Wireshark How To

About

Some tips to use Wireshark to find problems in SIP signaling and RTP media streams. Beware that these techniques dealing with RTP streams do not apply if those streams are encrypted.

The original site is no longer available.

Archive.org version: http://web.archive.org/web/20101204200659/http://www.panoramisk.com/151/analyzing-voip-with-wireshark/en/

Analyzing VoIP with Wireshark

When working in the IP telephony world it is crucial to know how to use a network analyzer, in order to understand how the traffic is propagating over the network. This article presents some interesting Wireshark features related to Voice over IP protocols.

Introduction

Wireshark (formerly Ethereal) is THE tool to have in your toolbox whenever working with applications that use the network. It is simple, efficient, and runs either on Microsoft Windows or Linux. Moreover it is free of charge. Even if you will find on some commercial products very powerful features, Wireshark has some good plug-ins targeting the VoIP space (as well as many others). This article focuses on SIP and RTP protocols which represent most of today's Voice over IP implementations.

How to capture frames

Prior to analyzing the network frames traversing the network, it is required to capture them. Analysis could be done either in real time when Wireshark is running on the probe itself, or it is also possible to capture the frames, store these in a file, and perform the analysis afterwards. More important is where to locate the probe in order to gather the appropriate frames containing voice related protocols. Since the SIP protocol is really distributed by nature, gathering voice traffic is a challenge, but there are solutions.

In order to collect voice frames we can either use Wireshark directly or use the application tcpdump, available on most Unixes and working directly from the command line.

With the chosen tool, two approaches are proposed:

- **using a mirror port on a switch:** this solution requires the use of network switches with a port mirroring feature. This feature is generally available on advanced enterprise products, some low end devices don't support it. The feature is sometime called port mirroring or SPAN port but the way it works is the same: we configure the switch to copy all frames transferred by a specific port to another dedicated port to which the analyzer is connected. If the voice traffic is separated from the data traffic onto a specific VLAN, it is sometimes possible to copy all traffic going through the VLAN towards the copy port. The port we will copy is where the SIP proxy is connected, but we can also copy an IP phone port if we know more precisely what to look for.
- **using directly the proxy server:** this second solution is about capturing traffic on the unix server hosting the proxy. For sure this requires running the tcpdump command directly from this server, which is not always possible. In the case of Asterisk, we are mainly using Linux as a hosting system and Linux does support tcpdump in most distribution, we will install it directly from the package if not yet present.

Capture with tcpdump

When using Asterisk, we can take advantage of two main points:

- the operating system is mainly Linux, so we can use tcpdump
- Asterisk is acting as a “back-to-back UA”, that is to say it stays by default in the middle of any session (phone call), even if this is not following the SIP distributed model, we will take advantage of it.

When capturing directly from the Asterisk server, we will have the full voice session, including signalling (SIP part) and the voice transport (RTP part).

tcpdump is a character-mode command, it doesn’t require any graphical interface to run and is very light, we could use it without putting too much stress on our voice gateway and altering the quality. As always with Unix, the command requires parameters and tcpdump proposes a lot of options. Here is a list of the useful ones in our case:

- -p : doesn’t start in promiscuous mode, only frames from or to the Asterisk node will be captured,
- -n : no name resolution, otherwise we will have a lot of DNS queries which is not useful at this stage, we could do name resolution afterwards if needed,

- -s 0 : we get the full frame, not only the first bytes. When working only at the protocol level it is enough to get only the start of each frames, in our case we are requiring the content of the SIP and RTP frames. 0 means getting the whole frame,
- -w output file : all the captured frames will be stored in the file. That way we will be able, afterwards, to analyze the frames from the file. If using '-' as the name of the file, the output is sent to the standard output.

In addition, we could specify to tcpdump a filter which will lower the number of frames we get in the capture. We can for example get only UDP frames and filter IP addresses in order to focus only on a specific host.

Local Capture

In order to perform the capture directly on the Asterisk host we should be connected to it. This can be achieve either directly on the console or through a remote terminal. I prefer ssh like many admins since it is simple and secure.

