Herramientas criptográficas

David Fernández Vaamonde <davidfv@alfa21.com>

Jornadas de Criptografía Aplicada

Universidade da Coruña

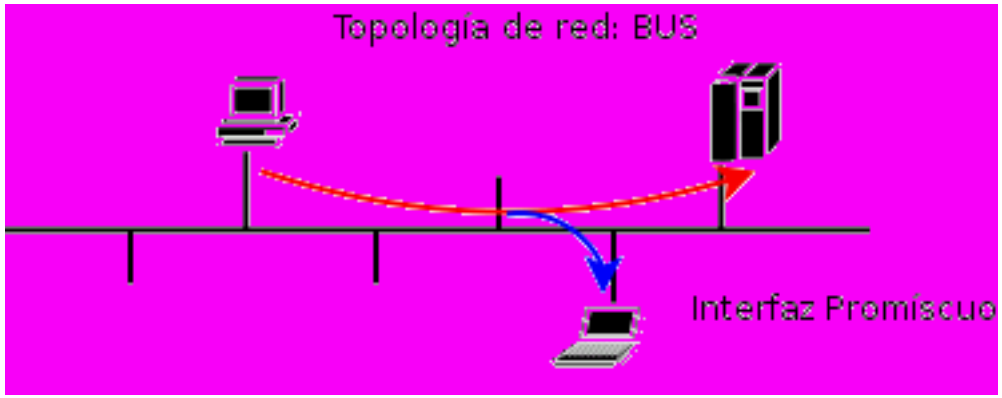
Guión

- 1) Experimento: Peligros del sniffing
- 2) Experimento: Demostración de Esteganografía
- 3) Sesiones seguras: CryptCat y SSH
  - Netcat y CryptCat
  - SSH: Usos habituales
  - Llaves SSH
  - Tuneles SSH
  - Trucos y atajos de teclado
- 4) Redes Privadas Virtuales: OpenVPN
  - Concepto de red privada virtual
  - Alternativas OpenSource
  - OpenVPN
    - Túneles sin seguridad
    - Túneles con clave
    - Túneles con certificados
  - Tunning de OpenVPN

Peligros de sniffing

¿Que es el "sniffing"?

- **Intercepción de las comunicaciones a través del medio**
- **Tipico en redes basadas en bus: Interfaces "promíscuos" (hubs)**

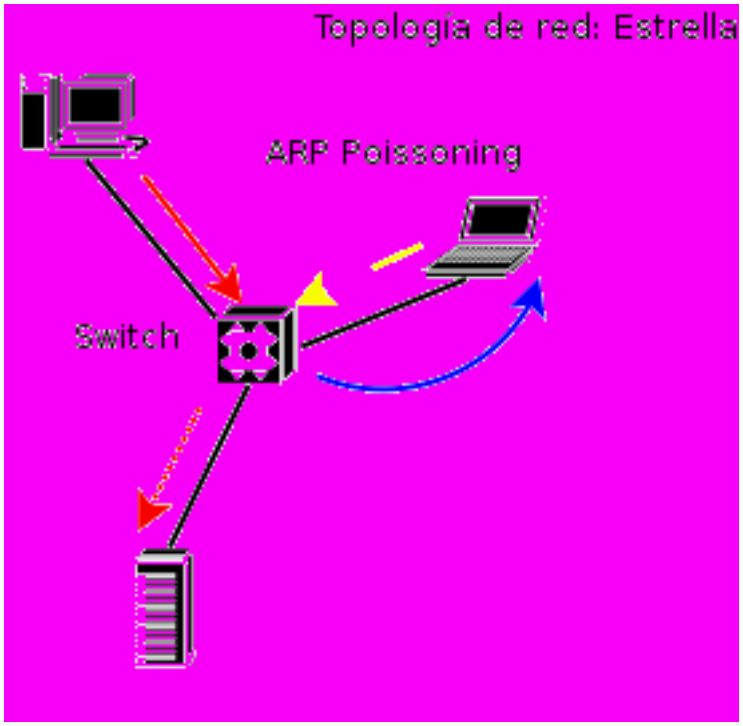


- **El interfaz toma como suya cada trama de la red**

Peligros del sniffing (y II)

