Call capturing

It allows you to receive .pcap.gz files with full dump of the call from CDR.

A problem of manual call capturing.

Very often, when the system is under high flow of calls you want to debug a particular call.
For example, when a new originator or terminator is set up.
Or when debugging a failed call, which was a few hours ago.
On a system that serves a small number of calls, you can use the instructions in Debug VoIP call with Wireshark and self-lift dump, and then decrypt it using *wireshark*.

However, if the system is already handling a large stream of calls, the above instructions for obtaining the dump may be illiquid.
For example, if you obtain the dump for all passing packets, your .pcap file becomes so large that *wirehshark* can not open it.
You can get around this problem by obtaining the dump with the filter on a specific IP address.

However, in this case, you will have problems with obtaining the dump, if the IP address for the media will not be the same for the IP addresses for signaling from the peer side.
Similarly, if the originator is a cluster, and the call can come from multiple IP.
You can get around this problem by specifying a filter for *tcpdump* to contain all possible IP address or network.

But here you might get a new challenge.
If you shoot a dump for the originator and want to see the form in which the call went to terminator.
We can not do without the *tcpdump* filter extension.
However, for complex routing rules, we can not even know where the call will be terminated for a particular originator.
Therefore to add terminator to the filter may be too complex.

But suppose you have managed to apply superhuman efforts and have generated appropriate filter.
Here a new issue waits for you, if you have multiple network interfaces, and calls are routed to a different interfaces.
You have to shoot a dump on each interface, and then merge the dump together.

There will also be a problem if the terminators that are involved in the routing for your origintors, receive traffic from other originators of yours.
In this case, your dump will acquire unwanted calls and may not be opened by *wireshark* again.

And even after all this, you may have difficulty with the isolation of the desired call flow, which is written to the file.
Because the file will contain the entire stream of calls from this peer.
Therefore, there remains a need to write to a dump for a very short time because *wireshark* simply would not open the file because of the size.

And if we have to do this operations many times a day as VoIP engineers have to?
We don't even speak about debugging calls "from the past", on which you might receive complaint from your customer.
This problem seems to be generally insoluble.

At this stage VoIP engineers understand that one can not manually capture calls for systems under heavy load.
Need an automated solution.

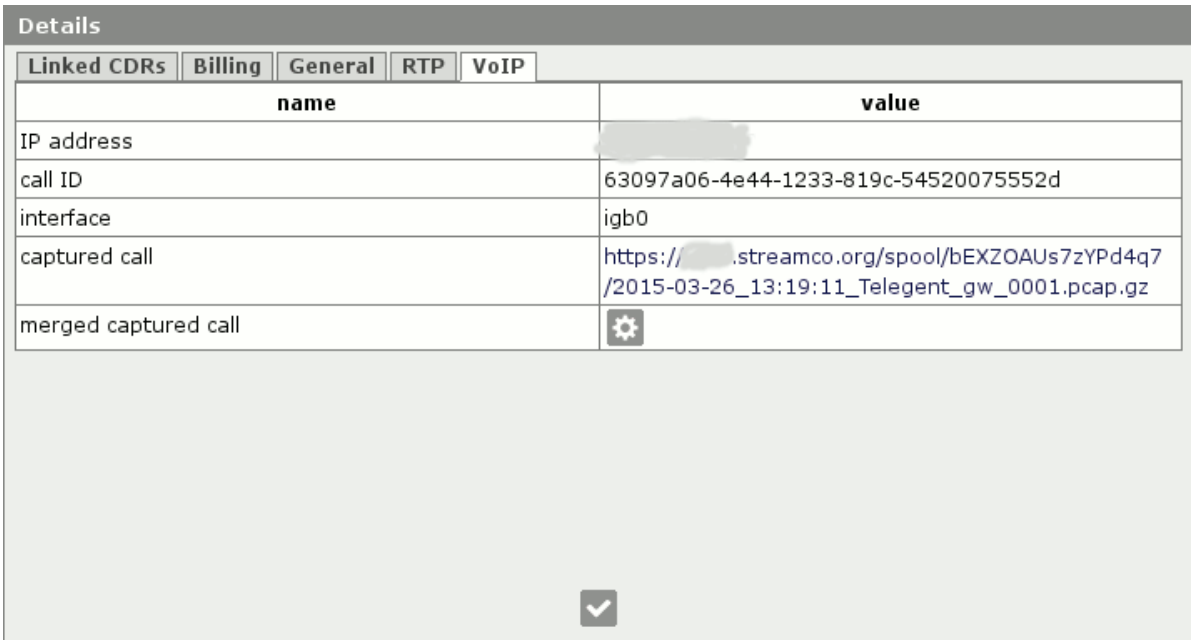
How does automated calls capturing works.

Call capturing feature of Smartswitch - this is an automated solution that helps to solve all the above problems.

To capture the calls an application daemon sniffer is used.
This application listens for traffic at the network interfaces.
However it's unreal to handle all the traffic passing through all interfaces.
Therefore, the application sets the special filters corresponding to configured peers, at the operating system level, for phishing just the appropriate traffic.
Thus, the application captures traffic for selected peers.
Afterwards, the capture of the traffic is divided into individual calls and stored in file dumps with the extension .pcap.
These files are then compressed to .pcap.gz format which is recognized by Wireshark.
Therefore the file dumps contain only information that are relevant to particular VoIP call.
This includes signaling and media (RTP streams, UDPTL faxes).
Extraneous information is not saved in them.

Files are attached to the details in the corresponding CDR database.

Thus, the call can be found in Call detail report, using the available filters, such as the caller's ID.
Then you can open the details and download the attached dump file.
Also CDR contains the name of the interface on which the dump was taken.
Further, since the incoming and outgoing legs of one call in Call detail report has linked CDR, you can automatically connect together dumps of all legs and download the file containing the entire dump of call legs, including the legs of the originator, terminators and all media streams.
An example is shown below:



Modes.

Application daemon has several modes of operation.
Mode is configured at the peer in the web interface.

- **enabled**.
  All calls related to this peer will be captured.

- **disabled**
  If this option is on the originator, then the call capturing will not be performed.
  If there is the terminator, the call capturing will occur if, if originator has option **chain** configured.

- **chain**.
  Allows you to record the call chain.
  If the originator is on, it begins to capture the call.
  If the call is routed to the termination, he also captured regardless of the options on the terminator.

Option **chain** is useful for debugging the full routing of calls from a particular originator.
Option **chain** leads to a small increase in PDD, as Asterisk needs some time to pass to the application daemon a command of turning on the call capturing feature for a specific terminator.

You can specify these options at every peer individually or can be set globally, and at peers leave the default.

If you think that this feature will be useful to you always (for example, for debugging calls "from the past" on customer complaints), it makes sense to set the option **enabled** globally.
Then capture calls will be activated for all existing peers.
This will not lead to an increase in PDD, as with option **chain**.
However this will require quite a lot of disk room to store all the dumps and will cause CPU usage increased.

Application daemon is able to process not only peers with static IP address and a dynamic and peers, such as dynamically registered SIP clients.
It also supports peers under NAT and peers under the specified ACL instead of a static IP address.

And with all this, the generated files will be small and fast to open in *wireshark*!

In your browser, you can customize automatic opening of files with the extension .pcap.gz with wireshark and the opening of a full dump should take a couple of seconds!
Application daemon very productive and can store hundreds of simultaneous calls.

Русский перевод

Files

captured_call.gif	13.1 KB	03/26/2015	Andrii Arsirii