


Implementation of a Virtual Private Network by Linux Operation System

Asmaa Salih Hamoudy Al-Sammara 

Received on: 8/9/2004

Accepted on:12/7/2005

Abstract

This paper deals with the implementation of a VPN in the Linux Operation System.

Linux was found to be a very appealing candidate for such applications for a number of reasons. Firstly the conventional Linux operation system is based on LINUX, which is renowned for its stability and high performance. Another benefit of Linux operation system is the availability of open source software. Not only is the Linux kernel source code open source but also almost all of the Linux based software is distributed as open source and General Public License (GPL).

In this paper, the system is implemented on a prototype of virtual private network that contains five computers .Then the analysis is performed using the result obtained from tests carried out on two VPN protocols and show that IPsec protocols is absolutely more preferable choice of tunneling protocols over openVPN in Linux OS.

تنفيذ شبكة افتراضية خاصة باستخدام نظام لينكس

الخلاصة

لقد تم تناول شبكة افتراضية باستخدام لينكس ، علما بانها المرشح المفضي في التطبيقات لانه :
اولا معروف باستقراره واداءه العاليين، وثانيا متوفر كبرنامج يمكن تغيير الشفرة الخاصة به
وتحديثه من قبل اي مستخدم. انجز في هذا البحث شبكة افتراضية تحتوى على خمسة حاسبات،
وتم انجاز التحليل باستخدام النتائج التي استخرجت من الاختبار الذي جرى على نوعين من
بروتوكولات VPN واثبت ان بروتوكول IPsec هو الافضل ل VPN 0

1.Introduction

1. VPN Protocols of the Prototype

There are many existing VPN protocol packages that are for the Linux operation system, in this prototype FreeS/WAN, which implements the IPsec protocol for Linux, and OpenVPN which is an application layer based VPN protocol. Other available tunneling protocols and VPN packages include PPTP protocols, L2TP and cIPE.

All these packages install in the Linux OS that implements the VPN gateway compatible with all VPN are installed protocol. In this work all testing and

analysis proposed use the FreeS/WAN IPsec and OpenVPN . The reason for these choices is that IPsec has emerged as the one of the standards in network security. It provides encryption and authentication services at the IP layer for Linux systems. Protocols used are the Authentication Header AH, Encapsulating Security Payload ESP and Internet Key Exchange IKE. The two primary components of FreeS/WAN are the kernel IPsec (KLIPS) and Pluto which is like IKE daemon. KLIPS implements the AH and ESP, handles packets within the kernel while Pluto

implements IKE, which negotiates connection with other systems. The FreeS/WAN is then used to configure VPNs of various topologies within the prototype [1].

On the other hand, choice is made of the OpenVPN which is a non-standard open-source package that utilizes OpenSSL for cryptographic methods to provide robust VPN services.

2. Prototype System Architecture

The implemented prototype system proposes a five inter-connected computers. Three of which possess two network interfaces with distinct IP addresses and are based on Linux operation system. The topology of this network is illustrated in the Figure 1.

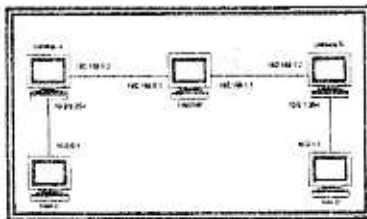


Figure 1 Prototype System Architecture

The middle computer (referred to as the Internet machine) with IP 192.168.0.1 and 192.168.1.1 has two network interfaces and it is acting as an Internet router. It also serves as the Certification Authority to sign certificates for the VPN computer/devices. Since this machine behaves as the public Internet, it will also perform traffic monitoring and "packet sniffing". This computer has been implemented in Linux platform.

Gateway A and Gateway B are two VPN gateways that provide all necessary security support to the 10.0.X.X internal networks. They

have both FreeS/WAN IPSec and OpenVPN configured that can provide host-host, host-network, and network-network topology VPN systems.

