Wireshark Tutorial: Decrypting HTTPS Traffic

By Brad Duncan

Published: 2020-08-21 · Archived: 2026-04-05 21:16:03 UTC

Executive Summary

This tutorial is designed for security professionals who investigate suspicious network activity and review packet captures (pcaps) of the traffic. The instructions assume you are familiar with Wireshark, and it focuses on Wireshark version 3.x.

When reviewing suspicious network activity, we often run across encrypted traffic. Why? Because most websites use the Hypertext Transfer Protocol Secure (HTTPS) protocol. But like most websites, various types of malware also use HTTPS. When reviewing pcaps from malware activity, it's very helpful to know what's contained within post-infection traffic.

This Wireshark tutorial describes how to decrypt HTTPS traffic from a pcap in Wireshark. Decryption is possible with a text-based log containing encryption key data captured when the pcap was originally recorded. With this key log file, we can decrypt HTTPS activity in a pcap and review its contents.

Today, we will examine HTTPS activity from a Dridex malware infection.

Note: Our instructions assume you have customized your Wireshark column display as previously described in "Customizing Wireshark – Changing Your Column Display.".

Here is a Github repository with a ZIP archive containing the pcap and a key log file used for this tutorial.

Warning: The pcap used for this tutorial contains Windows-based malware. There is a risk of infection if using a Windows computer. We recommend you review this pcap in a non-Windows environment like BSD, Linux or macOS if at all possible.

The Context Behind Encrypted Traffic

In the mid- to late-1990s, the most common protocol used by websites was Hypertext Transfer Protocol (HTTP), which generated unencrypted web traffic. However, as security became an increasing concern, websites started switching to HTTPS, and now we rarely see HTTP traffic from web browsing.

HTTPS is essentially an encrypted communications tunnel containing HTTP traffic. These tunnels first used Secure Sockets Layer (SSL) as an encryption protocol. Today most HTTPS traffic uses Transport Layer Security (TLS).

HTTPS Web Traffic

HTTPS traffic often reveals a domain name. For example, when viewing https://www.wireshark.org in a web browser, a pcap would show www.wireshark.org as the server name for this traffic when viewed in a customized Wireshark column display. Unfortunately, we don't know other details like the actual URL or data returned from the server. Following the Transmission Control Protocol (TCP) stream from a pcap will not reveal the content of this traffic because it is encrypted.

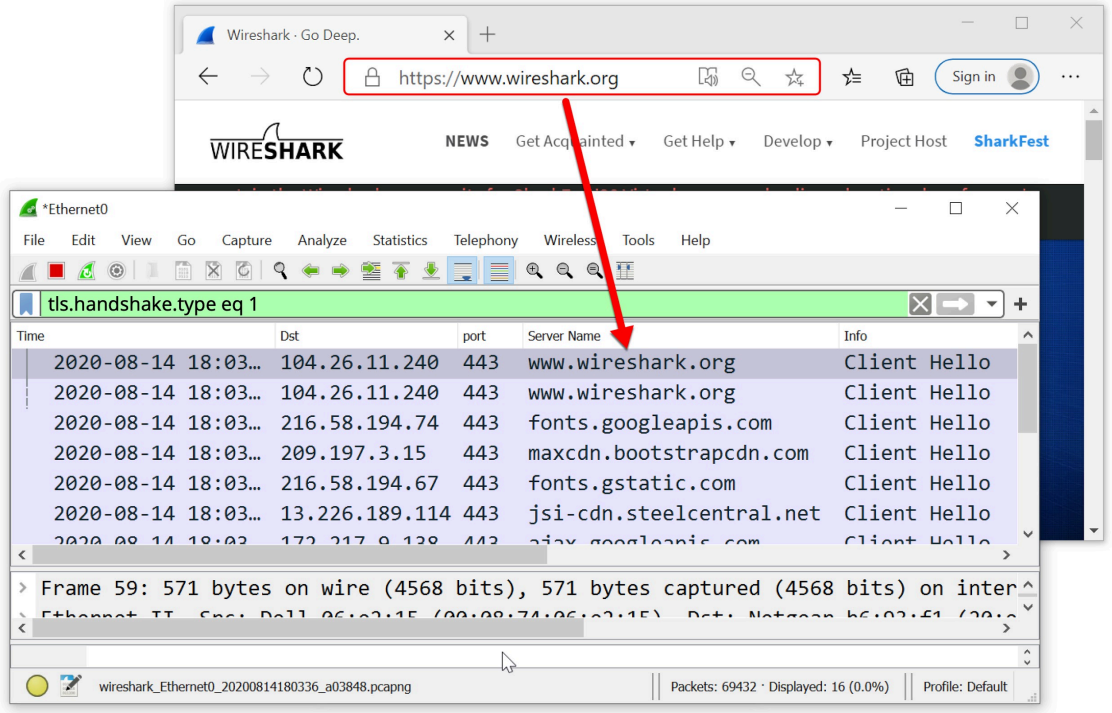

Figure 1. Traffic from HTTPS traffic to www.wireshark.org.