¿Solo en topologías de BUS? ... NO

- **También se puede hacer sniffing en topologias en estrella (switchs)**
- **Envenenamiento ARP (ARP Poissoning)**



- **Se "envenena" la cache ARP del switch hacia la máquina objetivo.**

Experimento: Peligros del sniffing

- **Herramientas:**
  - **Sniffit : Sniffer (modo promíscuo)**
  - **TCPDump : Visor de tráfico (modo promíscuo)**
  - **IPTraf : Visor "gráfico" de tráfico (modo promíscuo)**
  - **ArpSpoof (dSniff) : Envenenador de caches ARP**

- **1er Experimento: Sniffit en una máquina para ver su tráfico**
  - **Uso de SSH, uso de telnet, uso de netcat y cryptcat**

```
# sniffit -i eth0
```

- **2o Experimento: Sniffit y ARPspoof para envenenar la caché ARP**
  - **Uso de SSH, telnet, netcat, IPtraf**

```
# arpspoof -t 192.168.0.2
# sniffit -i eth0
```

Experimento: Demostración de esteganografía

¿Qué es la esteganografía?

Del griego: "steganos" (encubierto) + "grafos" (escribir)

- **Criptografía: Su potencia es la de no poder comprender el mensaje.**
- **Esteganografía: Su potencia es la de ni tan siquiera saber que este existe.**

Escondemos mensajes en ristras de bits (generalmente imagenes o ficheros de audio), estos podrán ser recuperados con una cierta clave.

Experimento: Demostración de esteganografía (y II)

¿Qué es la esteganografía?

- **Se intercala el mensaje en bits redundantes del objeto "huesped"**
- **Se recupera en base a una clave dada**
- **Se trata de que nadie sepa que en el objeto hay un mensaje oculto**
- **Existe ataques de "criptografía diferencial", que se basan en la búsqueda repetitiva de patrones en el objeto.**

Experimento: Demostración de esteganografía (y III)

Uso de software esteganográfico:

- **Programa: steghide**
- **Demostración... ¡LLevais toda la charla delante de ella!**
- **Extracción de un texto:**
  - **# steghide extract -sf fondo.jpg**
- **Creación de una imagen o audio esteganográfico:**
  - **# steghide embed -cf objeto -ef archivo**
- **Herramientas de ataque: xsteg, stegdetect, stegbreak**
  - **Xsteg: frontend sobre las otras, distintos tipos de ataque**

Sesiones seguras: Cryptcat

Cryptcat --> Versión segura de netcat

Netcat --> La "navaja suiza" de las redes ;)

- **Uso de NetCat:**
- **Para escuchar en un puerto y sacar lo que entra a la entrada standard**
  - **# nc -l -p 4500**
- **Para escribir en un puerto desde la entrada estandar**
  - **# nc localhost 4500**

Sesiones seguras: Cryptcat (y II)

- Algunos usos divertidos de Netcat:
  - # dd if=ficherito | nc localhost 4500
  - # nc -l -p 4500 > ficherito
- Pero no va encriptado. Solución: cryptcat
  - # dd if=ficherito | cryptcat -k clave localhost 4500
  - # cryptcat -k clave -l -p 4500 > ficherito
- Cryptcat nos da una capa encriptada para los "hacks" que se pueden hacer con netcat.

Sesiones seguras: SSH uso habitual

¿Qué es SSH?

