Tinc, the Userspace VPN Daemon

Own a Private Network

Tinc is a VPN daemon that can tunnel complete networks without modifying the kernel. Admins will particularly appreciate its simple installations, especially when faced with large-scale VPN installations: Tinc achieves far quicker integration of additional nodes than FreeS/WAN.

BY RALF SPENNEBERG

Tinc [1] has quite a few things going for it as an alternative to the ubiquitous FreeS/WAN: You do not need to patch the kernel as the program runs in userspace. It uses very simple configuration files making it easy to avoid errors. Tinc can also tunnel and thus protect non-IP traffic: you can use this technique to tunnel any Ethernet frame that can be encapsulated in IP. In contrast to this Freeswan can only protect IPv4 packets. Finally, Tinc has an extremely small footprint and does not waste host resources; the executable is a mere 80 KBytes.

Unfortunately, there is a downside to Tinc; it does not use a standard protocol and this often prevents interoperability between different operating systems, although native Win32 support was introduced with version 1.0. Userspace encryption takes longer than a comparable kernel feature. In the past the protocol has been prone to security holes that a standard protocol might have avoided (see “Vulnerabilities”).

Installation

Version 1.0 stable of Tinc has just been released (under the GPL). This version is compatible to Tinc 1.0pre8, but not to older versions. To compile Tinc you will need OpenSSL (0.9.7), a current *gettext* package (0.12), and the Zlib and LZO libraries.

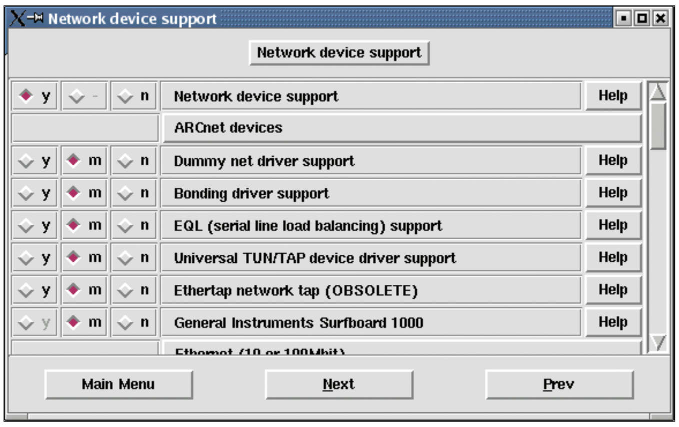
You can launch into the typical installation process:

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

As the installation is quite difficult, especially on Red Hat Linux 9, you might like to download the RPM packages I created for this distribution from [2].

Tinc uses either the *ethertap* driver (Linux 2.2) or the universal TUN/TAP driver (Linux kernel 2.4 or later, Figure 1) to communicate with the Linux kernel. It is easy to check whether your current kernel supports this driver. *modinfo tun* should output the path to the module.

Tinc also runs on FreeBSD, OpenBSD, NetBSD, Solaris, and MacOS X, as well as Windows. The latter uses the Cipe driver, which runs on both Windows 2000 and XP – older Windows versions will need a Cygwin environment.



```
Network device support

Network device support
y  -  n   Network device support                         Help
          ARCnet devices
y  m  n   Dummy net driver support                       Help
y  m  n   Bonding driver support                         Help
y  m  n   EQL (serial line load balancing) support       Help
y  m  n   Universal TUN/TAP device driver support        Help
y  m  n   Ethertap network tap (OBSOLETE)                Help
y  m  n   General Instruments Surfboard 1000             Help
          Ethernet (10 or 100Mbit)

Main Menu        Next        Prev
```

Figure 1: For the Linux 2.4 kernel, you need to enable *Universal TUN/TAP device driver support* below *Network device support*. This module is typically enabled

Digging Tunnels

In contrast to other VPN implementations, the admin

Listing 1: Tunnel Endpoints

```
Name = Berlin
ConnectTo = London
ConnectTo = Paris
Device = /dev/net/tun
PrivateKeyFile =
/etc/tinc/rsa_key.priv
```

