Diseño Y Aplicaciones de Sistemas Distribuidos **Seguridad Seguridad**

**Joan Vila** 

***DISCA / UPV***

**Departament d’Informàtica de Sistemes i Computadors Universitat Politècnica de València**

****

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

**Seguridad**

**Indice**

–Introducción

–Criptografía

 Claves simétricas

 Claves asimétricas

–Firmasdigitales

–Autentificaciónydistribucióndeclaves –Casosdeestudio:

 Kerberos

 SSL

 Java 1.2

*2*

****Introducción**

 **Hipótesis básicas**

Para que la seguridad sea efectiva se necesita:

**Diseño Y Aplicaciones deSistemas Distribuidos**

– Realizar una **hipótesis** de las posibles **amenazas** a la seguridad

– **Validar**, mediante pruebas formales, las técnicas de seguridad utilizadas (campo de investigación muy reciente y activo ...)

– Utilizar técnicas para **auditar**, puesto que ninguna lista de amenazas es exhaustiva

– Técnicas de seguridad = **mecanismos + políticas**

 Un sistema de cerraduras no sirve de nada si no existe una adecuada política de utilización.

**DYA**

*3*

****Introducción**

 **Amenazas a la seguridad**

Para producir un sistema seguro hay que especificar las hipótesis o amenazas que se van a tener en cuenta.

**Diseño Y Aplicaciones deSistemas Distribuidos**

Estas hipótesis pueden contemplar la prevención de los siguientes daños: – **Filtración** (*leakage*): obtención de información por personas no autorizadas. – **Adulteración** (*tampering*): alteración desautorizada de la información. – **Robo de recursos**: uso de facilidades sin autorización.

– **Vandalismo**: interferencia a la labor del sistema que no produce beneficio al que la perpetra.

**DYA**

*4*

****Introducción**

 **Métodos de ataque**

Para lograr los fines anteriores hace falta un método de ataque o *modus operandi*: ¿como se perpetra el ataque?

**Diseño Y Aplicaciones deSistemas Distribuidos**

Uno de los puntos mas vulnerables de un sistema distribuido son los canales de comunicación.

– **Fisgoneo** (*eavesdropping*): obtener copias de mensajes sin autorización escuchando sobre un bus de difusión.

– **Mascarada**: enviar o recibir mensajes utilizando la identidad de otro usuario. – **Adulteración de mensajes**: interceptar mensajes y alterar su contenidos antes de enviarlos al destino especificado.

**DYA**

– **Retransmisión diferida**: almacenar mensajes y reenviarlos más tarde, cuando la autorización de uso de un recurso haya caducado.

*5*

****Introducción**

 **Infiltración**

Para lanzar el método de ataque, lo primero es tener acceso a una cuenta para ejecutar el ataque. Los métodos más usuales son:

**Diseño Y Aplicaciones deSistemas Distribuidos**

– **Abuso** por parte de usuarios legítimos.

– **Reventar contraseñas** (*password cracking*)

– **Virus**: programa que va adosado a un programa legítimo y que se instala cuando este se ejecuta. Se puede activar por diversos medios.

– **Gusanos**: programa que aprovecha las facilidades de ejecución en el nodo remoto (rsh, correo-e que ejecuta macros, ...).

– **Caballos de Troya**: Programa que se ofrece como un programa útil pero con efectos colaterales nefastos (ej “*spoof login*”).

**DYA**

*6*

****Diseño Y Aplicaciones deSistemas Distribuidos**

**Introducción**

 **Métodos de ataque en un sistema cliente-servidor (i)Repetidor**

**Cliente**

**petición**

**respuesta**

**Fisgón**

**DYA**

**Servicio**

*7*

****Diseño Y Aplicaciones deSistemas Distribuidos**

**Introducción**

 **Métodos de ataque en un sistema cliente-servidor (ii)Cliente Servicio**

**impostor**

**petición**

**respuesta**

**Cliente Servidor**

**DYA**

**petición respuesta**

**impostor**

*8*

**

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

**Introducción**

 **Métodos de ataque en un sistema cliente-servidor (iii)**

Código

migrado

=

virus en potencia

**Cliente Servicio**

*9*

****Introducción**

 **Mecanismos de seguridad**

Los mecanismos de seguridad en un sistema distribuido se basan, en general, en técnicas basadas en la criptografía. Los más utilizados son:

