

Installing & Securing VoIP With Linux

By Bruce "Corky" Wink (cwink@softwink.com) &
Champ Clark (champ@softwink.com)

Successful businesses usually have the same goal, minimize costs to maximize profits. Today with the plethora of open source solutions, a small business can present a high tech image and still keep a lid on the expenses.



Bunch'O'Phones: No, they are not all VoIP capable! Astersisk.org, kphone and IAXcomm (PC). Western Electric (Black), Northern Electric Buttset (Red), Xact 900mhz handset (blue) and the Cisco 7940 VoIP phone.

Early last winter, we had the opportunity to present a proposal for a financial institution to add two new remote offices. The requirement was to find a low cost solution for both the voice and data connections between all bank branches and provide a seamless integrated phone system with branch to branch dialing by extension, voice mail and conference calling. All though the new branches were only a short distance

from the bank's main office, 27 miles and 49 miles respectively, dedicated leased lines were expensive because each office was located in separate rural area and each area was controlled by a different telco. Quotes from several carriers to provide enough bandwidth for both voice and data between the new branches and the main office would cost a minimum of \$2,500 per month per branch. Coupled with expensive router equipment, phone equipment not to mention computer workstations and servers was going to really push the ceiling on the budget requirements for the new offices.

We have been using Linux for years as firewalls and we knew there had to be a low cost open source solution to fulfill the requirements, and that's when we found "Asterisk" (<http://www.asterisk.org>). It's a full blown Linux based PBX system meeting all the requirements of voice mail, conference calling, caller ID, call parking, music on hold, PBX to PBX dialing and much more. Asterisk was created and founded by Mark Spencer, who later started Digium. Digium is now the primary sponsor and developer, and they also offer software support as well as compatible hardware for use with Asterisk.

We looked on Digium's website (<http://www.digium.com>) for the necessary hardware and found for a mere \$500 their T100P product which would replace the DSU/CSUs (this card plugs directly into the "smartjack") and the telco routing equipment. This card is simply amazing. It's no bigger than a ½ height pci network card with a RJ45 port to connect to a DS1/T1 circuit. It can then be used to split the DS1/T1 voice and data channels.



The T100P card from Digium.com

We priced bundled DS1/T1's with the local telcos for each of the new branches and found I could get 8 local voice line channels and 12 Internet channels (768KB) bandwidth to the Internet for \$1,200 per branch per month saving \$2,600 of reoccurring monthly costs. I purchased three Intel white-box computers for \$800 each containing 2.6Ghz processors 512MB ram and 40 GB hard drives and 3 T100P cards, one for the main office to integrate their existing Merlin Legend phone system, which the bank did not want to replace and one for each of the two branches. In order to give the branches a new updated professional look we ordered refurbished Cisco 7940 and 7960 phones which were SIP capable. SIP (Session Initialization Protocol) is fast becoming the VoIP standard. We ordered 5 7940s and one 7960 for each branch. The only difference is that the 7940's can handle two lines and the 7960's sport 6 lines. The phones come shipped with Cisco's proprietary "skinney" (SCCP) protocol but Cisco has SIP software that can be loaded on the phones easily with the use of a dhcp and a tftp server.

Once the circuit's were installed by the local telco. We loaded Slackware 9.1 on each of the white boxes, installed the T100P cards and loaded the Asterisk software. To bring up the 12 data channels with the T100P card, you use the generic HDLC layer for Linux. More information can be found at <http://hq.pm.waw.pl/hdlc>, and it should be noted that Asterisk comes with the HDLC utilities ("sethdlc", etc). The voice channels are brought up using the Asterisk T100P drivers, and adding them into your Asterisk's zaptel.conf, like this:

```
fxsls=1-8           ; Channels 1-8 are incoming voice.
nethdlc=13-24       ; Channels 13-24 are for data.
```

Once the Asterisk command “ztcfg -vvv” is issued, the zaptel.conf file is read and the system is configured to use the channels as you described.

We had a minor problem with the IP addressing that telco provided. They gave us a WAN address scheme which was non-routable and a local LAN address scheme which was routable. Since our Slackware box was doing the routing with a network card to our private LAN and the T100P card to the public Internet, when we set up the WAN address on the T100P, we couldn't masquerade our 10.x.x.x private LAN using that network! We did place a call to Digium for support on this issue but we were told essentially that the data network side of the circuit (nethdlc) was unsupported and our problem. So we brought up the public LAN address as a secondary virtual interface on the T100P card. We needed the firewalling services of “iptables” to guard our private network from the public Internet but it choked on network address translation using the virtual device “pvc0:1”. To resolve this, we used a little trickery. We brought up the 12 channel pvc0 without an IP address first, then added the virtual device pvc0:1 and gave it the WAN address and then we registered the public LAN address with the pvc0 device and everything worked beautifully. This is all a bit confusing, so here's a sample of our script (names/numbers have been changed to protect the innocent):

```
-----[Snip]-----
#!/bin/bash

echo "Staring Frame Relay Data Connection...."

/sbin/ztcfg -vvv           # Brings up the Data/Voice channels.
/sbin/sethdlc hdlc0 fr lmi ansi # Bring up the type, and all that.
/sbin/sethdlc hdlc0 create 33  # Brings up pvc0 for DLCI 33.
/sbin/ifconfig hdlc0 up      # Note: NO IP address

# This brings up the telco' internal (serial) frame. This _cannot_
# be seen from the outside.

/sbin/ifconfig pvc0 up
/sbin/ifconfig pvc0:1 x.x.x.y pointopoint x.x.x.z
/sbin/ifconfig pvc0 y.y.y.x pointopoint y.y.y.y
/sbin/route add default gw y.y.y.z
-----[Snip]-----
```

One of the biggest concerns when dealing with branch to branch VoIP communications is security. Normally, VoIP is not encrypted, so naturally, we looked at possible ways to encrypt the traffic. Building VPN tunnels between the offices was the first logical choice.

