criptografía aplicada

Herramientas Criptográficas II
(La venganza... ;))

David Fernández Vaamonde

david_fv<en>gpul<punto>org

II Taller de Criptografía aplicada
Universidade da Coruña

Guión

- Criptografía en autenticación
- Plugable Authentication Modules: PAM
- One Time Passwords: OPIE
- VPN: IPSec en Linux, OpenSWAN
- ...Bonus track...

Criptografía en autenticación

- Autenticación: Comprobar la identidad de algo
- Tokens de autenticación: Claves, Huellas,etc...
- Tokens más extendidos: Claves
- Claves en claro = Problema de seguridad

Criptografía en autenticación (y II)

- Solución: Cifrar las claves
- Ejemplos:
  - Claves UNIX: DES (8 char) o MD5 (255 char)
  - Cifrado en sentido único
- Problemas:
  - Autenticación “hardcoded” en los programas
  - Cada manera de autenticar distinta (“Cada maestrillo...”).
  - Se reprograma cada solución
  - No se pueden intercambiar en tiempo de ejecución

Plugable Authentication Modules

- Solución:
  - Desacoplar autenticación de los propios programas
- Plugable Authentication Modules: PAM
  - Separa autenticación de los programas
  - El programa pide autenticación y $DEITY proveerá ;)
  - General para todo el sistema, configurable en ficheros

Plugable Authentication Modules (II)

- Funcionamiento:
  - El programa requiere autenticación e indica el servicio.
  - Se resuelve en el fichero la autenticación para ese servicio.
  - Se llama a las librerías que implementa ese método.
  - Se devuelve el resultado al programa.

criptografía aplicada

Plugable Authentication Module (III)

- Ficheros de configuración:
  - `/etc/pam.conf`
  - `/etc/pam.d` (usado actualmente, un fichero por servicio).
- Elementos de configuración:
  - Nombre del servicio (samba, http, login...)
  - Tipo de módulo: auth, account, session o password
  - Flag de control: Reacción ante un acierto o fallo de aut.
  - Path al módulo
  - Argumentos del módulo
- Ejemplo:

```
auth   required  pam_unix.so nullok_secure
```

Plugable Authentication Modules (IV)

- Tipos de módulo:
  - **auth** : Establece que el usuario es quien dice ser. Asocia a un grupo.
  - **account** : Gestiona accesos que no se basan en autenticación (en horario, en recursos, etc...).
  - **session** : Tareas necesarias antes o despues de que sea dado el servicio (logging, montaje de directorios...)
  - **password** : Cuando se actualiza el token de autenticación.

Plugable Autentication Modules (V)

- Flag de control:
  - **required** : Se requiere éxito pero no se muestra el fallo hasta la ejecución completa.
  - **requisite** : Se requiere éxito pero si falla inmediatamente devuelve el control.
  - **sufficient** : Si tiene éxito asume éxito total y no ejecuta nada más.
  - **optional** : Se ejecuta pero su fallo no implica nada.

Plugable Autentication Modules (VI)

- Algunos módulos existentes en `/lib/security`:
  - `pam_unix`: Autenticación UNIX común (*auth*).
  - `pam_deny`: Deniega siempre (*auth*).
  - `pam_permit`: Permite siempre (*auth*).
  - `pam_securetty`: Solo terminal segura (*auth*)
  - `pam_mkdir`: Crea el home si no existe (*session*)
  - `pam_limits`: Permite o no según `/etc/security/limits.conf` (*account)*
  - `pam_time`: Acceso basado en tiempo `/etc/security/time.conf` (*account*).
  - `pam_env`: Carga una configuración por defecto (*session*)
  - `pam_cracklib`: Controla la calidad del password en un cambio (*password*)
  - `pam_motd`: Imprime el MOTD al entrar (*session*)

Plugable Authentication Modules (VII)

- Ejemplos de configuración:
  - Su:

```
auth       required   pam_wheel.so
auth       sufficient pam_rootok.so
@include common-auth
@include common-account
@include common-session
```

  - Password (en su cambio):

```
password   required   pam_unix.so nullok obscure min=4 max=8 md5
```

criptografía aplicada

criptografía aplicada

Plugable Authentication Modules (y VII)

- Ejemplos de configuración:
  - Un par de locuras:

```
auth        sufficient pam_permit.so
auth        required   pam_deny.so
```

  - Una política de denegación por defecto:

```
auth        sufficient pam_unix.so
auth        suffciente pam_deny.so
```

(Configuraciones en “fallback”)

One Time Passwords

- Soluciones aportadas por la criptografía en autenticación:
  - Intercepción (passwords cifradas)
  - Autenticación (X.509,...)
- Problema:
  - ¿Password siempre disponible en el sistema?
  - ¿Password anotada?
  - ¿Password demasiado debil?
  - ¿Queremos que esa passwords valga siempre?
  - ¿Otros usuarios conocen la password (keyloggers)?