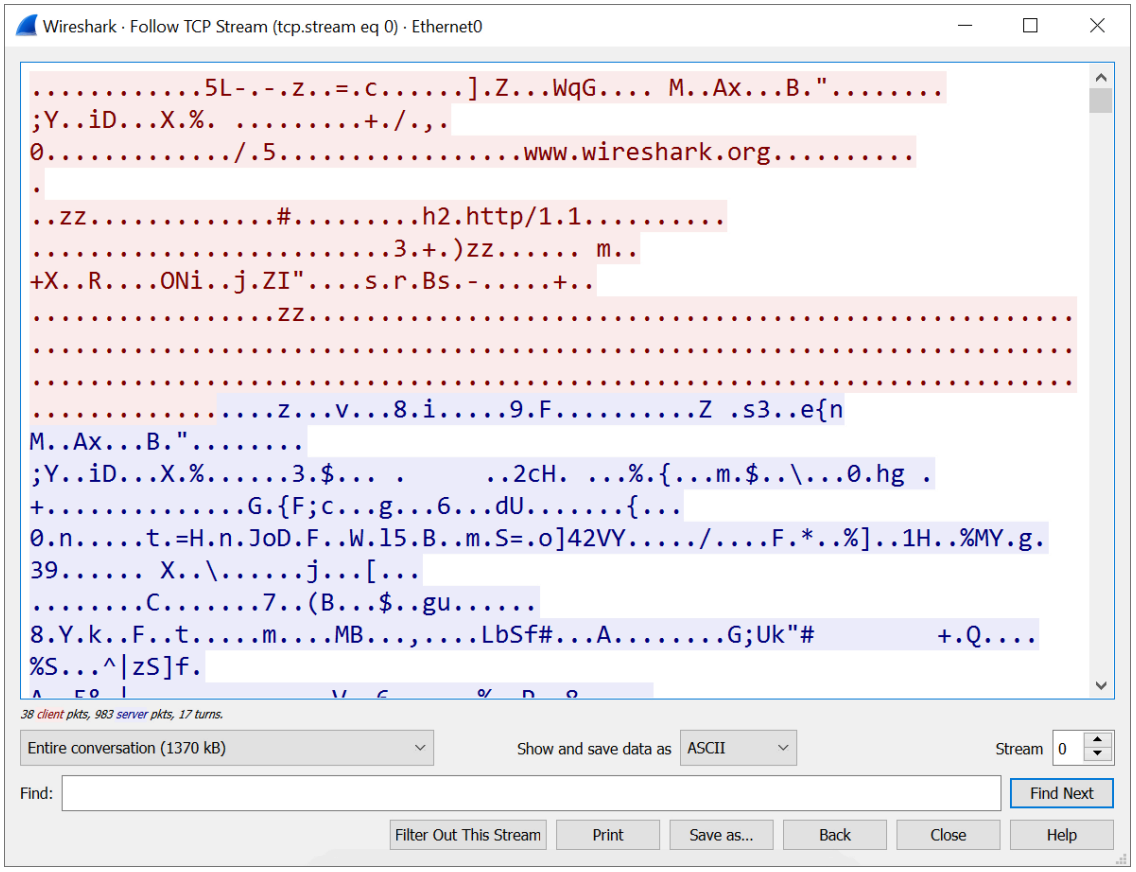

Figure 2. TCP stream of HTTPS traffic to and from server at www.wireshark.org.

Encryption Key Log File

An encryption key log is a text file. An example is shown in Figure 3.

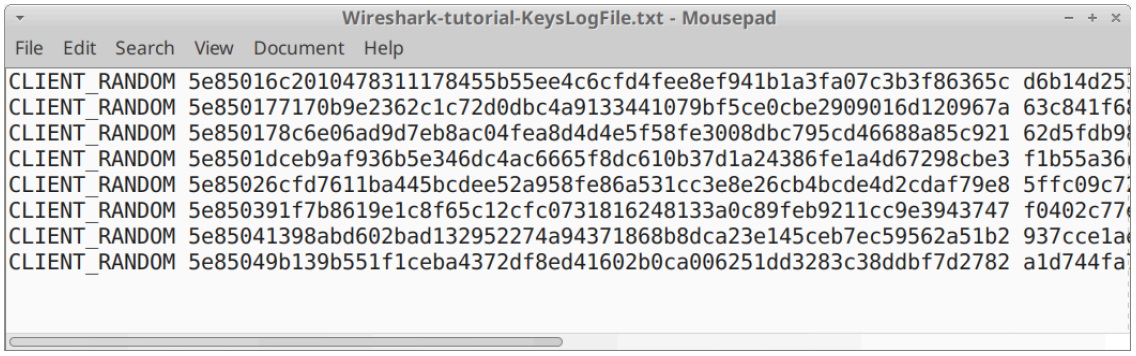

Figure 3. The key log file used in this tutorial.

These logs are created using a Man in the Middle (MitM) technique when the pcap is originally recorded. If no such file was created when the pcap was recorded, you cannot decrypt HTTPS traffic in that pcap.

Example of a Pcap With a Key Log File

A password-protected ZIP archive containing the pcap and its key log file is available at this Github repository. Go to the Github page, click on the ZIP archive entry, then download it as shown in Figures 4 and 5. Of note, the pcap contained in this ZIP archive provides access to a Windows-based malware sample when decrypted with the key

log. As always, we recommend you exercise caution and follow steps from this tutorial in a non-Windows environment.

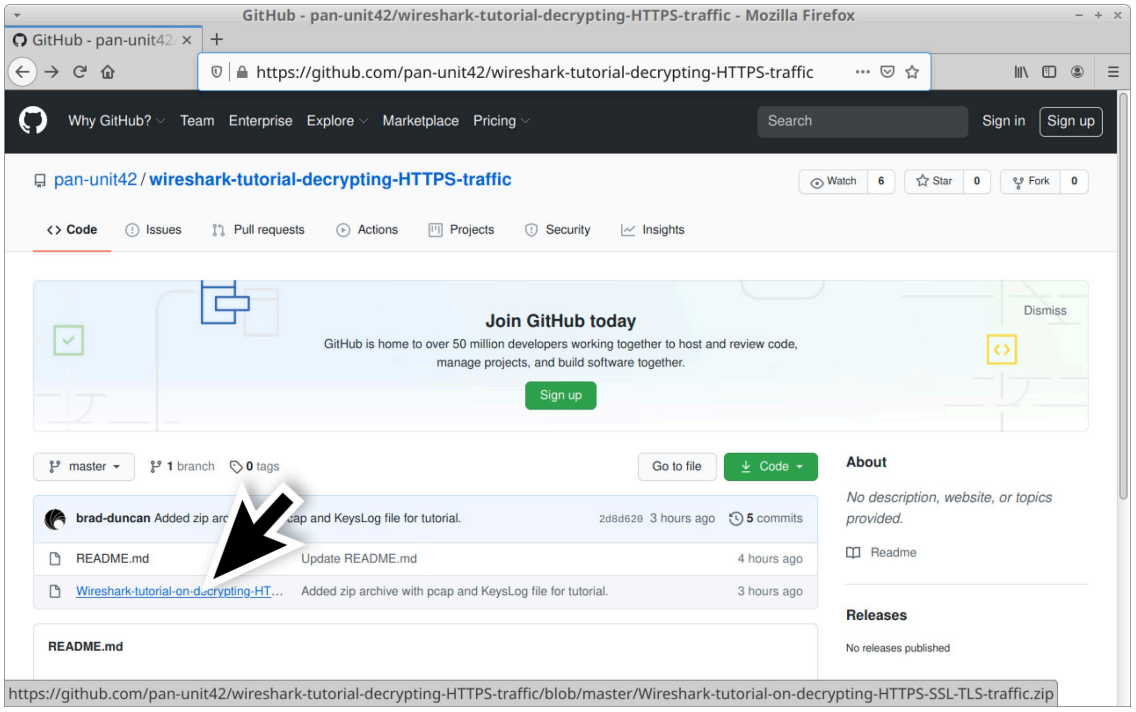

Figure 4. Github repository with link to ZIP archive used for this tutorial.

