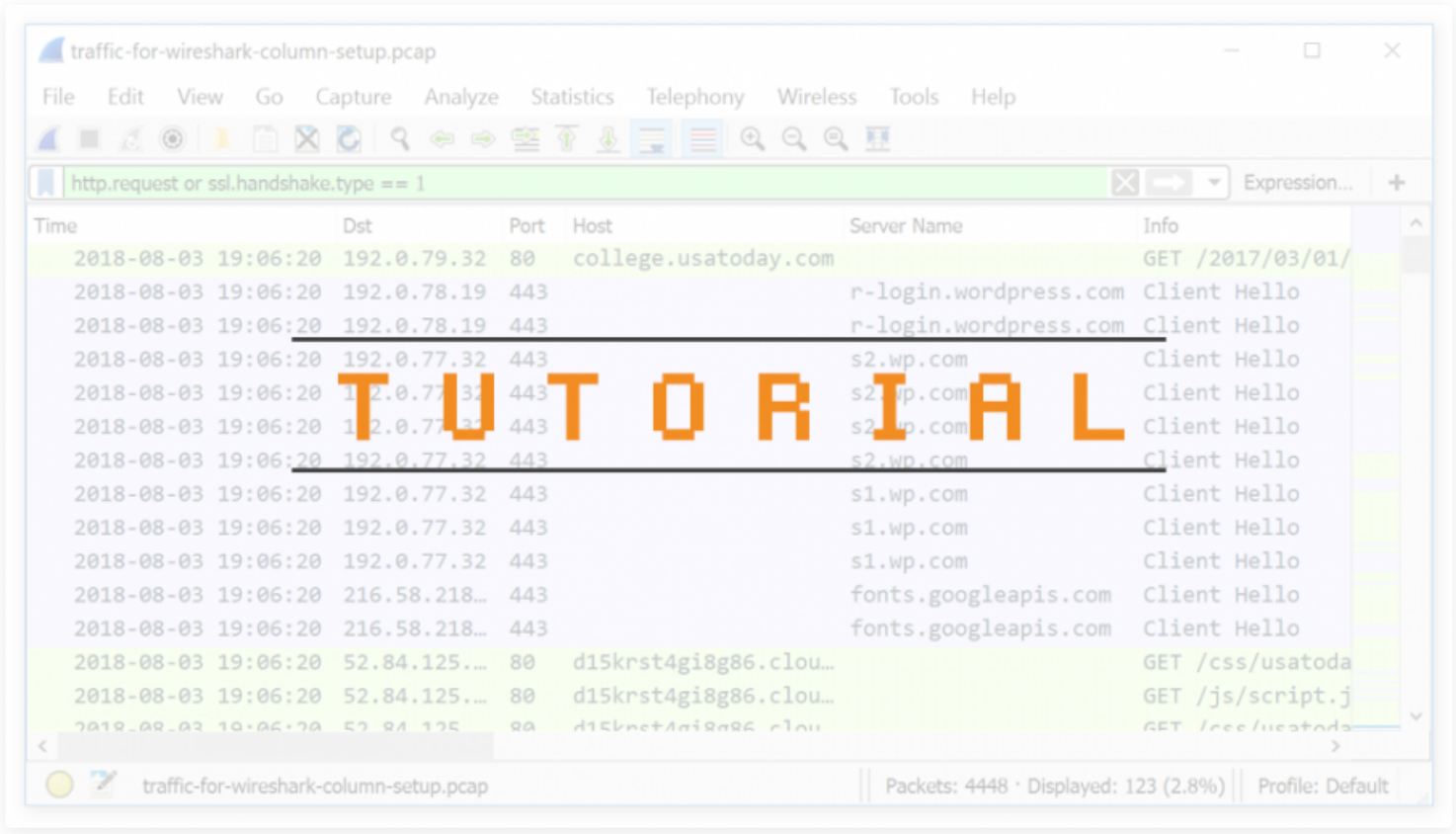
Using Wireshark: Exporting Objects from a Pcap

By Brad Duncan
July 10, 2019 at 6:00 AM
Category: Unit 42
Tags: tutorial, Wireshark



When reviewing packet captures (pcaps) of suspicious activity, security professionals may need to export objects from the pcaps for a closer examination.

This tutorial offers tips on how to export different types of objects from a pcap. The instructions assume you understand network traffic fundamentals. We will use these pcaps of network traffic to practice extracting objects using Wireshark. The instructions also assume you have customized your Wireshark column display as previously demonstrated in this tutorial.

Warning: Most of these pcaps contain Windows malware, and this tutorial involves examining these malicious files. Since these files are Windows malware, I recommend doing this tutorial in a non-Windows environment,

like a MacBook or Linux host. You could also use a virtual machine (VM) running Linux.

This tutorial covers the following areas:

- Exporting objects from HTTP traffic
- Exporting objects from SMB traffic
- Exporting emails from SMTP traffic
- Exporting files from FTP traffic

Exporting Objects from HTTP Traffic

The first pcap for this tutorial, ***extracting-objects-from-pcap-example-01.pcap***, is available here. Open the pcap in Wireshark and filter on ***http.request*** as shown in Figure 1.

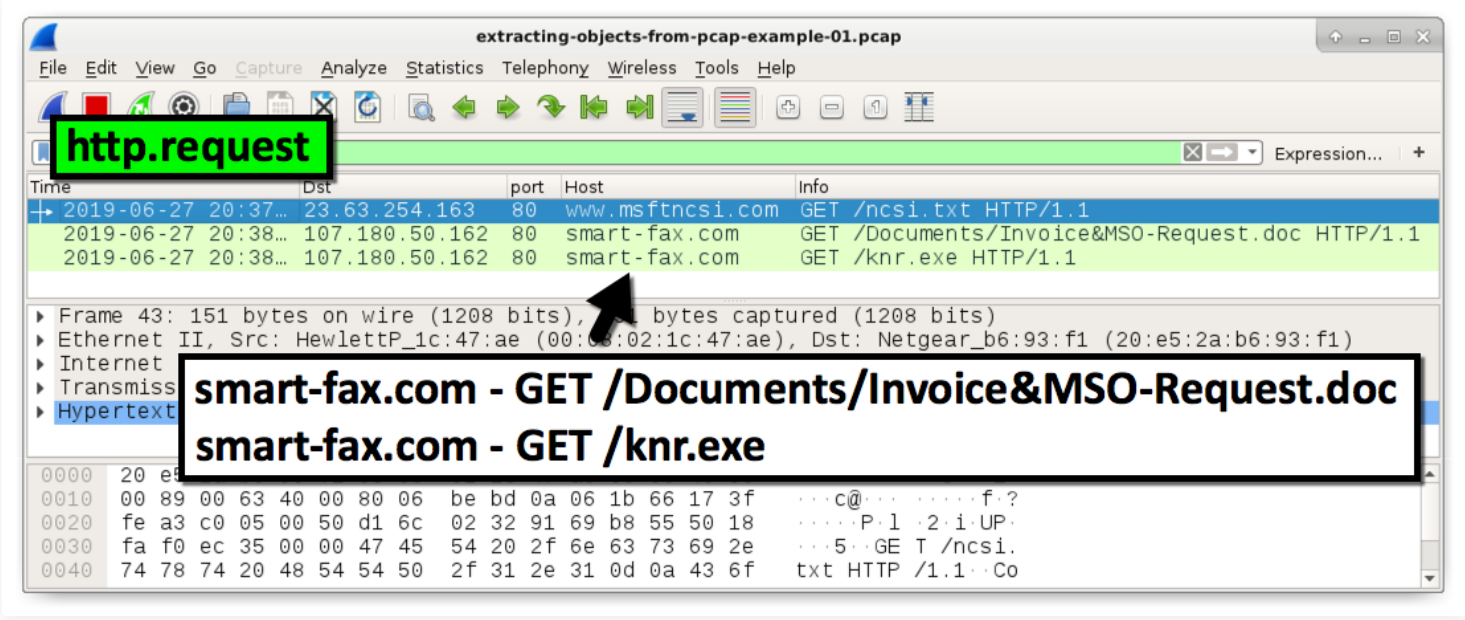
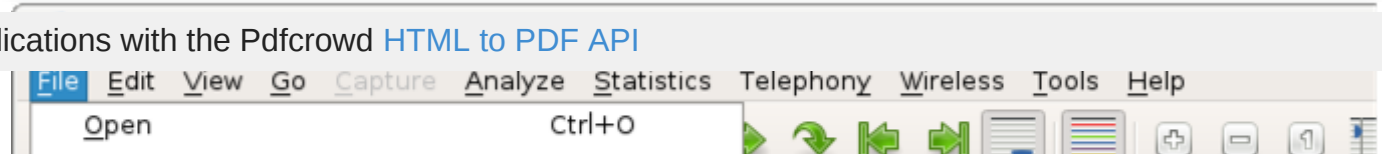


Figure 1. Filtering on the tutorial's first pcap in Wireshark.

After filtering on ***http.request***, find the two GET requests to ***smart-fax[.]com***. The first request ends with ***.doc***, indicating the first request returned a Microsoft Word document. The second request ends with ***.exe***, indicating the second request returned a Windows executable file. The HTTP GET requests are listed below.

- ***smart-fax[.]com*** – GET /Documents/Invoice&MSO-Request.doc
- ***smart-fax[.]com*** – GET /knr.exe

We can export these objects from the HTTP object list by using the menu path: ***File –> Export Objects –> HTTP...*** Figure 2 show this menu path in Wireshark.



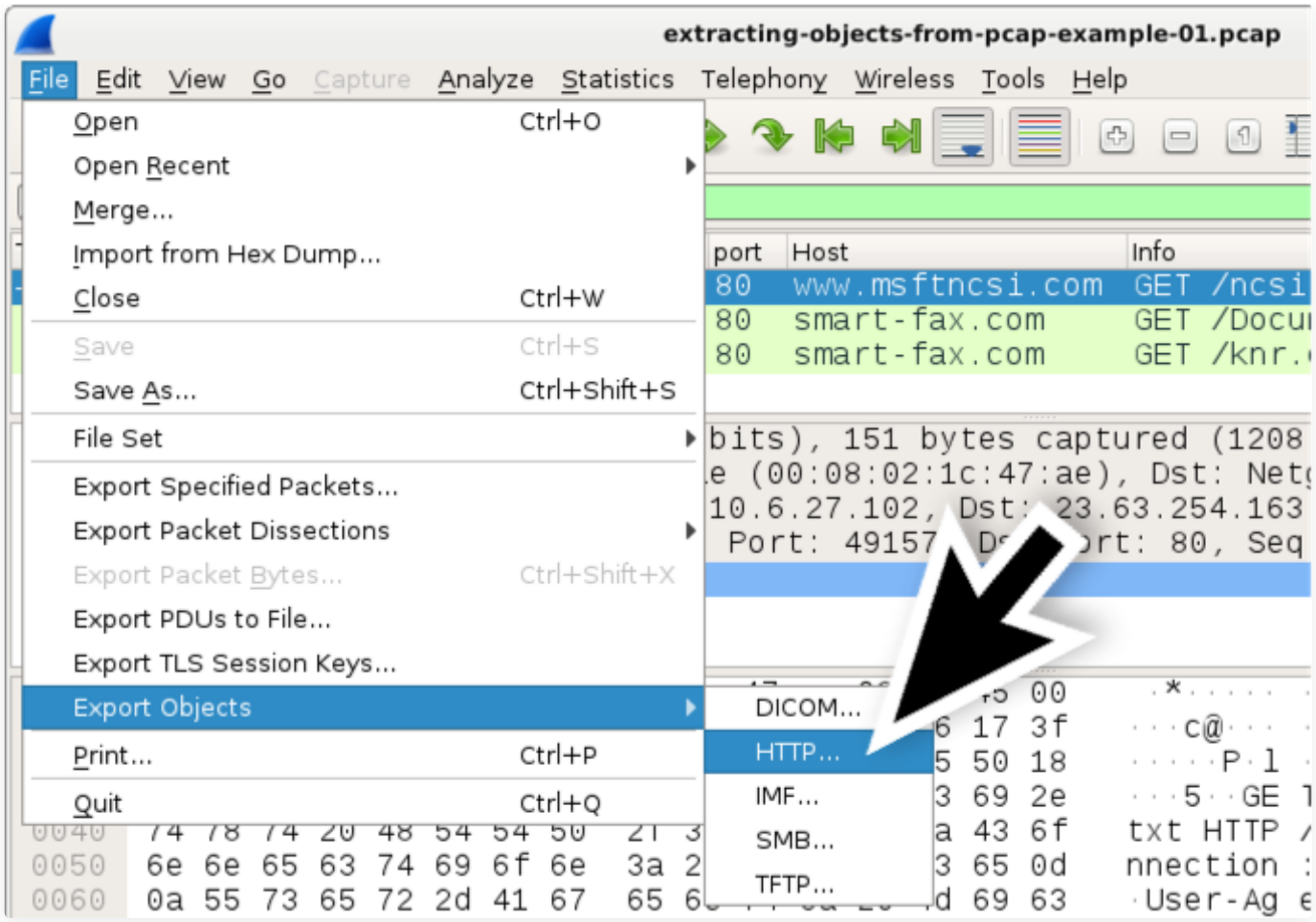
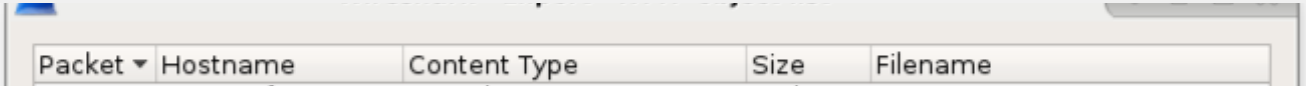