**Diseño Y Aplicaciones deSistemas Distribuidos**

– **Cifrado de mensajes:** consiste la transformación de mensajes mediante claves criptográficas. Tiene como objetivo:

 Ocultar los contenidos de un mensaje: evitar su lectura por una tercera parte. – **Firmas digitales:** acompañar el mensaje con un resumen cifrado (firma) que sólo el que lo envía puede generar. Tiene tres finalidades:

 Evitar que un mensaje (cuyo contenido no tiene porqué ser oculto) sea modificado.  Garantizar la identidad del que lo envía.

 Evitar el repudio del que lo envía.

**DYA**

*10*

****Introducción**

 **Mecanismos de seguridad (ii)**

– **Mecanismos de autentificación**: medio por el cual las identidades de un cliente y un servidor pueden ser establecido de manera fiable. Se establece **Diseño Y Aplicaciones deSistemas Distribuidos**

una clave de sesión para comunicar entre ambos.

 **Sistemas centralizados**: basados en contraseñas para sesiones. Requiere que todos los recursos estén bajo el control del mismo kernel.

 **Sistemas distribuidos**: basados en la criptografía: descifrar con éxito un mensaje con una clave secreta preacordada entre dos procesos indica que el mensaje es auténtico y proviene de un emisor autentificado.

– **Control de acceso:** son unos mecanismos para restringir los accesos de procesos sobre recursos.

**DYA**

*11*

**

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

**Seguridad**

**Indice**

–Introducción

–Criptografía

 Claves simétricas

 Claves asimétricas

–Firmasdigitales

–Autentificaciónydistribucióndeclaves –Casosdeestudio:

 Kerberos

 SSL

 Java 1.2

*12*

****Criptografía**  **Introducción**

Algoritmo de cifrado

**Diseño Y Aplicaciones deSistemas Distribuidos**

Texto en claro Texto cifrado (función, clave) 

**M{M}**

**(f, Ke) (f-1, Kd)**

**K**

– La efectividad depende del uso de funciones que resistan ataques para:  Dado {M}K obtener M (f puede ser pública)

 Dado {M}K y M obtener K (f puede ser pública)

– Dos técnicas:

**DYA**

 **Clave secreta (simétrica): Ke = Kd = K**

 **Clave pública (asimétrica): Ke != Kd**

*13*

**

**Criptografía**

 **Algoritmo de clave simétrica (secreta)** Solo un clave:

– **K** es secreta y compartida por emisor y receptor.

**Diseño Y Aplicaciones deSistemas Distribuidos**

 No debe enviarse nunca en un mensaje

– f y f-1 son públicas

**Emisor A Receptor B**

**DYA**

1.- Adquirir K

2.- f(K,M) → {M}K 3.- enviar {M}K

{M}K

1.- Adquirir K

2.- recibir {M}K 3.- f-1 (K,{M}K) → M

*14*

**

**Criptografía**

 **Algoritmo de clave simétrica (secreta)**

– La efectividad depende de que f y f-1 sean lo suficientemente complejas para

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

que obtener K a partir de M y {M}Ksea difícil.

– (f,K) definidas por DES (Data Encryption System) [National Bureau of Standards]

 K: clave de 56 bits

 f y f-1: mapean 64 bits planos en 64 bits cifrados usando 16 *rondas* de rotación dependientes de la clave y 3 transposiciones independientes de la clave.

– Descubrir K requiere una media de 255 a 3x1016 intentos ejecutando f-1. – Por otra parte, f tiene una complejidad computacional suficientemente alta para que cada intento cueste un tiempo de cómputo muy elevado y los intentos de descubrir la clave mediante “fuerza bruta” cuesten ,al menos, años.

