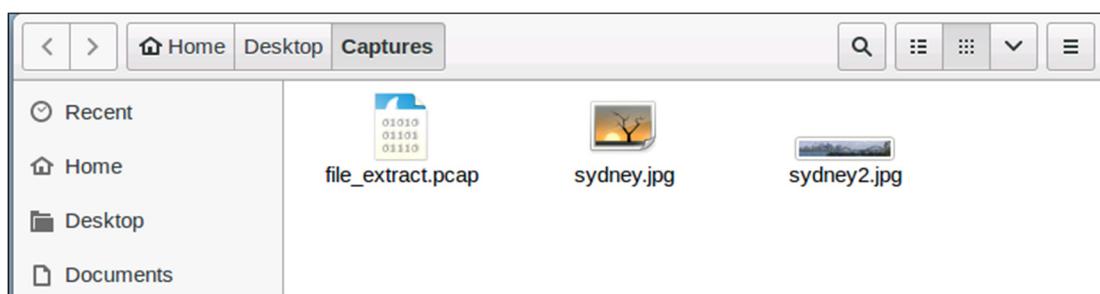




3.16 Save the file as a new jpg file.



As you can see, the file is now a jpg image instead of a broken file.



3.17 You can use the image viewer to open the file if you would like.

Great job, you have completed LAB006!

Thank You, you may now close this module.