Figure 2. Exporting HTTP objects in Wireshark.

This menu path results in an Export HTTP object list window as shown in Figure 3. Select the first line with ***smart-fax[.]com*** as the hostname and save it as shown in Figure 3. Select the second line with ***smart-fax[.]com*** as the hostname and save it as shown in Figure 4.



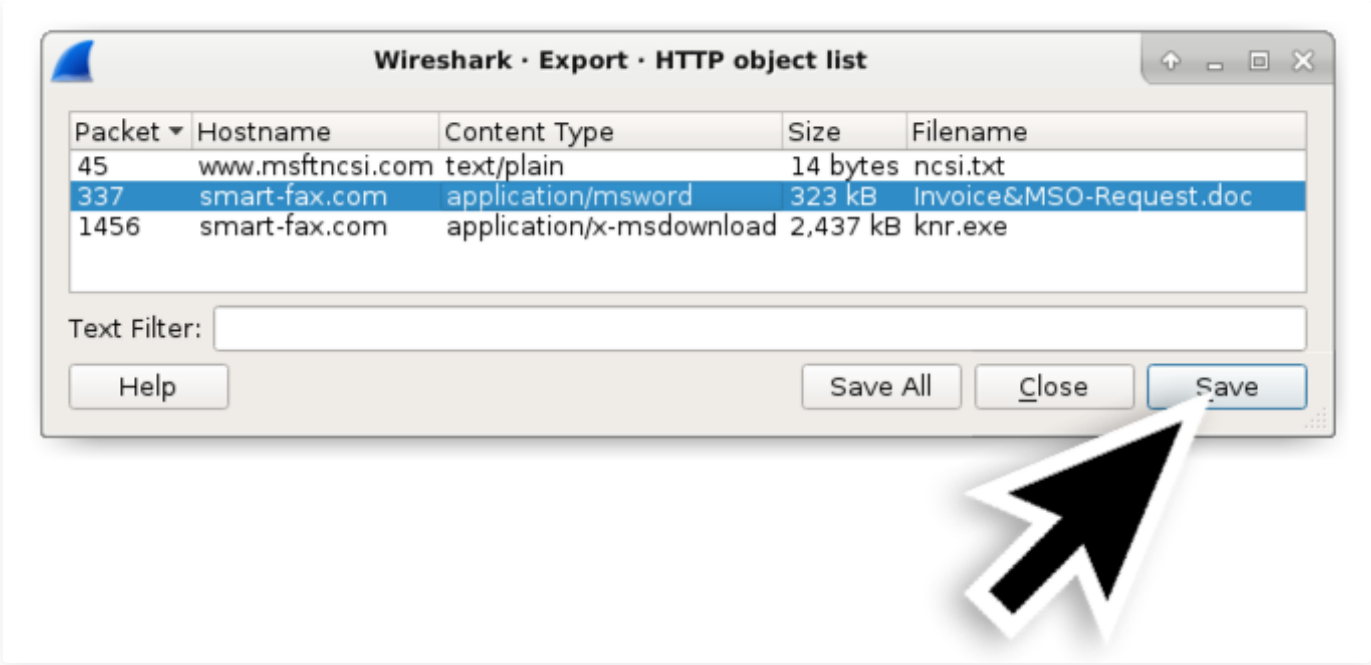
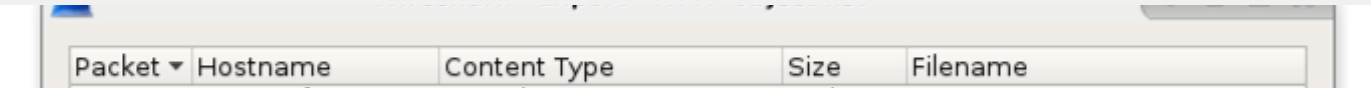


Figure 3. Saving the suspected Word document from the HTTP object list.



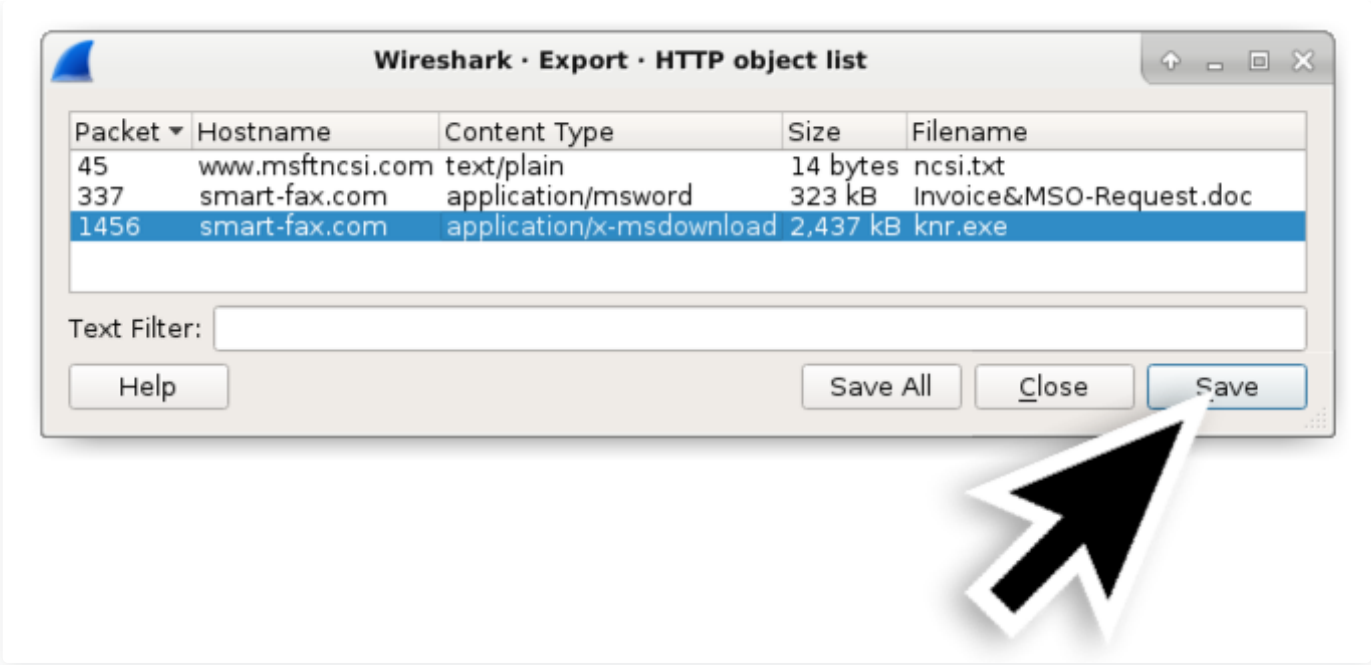


Figure 4. Saving the suspected Windows executable file from the HTTP object list.

Of note, the Content Type from the HTTP object list shows how the server identified the file in its HTTP response headers. In some cases, Windows executables are intentionally labeled as a different type of file in an effort to avoid detection. Fortunately, the first pcap in this tutorial is a very straight-forward example.

Still, we should confirm these files are what we think they are. In a MacBook or Linux environment, you can use a terminal window or command line interface (CLI) for the following commands:

- **file** *[filename]*
- **shasum -a 256** *[filename]*

The ***file*** command returns the type of file. The ***shasum*** command will return the file hash, in this case the SHA256 file hash. Figure 5 shows using these commands in a CLI on a Debian-based Linux host.

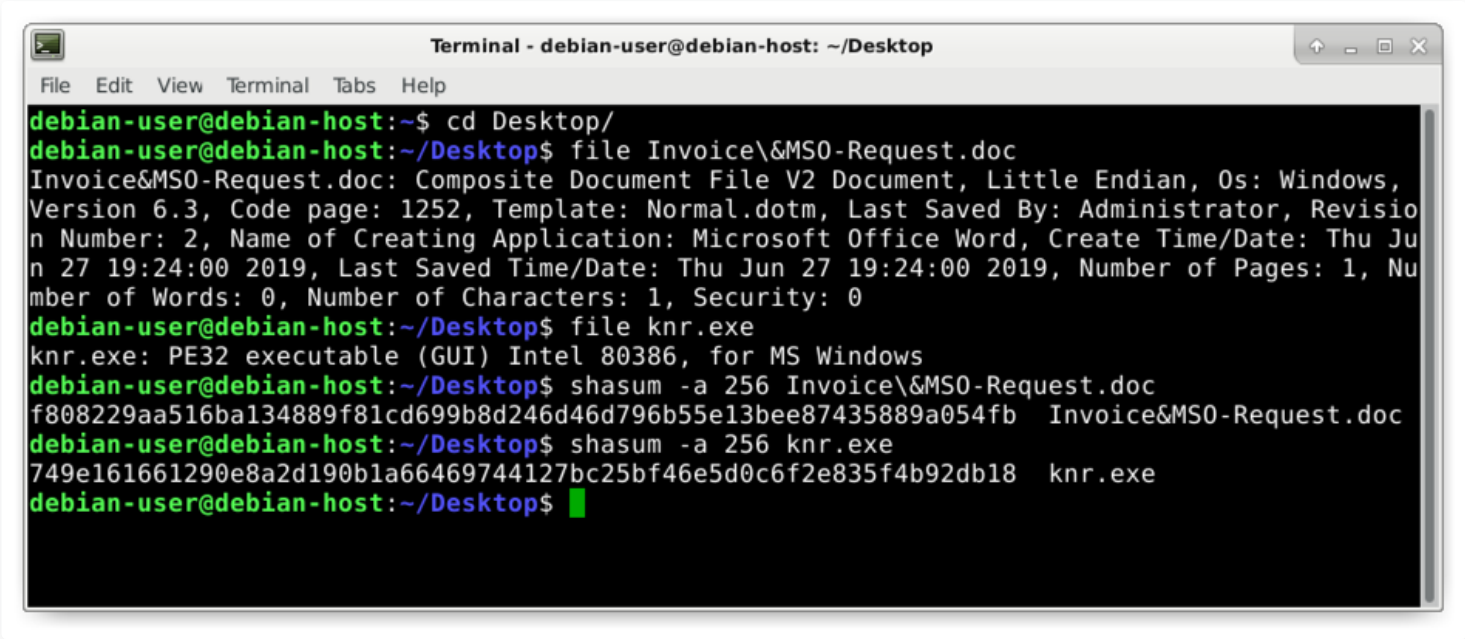


Figure 5. Determining the file type and hash of our two objects exported from the pcap.