In a host-host situation, gateways A and B will be treated as hosts that are connected to the Internet and create a secure tunnel for all communications between these two machines. Hosts C and D are not considered in this configuration. For a host-network configuration, one gateway behaves as a host that is connected to the other gateway and to the computer in the network behind the gateway. Finally, the network-network configuration enables the computers in the local networks (e.g. hosts C and D) can securely communicate through the VPN gateways (A and B) and the Internet using their private addresses (i.e. the 10.0.X.X address). All three topologies are illustrated in the Figure 1. The host C wants to securely talk to host D over the Internet and it can do that by using host D's private network address 10.0.1.1. Data from C up to A will be un-encrypted since it is in the physically private local network, but from A to B will be encrypted. This protects the data flowing across the "Internet" in this system. B will then decrypt all messages and deliver to D. Since VPN is completely transparent to hosts C and D, the operating system on these machines can be Linux or Window or any other operation systems that support TCP/IP networking protocols. This consequently makes the support of multiple platforms possible.

2.1 System Hardware Requirements

The hardware system requirements that were used in this prototype implementation, is listed in Table 1. It is not necessary to much all of the

following detail in order to rebuild the test system, it is only for reference.

	OS	RAM	CPU
Monitor	RedHat Linux 7.1	256 MB	Intel Pentium III
VPN Gateways A & B	RedHat Linux 7.1	256 MB	Intel Pentium III
Workstation C & D	Microsoft Windows Fast Ethernet (RTL 8139C)	128 MB	Pentium II

Table 1.

2.2 System Software Requirements

The prototype system implemented many software tools and customized configurations to perform the desired security services. Almost all tools and configurations are done on Linux platform computers. The Window machines required only minimal network setting.

2.2.1 Software Requirement on VPN Gateways A and B

A standard RedHat Linux 7.1 is installed on the VPN

Gateways A and B. During the installation, a number of installation options will be asked. When the installation asks what system type for

software packages to be installed on the machine, "Customize" should be selected. This will reduce hard disk space requirement for the Linux System. On the other hands, if hard disk space is not concerned. "Everything" option would be the most suitable option. If "Customize" option is selected, specific software packages must be selected in order to finish the installation. Here, the following package must be selected:

1. Kernel Development
2. Development platform
3. Web Server
4. Network Management

After the installation, the system kernel must be updated to support the latest VPN protocol.

(A) Update and Compiling System Kernel

The reason for updating the system kernel is to provide better compatibility. Download the last kernel package from following address

➤ <http://mirrors.kernel.org/redhat/redhat/linux/7.1/linux-2.5.9.tar.gz>

And unzip the package in /usr/src type by executing following command:

```
$ tar zxvf linux-2.5.9.tar.gz
```

After that from the same directory execute the following commands:

```
$ mv linux linux-2.5.9
```

```
$ ln -s linux-2.5.9
```

```
$ cd /usr/include
```

```
$ mv linux linux.old
```

```
$ mv asm asm.old
```

```
$ ln -s ../src/linux/include/linux linux
```

```
$ ln -s ../src/linux/include/asm asm
```

Then finally, configure the kernel:

```
$ cd /usr/src/linux
```

```
$ make menuconfig
```

```
$ make dep
```

```
$ make bzImage
```

(B) Install VPN Protocols

This paper uses only two protocols, OpenVPN and IPSec.

1) Install OpenSSL
OpenSSL package openssl-0.9.7.tar.gz is downloaded from <http://www.openssl.org> then follow these procedures: -

```
$ tar -xvf openssl-0.9.7.tar.gz
To unzip the package and in the
OpenSSL the directory should execute
the following command
```

```
$ ./config
$ make
$ make test
$ make install
```

2) Install OpenVPN

The OpenVPN package (openvpn-1.3.1-1.rh72.i386.rpm) was downloaded from

➤ <http://unc.dl.sourceforge.net/sourceforge/openvpn/openvpn-1.3.1-1.rh72.i386.rpm>

Then the following command was run

```
$ rpm -ivh openvpn-1.3.1-1.rh72.i386.rpm
```

```
$ ./configure
$ make
$ make install
```

Key Management with OpenSSL

OpenVPN can support different key management methods. TLS asymmetric key was used in the prototype. New CA certificate will generate using

```
$ openssl req -nodes -new -x509 -keyout ca.key -out ca.crt
```

The .key file is the private key and .crt file is the CA's certificate that is signed by it. Sample keys tmp-ca.key and tmp-ca.crt are provided in the package but new keys should be generated for a secure system.