With the following command:

```
tcpdump -w trace.cap -p -n -s 0 "udp”
```

Remote Capture

Since you could manage many Asterisk servers, it could be easier to start a capture from your desktop by using the remote command facility provided by ssh. The ssh command on the desktop is required, already installed on Mac OS X and on Linux, you can install the Cygwin version on Microsoft Windows, you will have exactly the same command.

The proposed command is:

```
ssh root@asterisk 'tcpdump -w - -p -n -s 0 udp' > capture-asterisk.cap
```

The capture will occur on the server named asterisk using the tcpdump command, the frames will not be stored on the server itself but redirected to the desktop machine through the ssh tunnel2. The file will be named capture-asterisk.cap and stored in the current directory of your local workstation. Using such a method, you can start a capture on multiple Asterisk servers without having to connect a console session to them and transfer the capture file afterwards; furthermore, there is no storage issue on the remote machines.

Analyze

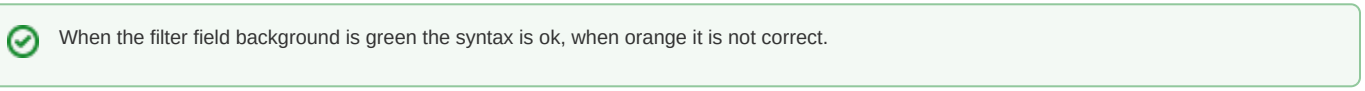
One we have a trace file we can now use the Wireshark tool to dive into it.

Wireshark is using a rich GUI and presents all the frames in the capture. On the upper window part you have the frame list, below that is the content of the selected frame either in decoded or raw format. Ethernet and IP protocols are decoded, what interests us today are SIP, IAX2, and RTP protocols since dedicated to voice over IP.

From the file explorer, we can open the file trace and analyze it. By default frames are ordered by arrival time, you can change this sort criteria if required.

Filtering

Depending on the filter used during capture, we can have noisy frames in it. The filtering feature of Wireshark allows to focus only on specific frames. It is activated through the 'Filter' dialog on the upper part of the window. The filter language is specific but easy to learn, for example if only SIP frames are needed you enter the 'sip' filter, if you are interested in IAX, the filter is 'iax2'.



If you need more complex filtering rules, you can directly select the field from the decode window. Each protocol field can be selected and added to the current filter through a right click and the “Apply as filter" / … and "Selected” menu.

SIP analysis

When looking at all the frames we have in a capture, it could be difficult to find quickly what we are looking for. Hopefully, Wireshark is bundled with some SIP tools, these are accessible through the “Statistics / VoIP calls” menu.

In the window you will see all the Voice over IP calls present in the capture file. If the capture is performed directly with Wireshark, this list will evolve in real time.

From the list, it is possible to show a specific conversation graphically. This view is really helpful to follow the conversation since all frames can be selected and inspected in the main window. It is also possible to analyze the content of the voice conversation since it's transported in the RTP files.

The “Player” feature decodes the RTP frames associated with the selected conversation. The graph is showing the shape of the voice flow for both ways. This tool is interesting when focusing on voice quality, if a user is complaining when talking with a specific phone, it is possible to listen to the conversation afterwards.

RTP analysis

The RTP4 protocol is not dedicated to voice over IP traffic. But this protocol is used aside H.323 and SIP signalling, Wireshark proposes a specific module to analyze the RTP flows. The feature is available through the “Statistics / RTP / Stream Analysis” menu.

The analysis window proposes information about the terminals engaged in the conversation, the codec used, and some statistics about the flow. You can focus for example on jitter and packet loss in order to understand why the voice quality wasn’t good enough. From this module, it is also possible to extract the content of the RTP frames and rebuild a voice sound file. This one will be saved in WAV format and you can listen to it with any sound player available.

Conclusion

IP telephony, like any application heavily using the network, should be analyzed whenever a problem occurs. In order to be reactive, my advice is to train yourself when everything is working fine and you have spare time to train. This will allow you to know the tools but also when looking at nothing special it is very common to find issue on the network. Finally, don’t be afraid of the protocols, even if they are complex. Their proper understanding will be a great advantage when facing a real issue on your system.