The commands and their results from Figure 5 are listed below:

```
$ file Invoice\&MSO-Request.doc
```

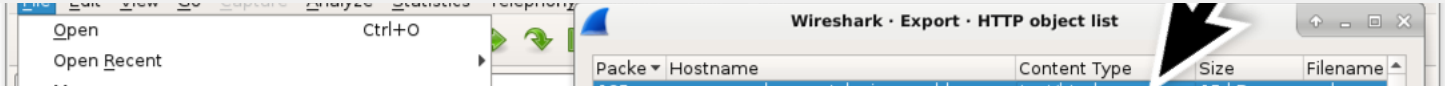
```
Invoice&MSO-Request.doc: Composite Document File V2 Document, Little Endian, Os:
Windows, Version 6.3, Code page: 1252, Template: Normal.dotm, Last Saved By:
Administrator, Revision Number: 2, Name of Creating Application: Microsoft Office
Word, Create Time/Date: Thu Jun 27 19:24:00 2019, Last Saved Time/Date: Thu Jun
```

```
27 19:24:00 2019, Number of Pages: 1, Number of Words: 0, Number of Characters:
1, Security: 0

$ file knr.exe

knr.exe: PE32 executable (GUI) Intel 80386, for MS Windows

$ shasum -a 256 Invoice\&MSO-Request.doc

f808229aa516ba134889f81cd699b8d246d46d796b55e13bee87435889a054fb Invoice&MSO-
Request.doc

$ shasum -a 256 knr.exe

749e161661290e8a2d190b1a66469744127bc25bf46e5d0c6f2e835f4b92db18 knr.exe
```

The information above confirms our suspected Word document is in fact a Microsoft Word document. It also confirms the suspected Windows executable file is indeed a Windows executable. We can check the SHA256 hashes against VirusTotal to see if these files are detected as malware. We could also do a Google search on the SHA256 hashes to possibly find additional information.

In addition to Windows executable or other malware files, we can also extract web pages. Our second pcap for this tutorial, ***extracting-objects-from-pcap-example-02.pcap*** (available here) contains traffic of someone entering login credentials on a fake PayPal login page.

When reviewing network traffic from a phishing site, we might want to see what the phishing web page looks like. We can extract the initial HTML page using the Export HTTP object menu as shown in Figure 6. Then we can view it through a web browser in an isolated environment as shown in Figure 7.



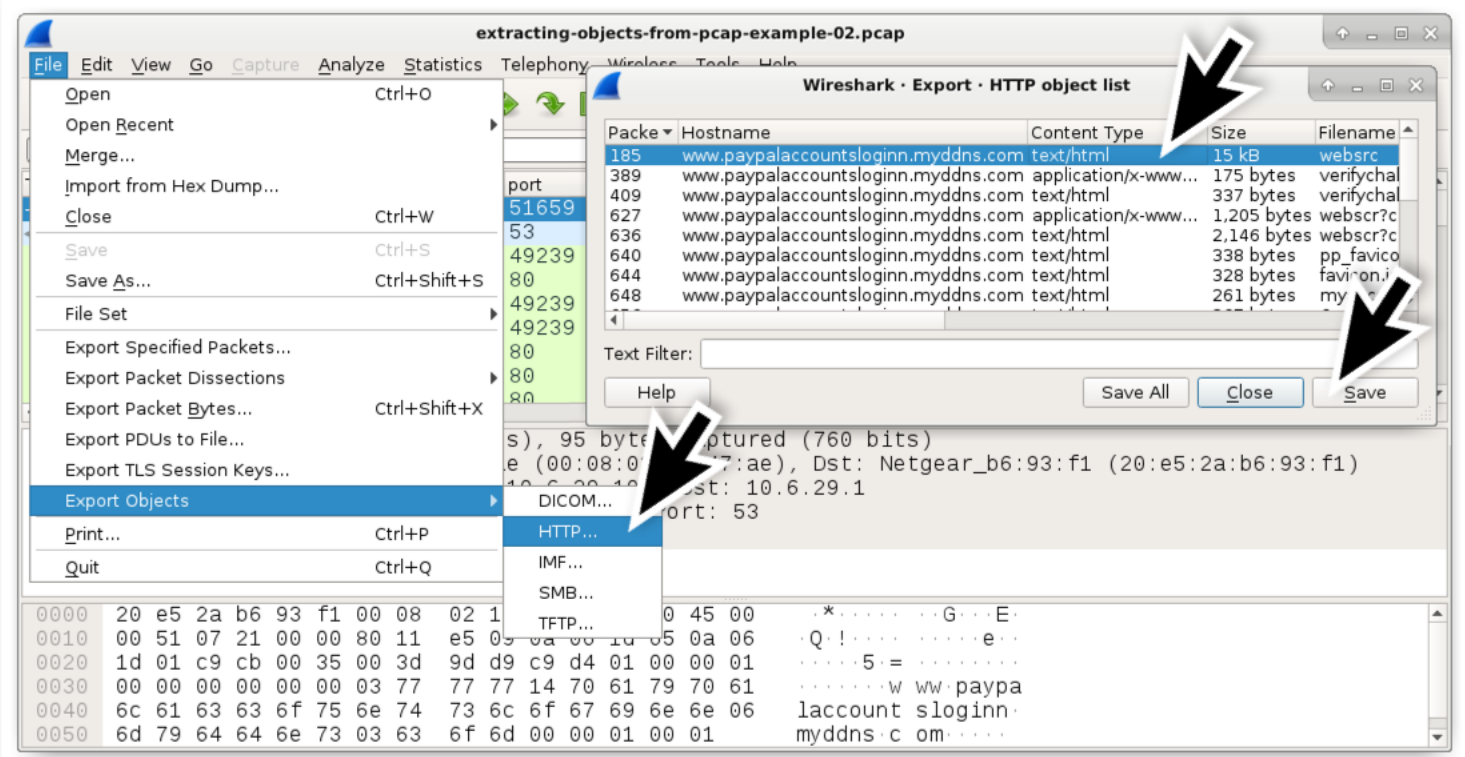
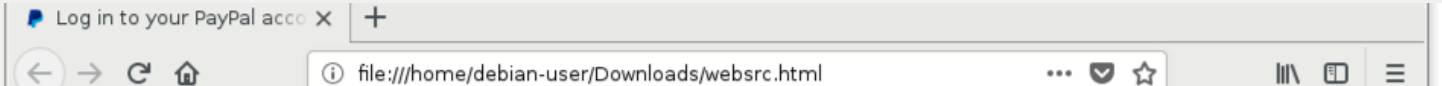


Figure 6. Exporting a fake PayPal login page from our second pcap.



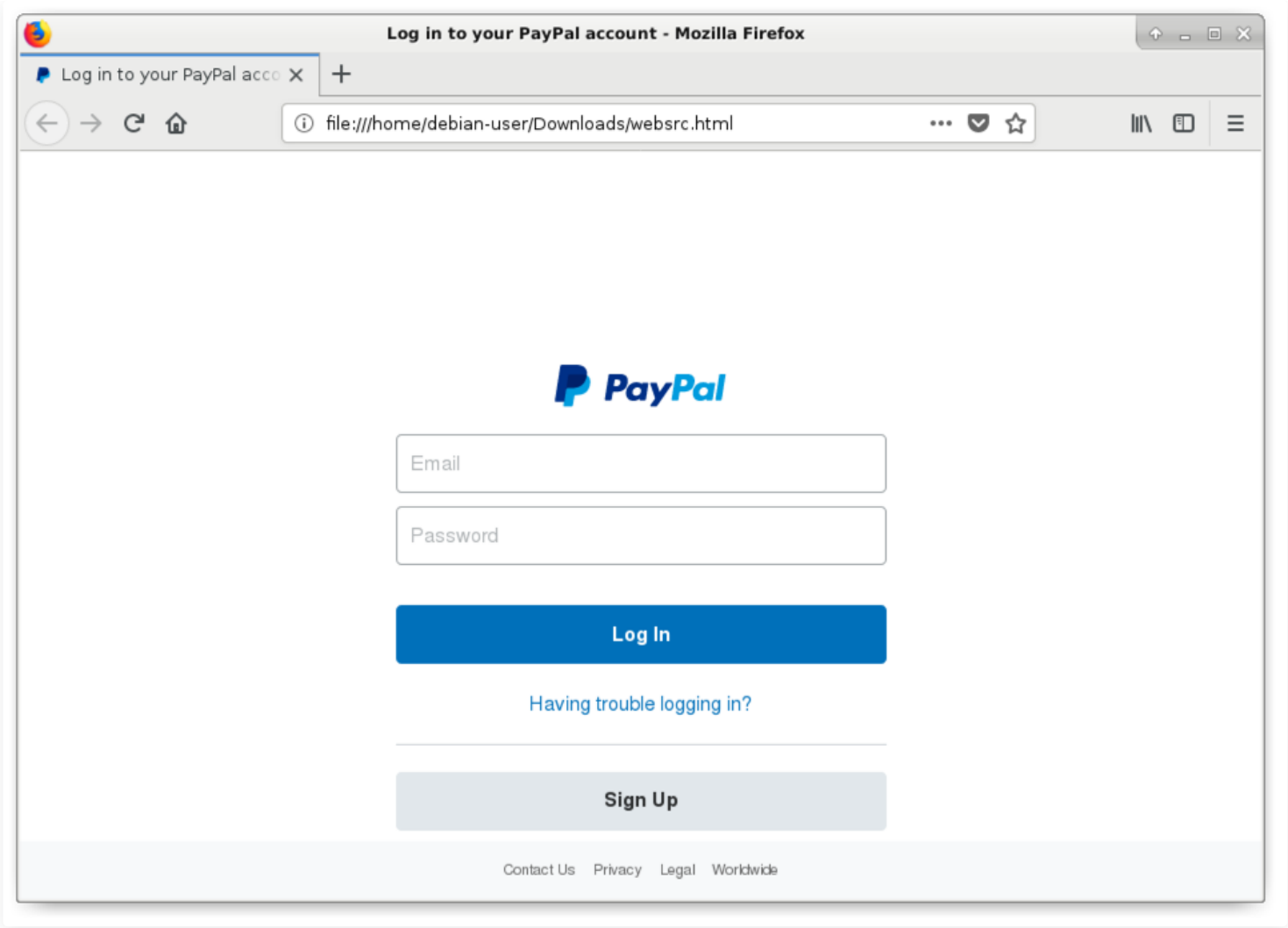
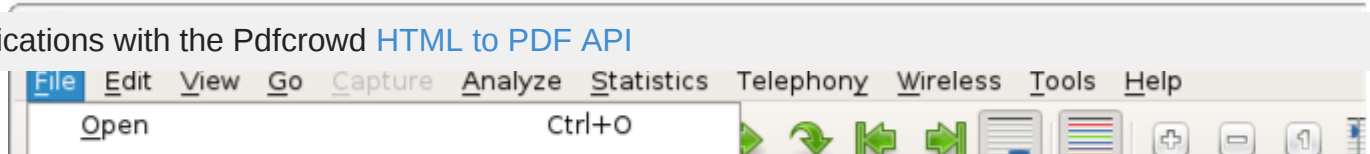


Figure 7. The exported fake PayPal login page viewed in a web browser.

Exporting Objects from SMB Traffic

Some malware uses Microsoft's Server Message Block (SMB) protocol to spread across an Active Directory (AD)-based network. A banking Trojan known as Trickbot added a worm module as early as July 2017 that uses an exploit based on EternalBlue to spread across a network over SMB. We continue to find indications of this Trickbot worm module today.