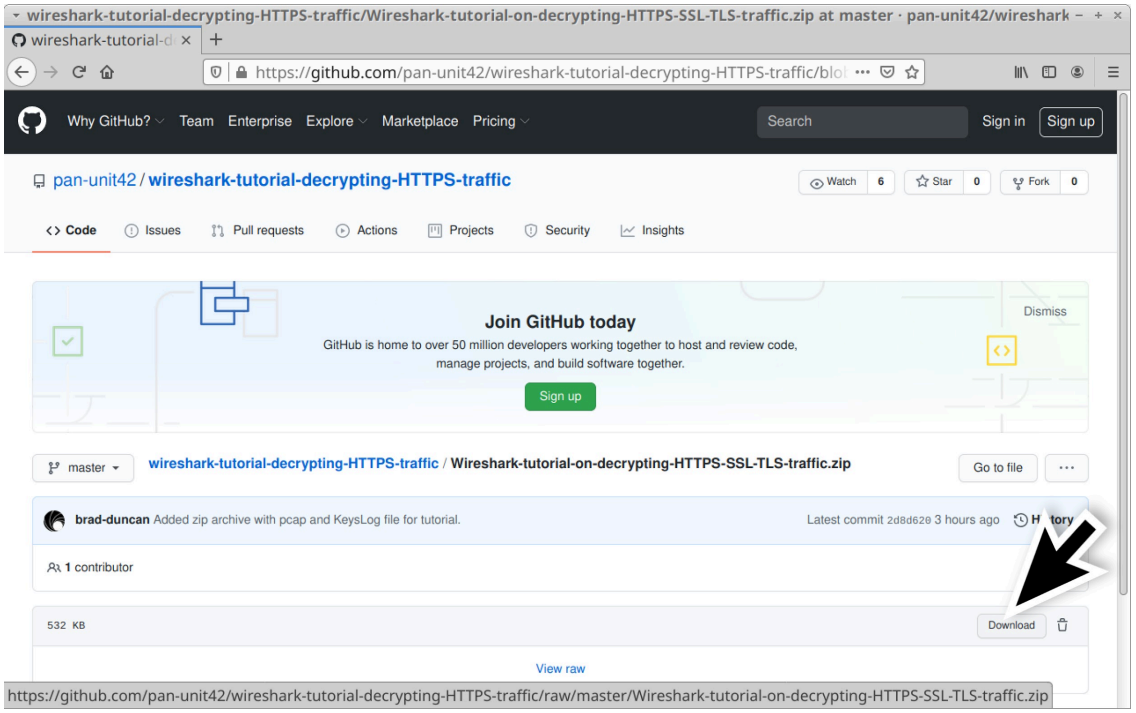

Figure 5. Downloading the ZIP archive for this tutorial.

Use ***infected*** as the password to extract the pcap and key log file from the ZIP archive. This will provide two files as shown in Figure 6:

- Wireshark-tutorial-KeysLogFile.txt
- Wireshark-tutorial-on-decrypting-HTTPS-SSL-TLS-traffic.pcap

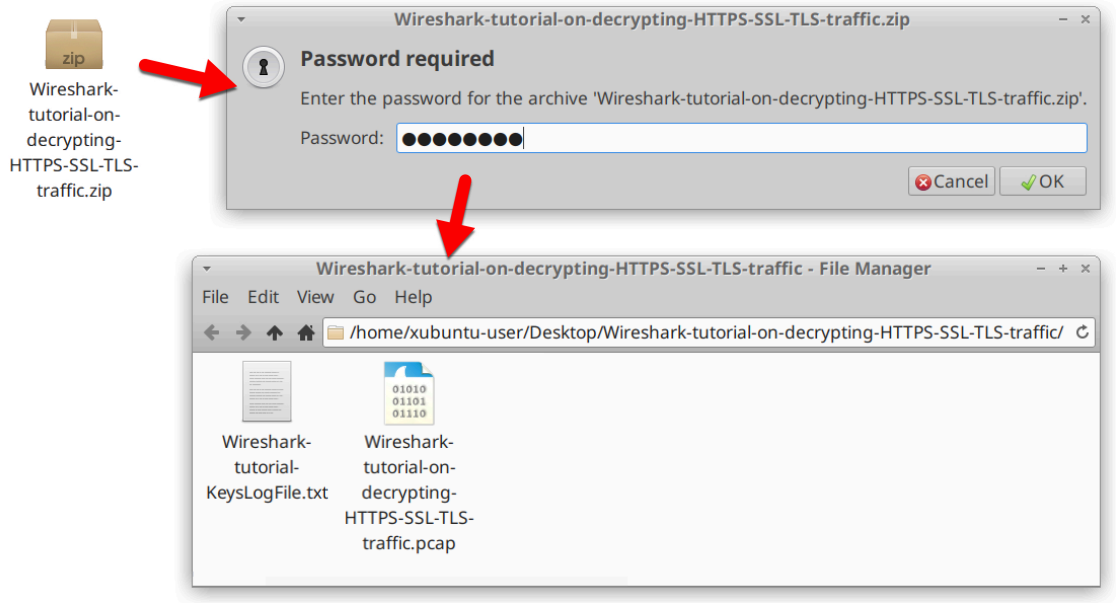

Figure 6. Key log file and pcap for this tutorial.