Then create a pem file (containing Diffie Hellman parameters) using:

```
$ openssl dhparam -out dh1024.pem 1024
```

This step is required for TLS server only.

Then generate keys for each VPN peer needed for the system:

```
$ open req -nodes -new -keyout mycert.key -out mucert.csr
```

Here the .csr file is a "Request to Sign" public key file to be sent to CA.

The CA can then sign the request by

```
$ openssl ca -out mycert.crt -in mycert.csr
```

Now the two peers (e.g. Omar and Mahad) are connected using

Omar:

```
$ openvpn --remote omar_IP --dev tun1 --ifconfig 10.4.0.1 10.4.0.2 --tls-client --ca ca.crt --cert omar.crt --key omar.key --reneg-sec 60 --verb 5
```

Mahad:

```
$ openvpn --remote mahad_IP --dev tun1 --ifconfig 10.4.0.2 10.4.0.1 --tls-client --ca ca.crt --cert omar.crt --key mahad.key --reneg-sec 60 --verb 5
```

This configuration gives Omar a virtual address 10.4.0.2 and Mahad 10.4.0.1.

Routing Table Configuration

Modification to the routing table and other settings are done to support the topologies network to network VPN.

To be able to allow internal traffic to transfer to the Internet over the VPN channel, first enable IP forwarding by:

```
$ echo 1 > /proc/sys/net/ipv4/ip_forward
```

Also it enables the package to forward through the firewall by:

```
$ iptables -A FORWARD -I tun+ -j ACCESP
```

Then enable secure outgoing traffic from the local network, add these entries to the routing table as follows: (10.0.0.0 and 10.0.1.0 are the local networks)

Omar:

```
$ route add -net 10.0.1.0 netmask 255.255.255.0 gw 10.4.0.2
```

Mahad:

```
$ route add -net 10.0.0.0 netmask
255.255.255.0 gw 10.4.0.1
```

3) Install FreeS/WAN IPSec

Download the package from the following:

➤ <http://download.freeswan.ca/binaries/RedHat-RPMs/2.5.9-24.7.x/freeswan-1.99-1.99.tar.gz>

First unzip the package in /usr/src by

```
$ tar zxvf freeswan-1.99
```

And inside the directory created

```
$ cd /usr/src/freeswan-1.99
```

```
$ make menugo
```

Finally installing the modules:

```
$ cd /usr/src/linux/arch/i386/boot
```

```
$ cp bzImage /boot/vmlinuz-2.5.9
```

Key Generation

For secure tunneling, generation of private key and public key is required for the end machines. In a secure working directory (permission set to 700), execute the following command:

```
$ ipsec rsaigkey -verbose 2048 >
rsakey.tmp
```

a pair of keys will be generated and shown in the `rsakey.tmp` file, the format of the file is:

```
: RSA {
<TAB> output of rsakeygen
<TAB>
<TAB> }
```

can put an index in front of the RSA key, for example

```
@elec.com : RSA {
<TAB> output of rsakeygen
<TAB>
<TAB> }
```

Then, copy the content of `rsakey.tmp` to `ipsec.secrets`:

```
$ cp -f rsakey.tmp /etc/ipsec.secrets
```

Network Configuration

After the RSA keys have been generated, IPSec is ready to have

configuration that is specified in the `ipsec.conf` file

Config setup

```
Interfaces="ipsec0=eth0"
```

```
Klipsdebug=none
```

```
Plutodebug=none
```

```
Plutoload=%search
```

```
Plutostart=%search
```

```
Uniqueids=yes
```

```
Conn %default
```

```
Keyingtries=0
```

```
Authby=rsasig
```

```
Conn VPNNetwork
```

```
left=192.168.0.2
```

```
leftsubnet=10.0.0.0/24
```

```
leftid=@elec.com
```

```
leftnexthop=192.168.0.1
```

```
right=192.168.1.2
```

```
rightsubnet=10.0.1.0/24
```

```
rightid=@elec.com
```

```
rightnexthop=192.168.1.1
```

```
Conn VPNhost
```

```
left=192.168.0.2
```

```
leftsubnet=10.0.0.0/24
```

```
leftid=@elec.com
```

```
leftnexthop=192.168.0.1
```

```
right=192.168.1.2
```

```
rightsubnet=10.0.1.0/24
```

```
rightid=@elec.com
```

```
rightnexthop=192.168.1.1
```

