# VPN CONSIDERATIONS & ANALYZING TRAFFIC TOOLS ON AN IPSEC COMMUNICATION LINK

**Marin LUNGU**
Computer Dept., Faculty of Automation, Computers and Electronics, University of Craiova
Str. Lapus Nr. 5, Craiova, Romania
Marin.Lungu@comp-craiova.ro

**KEYWORDS**
Communication, Client/Server, Chat, Linux Application, Sniffer

## ABSTRACT

The goal of this paper is to present a VPN type network security architecture based on the Internet connectivity model and IPsec communication approach. A **client/server application** (chat) and a **sniffer client** were chosed in a case study to implement security in a peer-to-peer communication link using Linux Slackware distribution platform [1] and Linux FreeS/WAN IPsec distribution [2]; **tcpdump** [3] which knows about IPSEC packets and how to print out network traffic in ASCII, lets us look at whats actually happening on the wire.

## Information over a network

The information sent over a network is divided up into smaller sections, called packets, put in some sort of virtual envelope, which then are sent on their way to destination. The type of "envelope" vary depending on the type of protocol being used: TCP, IPX, etc. However, each packet will contain the IP (Internet Protocol) address of origin, the IP of the addressee, and of course, information being sent. Sent packets are really open to anybody on the road across the information traffic way and must sometimes be processed by numerous computers and routers along the way. Theoretical and ideally, when another computer that isn't the intended recipient encounters a packet forward it to the next router or computer until it arrives at its intended destination. Upon arrival, the gateway reads each of these packets and decides what to do with them. At destination, the firewall parses the packets it receives and then relays the correct information according to certain rules to the intended computer within some sort of internal network. A gateway also pre-reads outgoing packets as well. However, this regular process can be interfered by a technique known as *sniffing*. Using a sniffer client someone can really "see" the passing packets through the Internet. System administrators can use sniffing to monitor their site, but also the hackers with malicious intentions can do sniff the packets and the results of their future actions can be quite annoying or even catastrophic.

## The basic idea of IPsec

**I**nternet **P**rotocol **sec**urity (**IPsec**) provides security functions, authentication and encryption, at the IP level. This requires a higher-level protocol, Internet Key Exchange (IKE) to set things up for the IP-level services:

- Encapsulating Security Payload (ESP) which encrypts and/or authenticates data
- Authentication Header (AH) which provides a packet authentication service.

Along with each of the above mentioned services comes a static IP address that never changes and allows you to host your own web site and administer your own server. Also, these services are used to build a VPN, a private network through un-secure networks.

## Operating Systems (OS) with IPsec support

The IPsec protocols are designed so that different implementations should be able to work together. As a result, the main commercial OS vendors put IPsec in their products: Microsoft in Windows 2000 and XP, IBM in OS390 via a crypto co-processor, Sun in Solaris 8, Hewlett Packard for their Unix machines, Certicom for the Palm. There are now Network cards with built-in IPsec acceleration (Intel, 3Com and Redcreek) and all the major open source OS support IPsec (i.e. FreeS/WAN as *the* Linux IPsec implementation).

## Road Warrior FreeS/WAN Config.

Most of problems arise when connecting from home to the office, especially if the telecommuter does not have a static IP address. In this case somewhat different set-up are necessary than VPN gateways with static addresses and with client systems behind them. In this context a "Road Warrior" is a traveller with dynamic IP address, which are connecting to home base from a laptop machine doing IPsec processing as a regular gateway. FreeS/WAN supports the connections very well but sometime there are difficulties at least in two situations:

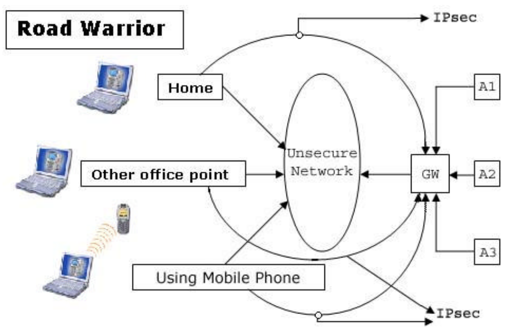


Figure 1. Some Road Warrior VPNs

- Road Wariors who get their addresses via **D**ynamic **H**ost **C** onfiguration **P**rotocol (DHCP); DHCP assign dynamic IP addresses, and provids additional information such as addresses of DNS servers and of gateways. FreeS/WAN can easy build and use a tunnel to such an address, but when the DHCP lease expires, FreeS/WAN does not know that. The tunnel fails, and the only recovery method is to tear it down and re-build it.
- If Network Address Translation (NAT) is applied between the two IPsec Gateways, this breaks IPsec. NAT allows multiple machines to communicate over the Internet when only one IP address is available for their use. IPsec authenticates packets on an end-to-end basis, to ensure they are not altered en route. NAT rewrites packets as they go by.