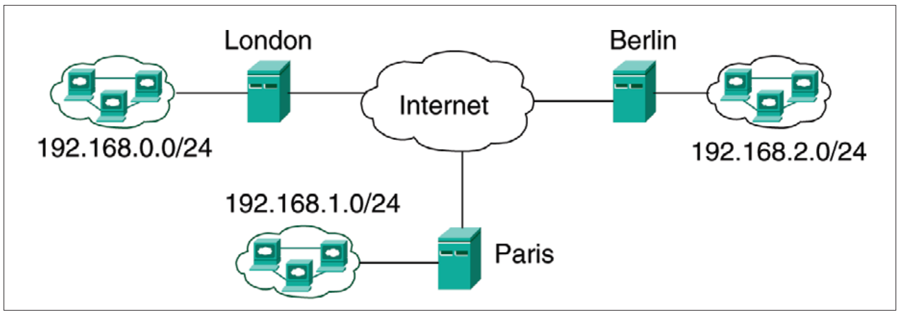


Figure 2: Tinc provides for simple deployment of VPNs with multiple endpoints. To add a node, the admin simple edits two configuration files

itself, when using Tinc does not define the tunnel but just its endpoints. This considerably simplifies the configuration process, as you would otherwise need to define six tunnels to support four endpoints, for example. Tinc does not distinguish between clients and servers: every Tinc daemon runs simultaneously in both client and server mode.

The configuration file, */etc/tinc/tinc.conf*, as shown in Listing 1, is the basis for the VPN shown in Figure 2. You also need to create a file in */etc/tinc/hosts/* for each name specified here (Berlin, Paris, and London). For example, */etc/tinc/hosts/Berlin* would contain the following two lines:

```
Address = berlin.spenneberg.org
Subnet = 192.168.2.0/24
```

The *Address* entry specifies the DNS name or IP address of the Berlin host. *Subnet* defines the network that will use this endpoint to communicate across the VPN. The VPN admin will need to create an RSA keypair to allow the endpoints to authenticate each other: *tincd -K* stores the private key in */etc/tinc/rsa_key.priv*; the tool appends the public key to a file below */etc/tinc/hosts/*.

After authenticating the partner node, Tinc creates a new network interface to support VPN communications. The configuration for this interface is stored in the */etc/tinc/tinc-up* file and contains the following lines.

```
ifconfig $INTERFACE 192.168.↵
0.1 netmask 255.255.0.0
```

The netmask is important here. It must support all the networks that will communicate across the VPN – a kind of supernetting.

After configuring all the hosts, creating all the keys and exchanging */etc/tinc/hosts/** files, the admin can now launch *tincd* on the hosts. The order is not important.

Peeking Behind the Curtains

Any tunnel that Tinc sets up comprises of two connections that both use port 665 (TCP and UDP). As the developers have registered this port with the IANA, you can even use the following */etc/services* entries:

```
tinc    665/tcp TINC
tinc    665/udp TINC
```

Tinc relies on the UDP protocol to exchange encrypted packets. It uses the Blowfish algorithm (this is configurable) with a 128-bit keylength in CBC mode (Cipher Block Chaining). Tinc also uses a 32-bit sequence number and a four byte Message Authentication Code (MAC, the code length is configurable). The MAC is computed by applying the SHA 1 algorithm to the packets. A combination of the sequence number and MAC protects the protocol from replay attacks.

Before Tinc can start packet encryption, it needs to authenticate the partner node and create symmetrical session keys. This meta-information is exchanged across the TCP connection, and this is why the Tinc documentation refers to it as a Meta connection.

Conclusion

Tinc is a mature application that allows admins to implement complex VPN scenarios quickly. Although interoperability with some operating systems cannot be guaranteed, Tinc is available for Linux, many BSD variants, Solaris, MacOS X, and Windows. ■

Vulnerabilities

In August 2000 a security hole was discovered in Tinc's key exchange routine. This vulnerability affected Tinc versions up to and including 1.0pre2, but has been patched since. The tried and trusted OpenSSL library now handles key exchanges, using 1024-bit RSA keys to authenticate.

Jerome Etienne investigated Version 1.0pre4 of the Tinc protocol and disclosed his findings on December 29 2001 [3]. He criticized the lack of a sequence number and packet authentication, which exposed Tinc to replay attacks or packet manipulation. These vulnerabilities were fixed in version 1.0pre5. Just like other VPN protocols (such as IPsec), Tinc now uses sequence numbers and a Message Authentication Code (MAC) that allows the receiver to detect modified packets.

Also, Jerome noted that Tinc used an extremely short (only 16-bit) random initialization vector (IV). The IV is required in CBC mode and should be as long and unique as possible. As Tinc used a random number, there was no way to guarantee the uniqueness of the vector. The protocol now uses the sequence number as the IV. The sequence number has a 32-bit length and is unique (within limits). Unfortunately, these changes have meant that protocol versions prior to 1.0pre8 are incompatible to the current version.

This goes to show that non-standardized protocols often have design flaws. Luckily, flaws are easily found and remedied in Open Source products, and Tinc is a good example of this.

INFO

[1] Tinc: *http://tinc.nl.linux.org*

[2] RPM packages: *http://www.spenneberg.org/VPN/Tinc/*

[3] Tinc security hole: *http://www.off.net/~jme/tinc_secu.html*