Sys Admin
the journal for UNIX and Linux systems administrators

Build IPSec VPNs Using the Linux Kernel 2.6

Ralf Spenneberg

Virtual private networks (VPNs) have been around for quite some time. Several protocols are available to implement VPN solutions. The most prominent protocols are the Point-to-Point-Tunneling-Protocol (PPTP) and the IP Security Protocols (IPSec). I have played around with both protocols during the past 5 to 6 years building both small and worldwide implementations. Most often I used open source operating systems like Linux or OpenBSD to implement the VPN gateways. Using Linux to build an IPSec VPN usually means using the FreeS/WAN IPSec stack.

A lot of my time working with VPNs is actually spent testing new implementations and products -- testing the interoperability, robustness, and speed. So, naturally I looked into the future Linux versions that are available as developer kernels 2.5.*.

If you are working in the security field and are using Linux, be prepared for big changes once Linux kernel 2.6 is released. A lot of features now available only as patches will be incorporated into the next Linux kernel. Among these features are:

- CryptoAPI, enabling the kernel to do cryptographic task
- Linux Security Module
  - Linux Intrusion Detection System (LIDS)
  - Security Enhanced Linux
- Native IPSec Stack

Having these features in the stock kernel will force these features into all the major distributions available. This article will cover the new IPSec Stack included in Linux.

This IPSec stack is be no means a FreeS/WAN stack. It is based on a totally different implementation by the USAGI team. The userspace tools of the KAME project (setkey and racoon) may be used to setup the connections. The KAME project implements an IPSec Stack for BSD. The Linux port was done by Dave Miller and Alexey Kuznetsov. If you know how to configure IPSec on BSD variants, most will be familiar to you.

If you have used IPSec in the past, you will understand the basic terminology and features of IPSec, so you might want to skip the next few paragraphs. For those not familiar with IPSec, I will try to cover the aspects you'll need to know to be able to use it.

IPSec introduces two new IP protocols: Authentication Header (AH, 51) and Encapsulating Security Payload (ESP, 50). These protocols can be used either in transport or in tunnel mode. In transport mode, the packets transferred between two distinct machines are secured. In tunnel mode, the computer using IPSec can function as gateways between two networks tunneling all packets going from one network to the other network and back. Usually the tunnel mode is used.

The AH protocol guarantees the integrity and the origin of the packet. To achieve this, a Hash Message Authentication Code (HMAC) is added to each packet. This HMAC is calculated using a hash function based on the contents of the packet and a secret shared key. The hash function even includes the immutable parts of the IP header, like IP addresses. Since the used key is secret, an attacker cannot modify the packet without the recipient noticing. The recipient has access to the same key and can check the integrity. The 96-bit long HMAC is usually implemented using either MD5 or SHA1.

The ESP protocol guarantees the integrity and confidentiality of the packet. It uses the HMAC

method like the AH protocol, but it just reads the actual payload of the protocol and not the immutable parts of the IP header. ESP is therefore not broken by network address translation (NAT) the way AH is. Additionally, it can encrypt the payload of the packet using different symmetric encryption algorithms like DES, 3DES, AES, etc. The used symmetric key needs to be shared by the two machines using IPSec.

The information needed for the setup of AH or ESP communications is called an IPSec Security Association (IPSec SA). This SA defines the HMAC and encryptions algorithms, IPSec protocols, IP addresses, and a unique Security Parameter Index (SPI) to identify the IPSec SA. All SAs are stored in the Security Association Database (SAD).

Connections, in which the symmetric keys, the algorithms, and protocols are manually chosen, are called manually keyed connections. These place a great burden on the administrator who must exchange the keys regularly by hand on all machines to ensure the security of the VPN.

The Internet Key Exchange protocol (IKE) can exchange these keys and re-key them automatically. Such connections are called automatically keyed connections. The IKE protocol uses UDP port 500. It exchanges proposals on the algorithms and then chooses the best protocol and the best algorithm available to both machines. The result of this process is called an Internet Security Association Key Management Protocol Security Association (ISAKMP SA). A Diffie-Hellman key exchange is used to create the symmetric keys needed for the authentication and encryption of the packets.