## Site-to-Site FreeS/WAN Configuration

Two different machines in two different places on Internet can communicate becoming their own gateways through a proper IPsec tunnel (any machine doing IPsec processing is a "gateway").

## FreeS/WAN Instalation

In order to instal the FreeS/WAN on a Linux platform we recomand the next six steps:

1. Download the last version of FreeS/WAN kit;
2. Unzip the kit;
3. Save the kernel configuration and run FreeS/WAN Compilation;
4. Copy ipsec.o in the modules' directory of Kernel;
5. Load it in kernel;
6. Configure IPsec;

Let choose the site-to-site model:

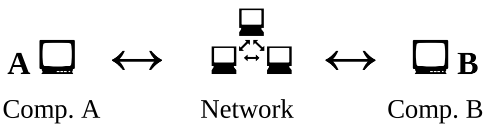


Figure 2. Site-to-site model

For simplicity and analyzing purposes, we used the next schema with adequat IP addresses:

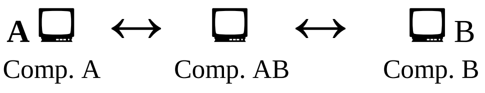


Figure 3. Simplefied Site-to-site model

| | |
|---|---|
| Comp. A: | eth0: 192.168.0.2/24 |
| Comp. AB: | eth0: 192.168.0.1/24 |
| | eth1: 192.168.1.2/24 |
| Comp. B: | eth0: 192.168.1.1/24 |

Next step, configure the */etc/ipsec.conf* file on the server, host and gateway, i.e. on computer A, *Site To Site* model:

```
    - /etc/ipsec.conf
# basic configuration
config setup
        interfaces="ipsec0=eth0"
        klipsdebug=none
        plutodebug=none
        plutoload=%search
        plutostart=%search
        uniqueids=yes

conn %default
          keyingtries=0
        keylife=8h
          disablearrivalcheck=no
        authby=secret

conn  site2site
        # left -- local
        left=192.168.0.2
        leftnexthop=192.168.0.1
        leftsubnet=192.168.0.2/32
        # right -- remote
        right=192.168.1.1
        rightnexthop=192.168.1.11
        rightsubnet=192.168.1.1/32
        auto=start
```

The server and a chat-client run on B, and only the chat-client on A; on computer AB the *Sniffer* catch the packet runing through port 3000. The chat is implemented in C++ using the API cross-platform wxWindows (www.wxwindows.org) in a client/server model with a TCP protocol communication;

The application was developped on Linux platform using the programming environment Kdevelop and GCC 2.95.3. compiler; on the Windows platform we used the DevC++ together with MinGW and GCC 2.95.3. compiler.

| | |
|---|---|
| wxWindows | - www.wxwindows.org |
| KDevelop | - www.kdevelop.org |
| DevC++ | - www.bloodshed.net |
| MinGW | - www.mingw.org |
| GCC | - www.gcc.org |

Communication protocol include six commands (AUTH, GET_USER_LIST, ADD_NICK, REMOVE_NICK, MSG, PUB_MSG) and every command has four parameters, i.e. COMMAND P1/P2/P3/P4.

**AUTH**: authenticates or not the users which send their user-name and password.

```
        Client
send:AUTH USER/PASS/NOP/NOP
recv:   AUTH OK/NOP/NOP/NOP
        AUTH DENY/MESSAGE/NOP/NOP
        Server
recv:   AUTH USER/PASS/NOP/NOP
send:   AUTH OK/NOP/NOP/NOP
        AUTH DENY/MESSAGE/NOP/NOP
```

**GET_USER_LIST**: authenticated user send to chat server GET_USER_LIST command. The answer is ADD_NICK NICK, which means accepted user on the users' lists.

```
        Client:
send:   GET_USER_LIST NOP/NOP/NOP/NOP
recv:   ADD_NICK NICK/NOP/NOP/NOP
        Server:
recv:   GET_USER_LIST NOP/NOP/NOP/NOP
send:   ADD_NICK NICK/NOP/NOP/NOP
```