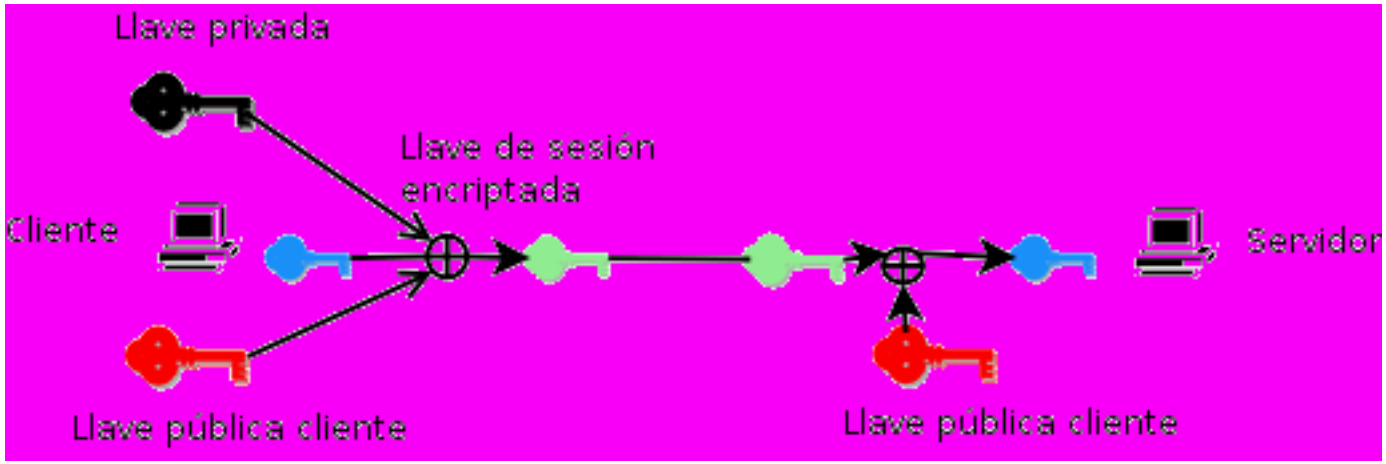
- Secure Shell: "¡Ah! Eso de conectarse seguro y tal..."
- ... NO, mentira, falso, caca!
- SSH permite hacer una sesion interactiva segura, y mucho más.
- Uso habitual:
  - # ssh davidfv@testral
  - # ssh davidfv@testral -v
  - # ssh testral
  - # scp davidfv@testral:fichero /home/davidfv

Sesiones seguras: Llaves SSH

- **Mecanismo:**
  - **Uso de llaves publicas para la autenticación**
  - **Uso de cifrado de clave pública:**
    - **Envío de una "llave de sesión" única y cifrada.**
    - **El cliente desbloquea la llave privada.**
    - **Se cifra con la llave publica del servidor y privada del cliente.**
    - **Si el servidor descifra la "llave de sesión", comienza la comunicación.**
- **Beneficios del uso de llaves:**
  - **Se usa la clave de la llave privada y no la del usuario**
  - **Se acepta una llave concreta desde un equipo concreto.**

Sesiones seguras: Llaves SSH (y II)

