Departamento de
Automática y Computación
Automatika eta
Konputazio Saila

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

Práctica 6: Herramientas criptográficas

1- Introducción

La seguridad es muy importante dentro del mundo de la información (Internet). En el intercambio de información personal o comercial, las personas desean saber con quién se establece la comunicación (autentificación), asegurarse de que la información que envían es la que se recibe sin haber sido modificada durante su trayecto (integridad) y prevenir que intrusos intercepten dicha información (privacidad).

El protocolo TCP/IP ha resuelto varios problemas respecto a la comunicación; pero hay que tener en cuenta que no fue diseñado pensando principalmente en la seguridad. Por lo tanto, es necesario incorporar tecnologías de seguridad imprescindibles para resolver problemas como los siguientes:

- Cómo llevar a cabo la autentificación sin la necesidad de enviar un nombre (login) y contraseña (password) sobre la red?
- Cómo autentificar usuarios para asegurarse de que ellos son realmente los que dicen ser?
- Cómo proteger la privacidad de la comunicación; es decir, el flujo de datos a través de la red entre un cliente (ej. "navegador") y un servidor (ej. "servidor web")?
- Cómo asegurar que archivos confidenciales sean accedidos solamente por personas autorizadas?

Existe una tecnología que es la base para la solución de los problemas de seguridad antes mencionados: **la criptografía**.

2- La utilidad OpenSSL

Tenemos instalado en los PCs A, B y C el paquete de software OpenSSL, que es una implementación open source de SSL que puede descargarse de http://www.openssl.org Además de una API criptográfica de propósito general para programar aplicaciones SSL, dispone de una utilidad de línea de comandos llamada `openssl` que permite realizar todas las tareas criptográficas que el protocolo conlleva:

- Calcular hashes criptográficos (MD2, MD4, MD5, RIPEMD-160, SHA, SHA-1)
- Cifrar con algoritmos de cifrado en bloque (AES, CAST, DES, Tiple DES, Blowfish,RC2) o en flujo (RC4), y usar aquéllos en múltiples modos de operación (ECB, CBC, CFB, OFB).
- Generar claves asimétricas para los algoritmos habituales (Diffie-Hellman, DSA, RSA)
- Utilizar los algoritmos para firmar, certificar y revocar claves.
- Manejar el laberinto de formatos de certificados que existen en el universo (X.509,PEM, PKCS7, PKCS8, PKCS12).

La sintaxis del comando `openssl` es genérica y por tanto será válida para cualquier SO en el que esté correctamente instalado el paquete OpenSSL.

Podemos consultar todas la opciones de OpenSSL:

$ openssl help

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

```
openssl:Error: 'help' is an invalid command.

Standard commands
asn1parse      ca             ciphers        crl            crl2pkcs7
dgst           dh             dhparam        dsa            dsaparam
enc            engine         errstr         gendh          gendsa
genrsa         nseq           ocsp           passwd         pkcs12
pkcs7          pkcs8          rand           req            rsa
rsautl         s_client       s_server       s_time         sess_id
smime          speed          spkac          verify         version
x509

Message Digest commands (see the 'dgst' command for more details)
md2            md4            md5            mdc2           rmd160
sha            sha1

Cipher commands (see the 'enc' command for more details)
aes-128-cbc    aes-128-ecb    aes-192-cbc    aes-192-ecb    aes-256-cbc
aes-256-ecb    base64         bf             bf-cbc         bf-cfb
bf-ecb         bf-ofb         cast           cast-cbc       cast5-cbc
cast5-cfb      cast5-ecb      cast5-ofb      des            des-cbc
des-cfb        des-ecb        des-ede        des-ede-cbc    des-ede-cfb
des-ede-ofb    des-ede3       des-ede3-cbc   des-ede3-cfb   des-ede3-ofb
des-ofb        des3           desx           idea           idea-cbc
idea-cfb       idea-ecb       idea-ofb       rc2            rc2-40-cbc
rc2-64-cbc     rc2-cbc        rc2-cfb        rc2-ecb        rc2-ofb
rc4            rc4-40         rc5            rc5-cbc        rc5-cfb
rc5-ecb        rc5-ofb
```

Comprobación y verificación de código

OpenSSL soporta varios tipos de "huellas digitales" o digest algorithms, como: MD2, MD4, MD5, SHA, SHA1, MDC2 y RIPEMD-160 tal y como acabamos de comprobar. Cada algoritmo puede ser invocado directamente o como opción del comando `openssl dgst`.