Our next pcap represents a Trickbot infection that used SMB to spread from an infected client at 10.6.26.110 to its domain controller at 10.6.26.6. The pcap, ***extracting-objects-from-pcap-example-03.pcap***, is available here. Open the pcap in Wireshark. Use the menu path ***File –> Export Objects –> SMB...*** as shown in Figure 8.



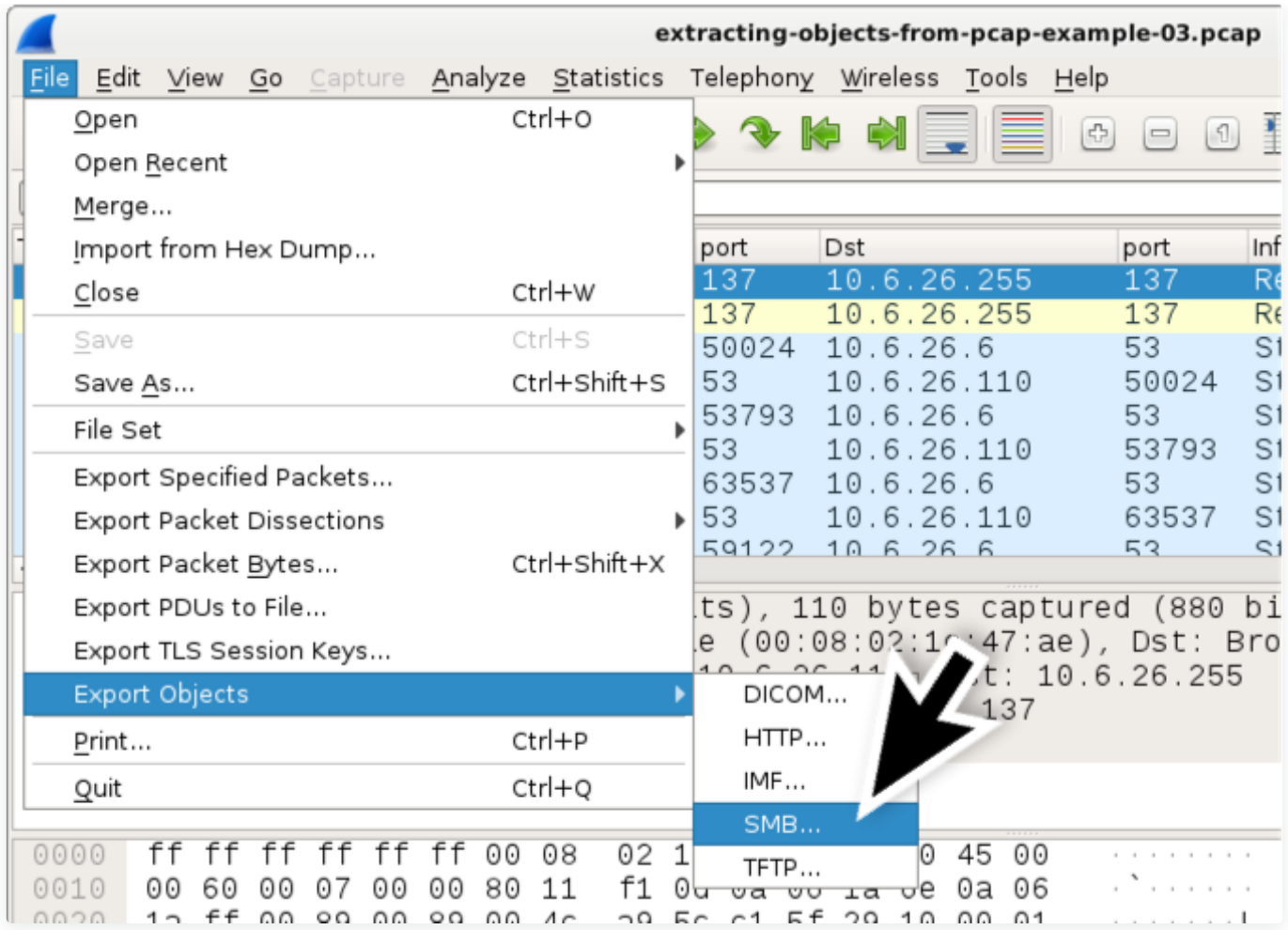
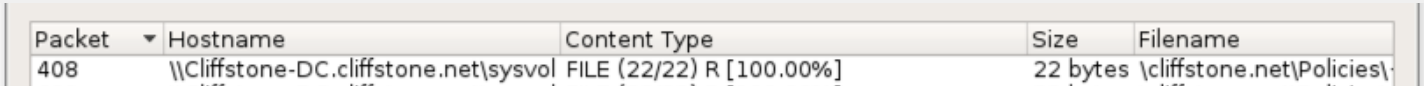


Figure 8. Getting to the Export SMB objects list.

This brings up an Export SMB object list, listing SMB objects you can export from the pcap as shown below in Figure 9.



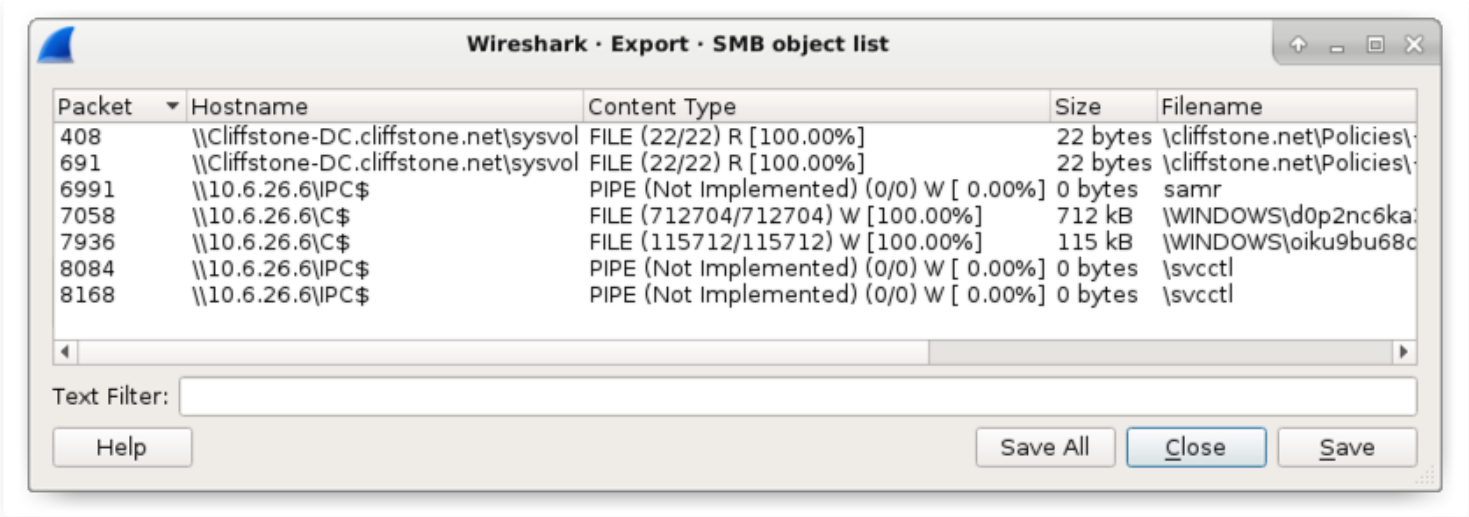


Figure 9. The export SMB object list.

Notice the two entries near the middle of the list with \\10.6.26.6\C$ as the Hostname. A closer examination of their respective Filename fields indicates these are two Windows executable files. See Table 1 below for details.

Packet number	Hostname	Content Type	Size	Filename
7058	\\10.6.26.6\C$	FILE (712704/712704) W [100.0%]	712 kB	\WINDOWS\d0p2nc6ka3f_fixhohlycj4ovqfcy_smchzo_ub83urjpphrwahjwhv_o5c0fvf6.exe
7936	\\10.6.26.6\C$	FILE (115712/115712) W [100.0%]	115 kB	\WINDOWS\oiku9bu68cxqenfmcsos2aek6t07_guuisgxhllixv8dx2eemqddnhyh46l8n_di.exe

Table 1. Data from the Export SMB objects list on the two Windows executable files.

In the Content Type column, we need [100.00%] to export a correct copy of these files. Any number less than 100 percent indicates there was some data loss in the network traffic, resulting in a corrupt or incomplete copy of the file. These Trickbot-related files from the pcap have SHA256 file hashes as shown in Table 2.

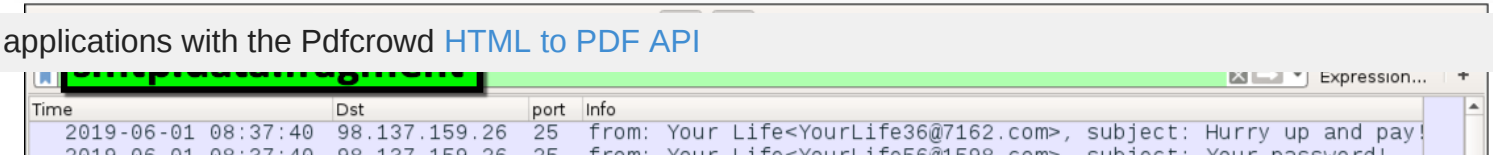
SHA256 hash	File size
59896ae5f3edcb999243c7bfdc0b17eb7fe28f3a66259d797386ea470c010040	712 kB
cf99990bee6c378cbf56239b3cc88276eec348d82740f84e9d5c343751f82560	115 kB

Table 2. SHA256 file hashes for the Windows executable files.

Exporting Emails from SMTP Traffic

Certain types of malware are designed to turn an infected Windows host into a spambot. These spambots send hundreds of spam messages or malicious emails every minute. In some cases, the messages are sent using unencrypted SMTP, and we can export these messages from a pcap of the infection traffic.

One such example is from our next pcap, ***extracting-objects-from-pcap-example-04.pcap*** (available here). In this pcap, an infected Windows client sends sextortion spam. Open the pcap in Wireshark, filter on ***smtp.data.fragment***, and you should see 50 examples of subject lines as shown in Figure 10. This happened in five seconds of network traffic from a single infected Windows host.