*15*

**

**Criptografía**

**Algoritmodeclaveasimétrica(pública)** Dos claves:

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

– **Ke**: que es pública y sirve para cifrar.

 Puede obtenerse de un servidor de claves.

– **Kd**: que es secreta y sirve para descifrar.

Elúnicoquelasabeeselreceptordeunmensaje. Dos funciones de cifrado:

– E: Función para cifrar. Es pública.

– D: Función para descifrar. Es pública.

– **Amenza**: Dados Ke, E y D, 🡺 obtener KD Resultacomputacionalmentemuydifícil(billonesdeaños).

*16*

**

**Criptografía**

 **Algoritmo de clave asimétrica (pública)**

**Diseño Y Aplicaciones deSistemas Distribuidos**

**Emisor A**

**Base de**

**datos de**

**claves**

**públicas** Pedir Ke Ke

**Receptor B**

1.- Adquirir Ke

2.- E(Ke,M) → {M}Ke

3.- enviar {M}Ke

**DYA**

{M}K

1.- Calcular Ke, Kd 2.- Publicar Ke

3.- recibir {M}Ke

4.- D (Kd,{M}Ke) → M

*17*

**

**Criptografía**

 **Ejemplo de algoritmo de clave asimétrica (pública)** – **Clave pública**: <e,N>

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

– **Cifrado de M**:

E(e,N,M) = M**e** mod N

– **Descifrado de M**:

D(d,N,C) =C**d** mod N

Donde:

 e: número (obtenido a partir de números primos muy grandes)

 d: número (obtenido a partir de números primos muy grandes)

 N: tamaño del bloque

 C: texto cifrado

– Obtener d a partir de e supone factorizar el producto de dos números primos muy grandes. En la práctica costaría 4 billones de años con un computador del año 1978.

*18*

**

**Criptografía**

 **Utilidades de los algoritmos de clave asimétrica (pública)** – **Mensajes cifrados (privacidad información)**: permite que los emisores

envíen información cifrada a un receptor que sólo el puede descifrar

**Diseño Y Aplicaciones deSistemas Distribuidos**

 **Ke** es pública entre los emisores

 **Kd** es secreta al receptor

– **Mensajes de sólo lectura (no modificación información)**: permite que un emisor envíe información cifrada que cualquier receptor puede descifrar, pero no modificar

 **Ke** es secreta al emisor

 **Kd** es publica entre los receptores

 **Combinación de técnicas**

**DYA**

– El algoritmo de clave pública puede utilizarse para enviar una clave secreta al receptor y luego utilizar esa clave secreta para la comunicación posterior. Resulta más eficiente.

*19*

****Seguridad**

**Indice**

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

–Introducción

–Criptografía

 Claves simétricas

 Claves asimétricas

–Firmasdigitales

–Autentificaciónydistribucióndeclaves –Casosdeestudio:

 Kerberos

 SSL

 Java 1.2

*20*

**

**Firmas digitales**

 **Fundamento**

– Es una información que se adosa a un mensaje cuya función es garantizar

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

que:

 Un emisor no pueda suplantar a otro emisor

 El cuerpo del mensaje no ha sido modificado.

– Se intenta que proporcione la misma forma de autentificación y responsabilidades que una firma escrita, pero goza de dos ventajas importantes frente a ésta:

 Es más difícil de falsificar y más fácil de detectar falsificaciones.

 Garantiza que no se puede modificar el texto del documento que ha sido firmado.

*21*

**

**Firmas digitales**

 **Implementación**

– **Implementación simple**: <M,A,{M}KA>

M: mensaje

**Diseño Y Aplicaciones deSistemas Distribuidos**

A: identidad de un sujeto

{M}KA: firma; puede ser con clave secreta o clave pública

– **Implementación con función sintetizadora**: <M,A,{D(M)}KA>

 La función de síntesis D realiza un resumen o *checksum* del mensaje M y garantiza que D(M)!=D(M’) si M!=M’

 Funciones sintetizadoras o de “*digest*” [Rivest], [Mitchell]. Existen algunas normalizadas: MD5 (RFC1321) y SHA-1

 **Notación**

**DYA**

– PKA: clave pública de A

– SKA: clave secreta de A

*22*

**

**Firmas digitales**

 **Firmas con claves públicas**

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

1. A → B: M,A, {D(M)}SKA ----------- 2. B → S: A ---------------------------- 3. S → B: A, PKA 



– A envía a B el mensaje original con su firma.