Cree un archivo de texto con la cadena "Laboratorio de telemática 1" y saque su digest MD5:

```
$ openssl dgst -md5 archivo
$ openssl md5 archivo
```

¿Le sale distinto que al compañero?¿Por qué puede ser?

Cifrando (encriptando) datos

Cifrar es el proceso de convertir datos (generalmente texto plano) a un formato alternativo (texto cifrado) que sea diferente al original. El proceso de cifrado de datos generalmente requiere una clave y usa una serie de algoritmos para realizar la transformación de texto plano a texto cifrado.

Los algoritmos de clave simétrica (algoritmos de clave compartida) usan la misma clave para cifrar y descifrar datos. Los algoritmos de clave pública (algoritmos de clave asimétrica) usan diferentes claves para el cifrado y el descifrado. Los algoritmos de clave pública toman su nombre

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

por una de las claves que se utilizan, la clave pública, que puede ser distribuida a otras personas, los datos que son cifrados con una clave pública sólo pueden ser descifrados con la clave privada asociada.

OpenSSL soporta varios algoritmos de clave simétrica como: DES, DES3, IDEA, Blowfish y AES. Cada algoritmo de clave simétrica puede ser invocado desde la línea de comandos pasando el nombre del algoritmo en el comando openssl.

Cifre con DES3 el archivo de texto anteriormente creado:

```
$ openssl enc -e –des3 -in archivo -out archivo.enc.des3
```

Descifrando datos

Para descifrar archivo.enc.des3 usaremos la opción –d tras el algoritmo de cifrado en el comando openssl.

```
$ openssl enc -e -d -des3 -in archivo.enc.des3
```

Consulte la ayuda online:

http://www.openssl.org

http://dns.bdat.net/documentos/certificados_digitales/x249.html

Descargue el archivo `cadena.enc` de la web de la asignatura y descífrelo sabiendo que está cifrado con CAST5 en modo CFB y la contraseña es "telemat"

Checkpoint 6.1: Mostrar al profesor la cadena original

Certificados digitales (claves pública/privada)

Para llegar a tener un certificado, primero tenemos que tener un par de claves pública/privada. OpenSSL trabaja con los algoritmos de clave pública RSA (de Rivest, Shamir y Adelman), DSA (Digital Signature Algorithm) y DH (Diffie-Hellman). Vamos a trabajar con RSA, puesto que es el más sencillo de los tres para esta tarea.

Genera un clave privada RSA de 1024 bits usando el subcomando genrsa (empieza con openssl genrsa -h para ver las opciones disponibles). La clave debe tener 1024 bits, y tendrás que guardarla en un fichero que llamaremos `priv.pem`.

Observa el encabezamiento de la clave privada obtenida.

Extrae de la clave privada que acabas de crear, la clave pública (consulta la ayuda del subcomando rsa, dispone de una opción para ello). Guarda el resultado en `pub.pem`. Qué diferencias aprecias respecto la clave privada.

Checkpoint 6.2: Mostrar al profesor las claves obtenidas.

Creación de un certificado autofirmado

En el mundo real, solicitaríamos de una Autoridad de Certificación (por ejemplo, Verisign) que firmase nuestra clave pública vinculándola con nuestros datos identificativos. Esa solicitud es un

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

documento digital que se denomina Certificate Signing Request (CSR). Una solicitud así se crea con el comando:

```
$ openssl req -new -key priv.pem -out solicitud.csr
```

Dicho comando nos va a pedir una serie de datos, que serán los que nos identifiquen en el certificado. Se trata de campos de un nombre X.509. Como ves, tenemos que dar como argumento la clave privada, porque la solicitud va firmada por nosotros, y eso requiere el uso de dicha parte privada. La pública se deduce de ella.

Crear una CSR para la clave pública/privada generada en la sección anterior. Échele un vistazo y comprueba qué dice la cabecera.

Este documento se enviaría, e.g., a Verisign, y tras la verificación oportuna de quiénes somos (y el pago de una buena cantidad) Verisign nos extendería un certificado. Para evitar esto, vamos a firmar nosotros mismos la solicitud. El comando sería:

```
$ openssl req -x509 -key priv.key -in solicitud.csr
```

y el resultado debe ser un certificado.

Genera un certificado autofirmado de acuerdo con la instrucción anterior y depositalo en `cert.pem`.