We looked at several possible virtual private network (VPN) solutions before deciding on "OpenVPN" (<http://openvpn.sourceforge.net>). There are several reasons we picked this. First off, it's open source with a good track record. It allows us to pick the authentication and encryption method we wanted to use since it's based on OpenSSL. We also liked it because it allowed us to build UDP based VPN tunnels. This is important when dealing with VoIP.

For the most part, VoIP is transmitted by UDP/RTP (Real Time Protocol) packets. There is a good reason for this. TCP provides verification of packet delivery. This sounds like a good thing, but in the VoIP world, it's not. With TCP, if a packet is dropped, the remote side will retransmit the packet at the receivers request. By the time the voice packet is retransmitted, it's too late to put it in our audio stream! If we don't receive the packet within the allotted time it's useless to us. This is why network latency is such a issue with VoIP. TCP was designed to correct dropped packets as well re-organized out of sequence packets. This is very, very bad for VoIP. UDP on the other hand, allows packets to be sent out of order and there is no verification if the packet arrived. It's sent, and if it makes it, it makes it. This is the reason you *do not* want to use a TCP based VPN. If you encapsulate UDP traffic within a TCP tunnel, and the TCP tunnel traffic gets dropped or corrupted, it will cause a retransmission which completely defeats the purpose of VoIP using UDP!

Next we set up Inter-Asterisk-eXchange (IAX2) between each of the Asterisk boxes. With our encrypted UDP based tunnels up, we now had to put each Asterisk PBX server in communications with each other. We decided to use the IAX2 protocol because it's a simple protocol those geniuses at Digium developed – kudos to the Mark Spencer and his company. It operates similar to SIP but has several advantages. IAX2 can easily travel through NAT (Network Address Translated) environments, where SIP sometimes has problems. IAX2 also has a nifty feature called, "trunking". "Trunking" in IAX2/VoIP terms means that when there is more than one call over a IAX2 connection, the "calls" can share some of the TCP/IP header overhead thus reducing our bandwidth utilization requirement for voice conversations between branches.

For voice traffic compression, we decided on the GSM (Global System for Mobile) codec. It's a fairly "open" codec, uses as little as 13K per call, built in to Asterisk and it's free. We looked at G.729 as an alternative because it uses slightly less bandwidth at 9.6K per call but it's not free, you have to purchase a license for each device on which you're going to use it and it didn't seem worth the hassle to install.

Configuration of the Asterisk software wasn't all that complicated. We set up the extension for each of the new branches 200s for one and 300s for the other. The main office running the old Merlin Legend system was already configured for 100s.

We did have to call for support to the folks at Avante for help with the Legend system. The cost for their support was \$200 per hour, however it only took about an 1 ¼ hours of support time to tell the system that when a user dials a 200 or 300 extension to send the call out it's D100 CSU channel to the Digium T100P card located in the asterisk box at the main office. We enabled call conferencing (also built in to Asterisk) so that the managers from each branch could dial into the conference center hosted on the main office's Asterisk box.

The call quality between offices have been good once we instituted QoS. We initially had issues with crackle and break up when users data demands between offices started interfering with the call transport. We knew that being able to do QoS over the Internet was unlikely, because routers on the Internet typically won't honor it, however we did know that the VPNs between the office were going to be used for other purposes (web browsing, file sharing, etc) so we setup QoS on the internal VPN devices. Setting up QoS (Quality of Service) is a important thing and once it is set up at the very least, you're calls won't get interrupted by heavy web browsing and file transfers over the VPN. After we completed the configuration of the QoS on the VPN devices the crackle and breakup disappeared. More QoS information can be found at: <http://lartc.org/>.

Considering these installations where being preformed at financial institutions, 911 services where crucial. At each office, we had one standard POTS (Plain Old Telephone Service) line. This served a dual purpose, as a fax line and a backup for 911 services in the event of power outage.

There you have it, a Firewall, PBX and Internet Telephony all in one box and for the price of a \$500.00 dollar card and a generic white box computer. You could install your main server machine, a backup machine and still save money! All you need is a T-1 and you're off and running.

Rather than clutter this article with the numerous configuration files, we've posted slightly modified versions (to protect the innocent!).

These can be viewed and downloaded at:

<http://www.softwink.com/example-asterisk-article-1/>.

Champ Clark III & Bruce M. Wink

Both are partners in softwink.com an Internet security company that specializes in Intrusion Detection Monitoring for the financial industry.

They both can be reached at champ@softwink.com and cwink@softwink.com respectively.