HTTPS Traffic Without the Key Log File

Open ***Wireshark-tutorial-on-decrypting-HTTPS-SSL-TLS-traffic.pcap*** in Wireshark. Use a basic web filter as described in this previous tutorial about Wireshark filters. Our basic filter for Wireshark 3.x is:

(http.request or tls.handshake.type eq 1) and !(ssdp)

This pcap is from a Dridex malware infection on a Windows 10 host. All web traffic, including the infection activity, is HTTPS. Without the key log file, we cannot see any details of the traffic, just the IP addresses, TCP ports and domain names, as shown in Figure 7.

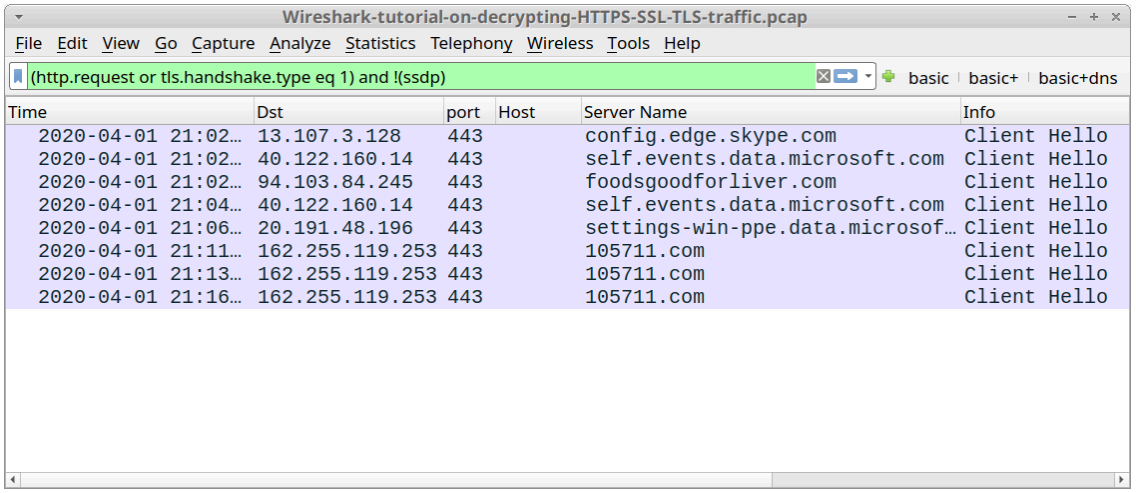

Figure 7. Viewing the pcap in Wireshark using the basic web filter without any decryption.

Loading the Key Log File

Open ***Wireshark-tutorial-on-decrypting-HTTPS-SSL-TLS-traffic.pcap*** in Wireshark. Then use the menu path ***Edit --> Preferences*** to bring up the Preferences Menu, as shown in Figure 8.

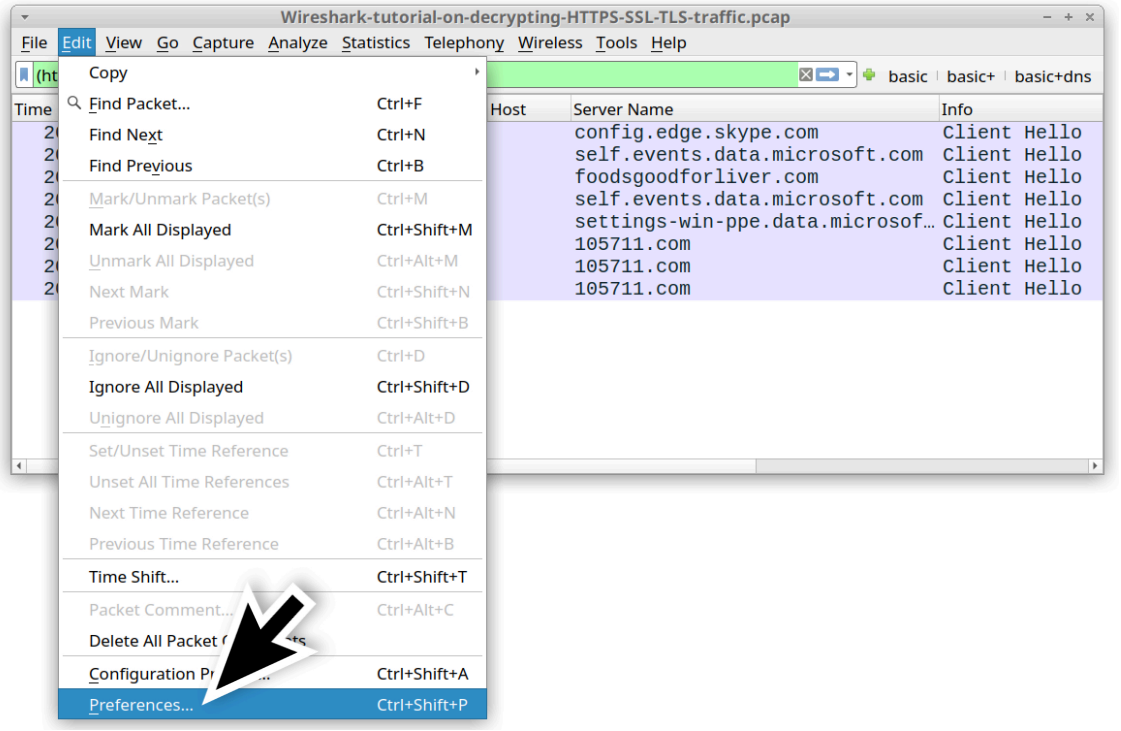

Figure 8. Getting to the Preferences Menu in Wireshark.

On the left side of the Preferences Menu, click on Protocols, as shown in Figure 9.