– B pide a un servidor de firmas la clave pública de A.

– S proporciona a B la clave pública de A (PKA) y B la utiliza para descifrar la firma recibida en el mensaje 1 y comparar el resultado con el valor con el valor recién calculado de D(M).

*23*

**

**Firmas digitales**

 **Claves públicas: ¿quien emite la firma del mensaje?** – **Autofirmado**: el que realiza la firma es el propio emisor del mensaje. Se

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

utiliza cuando el receptor conoce al emisor y confía en él. Lo único que garantiza es que el mensaje no ha sido modificado.

– **Entidad certificadora**: es una tercera parte, que garantiza la identidad del emisor del mensaje

 El emisor se autentifica ante la entidad certificadora.

– Esto se puede realizar, por ej., enviándole un mensaje autofirmado que la entidad verifica (por ej. comprobando su DNI, sus huellas digitales).

 La entidad certificadora emite un **certificado** de la identidad del emisor con la firma de la propia entidad certificadora. La firma de la identidad certificadora puede ser comprobada ante un servidor de firmas.

 Una entidad certificadora ha de ser admitida como “fiable” por el receptor. Si no es así, debe de ser autentificada por otra tercera parte, dando lugar a una **cadena de autentificación** que puede ser arbitrariamente larga.

*24*

**

**Firmas digitales**

 **¿Cual es la finalidad de la entidad certificadora?**

– Establecer, a efectos legales, una correspondencia entre la identidad del

emisor y la identidad de una persona física, según consta en su DNI o similar. **Diseño Y Aplicaciones deSistemas Distribuidos**

– Evitar el **repudio** de la información: una persona física no puede alegar que un mensaje enviado por ella a un receptor no fue realmente enviado por ella.

 **Revocación de firmas digitales**

– Una firma digital puede ser revocada por una entidad certificadora que no identificó correctamente a una persona física.

– Existen **listas de revocaciones** con los números de las firmas revocadas.

– Las revocaciones son difíciles de detectar automáticamente.

**DYA**

*25*

****Firmas digitales**

---------------------------------------------

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

 **Firmas con claves públicas y entidad certificadora** 

1. A genera PKA SKA (pública y secreta) 2. A se autentifica ante S 

Le proporciona su identidad A y PKA.

3. S → A: {A, PKA}SKS

4. A→ B: A, M, {D(M)}SKA , {A, PKA}SKS 5. B →S: {clave pública S?}

6. S → B: PKS.

7. B descifra {A, PKA}SKS y {D(M)}SKA

------

– Utilizando un generador de claves – Por diversos medios.

 Puede requerir presencia física ante S y firma de un contrato.

– S envía a A un certificado. – A incluye el certificado en todos sus mensajes

– B reconoce la entidad certificad. – La entidad le proporciona PKS – B es capaz de descifrar el

certificado, obtener PKA y con ésta, descifrar el resumen D(M).

*26*

**

**Firmas digitales**

 **Firmas con clave secreta** 1. A → S: A, {D(M)}KA 

**Diseño Y Aplicaciones deSistemas Distribuidos** 

2. S → A: {A, D(M),t} Ks

3. A → B: M, {A, D(M),t} Ks

4. B → S: B, {A, D(M),t} Ks

– A calcula D(M), lo cifra con su clave secreta y se lo envía a S

– S elabora un certificado fechado y firmado con su clave secreta de:

 la identidad de A

 la síntesis D(M)

 un *timestsamp*

– A envía a B el mensaje y el certificado – B guarda una copia del mensaje y envía a S el certificado para que lo descifre

**DYA**

5. S → B: {A, D(M),t} KB

– S descifra el certificado y se lo devuelve a B cifrado con la clave secreta de B. B lo descifrará, generará D(M) y lo comparará

*27*

**

**Firmas digitales**

 **Firmas con clave secreta**

La autentificación de A se realiza generando D(M), descifrando el D(M)

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

del mensaje y comparándolos. Si coinciden, B tendrá que aceptar que A es el originario puesto que:

– B confía en el servidor de autentificación y este le envía un mensaje diciendo que ha verificado la firma de A (paso 2).