Usually the IKE protocol is not implemented by the operating system itself but by an external daemon. For example, FreeS/WAN uses Pluto. The new Linux kernel uses Racoon, the IKE daemon of the KAME project. This daemon communicates using the IKE protocol and creates IPSec SAs as needed.

The operating system must know when to use which SA and whether to allow unencrypted traffic. This information is stored in the Security Policy (SP), which in turn is stored in the Security Policy Database (SPD).

How to Configure the Kernel

When configuring the Linux kernel, make sure that you have at least version 2.5.47 or later. Then, extract the Linux kernel and start the kernel configuration using the following commands:

```
cd /usr/src
tar xvjf /path-to-kernel-archive/linux-2.5.<version>.tar.bz2
cd linux-2.5.<version>
make xconfig
```

When configuring the Linux kernel, choose at least the features shown in Figures 1 and 2. Depending on your kernel version, you might need to turn on the IPv6 support for the kernel to compile successfully. Then compile and install the new Linux kernel:

```
make bzImage
make modules
make install
make modules_install
```

Check your boot loader. Most often the **make install** command already took care of the boot loader configuration. Reboot your computer using the new Linux kernel.

For the following test setup, you will need at least two machines. These can either be real computers or you can simulate the machines using either VMware or User-Mode Linux. Once your machines are up and running, you need the user space tools. These tools are now located at:

```
http://ipsec-tools.sf.net
```

When compiling these tools from source, your kernel source tree must appear under /usr/src/linux:

```
ln -s /usr/src/linux-2.5.<version> /usr/src/linux
```

The rest of the compile process boils down to:

```
./configure
```

```
make
make install
```

If you are using an RPM-based Linux distribution, like Red Hat Linux, you can download an RPM package at:

```
http://www.spenneberg.org/VPN/kernel-2_5/.
```

Once the tools are installed, you can begin to set up your VPN. The following pages will demonstrate first how to set up a manually keyed connection. Once that is working, I will describe automatically keyed connections.

Manually Keyed Connections

The main command to set up IPSec in the new Linux kernel is **setkey**. The easiest way to use this command is a script, which is then executed using **setkey**. Here is a typical file:

```
#!/usr/sbin/setkey -f
#
# File /etc/ipsec.conf

# delete the SAD and SPD
flush;
spdflush;

# manually keyed AH connection
# add <IP> <IP> <protocol> <SPI> -A <algorithm> <key>
add 192.168.1.5 192.168.2.5 ah 0x300 -A hmac-md5 \
 0xdf84cd88405b8faed89031e4118e6cf6;
add 192.168.2.5 192.168.1.5 ah 0x400 -A hmac-md5 \
 0xdf84cd88405b8faed89031e4118e6cf6;

# manually keyed ESP connection
# add <IP> <IP> <protocol> <SPI> -E <algorithm> <key>
```

```
add 192.168.1.5 192.168.2.5 esp 0x301 -E 3des-cbc \
 0x78bab1a6380fa764e4be09da2e13d65076d503cf3089ea82;
add 192.168.2.5 192.168.1.5 esp 0x401 -E 3des-cbc \
 0x78bab1a6380fa764e4be09da2e13d65076d503cf3089ea82;

# Security policies define when to use which SA
spdadd 192.168.1.5 192.168.2.5 any -P out ipsec
           esp/transport//require
           ah/transport//require;

spdadd 192.168.2.5 192.168.1.5 any -P in ipsec
           esp/transport//require
           ah/transport//require;
```

This file must be created on one of the machines (192.168.1.5). The computers must use the configured IP addresses 192.168.1.5 and 192.168.2.5, or these need to be modified. Unfortunately, the configuration of the second computer must reflect its point of view. The direction (in/out) in the security policies has to be exchanged.