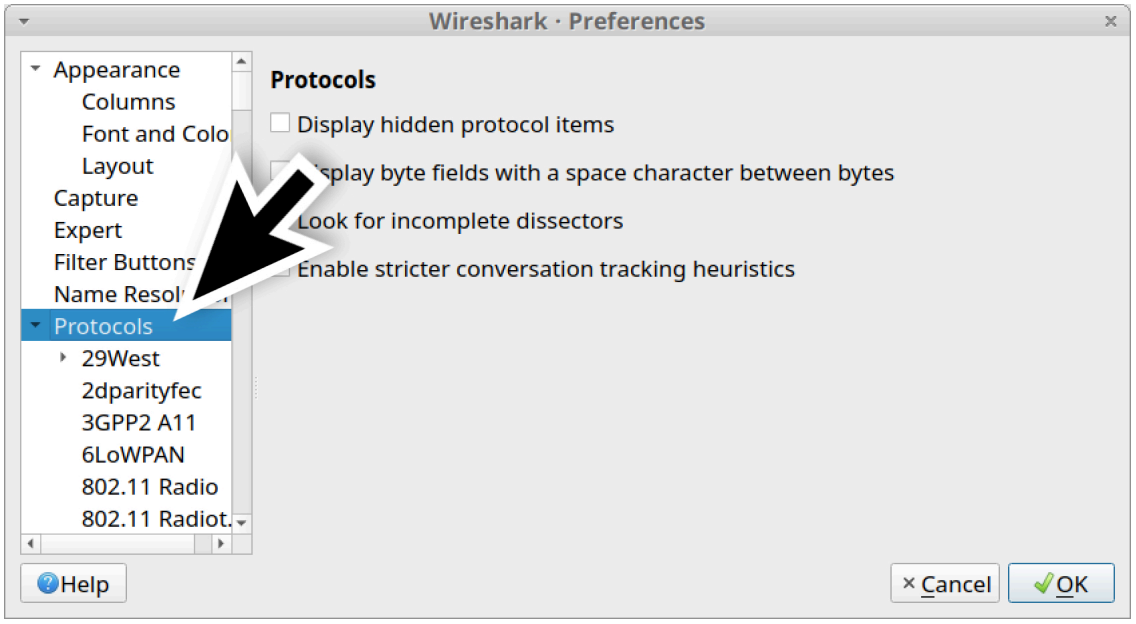

Figure 9. Selecting Protocols in the Preferences Menu.

If you are using Wireshark version 2.x, scroll down until you find ***SSL*** and select it. If you are using Wireshark version 3.x, scroll down to ***TLS*** and select it. Once you have selected SSL or TLS, you should see a line for ***(Pre)-Master-Secret log filename***. Click on the "Browse" button and select our key log file named ***Wireshark-tutorial-KeysLogFile.txt***, as shown in Figures 10, 11 and 12.

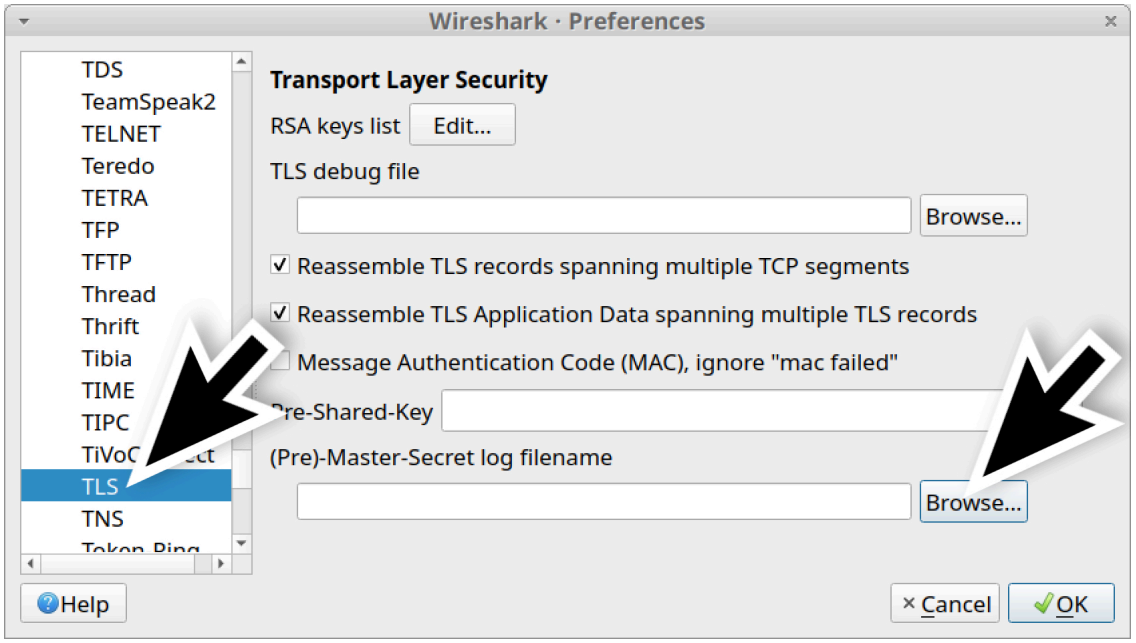

Figure 10. Finding the (Pre)-Master-Secret log filename field under TLS in Wireshark 3.x.

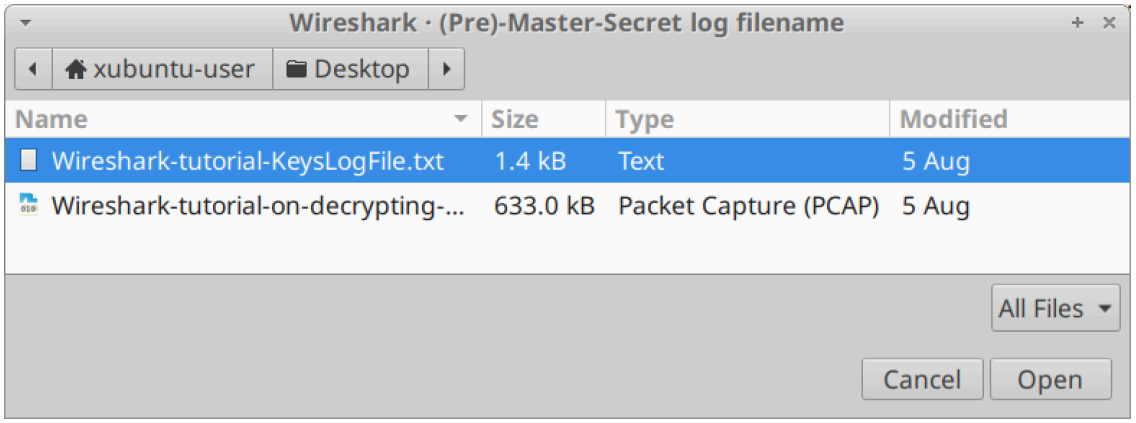

Figure 11. Selecting our key log file for this tutorial.