– A no puede alegar que la firma ha sido falsificada, ya que B tiene una copia del certificado original que puede ser contrastado con el servidor de autentificación. A no puede alegar que B la ha falsificado pues B no conoce la clave secreta del servidor.

*28*

****Seguridad**

**Indice**

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

–Introducción

–Criptografía

 Claves simétricas

 Claves asimétricas

–Firmasdigitales

–Autentificaciónydistribucióndeclaves –Casosdeestudio:

 Kerberos

 SSL

 Java 1.2

*29*

**

**Autentificación y**

**distribución de claves**

 **Planteamiento del problema**

– A se autentifica ante S (por ejemplo mediante una variación segura del protocolo *login*)

**Diseño Y Aplicaciones deSistemas Distribuidos**

y le solicita una clave KAB de sesión con B.

– S entrega a A un ticket para autentificarse con B y la clave KAB cifrada. – Con este ticket A y B ejecutan un protocolo, al final del cual ambos están seguros de la identidad del otro y conocen la clave KAB

**DYA**

**login**

**ticket**

**S**

**ticket**

**protocolo**

**CANAL SEGURO**

**A B**

**KAB**

*30*

**

**Autentificación y**

**distribución de claves**

 **Canal seguro**

– El servidor (y el cliente) conocen la identidad del cliente (servidor) con el que

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

comunican y puede comprobar sus derechos de acceso antes de realizar una comunicación.

– Asegura la privacidad y la integridad (manipulación) de los datos transmitidos a través de el.

– Cada mensaje incluye un sello de carácter temporal de tipo físico o lógico para prevenir el reenvío o la reordenación de los mensajes.

*31*

**

**Autentificación y**

**distribución de claves**

 **Introducción**

Los problemas de autentificación y distribución de claves suelen ser implementados por un único servidor.

**Diseño Y Aplicaciones deSistemas Distribuidos**

– **Distribución de claves:** encargado de manufacturar y distribuir claves – **Autentificación**: un servicio de autentificación es capaz de verificar la identidad un proceso (por mecanismo de claves públicas o privadas). Normalmente, se basa en un ticket para que el proceso lo use en su interacción con otros servidores

 **Algoritmos de autentificación**

**Needham y Schroeder**: en 1978 proporcionan el primer algoritmo de autentificación. Existen dos versiones:

**DYA**

– **Clave secreta**

– **Clave pública**

*32*

**

**Autentificación y**

**distribución de claves**

 **Algoritmo de Needham-Schroeder (i)**

– Basado en un servidor de autentificación que mantiene una tabla de <nombres, clave-secreta>

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

**Notación:**

– **A**: Cliente que inicia la comunicación.

– **B**: Servidor al que se dirige A

– **S**: Servidor de claves

– **KA**: Clave secreta de A (*password*). La saben A y S. Nunca se publica cifrada o descifrada por la red.

– **KB**: Idem para B.

– **KAB**: Clave secreta proporcionada por S para la comunicación entre A y B. Sólo se publica cifrada por la red

– **NA**: Una prueba de “frescura” (actualidad) generada por A.