Checkpoint 6.3: Mostrar al profesor mediante las distintas opciones del comando `openssl` los distintos campos x509, uno por cada opción `openssl`, del certificado `cert.pem`

Consulte la ayuda online:

http://dns.bdat.net/documentos/certificados_digitales/x249.html

Desde PC A, B ó C abra su navegador e indíquele la dirección https://127.0.0.1

¿Qué puede decir acerca del certificado SSL de su servidor Apache? ¿Qué tendría que hacer para que el navegador lo considerara como seguro?

3- Comunicaciones seguras con SSH

Conecte PCA y PCC a un conmutador de switch0(consulte la documentación de los armarios).

Asigne una dirección IP a ambos dentro del rango 10.3.armario.0/24

Analice, mediante wireshark, una conexión telnet desde PCA a PCc. ¿Puede ver los datos de login y contraseña?

Repita la operación mediante una conexión SSH ¿Qué ha cambiado? ¿Puede ver ahora la contraseña? ¿Qué protocolo se está utilizando?

- Analice el proceso de autenticación:

En su PCA elimine, si existiera, el archivo /home/ssi/.ssh/known_hosts

Vaya a PCC mediante el KVM(conmutador de teclado-ratón-monitor) y obtenga el fingerprint de las claves de PCC mediante el comando ssh-keygen(man ssh-keygen). Las encontrará en /etc/ssh/ ¿Tendría sentido calcular el fingerprint de una clave privada?

Conéctese ahora desde PCA mediante ssh a PCC ¿Qué fingerprint le muestra su cliente SSH? ¿Cómo lo ha obtenido? ¿De qué es ese fingerprint?

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

¿Qué clave pública nos está mandando PCC; RSA ó DSA? Calcular, ahora, en PCA el fingerprint de /home/ssi/.ssh/known_hosts le puede dar una pista.

¿Qué pasaría al conectarnos a PCC desde PCA mediante SSH, si borrásemos en PCC el par de claves privada/pública RSA y creásemos un nuevo par de claves?

Utilice wireshark, tcpdump o ettercap (sudo ettercap –T –z //22) para analizar la comunicación entre PCA - PCC y averiguar qué es lo que está pasando si es que no lo sabe ya.

Checkpoint 6.4: Muestre al profesor cuál es el mecanismo de encriptación que se está empleando y a qué corresponde el fingerprint obtenido al conectarse desde PCA a PCC mediante ssh.

SSH sin contraseña

SSH se puede configurar para que utilice certificados o claves DSA/RSA, de tal forma que podremos autenticarnos automáticamente en el servidor.

Para ello deberá generar un par de claves privada/pública y transferir una de ellas al servidor donde deseemos autenticarnos.

Utilice PCA como cliente desde el que conectarse mediate SSH al PCC que hará las veces de servidor.

Consulte el manual del comando `ssh-keygen` para generar un par de claves privada/pública DSA. Lance dicho comando desde `/home/ssi` y durante la generación del par de claves, acepte las opciones por defecto(pulsando enter) y no indique contraseña alguna (`passphrase`).

Una vez terminado el proceso, en la carpeta `/home/ssi/.ssh`, tendrá el par de claves privada/pública. Verifique el fingerprint obtenido. ¿A cuál de las claves generadas corresponde?

Ahora tiene que transferir una de sus claves al servidor(PCC) ¿Cuál; la privada o la pública?¿Por qué? Utilice para ello el comando `ssh-copy-id`(`man ssh-copy-id`).

En el servidor ¿Dónde se guarda la clave transferida? Verifique el fingerprint del archivo `authorized_keys`; debería coincidir con el fingerprint de su clave pública.

Con ésto `ssh-copy-id` habrá agregado la clave pública al archivo `/home/ssi/.ssh/authorized_keys` del usuario en el servidor remoto y la siguiente vez que nos conectemos la autenticación será automática sin necesidad de ingresar la contraseña nunca más.

Checkpoint 6.5: Mostrar al profesor que es capaz de conectarse desde PCA a PCC, sin introducir contraseña, y obtener directamente, sin teclear ningún comando:

-La configuración de la interfaz de red activa del PCC

-El nombre del equipo

-El directorio remoto

-Un listado de dicho directorio, con los permisos de los directorios y archivos correspondientes.

Consulte el enlace: http://www.uv.es/sto/articulos/BEI-2003-01/ssh_np.html para saber cómo restringir, en función de la identidad remota, los comandos que ésta puede ejecutar en el servidor.