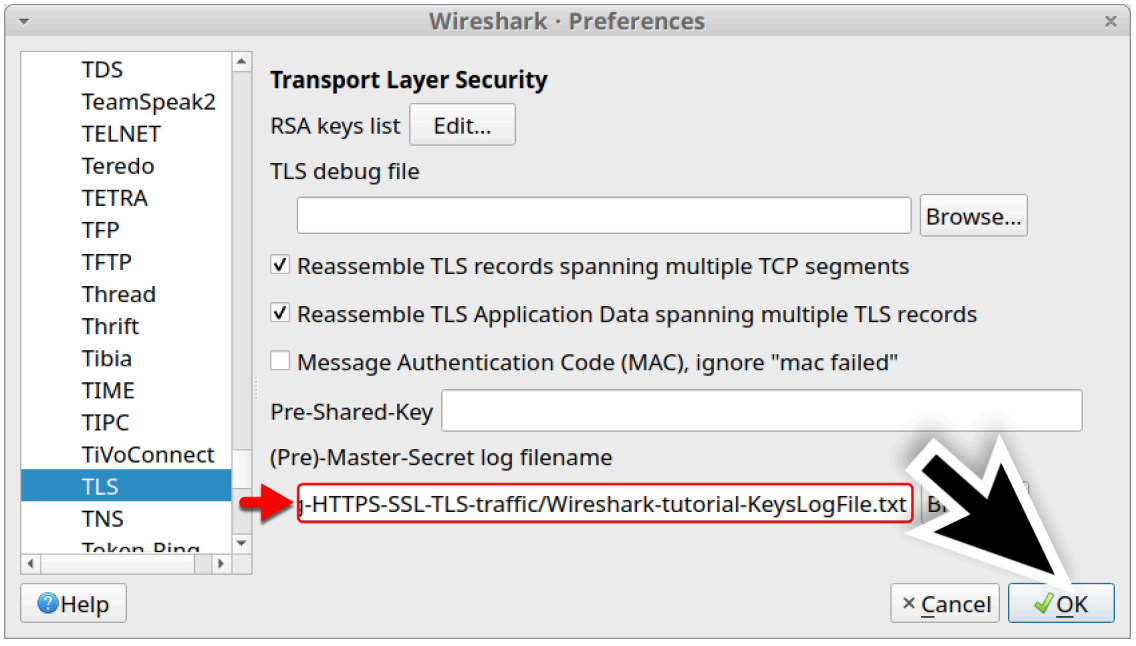

Figure 12. Once the file has been selected as the (Pre)-Master-Secret log filename, click “OK.”

HTTPS Traffic With the Key Log File

Once you have clicked “OK,” when using the basic filter, your Wireshark column display will list the decrypted HTTP requests under each of the HTTPS lines, as shown in Figure 13.

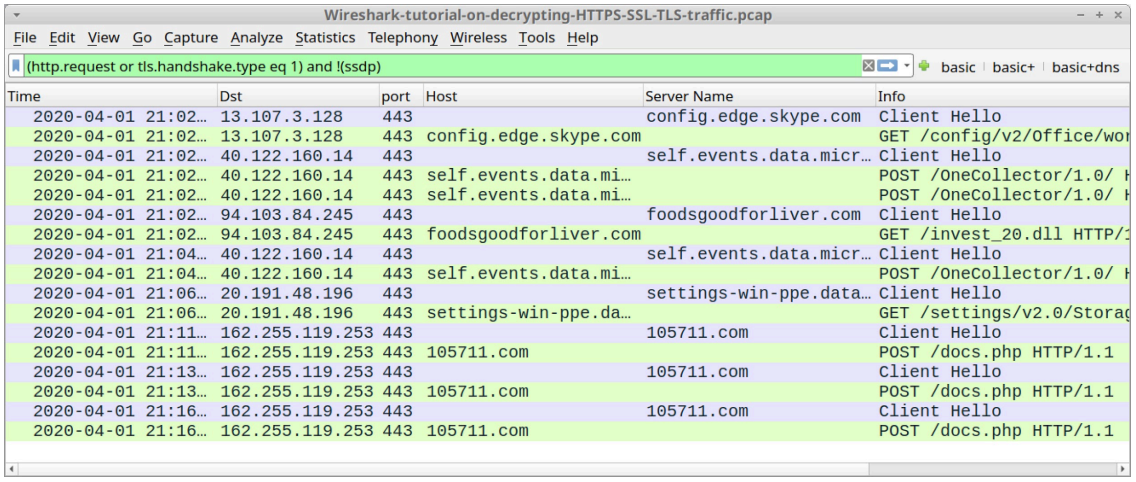

Figure 13. HTTPS decryption in Wireshark after using the key log file.

In this pcap, we now see HTTP requests to microsoft.com and skype.com domains previously hidden in the HTTPS traffic. We also find the following traffic caused by the Dridex infection:

- foodsgoodforliver[.]com - GET /invest_20.dll
- 105711[.]com - POST /docs.php

The GET request to foodsgoodforliver[.]com returned a DLL file for Dridex. The POST requests to 105711[.]com are command and control (C2) traffic from the Dridex-infected Windows host.

We can review the traffic by following HTTP streams. Right-click on the line to select it, then left-click to bring up a menu to follow the HTTP stream. Figures 14 and 15 show following the HTTP stream for the HTTP GET request to foodsgoodforliver[.]com.

