VPN with Tinc

Tinc is open source software for creating VPNs, virtual private networks over other physical channel such as the Internet, where individual participating hosts (nodes) appear to applications as if connected by wire in LAN.

Overview

Tinc utilizes asymmetric cryptography. Each node has its own private key, a public key and another public key; one for each participating node. These files are, together with a few configuration files, stored in /etc/tinc/<VPN name> directory.

Each node also runs a daemon (or multiple daemons, one for each separate VPN). Daemon listens on set port (default is 655) for incoming connections from other nodes. Only nodes with valid private keys can produce data decipherable with matching public keys and are thus granted access.

Public key file may contain not only key itself, but also public IP address (and port) of node to which it belongs. If set to, daemon will not wait for connections, but will attempt to connect to these known nodes.

Each node has its own IP address (in private address space) which, once the daemon is running, is assigned to virtual network interface. Any traffic coming from VPN is processed by the daemon and made come from that network interface, and any traffic send through that interface is also processed by the daemon and sent to VPN, all behind the scenes, transparent to applications.

Important feature of Tinc is that daemon can (and by default does) forward traffic for other nodes, e.g. if nodes A and B are behind NAT and can directly communicate with only node C, which has unrestricted internet access, or even do not know public key of each other, but C knows them both, C will happily forward traffic between/for them. They just need to know IP addresses (in private address space).

Installation

Compile using SlackBuild

- Tinc is now apparently maintained as SlackBuild.

Compile from source

- Download sources from http://www.tinc-vpn.org/download/
- Unpack and compile.

```
# ./configure --prefix=/usr --sysconfdir=/etc --localstatedir=/var
# make
```

```
# make install
```

- If you prefer to have Tinc somewhere aside, change prefix. Try DESTDIR if you want to temporarily install somewhere else:

```
$ make DESTDIR=/somewhere/else install
```

Configuration

- Either create initial configuration (also generates private and public keys for node), which means directory in /etc/tinc named after VPN with some initial files it.

```
# tinc -n VPNtest init node1
```

- Or generate only private/public key pairs. Place private keys to /etc/tinc/<VPN name>, merge public keys into one file and place it further into subdirectory /etc/tinc/<VPN name>/hosts.

```
$ tinc -c . generate-keys
$ mkdir -p VPNtest/hosts
$ mv *.priv VPNtest/.
$ cat rsa_key.pub ecdsa_key.pub > VPNtest/hosts/node1
$ rm rsa_key.pub ecdsa_key.pub
```

- Fine-tune configuration in /etc/tinc/<VPN name>/tinc.conf. Skip ConnectTo if deamon should passively wait for connections. Interface is name of virtual network card, see more below. Optionally set listening port, especially if you intend to run multiple daemons/VPNs.

tinc.conf

```
Name = node1
ConnectTo = node2
Interface = vpnNIC
Port = 6655
```

- Configure virtual network interface in /etc/tinc/<VPN name>/tinc-up. Do not manually create interface (via ip command), Tinc daemon will do that for you, just write down configuration for IP level. Also, make tinc-up file executable.

tinc-up

```
#!/bin/sh
ip addr add 192.168.1.1/24 dev vpnNIC
ip route add 192.168.1.0/24 dev vpnNIC
ip link set vpnNIC up
```

- Create tinc-down script that deconfigures VPN interface when VPN shuts down. Make the file executable too.

tinc-down

```
#!/bin/sh
ip link set vpnNIC down
ip route del 192.168.1.0/24 dev vpnNIC
ip addr del 192.168.1.1/24 dev vpnNIC
```

- Fine-tune public key file in /etc/tinc/<VPN name>/hosts/<this node>. Public IP may be also a hostname/domain, which is convenient in case you e.g. change ISP, but keep DNS name. Port should be same as in tinc.conf, but may differ if e.g. you are behind NAT with port forwarding from one port number to different port number. Let other nodes have this file and place their public key files here.

node1

```
Address = <public IP address> [port]
Subnet = 192.168.1.1/32
-----BEGIN RSA PUBLIC KEY-----
...
```

- Repeat process on (or for) other nodes, use different names for nodes and different private space IPs. Again, let nodes have each other's public key (or host) file.
- Start daemon, optionally specify debug level (0-5 where 5 is most eloquent) and where-to-log file.

```
# tincd -n VPNtest --debug=5 --logfile=/var/log/VPNtest.log
```

Windows

For sake of completeness, as you might want to e.g. build a VPN with Linux machine as fileserver accessed by Windows, remote-manage bunch of Windows behind NAT from Linux, play games, whatever, let's cover also Windows (XP, 7 and 8 are known to work).

Installation

- Download binary package.
- Install Tinc software.
- Preferably uninstall any possible existing TUN/TAP devices (virtual NICs). Tapinstall utility is part of Tinc package, should be in its install dir somewhere.

```
C:\path\to\tapinstall.exe remove tap0901
```

- Install new TUN/TAP device.

```
C:\path\to\tapinstall.exe install OemWin2k.inf tap0901
```

- Device drivers actually seem to come from OpenVPN project. Which is good, because they are signed; Windows are quite hostile towards unsigned drivers lately.

Configuration

There are a few differences in Windows configuration.

- You still generate initial configuration files, but place them into where Tinc is installed, which should be something like C:\Program Files\tinc\<VPN name>
- In tinc.conf, omit Interface directive, because Tinc daemon will then automatically select TUN/TAP device and directive may do more harm than good. Especially if Tinc service starts and fails immediately, check that Interface is not set.
- Tinc-up script is not used on Windows. You created persistent TUN/TAP device during installation (did you?) and now only manually configure IP (run ncpa.cpl, see properties of device and so on). This can be also scripted with command such as:

```
netsh interface ip set address name="Local Area Connection number" static
<IP address> <mask>
```

- But be warned: when created, TUN/TAP device can aquire pretty much any number in name, most likely 2, but not always.
- Finally, install (and start) Tinc service:

```
C:\path\to\tincd.exe --debug=5 --logfile=C:\path\to\file.log -n VPNtest
```

- Or, if service already exists, start service:

```
cmd> net start tinc.VPNtest
```

RC script

Here is some script to start all VPNs on boot. Note that stop command differs between 1.0 and 1.1 (prerelease) branches; 1.0 calls tinc**d**, 1.1 calls tinc (no d).

```
#!/bin/sh

VPNS=$(ls /etc/tinc)

start () {
    for VPN in $VPNS; do
        echo "Starting tinc daemon for $VPN..."
        /usr/sbin/tincd -n "$VPN" -d1 --logfile=/var/log/tinc."$VPN"
    done
}
```

```
stop () {
    for VPN in $VPNS; do
        echo "Stopping tinc daemon for $VPN..."
        /usr/sbin/tinc -n "$VPN" stop
    done
}

restart () {
    stop
    sleep 1
    start
}

case "$1" in
    ("start")
        start
        ;;
    ("stop")
        stop
        ;;
    ("restart")
        restart
        ;;
    (*)
        echo "Usage: $0 <start|stop|restart>"
        exit 1
esac

exit 0
```

Save it as e.g. /etc/rc.d/rc.tinc, make executable and then add line to rc.local.

rc.local

```
/etc/rc.d/rc.tinc start
```

Sources

Tinc website/documentation http://www.tinc-vpn.org

howtos, network