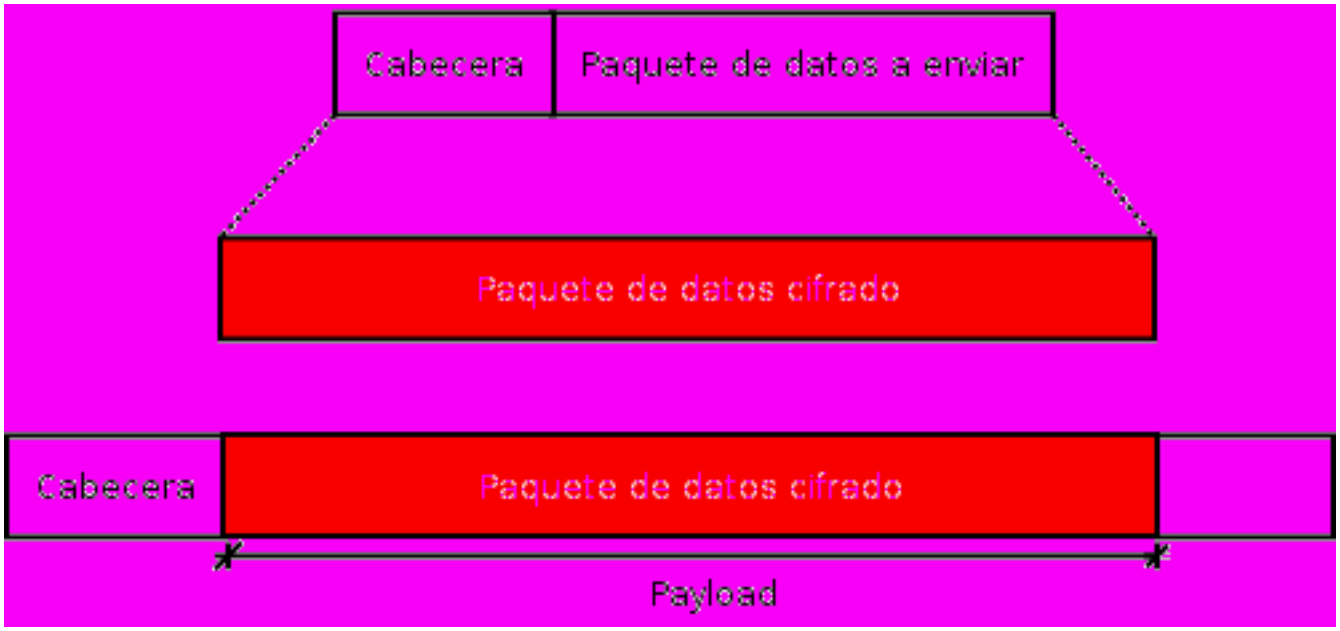
- **Generación de una llave:**
  - **# ssh-keygen -t dsa**
  - **Llaves: id_dsa, id_dsa.pub**
- **Cliente:**
  - **Fichero "config"**
    - **Host testral**
    - **StrictHostKeyChecking ask**
    - **IdentityFile ~/.ssh/id_dsa**
- **Servidor:**
  - **Copiamos .pub a .ssh del "servidor"**
  - **Añadimos .pub al fichero .ssh/authorized_keys**
    - **# cat id_dsa.pub >> authorized_keys**



Sesiones seguras: Túneles SSH

- **Problemática:**
  - Transmisión de datos en un medio inseguro: Internet
    - Intercepción de la comunicación
    - Suplantación de la comunicación
  - Figura de los "road-warrior"

- **Solución:**
  - Túneles cifrados.



Sesiones seguras: Túneles SSH (y II)

Túneles SSH

- **Características:**
  - Solución de tunel totalmente ad-hoc (para algo puntual)
  - Forwarding cifrado de puertos hacia la máquina local.
  - Opciones para tuneling:
    - Túnel encriptado para X (-X)
    - Túneles directos
    - Túneles inversos

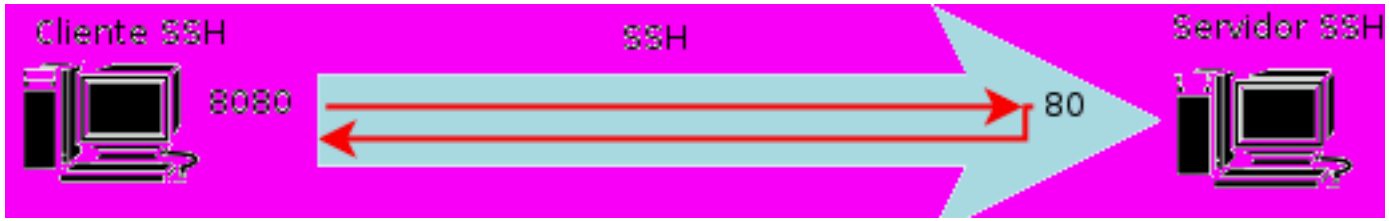
Sesiones seguras: Túneles SSH (y III)

- **Túnel cifrado para X**
  - Cifra conexiones de X
  - No es necesario exportar el display
  - Ha de estar permitido por el servidor

  - $ ssh -X davidfv@testral

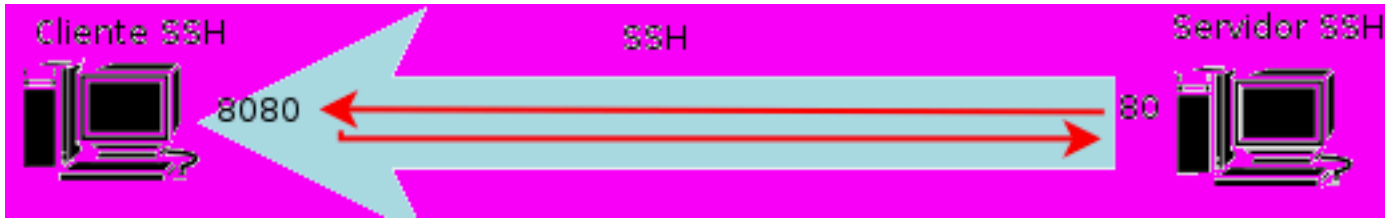
- **Túnel directo**
  - Forwarding de un puerto a local.
  - El cliente inicia la conexión.