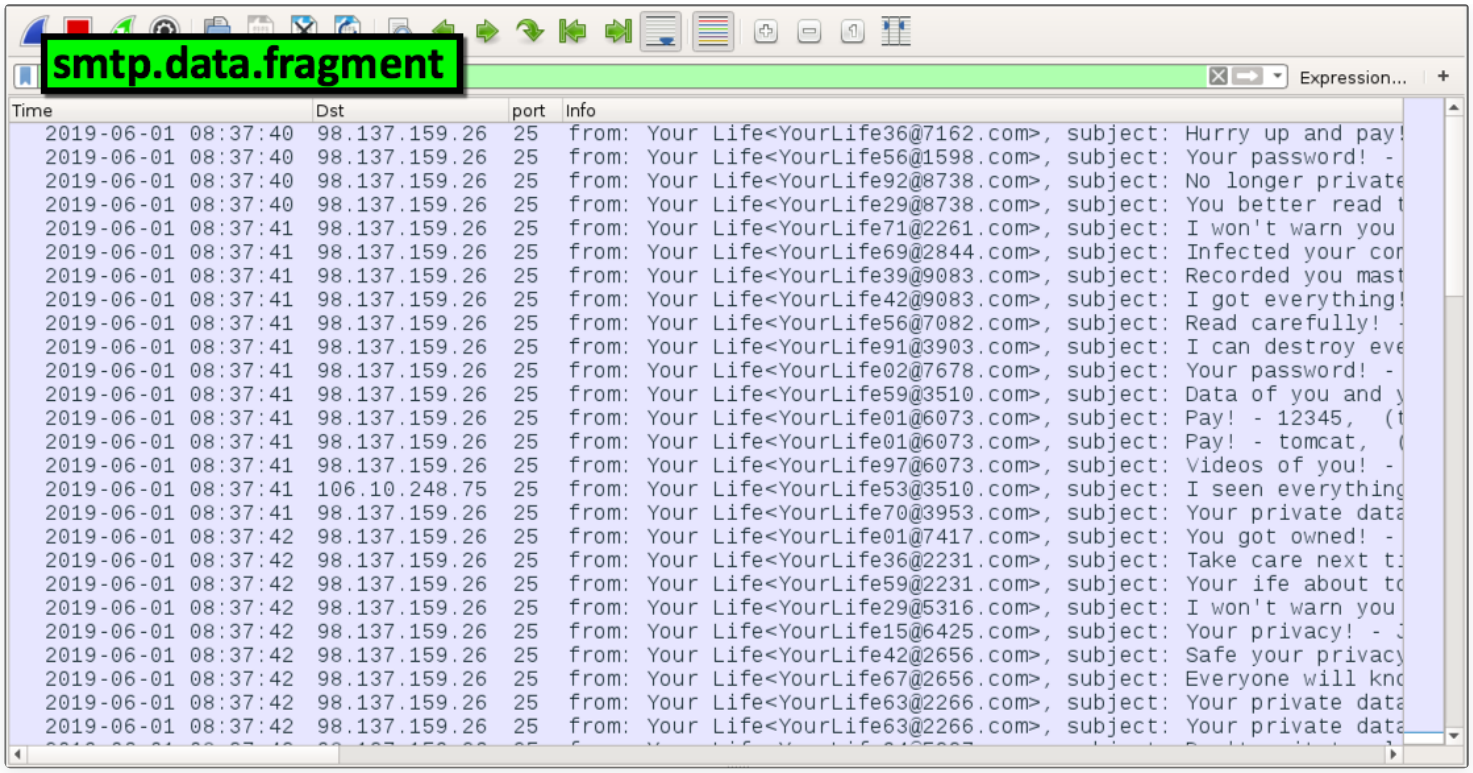
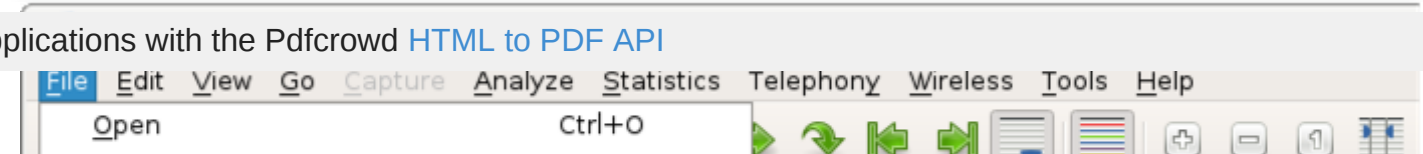


Figure 10. Filtering for email senders and subject lines in Wireshark.

You can export these messages using the menu path ***File –> Export Objects –> IMF...*** as shown in Figure 11. IMF stands for Internet Message Format, which is saved as a name with an .eml file extension.



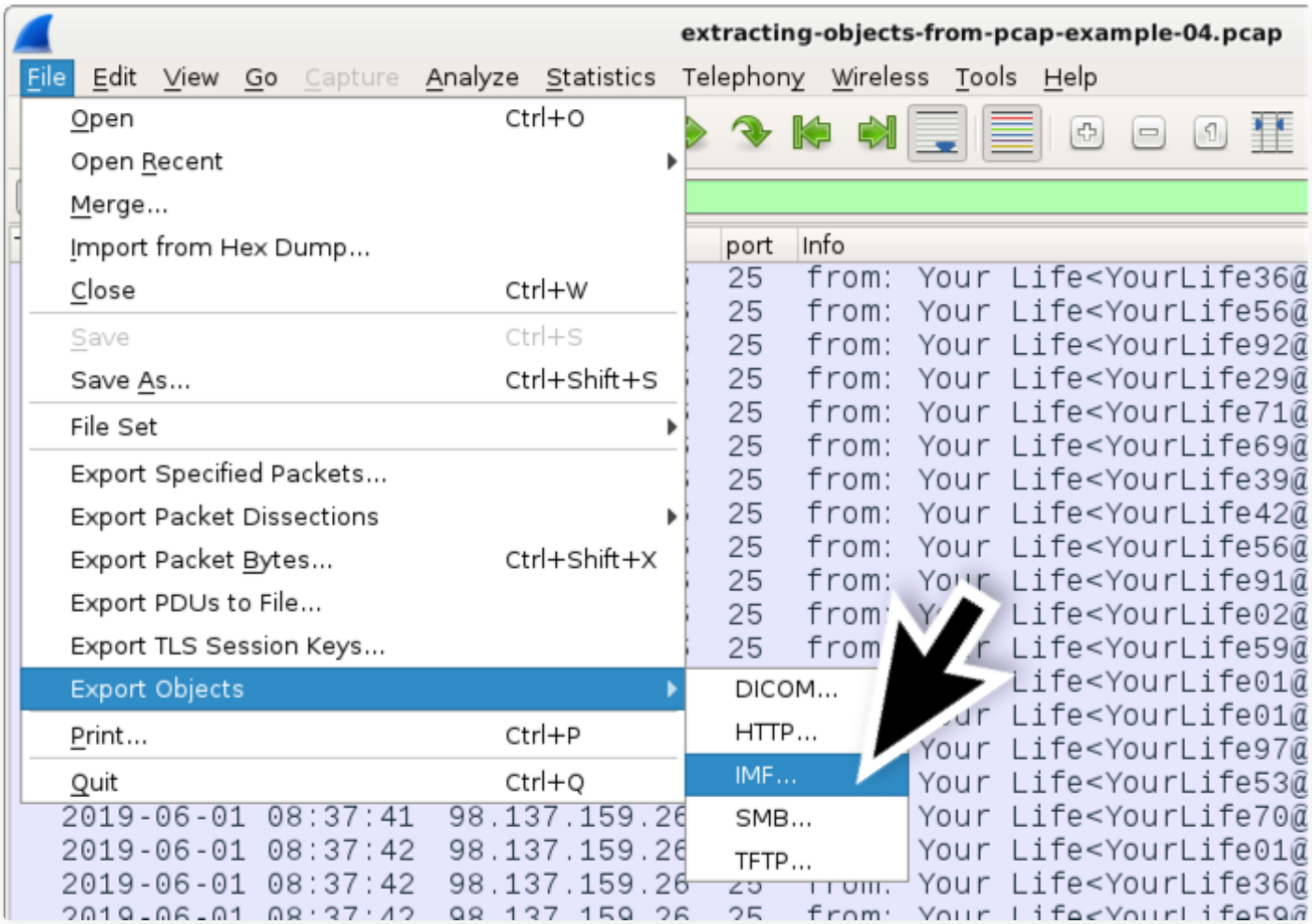
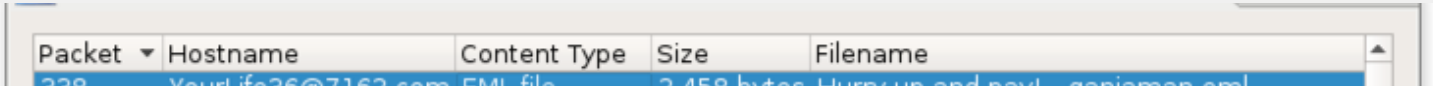


Figure 11. Exporting emails from a pcap in Wireshark.

The sextortion spam messages are all listed with an .eml file extension in the IMF object list as shown in Figure 12.



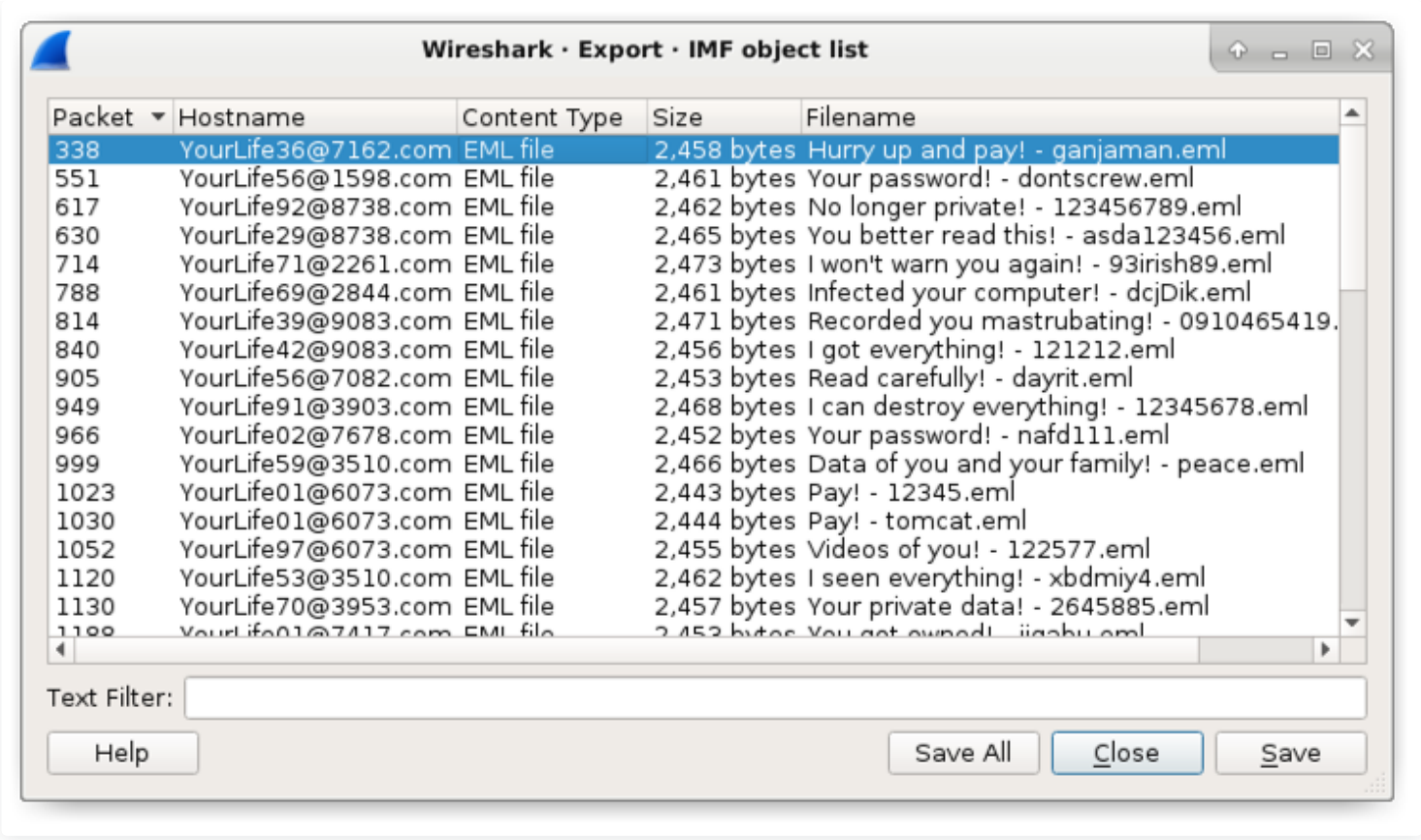
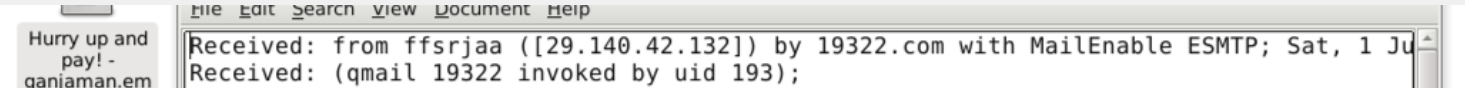


Figure 12. List of spam messages in the IMF object list.

After they are exported, these .eml files can be reviewed with an email client like Thunderbird, or they can be examined in a text editor as shown in Figure 13.