*33*

**

**Autentificación y**

**distribución de claves**

Alg. de Needham-Schroeder con clave

 **Alg. de Needham-Schroeder**

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

**con clave secreta (i)** 1. A → S: A,B, NA 

2. S → A: {NA, B, KAB {KAB,A}KB}KA

3. A → B: {KAB, A}KB

4. B →A: {NB}KAB

secreta (i)

– A pide a S una clave para comunicar con B – S devuelve un mensaje en clave de A con:  KAB: clave para comunicar A con B

 ticket cifrado en clave de B

La marca temporal NA demuestra que el mensaje es contestación al anterior. **A cree a S porqué sólo S sabe la clave secreta de A**

– A envía el ticket a B

– B descifra el ticket, averigua KAB, y lo usa para cifrar una prueba de frescura

**A cree a B porqué sólo B puede descifrar un mensaje con su clave**

**secreta**

*34*

**

**Autentificación y**

**distribución de claves**

Alg. de Needham-Schroeder con clave

 **Alg. de Needham-Schroeder**

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

**con clave secreta (ii) **5. A → B: {NB-1}KAB



secreta (ii)

– A demuestra a B que él fue quien envió el mensaje anterior siendo capaz de cifrar con KAB una transformación preacordada de NB.

*35*

**

**Autentificación y**

**distribución de claves**

Alg. de Needham y Schroeder con

 **Alg. de Needham y Schroeder con**

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

**clave pública (i)** 1. A → S: A,B 

2. S →A: {PKB, B}SKS

3. A →B: {NA, A}PKB

4. B → S: B, A

(1’)

clave pública (i)

– A pide a S la clave pública B

– S devuelve a A un mensaje cifrado con su clave secreta que contiene:  la clave pública de B.

**Puede ser descifrado por cualquiera pero no adulterado.** – A envía a B un mensaje cifrado con la clave pública de B que contiene:  una marca temporal

 su identificador

**Sólo B puede descifrarlo y obtener el identificador de A.**

– B solicita a S la clave pública de A

*36*

**

**Autentificación y**

**distribución de claves**

Alg. de Needham y Schroeder con clave pública

 **Alg. de Needham y Schroeder**

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

**con clave pública (ii)** 5. S→ B: {PKA, A}SKS (2’) 

6. B → A: {NA, NB}PKA (3’)

7. A → B: {NB}PKB

(ii)

– S devuelve a B un mensaje cifrado con su clave secreta que contiene:

 la clave pública de A.

**Puede ser descifrado por cualquiera pero no adulterado.**

– B envía a A un mensaje cifrado con la clave pública de A con un par de marcas temporales

 Sólo B puede devolverlo pues es el único que pudo descifrar (3)

 Sólo A puede descifrarlo

– A envía a B un mensaje cifrado con la clave pública de B con la marca

 Solo A puede enviarlo (descifrando (6))

– La comunicación es fresca.

*37*

****Seguridad**

**Indice**

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

–Introducción

–Criptografía

 Claves simétricas

 Claves asimétricas

–Firmasdigitales

–Autentificaciónydistribucióndeclaves –Casosdeestudio: 

 Kerberos

 SSL

 Java 1.2

*38*

**

**Kerberos**

 **¿Qué es Kerberos?**

– Protocolo de autentificación basado en algoritmo de Needham y Schroeder

con clave secreta desarrollado en el MIT. Disponible para UNIX e incluido en **Diseño Y Aplicaciones deSistemas Distribuidos**

el entorno de OSF para sistemas distribuidos.

 **Arquitectura de Kerberos**

– Un servidor Kerberos se conoce como: KDC: Kerberos Distribution Center y ofrece dos servicios:

 **AS:** Authentication Service

 **TGS:** Ticket Granting Service

– Basada en tres objetos de seguridad:

 **Clave de sesión**

 **Ticket**

**DYA**

 **Autentificador**

*39*

**

**Kerberos**

 **Arquitectura de Kerberos**

– **Clave de sesión**: clave secreta generada por Kerberos y expedida a un

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

cliente para uso con un servidor durante una sesión. No es obligatorio utilizarla en toda la comunicación con el servidor; sólo si:

 el servidor lo requiere (los datos son confidenciales)

 el servidor es un servidor de **autentificación**

**Notación:**

– **KCS**: clave para la comunicación entre un un cliente C con un servidor S

*40*

**

**Kerberos**

 **Arquitectura de Kerberos**

– **Ticket**: un testigo expedido a un cliente del servicio de tickets de Kerberos (TGS) para solicitar los servicios de un servidor. Garantiza que el cliente ha

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

sido autentificado recientemente.