  - $ ssh testral -L 8080:localhost:80
  - $ ssh testral -L 8080:sidh:80

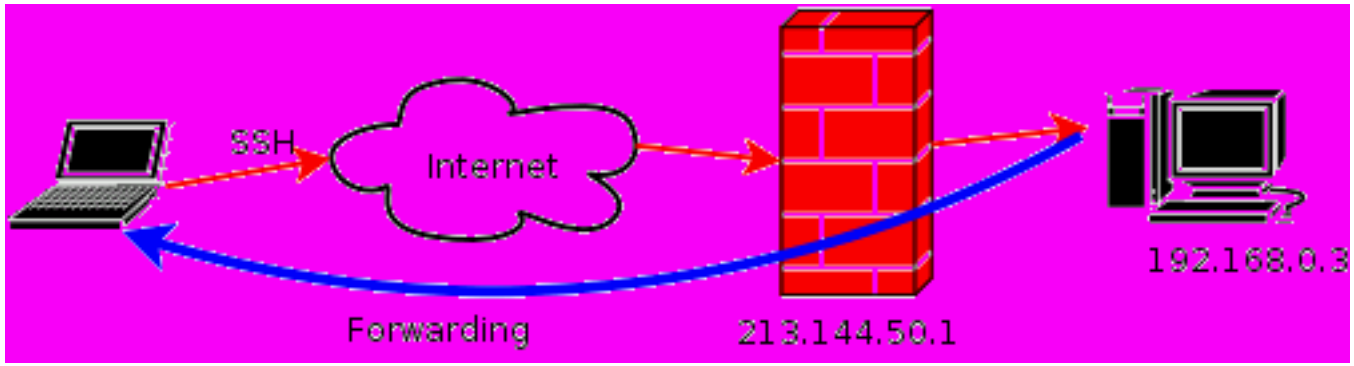


Sesiones seguras: Túneles SSH (y IV)

- **Túnel inverso**
  - **Forwarding de un puerto desde el servidor al cliente**
  - **Permite establecer el forward desde el servidor**
  - **$ ssh testral -R 8080:localhost:80**



- **Paso a través de un firewall:**
  - **$ ssh 213.144.50.1 -L 8080:192.168.0.3:80**