IP forwarding

IP forwarding must be enabled in all gateway machines, by executing the following command:

```
$ echo 1 >
/proc/sys/net/ipv4/ip_forward
```

Now the IPSec tunnel will start by executing this command:

```
$ ipsec auto -up <connection-name>
```

4) Apache Web Server with PHP support

Use those files that come with Red Hat Linux 7.1 CD. Find those files `apache-1.3.20-16.i386.rpm`, and `php-4.0.6-7.i386.rpm`, then execute the following command installation:

```
$ rpm -ivh apache-1.3.20.i386.rpm
```

```
$ rpm -ivh php-4.0.6-7.i386.rpm
```

After install all the files, go to "/etc", edit php.ini files at "file_uploads" change from "off" to "on"

- 5) Testing Apache Server with PHP
 1. go to /var/www/html
 2. remove file "index.html"
 3. Create a new "index.php" with following content
<?php phpinfo(); ?>
 4. Then load the html page with any browser



Figure (2) PHP and Apache SCREEN

A setup screen shown in Figure 2 should be received indicating that apache and PHP have been successfully installed.

2.2.2 Software Requirement on Monitor Machine

This is a Linux machine, which requires the following modules during Linux installation. The complete installation option should be selected if there is enough hard disk space.

1. Network Management
2. Web Server
3. Network Monitoring Package

This machine does not require any support for VPN. Only network management is needed. The main purpose of this machine is to act as the Certification Authority for public computers (or the Internet) to perform monitoring processes. Required

software packages on this machine are:

- ❖ OpenSSL
- ❖ Apache web server & PHP

2.3 Test Execution Procedure
Insecure channel & IPSec:

On the gateway machine with IP address 192.168.0.2 execute the sell script written for the test by entering "/pingtest1" to the command line.

Content of the file pingTest1
 Ping -c 30 -s 8 192.168.1.2 > output
 Ping -c 30 -s 64 192.168.1.2 > output
 Ping -c 30 -s 128 192.168.1.2 > output
 Ping -c 30 -s 256 192.168.1.2 > output
 Ping -c 30 -s 512 192.168.1.2 > output
 Ping -c 30 -s 1024 192.168.1.2 > output
 Ping -c 30 -s 2048 192.168.1.2 > output
 Ping -c 30 -s 4096 192.168.1.2 > output
 Ping -c 30 -s 8192 192.168.1.2 > output
 Ping -c 30 -s 16384 192.168.1.2 > output
 Ping -c 30 -s 32768 192.168.1.2 > output
 Ping -c 30 -s 65507 192.168.1.2 > output

OpenVPN:

On the gateway machine with IP address 10.4.0.1 execute the sell script written for the test by entering "/pingtest2" to the command line.

Content of the file pingTest2
 Ping -c 30 -s 8 10.4.0.2 > output
 Ping -c 30 -s 64 10.4.0.2 > output
 Ping -c 30 -s 128 10.4.0.2 > output
 Ping -c 30 -s 256 10.4.0.2 > output
 Ping -c 30 -s 512 10.4.0.2 > output
 Ping -c 30 -s 1024 10.4.0.2 > output
 Ping -c 30 -s 2048 10.4.0.2 > output
 Ping -c 30 -s 4096 10.4.0.2 > output
 Ping -c 30 -s 8192 10.4.0.2 > output
 Ping -c 30 -s 16384 10.4.0.2 > output
 Ping -c 30 -s 32768 10.4.0.2 > output
 Ping -c 30 -s 65507 10.4.0.2 > output

3.0 Performance and Analysis

This section outlines the analysis performed using the results obtained from tests carried out on the prototype system.

3.1 Performance Test

As described in implementation, two tunneling protocols are selected in this performance.

- 1) IPSec (standard tunneling protocol) and
- 2) OpenVPN (non-standard tunneling protocol).

Networking command "ping" is used for testing the performance of the 2 protocols. This test is also performed on insecure network (i.e. no tunneling) to provide some guidelines for the results.