As soon as the command **setkey -f /etc/ipsec.conf** has been executed, the machines exchange encrypted and authenticated packets. A typical ping will be recorded by tcpdump as:

```
12:45:39.373005 192.168.1.5 > 192.168.2.5: AH(spi=0x00000300,seq=0x1):
ESP(spi=0x00000301,seq=0x1) (DF)
12:45:39.448636 192.168.2.5 > 192.168.1.5: AH(spi=0x00000400,seq=0x1):
ESP(spi=0x00000401,seq=0x1)12:45:40.542430
```

This was a very simple example of a manually keyed connection. Unfortunately, the administrator must re-key the connection regularly to ensure the integrity of the keys. The keys must be exchanged in a secure way. The VPN at least cannot be used for the first exchange. The set up creates a high administrative burden and is error prone.

Automatically Keyed Connections

The solution is the usage of the IKE protocol. The IKE daemon, **racoon**, is part of the above-mentioned ipsec-tools package. It is installed when the **setkey** command is installed.

This IKE daemon first authenticates its peer. It then negotiates the IPSec protocols and algorithms using proposals. Then it creates the symmetric keys for the authentication and encryption of the protocols. These keys are automatically re-keyed by **racoon**.

The authentication can be based on shared secrets (like passwords), Kerberos, and X.509 certificates. It supports the Main, Aggressive, and Base Mode in phase 1 of the IKE protocol. Almost all known encryption and authentication algorithms are supported by Racoon.

To use Racoon, you must first create the configuration file **/etc/racoon.conf**. For this demonstration, I will use pre-shared keys (PSK) for the authentication of the peers. The PSK has to be stored in a separate file **/etc/psk.txt**. A typical PSK file is shown below. The first column can either hold an IP address, an email address, or an fqdn. Everything starting at the second column is used as secret and can either be ASCII or a hexadecimal starting with 0x:

```
# IPv4 addresses
192.168.2.5            bad simple psk
5.0.0.1                0xe10bd52b0529b54aac97db63462850f3
# USER_FQDN
ralf@spenneberg.net    kfaizeafasdf
# FQDN
www.spenneberg.net     This PSK identifies a fully qualified domain name
```

A typical Racoon configuration file for the setup used in the manually keyed connection is shown here. This configuration applies to the machine 192.168.1.5:

```
path pre_shared_key "/etc/psk.txt";

remote 192.168.2.5 {
        exchange_mode main;
        proposal {
                encryption_algorithm 3des;
```

```
                hash_algorithm md5;
                authentication_method pre_shared_key;
                dh_group modp1024;
        }
}

sainfo address 192.168.1.5 any address 192.168.2.5 any {
        pfs_group 768;
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

Again, the configuration on the second machine must be slightly modified replacing every occurrence of 192.168.1.5 by 192.168.2.5 and vice versa. Racoon will use the configured parameters for its proposal to its peer.

Racoon does not start the connection by itself, like FreeS/WAN does. For Racoon to start the connection, it must be asked. This is accomplished by the Linux kernel. Whenever the Linux kernel encounters a packet that must be encrypted according to the security policies, but no security association defining the keys and algorithms exists, the kernel calls Racoon. It is now Racoon's task to authenticate the peer and to create the appropriate SAs.

The required security policies shown here must be created using the command **setkey** before Racoon has been started:

```
#!/usr/sbin/setkey -f
#
# Flush the SAD and SPD
flush;
spdflush;

# Security policies for racoon
spdadd 192.168.1.5 192.168.2.5 any -P out ipsec
```

```
            esp/transport//require;

spdadd 192.168.2.5 192.168.1.5 any -P in ipsec
            esp/transport//require;
```

Again the configuration in this example is prepared for the machine 192.168.1.5. The directions (in/out) of the security policies must be exchanged for the machine 192.168.2.5.

Now the administrator can fill the SPD and start Racoon:

```
# setkey -f /etc/ipsec.conf
# racoon -c /etc/racoon.conf
```

As soon as the connection is needed, Racoon will start negotiating the connection. When started using the option **-F**, Racoon will stay in the foreground and will log everything on the console. In this mode, errors can easily be found:

```
2003-01-21 21:11:17: INFO: main.c:170:main(): @(#)racoon
  20001216 20001216 sakane@kame.net
2003-01-21 21:11:17: INFO: main.c:171:main(): @(#)This product
  linked OpenSSL 0.9.6b [engine] 9 Jul 2001 (http://www.openssl.org/)
2003-01-21 21:11:17: INFO: isakmp.c:1365:isakmp_open():
  127.0.0.1[500] used as isakmp port (fd=7)
2003-01-21 21:11:17: INFO: isakmp.c:1365:isakmp_open():
  192.168.1.5[500] used as isakmp port (fd=9)
2003-01-21 21:11:37: INFO: isakmp.c:1689:isakmp_post_acquire():
  IPsec-SA request for 192.168.2.5 queued due to no phase1 found.
2003-01-21 21:11:37: INFO: isakmp.c:794:isakmp_ph1begin_i():
  initiate new phase 1 negotiation: 192.168.1.5[500]<=>192.168.2.5[500]
2003-01-21 21:11:37: INFO: isakmp.c:799:isakmp_ph1begin_i():
  begin Identity
Protection mode.
2003-01-21 21:11:37: INFO: vendorid.c:128:check_vendorid():
  received Vendor
```

```
ID: KAME/racoon
2003-01-21 21:11:37: INFO: vendorid.c:128:check_vendorid():
  received Vendor
ID: KAME/racoon
2003-01-21 21:11:38: INFO: isakmp.c:2417:log_ph1established():
  ISAKMP-SA
established 192.168.1.5[500]-192.168.2.5[500]
  spi:6a01ea039be7bac2:bd288ff60eed54d0
2003-01-21 21:11:39: INFO: isakmp.c:938:isakmp_ph2begin_i():
  initiate new
phase 2 negotiation: 192.168.1.5[0]<=>192.168.2.5[0]
2003-01-21 21:11:39: INFO: pfkey.c:1106:pk_recvupdate():
  IPsec-SA established:
ESP/Transport 192.168.2.5->192.168.1.5 spi=68291959(0x4120d77)
2003-01-21 21:11:39: INFO: pfkey.c:1321:pk_recvadd():
  IPsec-SA established:
ESP/Transport 192.168.1.5->192.168.2.5 spi=223693870(0xd554c2e)
```

Further Options Using IPSec and Racoon

Racoon can additionally use X.509 certificates for the authentication of the peer. The peer can have a static IP address, as shown in the examples, or a dynamic IP address (roadwarrior). Then the IP address of the peer in the Racoon configuration file is replaced by **anonymous**. Racoon can then even create the required IPSec SAs itself.

The newest Linux kernels even start supporting NAT-traversal. Unfortunately this support is not fully functional yet. Applying network address translation to AH packets breaks the AH protocol because the AH protocol authenticates even the IP addresses. The ESP protocol can be handled by NAT devices and does not break. But usually all NAT devices keep the state information about the NAT'ed connections in a state table, which is organized by the used protocol and the port. ESP does not have any ports like TCP or UDP. Therefore, most NAT devices only support one NAT'ed ESP connection. A second separate connection breaks the first. The NAT-traversal solves this problem by encapsulating the IKE and the ESP packets in UDP packets. The UDP protocol supports the ports and can easily be NAT'ed by most devices.

Ralf Spenneberg has used Linux since 1992 and worked as a systems administrator since 1994. For the past five years, he has been working as a freelancer in the Linux/Unix field. Most of the time he provides Linux/Unix training. His specialty is network administration and security (firewalling, VPNs, intrusion detection). He has developed several training classes used by Red Hat and and other IT training companies in Germany. He has spoken at several SANS conferences and even more Linux/Unix specific conferences. He was chosen to be member of the program committee of the Linux Kongress and the GUUG Frühjahrsfachgespräch. Last year, he published his first German book, Intrusion Detection für Linux Server*.*