One Time Passwords (II)

- Solución:
  - Passwords de un solo uso. Nuevo password en cada conexión.
- Modo de funcionamiento:
  - Desafío para entrar
  - Cálculo del password en local/Petición de clave a un administrador
  - Resolución del desafío
  - Próxima vez: Desafío distinto.

criptografía aplicada

One Time Password (III)

- Ventajas de OTP:
  - El password no viaja por la red, solo desafio.
  - Desafios dificiles de descifrar (MD5)
  - No es necesario forzar el cambio de password (implícito ;))
  - En algunos casos se une el desafio con un token de autenticación.
  - Se puede "controlar" a un usuario.

One Time Password (IV)

- Sistemas OTP libres:
  - **S/Key** → *Bellcore* (OpenSource), 1994 → Comercial (MD4)
  - **OPIE** → *Navy Research Lab* (MD4, MD5, SHA1), (solo openbsd)
  - ¿Port a Linux? :)

One Time Password (V)

- Funcionamiento:
  - `opiepasswd -c usuario`
  - Se introduce una clave y se inicializa el contador.
  - `opiekey numero_secuencia seed`
  - Nos da el one time password (calculado en local).
- Integración:
  - *Problema*: ¿Cambiamos todas las aplicaciones?
  - *Solución*: No → `pam_opie.so ;)`

One Time Password (y VI)

- Ejemplo:
  - PAM:

```
auth     required          pam_opie.so
```

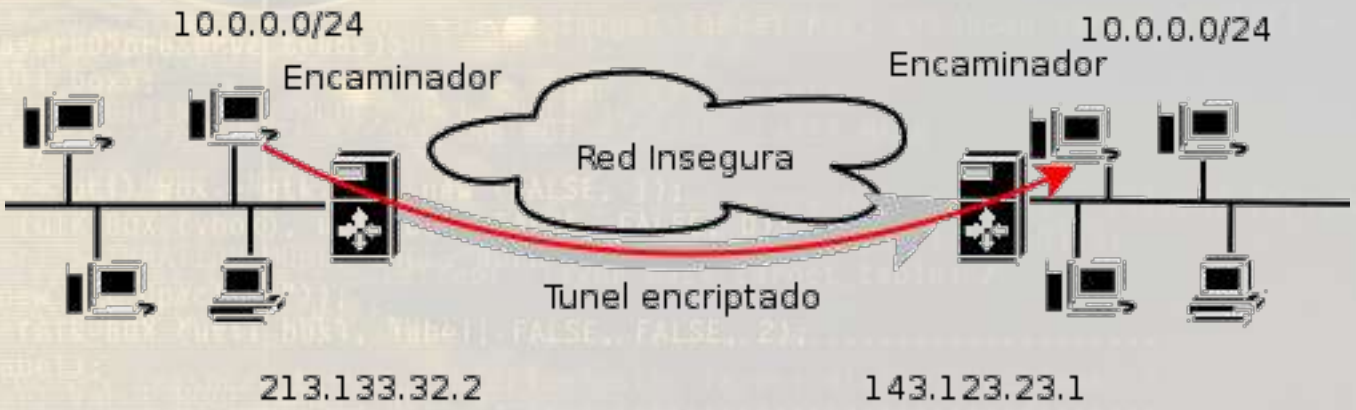
  - o:

```
auth     sufficient        pam_opie.so
auth     sufficient        pam_unix.so
```

- Excepción:
  - Algunas aplicaciones pueden dar problemas : SSH

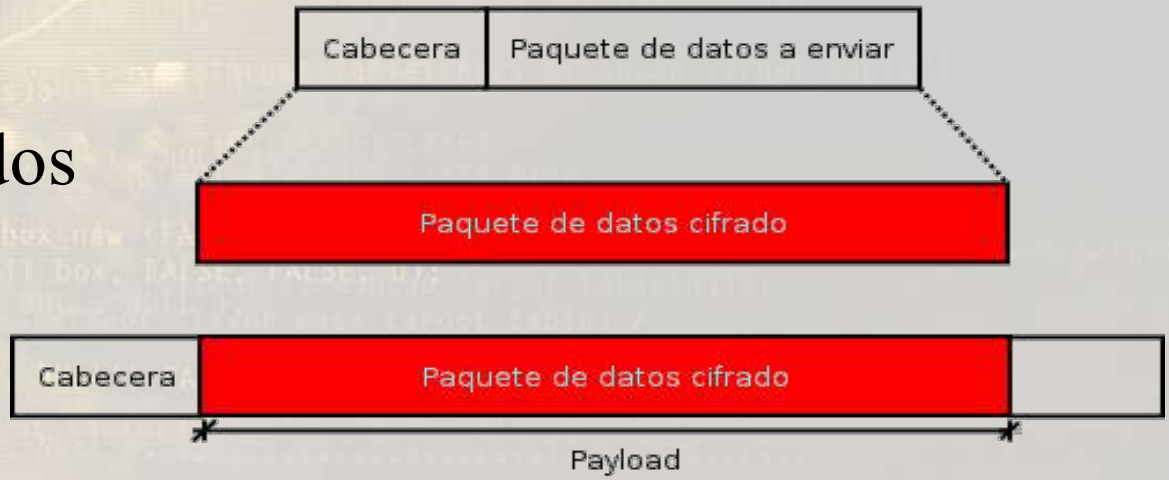
Redes Privadas Virtuales: Concepto de VPN