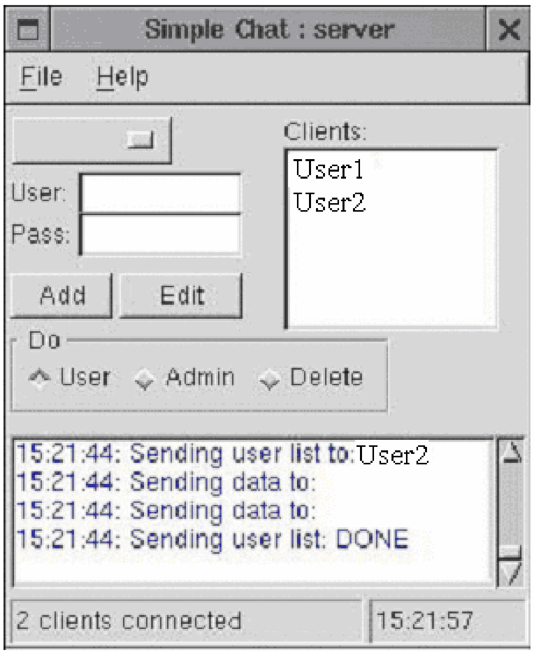
**ADD_NICK**: sent by server answering to a client GET_USER_LIST command or when ther is a new connected user and that means all the other users must up-date their lists.

**REMOVE_NICK**: sent by server to all users when someone is disconnected in order to delete it from their lists.

**MSG**: sent together with a message by a client to the server, which in turn re-send the message to all the connected users using **PUB_MSG** command.

## The Chat Server

When power-up, the chat server (Figure 4) open and wait on all IP addresses the clients' requests on port number 3000.



User:
Pass:

Clients:
User1
User2

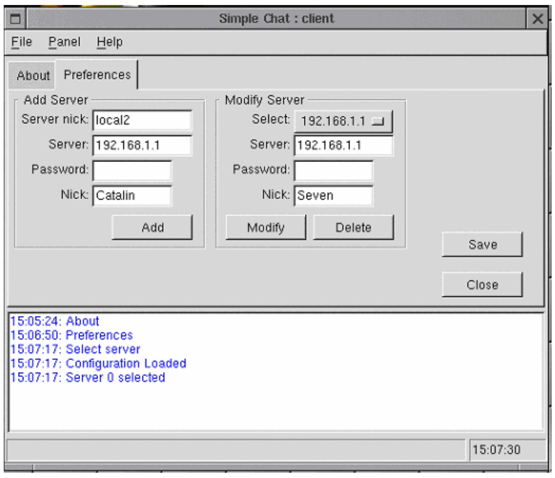
```
15:21:44: Sending user list to:User2
15:21:44: Sending data to:
15:21:44: Sending data to:
15:21:44: Sending user list: DONE
```

2 clients connected | 15:21:57

Figure 4. The main window of chat server

## The Chat Client

When power-up, the chat client display the configuration window (Figure 5) and goes through the following steps:



Add Server
Server nick: local2
Server: 192.168.1.1
Password:
Nick: Catalin

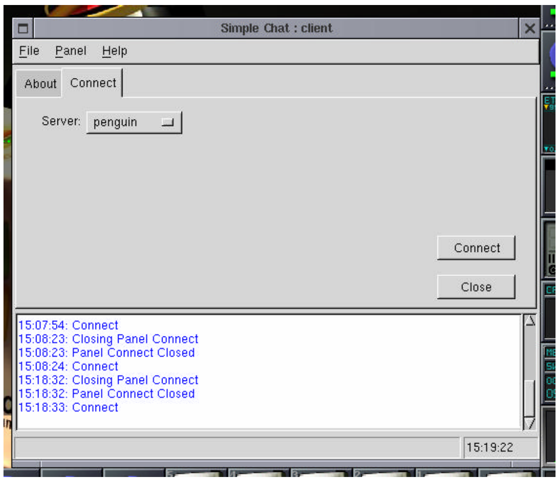
Modify Server
Select: 192.168.1.1
Server: 192.168.1.1
Password:
Nick: Seven

```
15:05:24: About
15:06:50: Preferences
15:07:17: Select server
15:07:17: Configuration Loaded
15:07:17: Server 0 selected
```

15:07:30

Figure 5. The configuration window of chat client

1. Configuration: if there is not yet an installed server, is needed some modifications on the existing one (**File/Preferences** windoew or **Alt-P**) or is intended to add a new one (Figure 5).