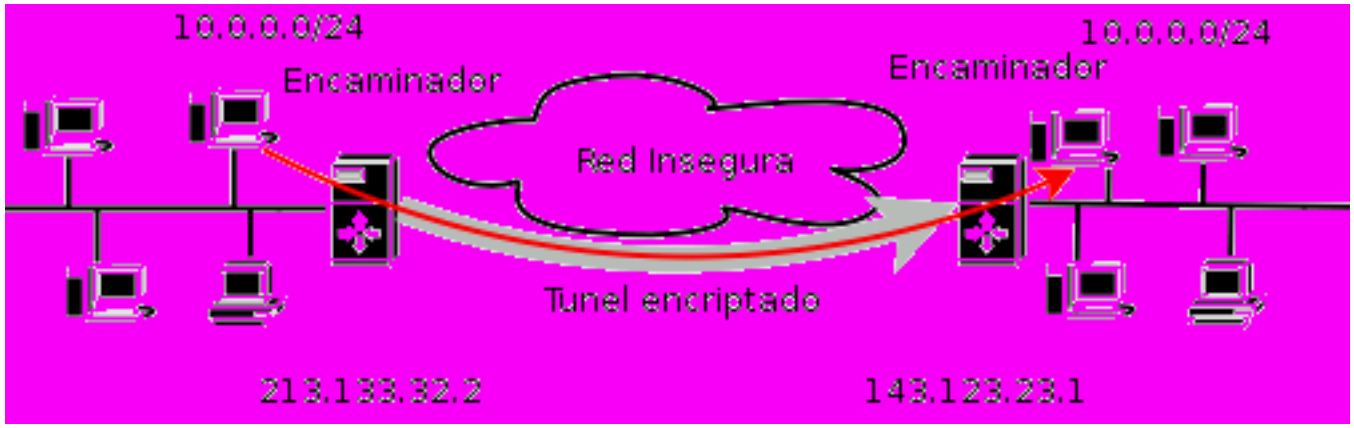


Sesiones seguras: Trucos y atajos de teclado

- **Ejecución remota de un comando**
  - **$ ssh davidfv@testral -C "ls -la"**
- **Backup remoto**
  - **$ ssh davidfv@testral -C "tar cz directorio" > p.tar.gz**
- **Teclas:**
  - **~. Aborta una sesion**
  - **~? Ayuda**
  - **~# Ver conexiones forward**

Redes Privadas Virtuales: Concepto de VPN

- Configuradas con túneles (visto anteriormente).
- Rutan tráfico encriptado entre direcciones locales.
- Generalmente une máquinas en el mismo rango de red.
- Son transparentes al usuario (al contrario que SSH).
- Caso tipico:



Redes Privadas Virtuales: Concepto de VPN

- **Seguridad en redes privadas:**
  - Encriptación del tráfico
  - Autenticación entre extremos

- **Alternativas OpenSource**
  - IPSec en el kernel
    - Estandar mas robusto
    - Modifica el stack TCP/IP
    - De serie en IPv6
  - Cipe
    - Alternativa basada en UDP
    - Necesario compilar contra el kernel
  - PPTPd
    - Totalmente compatible con sistemas Microsoft

OpenVPN

Características de OpenVPN

- Se basa en UDP e interfaces tun
- No es necesario recompilar
- Usa OpenSSL
- Da encriptación y autenticación
- Alternativa que mostraremos.
- Multiples túneles basados en puertos
- Tres tipos de túneles:
  - Sin seguridad
  - Con clave
  - Con certificados

OpenVPN

- **Características de OpenVPN. Uso de tun.**
  - **Uso del dispositivo tun**
  - **Capa previa de red (tun0, tun1,...)**
  - **Permite que las aplicaciones lo usen como un ifaz de red.**
  - **Hace los cambios pertinentes sobre los paquetes**
  - **Interfaces "tun" punto a punto entre dos máquinas**

  - **Configuración**

    - **modprobe tun**
    - **mknod /dev/net/tun c 10 200**
    - **echo 1 > /proc/sys/net/ipv4/ip_forward**

OpenVPN

- Túneles sin seguridad
  - Ficheros de configuración en /etc/openvpn
  - /etc/openvpn/test (nodo 1)
    - dev tun
    - lport 5000
    - rport 5001
    - remote 192.168.0.7
    - ifconfig 10.0.0.1 10.0.0.2
    - up ./test-up
  - /etc/openvpn/test-remoto (nodo2)
    - dev tun
    - lport 5001
    - rport 5000
    - remote 192.168.0.40
    - ifconfig 10.0.0.2 10.0.0.1
    - up ./test-up
  - /etc/openvpn/test-up (script de arranque)
    - route add -net 10.0.0.0 netmask 255.255.255.0 gw $5

Túneles sin seguridad (y II)

- **Comentarios:**
  - dev -> Tipo de dispositivo a usar
  - remote -> Nodo remoto
  - ifconfig -> ip local, ip remota (del tunel)
  - up -> Script de configuración.
    - Parametros
      - interfaz
      - mtu local
      - mtu remoto
      - ip túnel local
      - ip túnel remota
      - init
    - Configuración de rutas y firewall
  - lport -> Puerto local
  - rport -> Puerto remoto

Redes privadas virtuales: OpenVPN (y II)