- Configuradas con túneles
- Rutan tráfico cifrado entre direcciones locales
- Generalmente une máquinas en el mismo rango de red.
- Son transparentes al usuario.
- Caso tipico:



Redes Privadas Virtuales: Concepto de VPN

- 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



Redes Privadas Virtuales: Concepto de VPN

- Seguridad en redes privadas:
  - Cifrado del tráfico
  - Autenticación entre extremos
- Alternativas Open Source:
  - OpenSWAN (Ipsec en el kernel)
    - Estandar más robusto
    - Modifica el stack TCP/IP
    - De serie en IPv6
  - CIPE
    - Alternativa basada en UDP
    - Necesario compilar contra el kernel

criptografía aplicada

Redes Privadas Virtuales: Concepto de VPN

- PPTP:
  - Totalmente compatible con sistemas Microsoft
- OpenVPN:
  - Más sencillo.
  - Interfaces tun como wrappers.
  - No modifica el stack.
  - No corre en el kernel.
  - Es complejo hacer un servidor de túneles

Redes Privadas Virtuales: IPSec/OpenSWAN

- Estándar IPSec:
  - Cifrado y autenticación a nivel IP
  - Protege todo el flujo de red (frente a SSL,PGP,SSH)
  - Otros operativos lo implementan (Windows, MAC)
- Implementación OpenSWAN:
  - Protocolo ***ESP*** (*Encapsulation Security Payload*)
    - Cifrado y autenticación
    - Implementado en el kernel (*KLIPS*)
  - Protocolo ***IKE*** (*Internet Key Exchange*)
    - Intercambio de claves (RSA,X.509)
    - Negociado de conexión
    - Implementado por *Pluto*

Ipsec / OpenSWAN. Ejemplos.

- Pasos para establecer una VPN:
  1. Generar una llave o un certificado para el equipo:
     - *ipsec newhostkey --output /etc/ipsec.secrets*
  2. Obtener la llave del equipo:
     - *ipsec showhostkey –left*
  3. Generar una nueva configuración de conexión:
     - *config setup*
     - *interfaces=”ipsec0=eth0”*
     - *conn ...*

Ipsec / OpenSWAN. Ejemplos.

- Configuración sencilla:

```
conn emain-tsunade
        left=192.168.1.2
        leftsubnet=192.168.1.0/24

leftrsasigkey=0sAQNoWz6LJf97Zhv+6ljpCCeaFr...

        right=192.168.1.1
        rightsubnet=192.168.1.0/24

rightrsasigkey=0sAQN9Znlz+wS/68CFW4/zJBPfL...
        auto=add
```

Ambos en la misma red (No es la configuración habitual)

Ipsec / OpenSWAN. Ejemplos.

- Configuración compleja:

```
conn emain-tsunade
        left=192.168.1.2
        leftsubnet=192.168.1.0/24

leftrsasigkey=0sAQNoWz6LJf97Zhv+6ljpCCeaFr...
        leftnexthop=
        right=192.168.1.1
        rightsubnet=192.168.1.0/24

rightrsasigkey=0sAQN9Znlz+wS/68CFW4/zJBPfL...
        rightnexthop=
        auto=start
```

Se unen elementos en distintas redes

criptografía aplicada

...Bonus Track... Peligros del Tunneling

- Beneficios del tunneling:
  - Túneles para seguridad:
    - *VPN,GRE,SSH*
  - Túneles para adaptación de redes
    - IPv6 (túneles *sit*)
- Peligros del tunneling:
  - Mas dificil restricción al paso de tráfico en firewall
  - Tráfico “camuflado” en un túnel
  - Es necesario una máquina “al otro lado” que desencapsule el tráfico
  - *Piercing* firewall

...Bonus Track... Peligros del Tunneling (II)

- Caso práctico: *`httptunnel`*
- Escenario:
  - Red tras un firewall que lo filtra todo
  - Solo se permite acceso a un proxy web o ftp
  - El acceso al exterior se realiza a través de un proxy.
- Piercing Firewall
  - Encapsular tráfico en peticiones HTTP o FTP
  - Algo al otro lado tendrá que reconocerlas
- HTTPTunel permite realizar ese túnel contra un proxy

...Bonus Track... Peligros del Tunneling (y III)

- Caso práctico. Realización.
  - Se arranca un servidor (*hts*) en la máquina externa.
  - El servidor entiende las peticiones y las redirige
    - `# hts -F emain.davidfv.net:22 2080`
  - En el cliente se arranca un servidor local
  - Toma la petición, la transforma en petición HTTP
  - La petición HTTP a emain.davidfv.net:2080 la envía al proxy
  - El proxy envía la comunicación a la máquina externa y devuelve resultado
    - `# htc -P proxy.uc...:3128 -F 15000 emain.davidfv.net:2080`
  - La conexión se hace contra un puerto local que está redirigido, por ejemplo:
    - `# ssh -p 15000 davidfv@localhost`