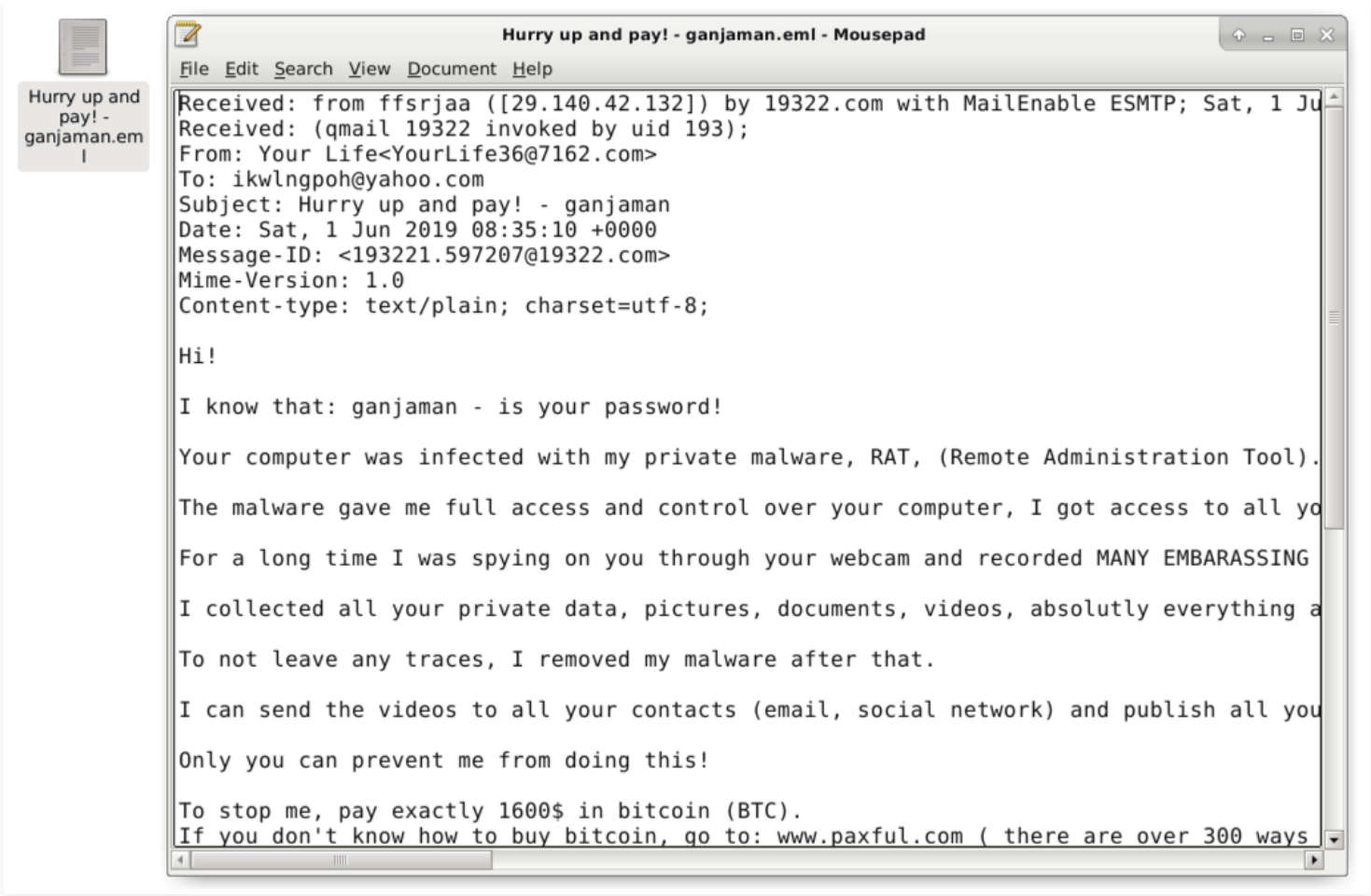


Figure 13. Using a text editor to view an .eml file exported from the pcap.

Exporting files from FTP Traffic

Some malware families use FTP during malware infections. Our next pcap has malware executables retrieved from an FTP server followed by information from the infected Windows host sent back to the same FTP server.

The next pcap is ***extracting-objects-from-pcap-example-05.pcap*** and is available here. Open the pcap in Wireshark. Filter on ***ftp.request.command*** to review the FTP commands as shown in Figure 14. You should find a username (USER) and password (PASS) followed by requests to retrieve (RETR) five Windows executable files: ***q.exe***, ***w.exe***, ***e.exe***, ***r.exe***, and ***t.exe***. This is followed by requests to store (STOR) html-based log files back to the same FTP server approximately every 18 seconds.

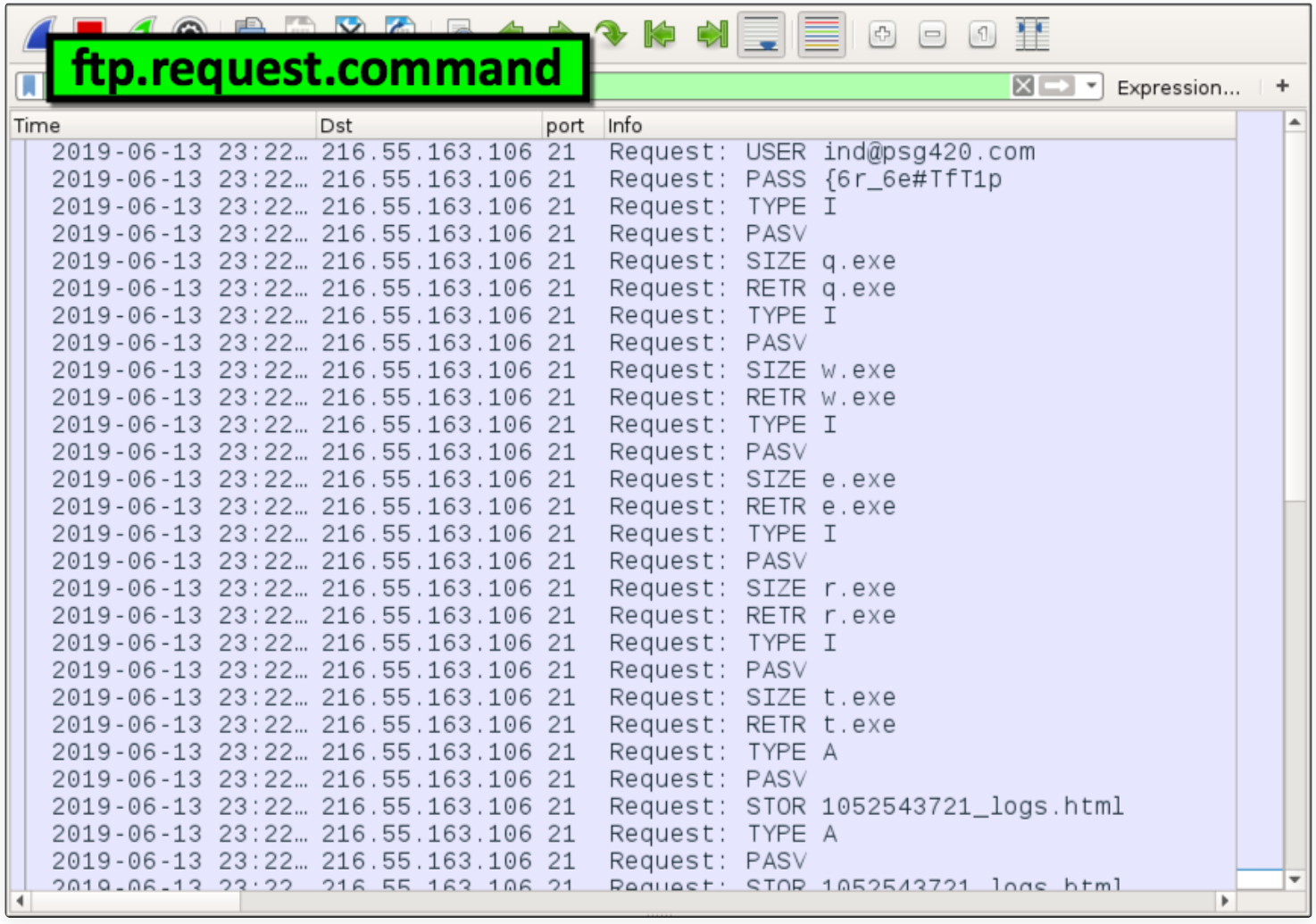


Figure 14. Filtering for FTP requests in Wireshark.

Now that we have an idea of the files that were retrieved and sent, we can review traffic from the FTP data channel using a filter for ***ftp-data*** as shown in Figure 15.

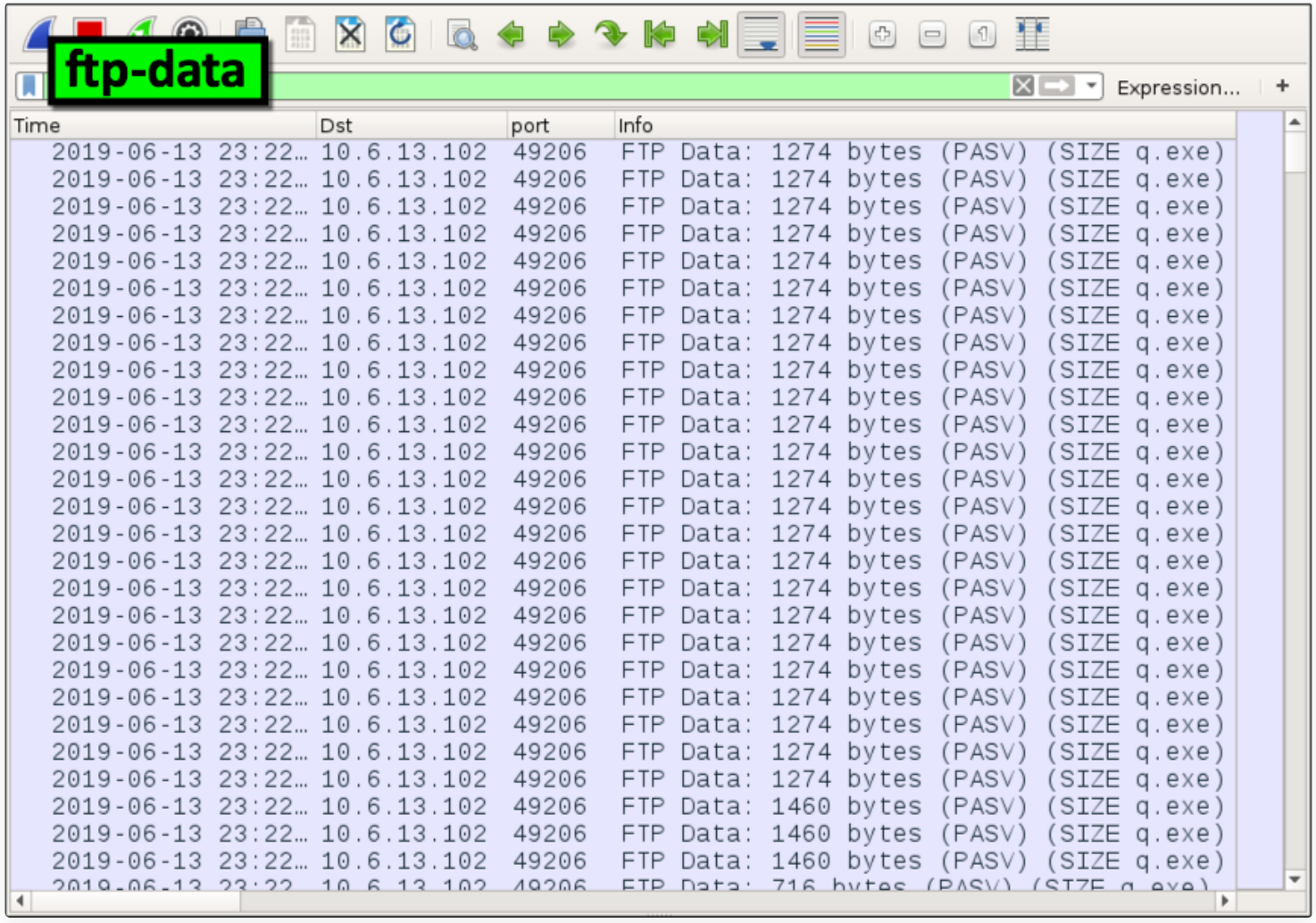


Figure 15. Filtering on FTP data traffic in Wireshark.