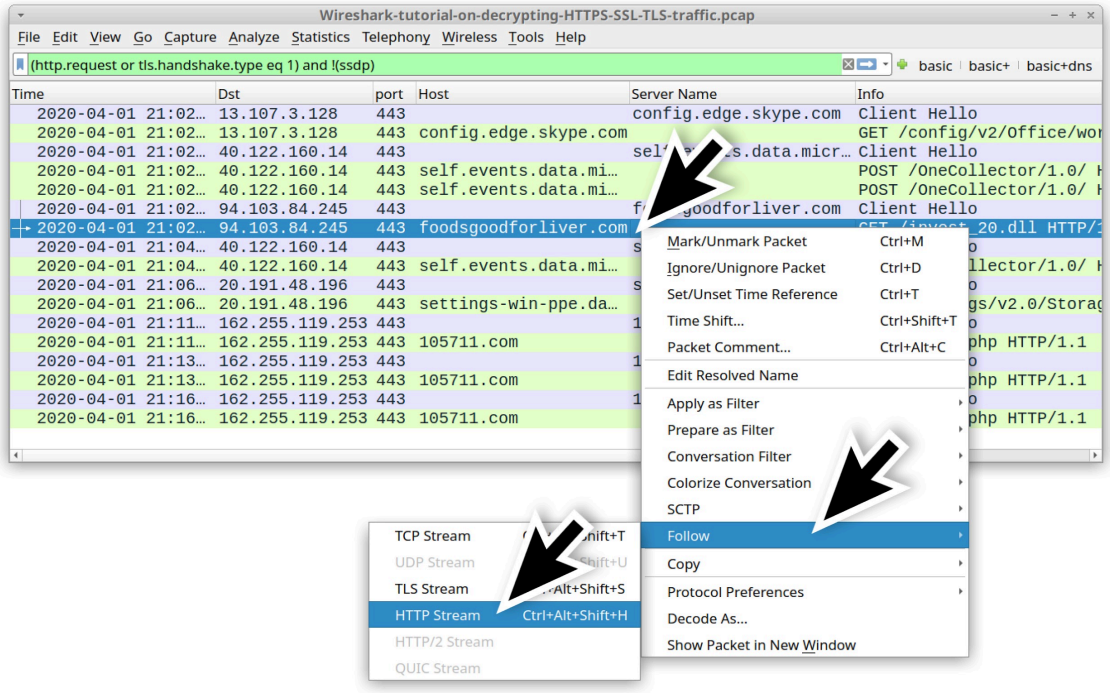

Figure 14. Following HTTP stream for the GET request to foodsgoodforliver[.]com.

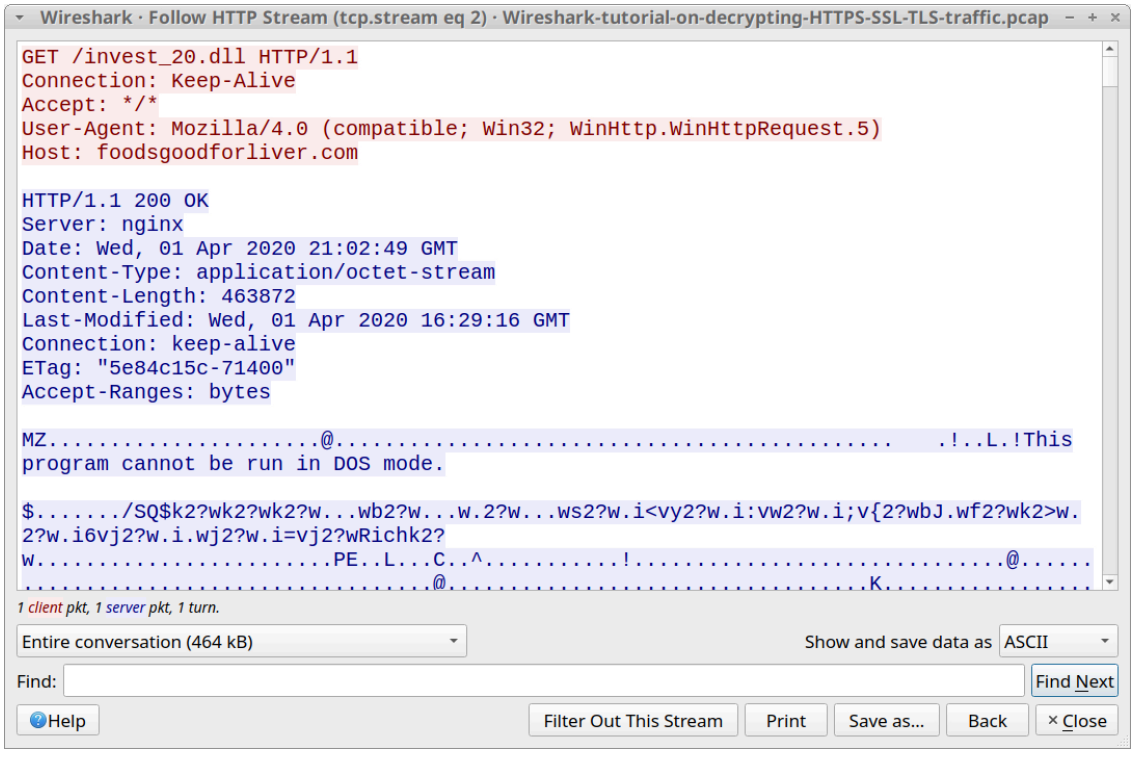

Figure 15. HTTP stream indicates an EXE or DLL returned from the server.

Since we have the key log file for this traffic, we can now export this malware from the pcap. Use the menu path ***File --> Export Objects --> HTTP*** to export this file from the pcap, as shown in Figure 16.

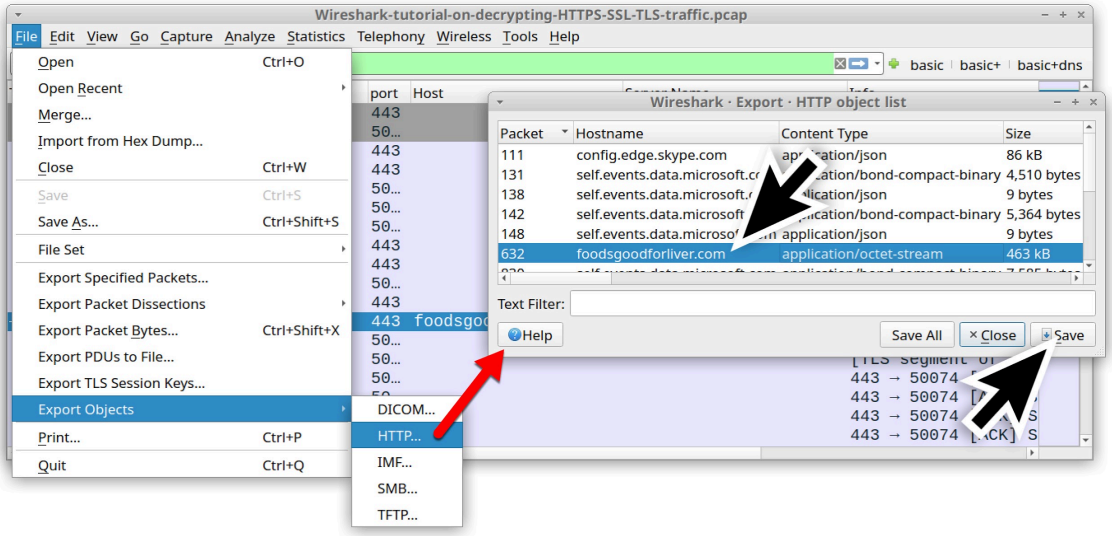

Figure 16. Exporting the malware binary returned from foodsgoodforliver[.]com.