3.2 Results

The graphical representations of the results are shown in the following diagrams

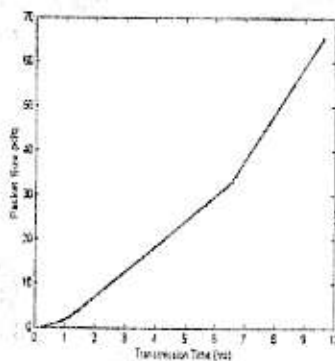


Figure 3 performance of Insecure Channel

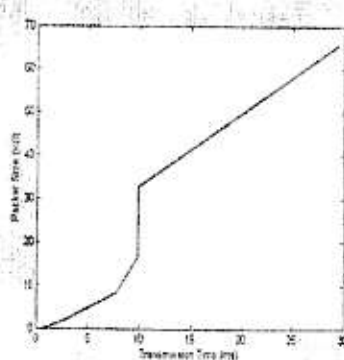


Figure 4 Performance of IPSec

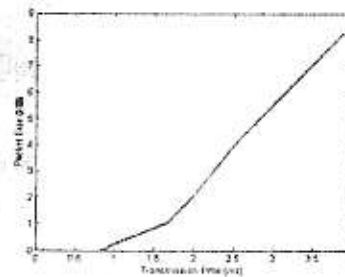


Figure 5 Performance of OpenVPN

3.3 Analysis

As shown in the above Figures, IPSec is by far the better tunneling protocol option over OpenVPN.

IPSec

The performance of IPSec goes downhill as the packet size grows. This is due to the fact that IPSec adds additional headers as well as performs encryption on the original packets – the time for IPSec to encapsulate the packet keeps getting longer and longer as the package size gets bigger. One interesting note is that the performance keeps almost constant when dealing with packets that are between 16384 and 32768 bytes – after this range, the performance goes back to its downhill trend again.

OpenVPN

Similar to IPSec, the performance of OpenVPN goes downhill as the packet size grows, but in a much more dramatic scale. The time required for transmission grows wildly after the package size reaches 8192 bytes – only two out of thirty package sent were received when the package is set to 16384 bytes, and the time required for those transmissions are around 2 seconds; the remaining 28 packages are regarded as loss since they took much longer than the specified interval of "ping" command to transmit. The 100% loss of packet with packet size = 32768 and 65507

can be explained with the same reason- they just took too long to transmit even the "ping" command stopped waiting for them.

3.4 Discussion of the Results

Although the performance of both IPSec and OpenVPN suffers greatly as the transmit packet size grows, it is not difficult to see which one has the more catastrophic effect. The primary cause to such occurrence is the difference in the layers that the two protocols function in: IPSec works in Network Layer whereas OpenVPN works in Session Layer. IPSec performs its routines in a very deep level (kernel level) of the system. The time required for operations such as encryption is kept as minimal as possible; OpenVPN operates in a much higher level- the need to call external software packages (e.g. OpenSSL) for the necessary operation makes the performance suffer deeply, let alone the significant amount of overload generated. Overall, IPSec is absolutely the more preferable choice of tunneling protocol over OpenVPN.

4. Conclusion

From this work, it can be concluded that IPSec is more preferable for a tunneling protocol to Open VPN and that it's possible to apply the results of this work to an actual site such as university campus, to make easy flow of the information between departments.

5. Reference

1. Enjamin Lee, "Security Protocols in VPN Technologies" DigiSAFA Pte Ltd, August 2002, available at [http://www.digisafe.com/news/pdf/VPN_Protocols\(Ver1_1b\).pdf](http://www.digisafe.com/news/pdf/VPN_Protocols(Ver1_1b).pdf)
2. Brian Proffitt, "RedHat Linux 7)", Prentice Hall of India, 2001.
3. IETF IPSec WG, "IP Security Protocol (IPSec)", April 2003, available at <http://www.ietf.org/html.charters/ipsec-charter.html>
4. James Yonan, "Open VPN", Source Forge Co., 2003, available at <http://openvpn.sourceforge.net/>
5. William Valella, "Securing Open Source Virtual Private Networks", MSc. Thesis, University of Florida, 2001.