**Notación**: un ticket de un cliente C para acceder un servidor S toma la forma: – {ticket(C,S)}**KS=** {C, S, t1, t2, KCS}**KS**

– Incluye:

 El nombre del cliente C para evitar posible uso por impostores

 Un periodo de validez [t1, t2]

 Una clave de sesión KCS asociada para uso de cliente y servidor

– Kerberos siempre proporciona el ticket ya cifrado con la clave secreta del servidor al que se le entrega.

*41*

**

**Kerberos**

 **Arquitectura de Kerberos**

– **Autentificador**: un testigo construido por el cliente y enviado a un servidor

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

para probar su identidad y la actualidad de la comunicación. Sólo puede ser utilizado una vez.

**Notación**: un autentificador de un cliente C ante un servidor S toma la forma: – {auth(C)}**KCS=** {C, t}**KCS**

– Contiene, cifrado en la clave de la sesión,:

 el nombre del cliente

 timestamp

*42*

**

**Kerberos**

 **Arquitectura de Kerberos**

– Al iniciar una sesión (*login*) los usuarios se autentifican con el servicio de

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

autentificación AS utilizando una variación segura del protocolo de *passwd*, el cual proporciona:

 un ticket para el servicio de tickets TGS

 una clave de sesión para TGS

– Utilizando el ticket y clave anterior el cliente solicita a TGS

 un ticket para un determinado servidor

 una clave para un determinado servidor

– Kerberos sigue de cerca el protocolo de Needham y Schroeder con clave secreta utilizando *timestamps* como pruebas de frescura con dos propósitos:

 Evitar reenvíos de viejos mensajes capturados en la red o reutilización de viejos tickets obtenidos de zonas de memoria del usuario autorizado.

 Poder revocar a los usuarios los derechos al cabo de un tiempo.

*43*

**

**Kerberos**

 **Arquitectura de Kerberos**

**Centro de distribución**

**de claves Kerberos**

**TGS** 1. Petición ticket

**KDC**

**Diseño Y Aplicaciones deSistemas Distribuidos**

para TGS

**Cliente C**

Establecer

sesión

2. ticket para TGS

**AS**

**Servicio**

**Autentific. A**

**Servicio Tickets T**

**Servidor S**

**DYA**

de login ----------------- Establecer sesión

Servidor ----------------- Solicitar servicio ...

3. Petición ticket

para S

4. ticket

para S

5. **Petición** cifrada con clave de sesión 6. **Respuesta** cifrada con clave de sesión

Realizar servicio

*44*

**

**Kerberos**

 **Protocolo seguro de *login* (Kerberos)**

– Cuando un usuario abre una sesión el programa de *login* envía el nombre del

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

usuario al servicio de autentificación AS de Kerberos

– Si el usuario es conocido, AS responde con:

{KAB, n}passwd + {ticket(A,B)}**K**T

– El programa *login* intenta descifrar {KAB, n}passwd con el passwd que el usuario proporciona y si, éste es correcto, obtiene KAB y n.

 **El passwd nunca circula por la red.!!**

– Una vez obtenido KAB el programa *passwd* lo guarda para su subsiguiente comunicación con B y borra el *passwd* del usuario de memoria. – **Ticket**: un testigo expedido a un cliente del servicio de tickets de Kerberos (TGS) para solicitar los servicios de un servidor. Garantiza que el cliente ha sido autentificado recientemente.

*45*

**

**Kerberos**

 **Protocolo de *login* con Kerberos**

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

**1.** C → A: C, T, n

Petición ticket para TGS

2. A → C: {KCT, n}Kc

{ticket(C,T)}KT

– C pide al AS de Kerberos un ticket para comunicar con TGS.

– AS retorna un mensaje, que contiene:

 Una clave para comunicación con TGS y un timestamp cifrado con la clave secreta del cliente  Un ticket para comunicación con TGS (como siempre, cifrado con la clave secreta del receptor del ticket)

A sólo podrá descifrar el mensaje si conoce su clave secreta KC

*46*

**

**Kerberos**

 **Protocolo Kerberos**

3. C → T: {auth(C)}KCS,

– C pide al TGS de Kerberos un ticket para

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

{ticket(C,T)}KT, S, n} Petición ticket para S

4. T → C: {KCS, n}KCT,

{ticket(C,S)}KS

comunicar con S.