We cannot use the ***Export Objects*** function in Wireshark to export these objects. However, we can follow the TCP stream from the data channels for each. Left-click on any of the lines that end with (SIZE q.exe) to select one of the TCP segments. Then right-click to bring up a menu and select the menu path for ***Follow –> TCP stream*** as shown in Figure 16.

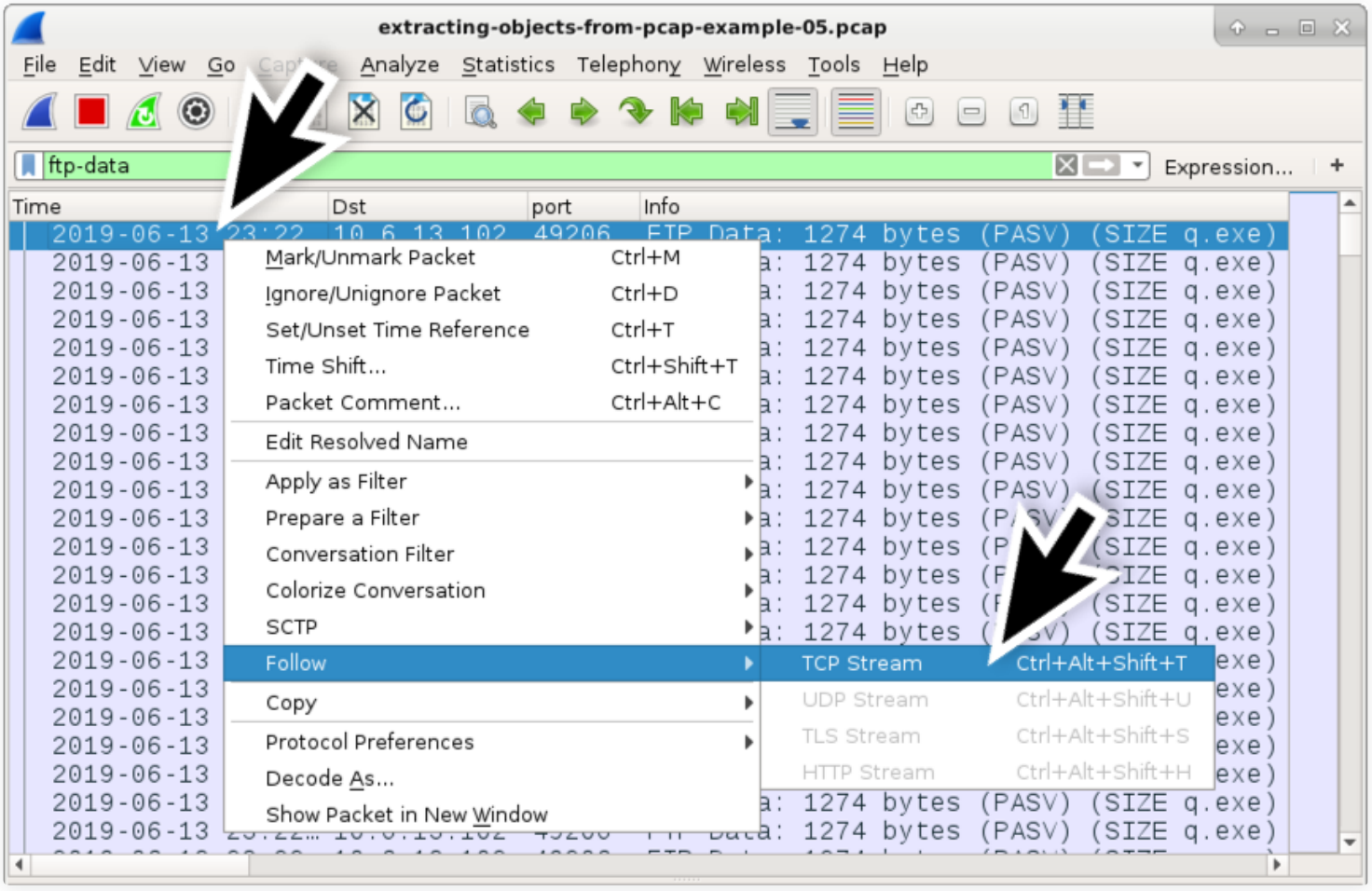


*Figure 16. Following the TCP stream of an FTP data channel for **q.exe**.*

This will bring up the TCP stream for ***q.exe*** over the FTP data channel. Near the bottom of the window is a button-style menu labeled “Show and save data as” which defaults to ASCII as shown in Figure 17. Click on the menu and select “Raw” as shown in Figure 18.

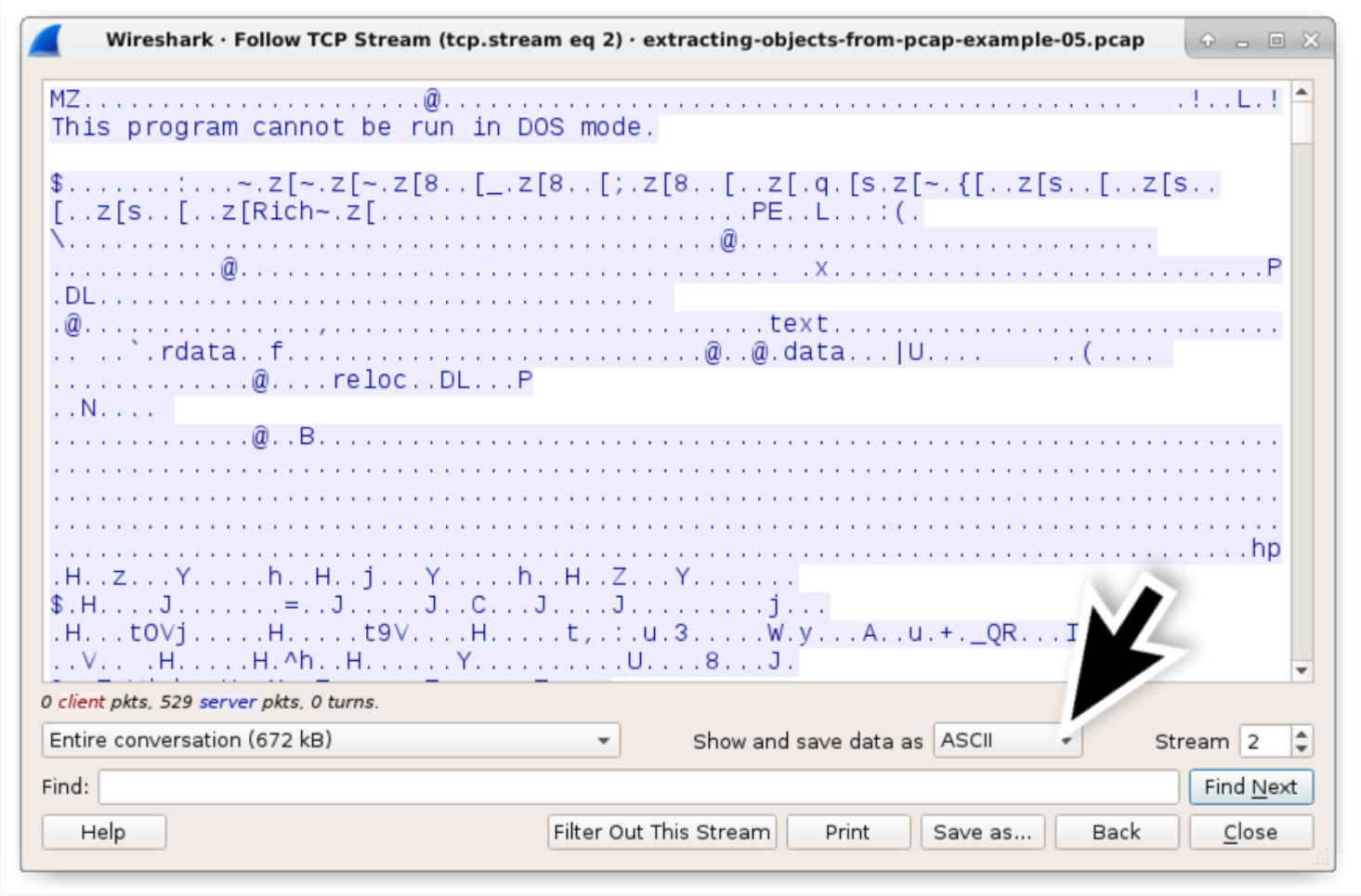
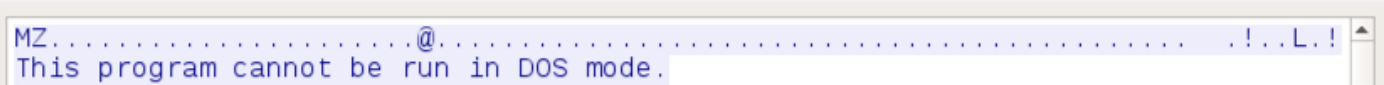


*Figure 17. The TCP stream window for **q.exe**. Note the “Show and save data as” button-style menu.*



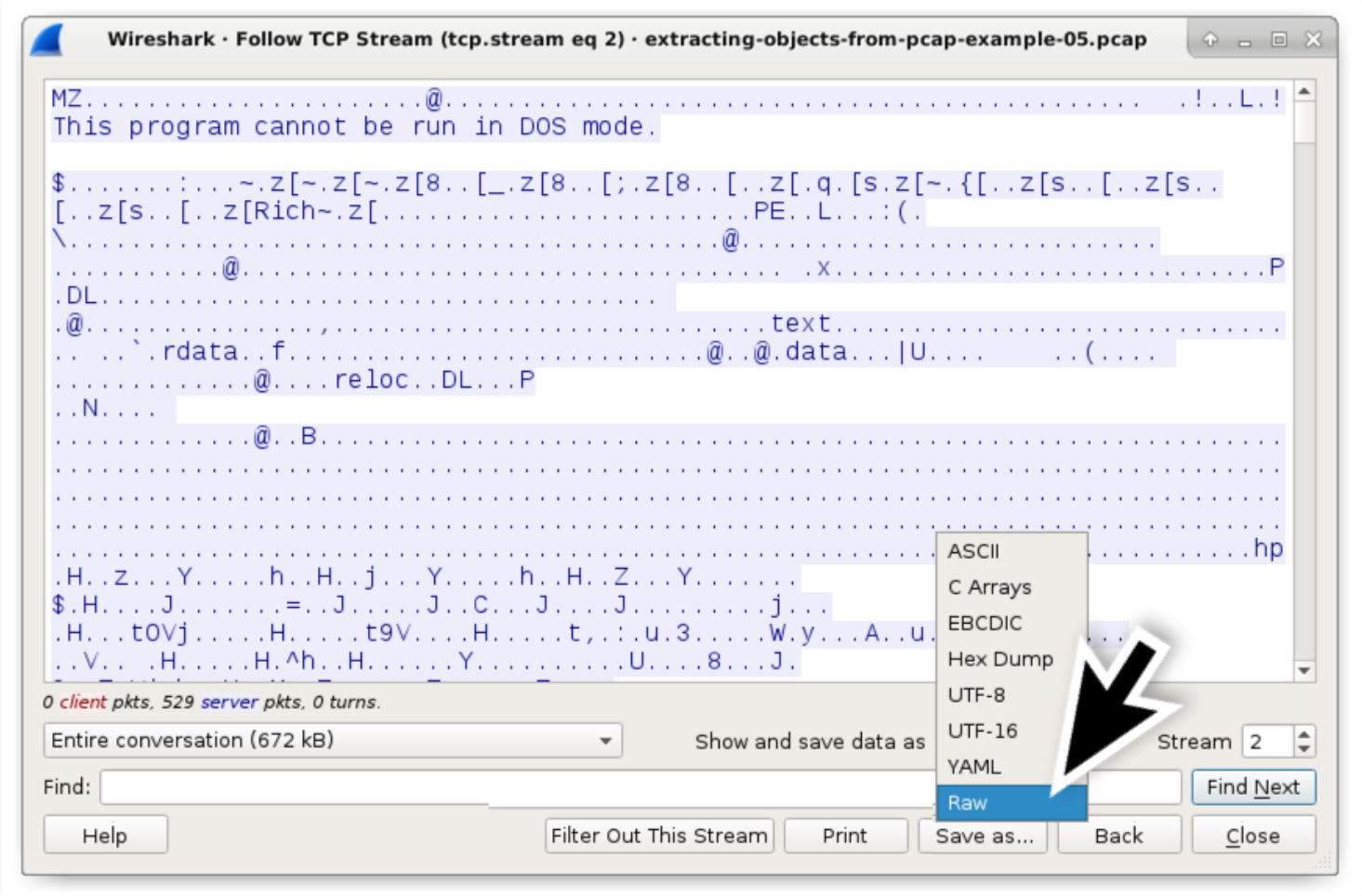


Figure 18. Selecting "Raw" from the "Show and save data as" menu.