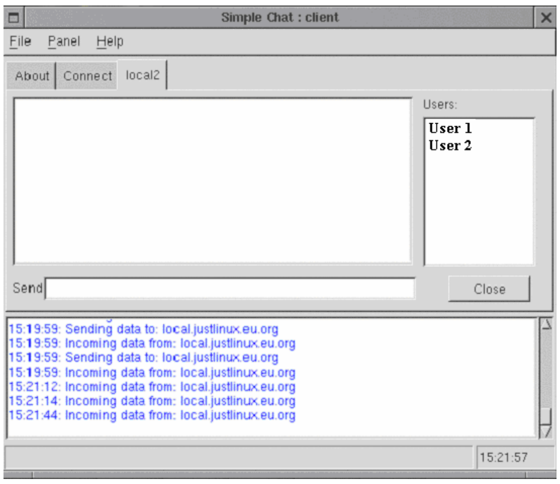
Server: penguin

```
15:07:54: Connect
15:08:23: Closing Panel Connect
15:08:23: Panel Connect Closed
15:08:24: Connect
15:18:32: Closing Panel Connect
15:18:32: Panel Connect Closed
15:18:33: Connect
```

15:19:22

Fig. 6. The connect window of chat client

2.Conexion to the server: use the Connect option (Alt-C) from the File menu, choose the server and push the Connect button (Figure 6). From now on we see and work in the main chat client window (Figure 7).



Users:
User 1
User 2

```
15:19:59: Sending data to: local.justlinux.eu.org
15:19:59: Incoming data from: local.justlinux.eu.org
15:19:59: Sending data to: local.justlinux.eu.org
15:19:59: Incoming data from: local.justlinux.eu.org
15:21:12: Incoming data from: local.justlinux.eu.org
15:21:14: Incoming data from: local.justlinux.eu.org
15:21:44: Incoming data from: local.justlinux.eu.org
```

15:21:57

Figure 7. The main chat client window

Without IPsec tunnel running, we catch something like that:

```
.......................................
SourceIP: 192.168.1.1,3000
DestIP  : 192.168.0.2,1032
.....A.:....AUTH OK/NOP/NOP/NOP

SourceIP: 192.168.0.2,1032
DestIP  : 192.168.1.1,3000
.......A.:GET_USER_LIST OP/NOP/NOP/NOP

SourceIP: 192.168.1.1,3000
DestIP  : 192.168.0.2,1032
.....A.H....ADD_NICK Sorin/NOP/NOP/NOP
.................................................................
```

The Sniffer see the packets passing through AB computer and display them if some conditions are met (a particular source/destination port and/or IP).
If the IPsec tunnel works, the packets' header is modiffied, the running server source/destination port is not 3000, as before, and the Snifer can't see the semnificance of packets anymore; using "tcpdump" Linux command we will see only a string of cripted packets through network (AB computer in this particular case), i.e:

```
..........................................
tcpdump -n -i eth1
tcpdump: listening on eth1
21:50:22.840769 192.168.0.2.1031 >
192.168.1.1.3000: F 356836452:356836452(0)
ack 198565953 win 5840 <nop,nop,timestamp
369590 4294419> (DF)
..............................................
21:52:29.855779 192.168.0.2.500 >
192.168.1.1.500: isakmp: phase 1 I ident: [|sa]
(DF)
...........................................................
21:52:41.473215 192.168.0.2 > 192.168.1.1:
ESP(spi=0x382a31ee,seq=0x2)
21:52:41.474256 192.168.1.1 > 192.168.0.2:
ESP(spi=0x71a6b0c9,seq=0x2)
........................................................................
158 packets received by filter
0 packets dropped by kernel
........................................................................
```

## Conclusion: VPN, your system and its security

A VPN (Virtual Private Network) connection protects your information between your applications and the VPN server itself. It does not provide complete end-to-end security. Wireless users are required to use the VPN system because unencrypted wireless communication can be intercepted in the air far more easily than wires can be tapped and interpreted. VPN is recommended for a remote access user, who needs an on-site IP address to authenticate him/herself. But anti-virus software, system patches, additional layers of encryption to finish the link between user applications and server applications, and vigilance are still required for system security. In addition, firewalls and other security measures are still recommended.
Though not all of the VPN device software supports the same protocols, the two that are most commonly used are IPSec (Internet Protocol security) and Microsoft's widely used PPTP (Point to Point Tunnelling Protocol).

## References

1. www.slackware.org
2. www.freeswan.org
3. http://www.tcpdump.org
4. www.wxwindows.org
5. www.kdevelop.org
6. www.bloodshed.net
7. www.mingw.org
8. www.gcc.org