If you are in a BSD, Linux or macOS environment, open a terminal window and use the

file

command to confirm this is a DLL file. Then use

shasum -a 256

to get the SHA256 hash of the file, as shown in Figure 17.

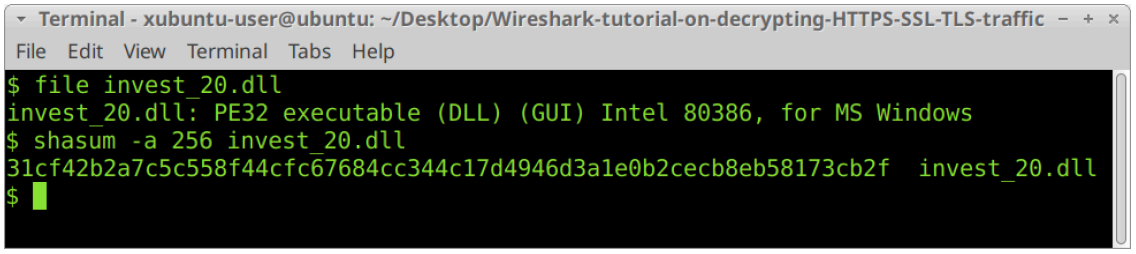

Figure 17. Getting the SHA256 hash of this malware in a Linux environment.

The SHA256 hash of this malware is:

31cf42b2a7c5c558f44cfc67684cc344c17d4946d3a1e0b2cecb8eb58173cb2f

If you search for this hash online, you should find results from at least two publicly available online sandbox environments.

Finally, we can review C2 traffic from this Dridex infection. Use your basic web filter, then follow an HTTP stream from one of the POST requests to 105711[.]com. An example from one of the HTTP streams is shown in Figure 18.

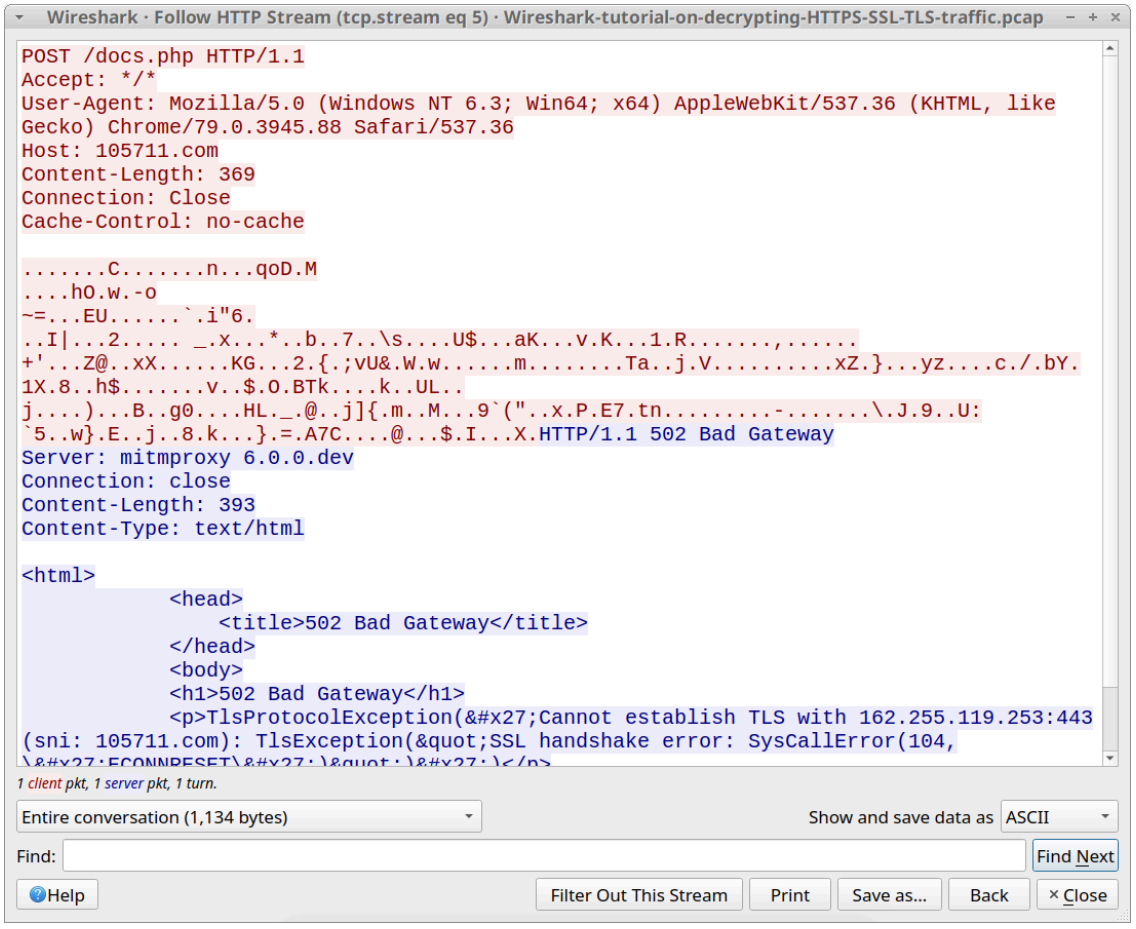

Figure 18. HTTP stream from one of the Dridex C2 POST requests.

Source: https://unit42.paloaltonetworks.com/wireshark-tutorial-decrypting-https-traffic/