- Túneles con clave
  - Generación de las claves
    - # openvpn --genkey --secret llave.key
    - Llave de 2048 bits
    - Entropía:
      - 512 bits cifrado
      - 512 bits HMAC autenticación
  - Copia de la llave al otro extremo por medio seguro (disquete, scp...)
  - Nuevo parámetro en ambos ficheros de configuración:
    - secret /path/llave.key
  - Ventaja: Muy simple y relativamente seguro.
  - Desventaja: No tan seguro como la última opción :)

Redes privadas virtuales: OpenVPN (y III)

Túneles con certificados

- **Ventajas**
  - **Autenticamos ambos puntos del tunel con certificados**
  - **Ciframos la conexión (Usamos Diffie-Hellman para el intercambio de la clave de sesión).**
  - **Los certificados son firmados por una Autoridad Certificadora (CA)**
  - **Diferenciamos servidor y cliente.**

Redes privadas virtuales: OpenVPN (y IV)

Túneles con certificados (y II)

Generación de certificados y llaves.

- Certificado -> Público (.crt)
- Llave -> Privada (.key)

- Creación de una autoridad certificadora (CA):
  - # openssl req -nodes -new -x509 -keyout my-ca.key -out my-ca.crt
  - Tomamos este certificado como autoridad certificadora.

- Creación de los certificados de ambos lados del túnel:
  - # openssl req -nodes -new -keyout local.key -out local.csr
  - Firma del certificado por la CA:
  - # openssl ca -out local.crt -in local.csr

- Parametros Diffie-Hellman (Solo para el servidor)
  - # openssl dhpram -out dh1024.pem 1024

Redes privadas virtuales: OpenVPN (y V)

Túneles con certificados (y III)

Ficheros de configuración:

- **En el servidor:**
  - **tls-server**
  - **dh dh1024.pem**
  - **ca my-ca.crt (Certificado de la CA)**
  - **cert local.crt**
  - **key local.key**
- **En el cliente**
  - **tls-client**
  - **ca my-ca.crt**
  - **cert remote.crt**
  - **key local.key**

Tunning y otras directivas de OpenVPN

Directivas:

- persist-tun -> No cierra el dispositivo tun al perder conexión o relanzar
- persist-key -> No relee la llave al perder conexión o relanzar
- down <script> -> Ejecuta un script al finalizar la conexión
- inetd <wait|nowait> -> Configuración para inetd/xinetd
- comp-lzo -> Comprime los datos del túnel con lzo

Tunning y otras directivas de OpenVPN

- askpass -> Pregunta password de la llave privada (sin -nodes)
- ping <segundos> -> Ping que comprueba la conexión si no hay paquetes en <segundos>
- ping-exit, ping-restart <segundos> -> Sale/Reinicia si no hay datos en cierto tiempo
- Ejemplo: Líneas para road-warriors
  - ping 15
  - ping-restart 45
  - persist-tun
  - persist-key

Servidor de túneles

- Uso de xinetd (o inetd, con /etc/services modificado)
- Ejemplo:

```
service vpn_auth_1
{
 type          = UNLISTED
 port          = 2011
 socket_type    = dgram
 protocol       = udp
 wait          = yes
 user          = root
 server         = /usr/sbin/openvpn
 server_args    = --inetd --config /root/tuneles/vpn_auth_1 --inactive 600 --user nobody
}
```

Redes privadas virtuales: OpenVPN (y IX)

Servidor de túneles (y II)

- No se puede usar "remote" (ya la sabe ;))
- --inactive -> Para el daemon despues de N segundos de inactividad
- --user -> Corre el daemon como un usuario determinado
- --config -> Arranca con una configuracion determinada
- --inetd -> Indica que se correrá en inetd o xinetd

Conclusiones

- Miles (o millones ;)) de formas de asegurar comunicaciones
- Software libre y criptografía han de ser filosofías paralelas
- Las formas generales de asegurar comunicaciones: túneles encriptados
- SI, se puede hacer VPNs con Linux (y de muchas maneras ;))

Referencias

- http://openvpn.sf.net
- http://www.openssh.com
- http://www.mindrot.org/~djm/auug2002/ssh-tutorial.pdf
- http://www.robertgraham.com/pubs/sniffing-faq.html
- http://www.gpul.org

¡Gracias por venir!