The window should now show hexadecimal characters instead of ASCII as shown in Figure 19. Use the ***Save as...*** button near the bottom of the window to save this as a raw binary, also shown in Figure 19.

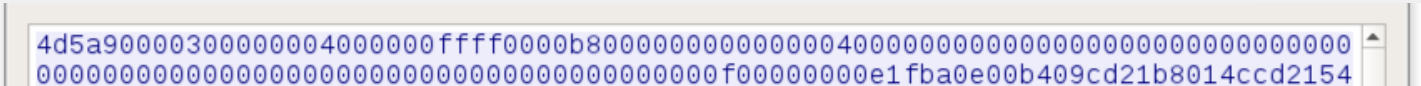


Figure 19. Saving the data from a TCP stream as a raw binary.

Save the file as ***q.exe***. In a Linux or similar CLI environment, confirm this is a Windows executable file and get the SHA256 hash as shown below.

```
$ file q.exe
```

```
q.exe: PE32 executable (GUI) Intel 80386, for MS Windows
```

```
$ shasum -a 256 q.exe
```

```
ca34b0926cdc3242bbfad1c4a0b42cc2750d90db9a272d92cfb6cb7034d2a3bd q.exe
```

The SHA256 hash shows a high detection rate as malware on VirusTotal. Follow the same process for the other .exe files in the pcap:

- Filter on ***ftp-data***
- Follow the TCP stream for a TCP segment with the name of your file in the Info column
- Change “Show and save data as” to “Raw”
- Use the “Save as…” button to save the file
- Check to make sure your saved file is, in fact, a Windows executable file.

This should give you the following files as shown below in Table 3.

SHA256 hash	File name
32e1b3732cd779af1bf7730d0ec8a7a87a084319f6a0870dc7362a15ddbd3199	e.exe
ca34b0926cdc3242bbfad1c4a0b42cc2750d90db9a272d92cfb6cb7034d2a3bd	q.exe
4ebd58007ee933a0a8348aee2922904a7110b7fb6a316b1c7fb2c6677e613884	r.exe
10ce4b79180a2ddd924fdc95951d968191af2ee3b7dfc96dd6a5714dbeae613a	t.exe
08eb941447078ef2c6ad8d91bb2f52256c09657ecd3d5344023edccf7291e9fc	w.exe

Table 3. Executable files from the FTP traffic.

We have to search more precisely when exporting the HTML files sent from the infected Windows host back to the FTP server. Why? Because the same file name is used each time. Filter on ***ftp.request.command***, and scroll to the end. We can see the same file name used to store (STOR) stolen data to the FTP server as an HTML file as shown in Figure 20.

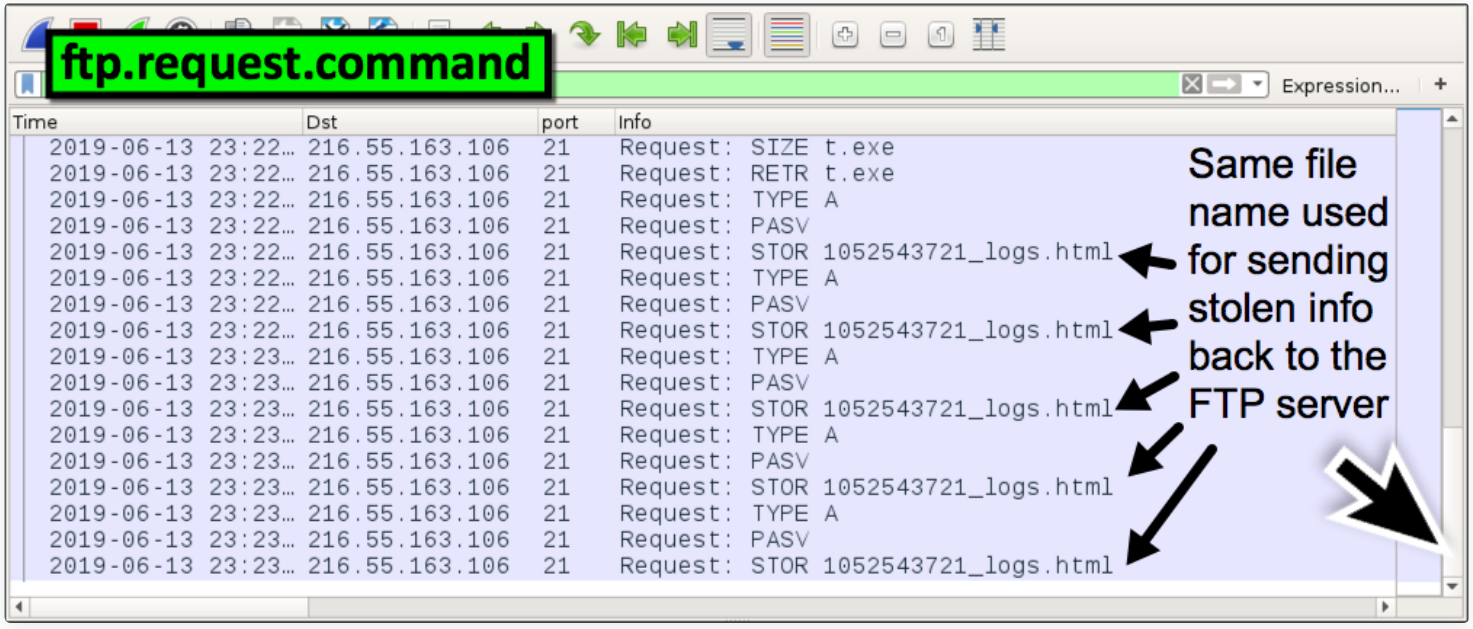
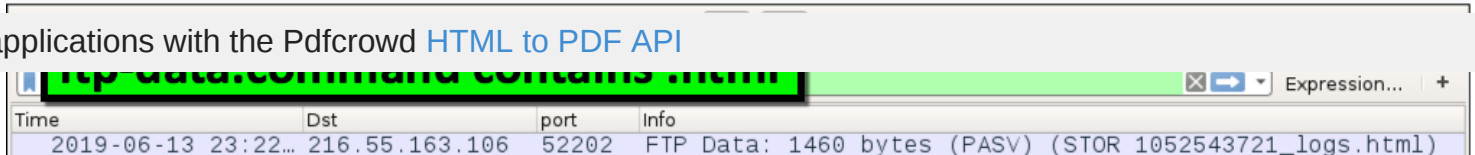


Figure 20. The same file name used for sending stolen info back to the FTP server.

To see the associated files sent over the ftp data channel, use the filter ***ftp-data.command contains .html*** as shown in Figure 21.



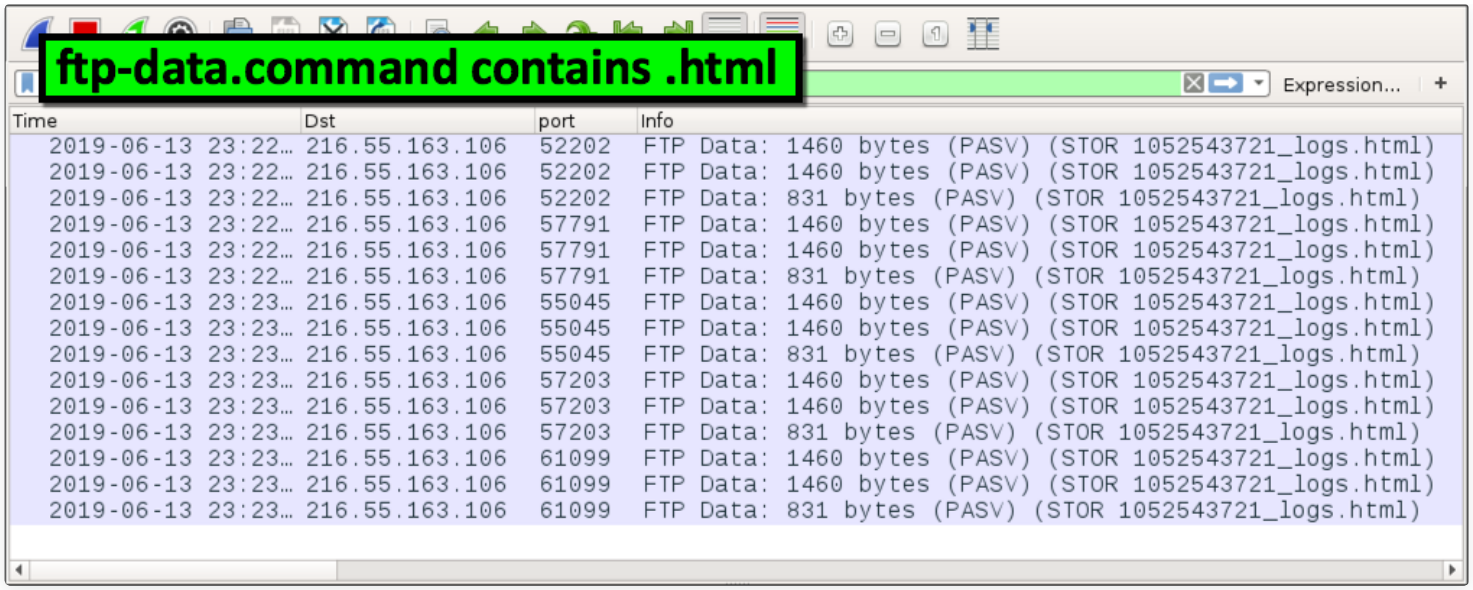


*Figure 21. Filtering on files with **.html** in the file name over the FTP data channel.*

In Figure 21, the destination port changes each time the file is stored (STOR) to the FTP server. The first time has TCP port 52202. The second time has TCP port 57791. The third time has TCP port 55045. The fourth time has 57203. And the fifth time has 61099.

We use the same process as before. Instead of focusing on the file names, focus on the TCP ports. Follow the TCP stream for any of the TCP segments using port 52202. In the TCP stream window, change “Show and save data as” to “Raw.” Then save the file. Do the same for the HTML file over TCP port 57791.

If you do this for all five HTML files, you’ll find they are the same exact file. These text-based HTML files contain data about the infected Windows host, including any passwords found by the malware.

Summary

Using the methods outlined in this tutorial, we can extract various objects from a pcap using Wireshark. This can be extremely helpful when you need to examine items from network traffic.

For more help using Wireshark, please see our previous tutorials:

- Customizing Wireshark – Changing Your Column Display
- Using Wireshark: Display Filter Expressions
- Using Wireshark: Identifying Hosts and Users