– TGS comprueba el ticket y, si es válido, retorna un mensaje, que contiene:

 Una clave para comunicación con S y un *timestamp* cifrado con la clave de sesión del par C-TGS

 Un ticket para comunicación con S

A sólo podrá descifrar el mensaje si conoce la clave secreta KCT

*47*

**

**Kerberos**

 **Protocolo Kerberos**

5. C → S: {auth(C)}KCS,

– C envía a S un recién generado autentificador,

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

{ticket(C,T)}KS, request, n Petición servicio

6. S → C: {n}KCS

Autentificación del cliente (opcional)

el ticket y una petición. La petición puede ir cifrada si el servidor lo requiere.

– S envía a C la prueba de actualidad cifrada en la clave secreta de la sesión.

Sólo S pudo obtener KCS y, por tanto, enviar este mensaje.

*48*

**

**Kerberos**

 **Utilización**

– En el **login**

– Acceso a otros servers, p.e, **rlogind**

**Diseño Y Aplicaciones deSistemas Distribuidos**

– Acceso a sistemas de ficheros remotos: **NFS**

 **Implementación de Kerberos**

– Servidor que se ejecuta en máquina segura +

– Conjunto de bibliotecas para uso por clientes y aplicaciones +

– Algoritmos de cifrado del DES ( son fácilmente reemplazables):

– **Servicio escalable** a todo el mundo: el mundo esta dividido en *realms*, cada cual con su autoridad de identificación. Los TGS’s están registrados en todos los *realm*s. – **Admite replicación** en las bases de datos

 **Críticas a Kerberos**

**DYA**

– Derivadas de la utilización de *timestamps* como prueba de actualidad. Requiere sincronización de relojes.

*49*

****Seguridad**

**Indice**

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

–Introducción

–Criptografía

 Claves simétricas

 Claves asimétricas

–Firmasdigitales

–Autentificaciónydistribucióndeclaves –Casosdeestudio:

 Kerberos

 SSL

 Java 1.2

*50*

**

**SSL**

 **Secure Sockets Layer**

– Estándar de facto (RFC 2246) diseñado originalmente por Netscape (1996)

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

para proporcionar seguridad en transacciones electrónicas en Internet.  *https:*

 www.openssl.org

– Comunicación segura basada en el concepto de **canal seguro**. – El canal seguro es completamente configurable

 Permite que la comunicación en ambos sentidos sea encriptada y autenticada, aunque ambas características son opcionales, evitando así realizar cifrados y consumir recursos innecesarios.

– Algoritmos de encriptación negociables.

– Opcionalmente puede incluir compresión.

– Proporciona seguridad a http, pero también FTP y otros

– Interfaces Java y CORBA

*51*

**

**SSL**

**Componentes**

– Consta de dos capas:

**Diseño Y Aplicaciones deSistemas Distribuidos**

 Protocolo de nivel desesión:RegistroSSL –Implementaelconceptodecanalseguro Capade**handshake**

–Estableceyconfiguranelcanalseguro

**Protocolode handshake SSL**

**Cambiode**

**especificaciones CifradoSSL**

**Protocolode alerta**

**SSL**

**HTTP**

**DYA**

**ProtocoloderegistroSSL**TCP/IP

*52*

**

**SSL**

 **Protocolo de handshake**

– El establecimiento de un canal seguro se basa en un esquema en tres fases:

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

 Comunicación no encriptada en intercambios iniciales

 Criptografía de clave pública para autenticar a las partes

– Certificados X.509 de una autoridad o generados de modo temporal

 Finalmente se conmuta a criptografía de clave secreta una vez se ha establecido una clave de secreto compartida.

– El cambio a cada fase es opcional y viene precedido por una negociación.

*53*

**

**SSL**

 **Protocolo de handshake** HolaCliente

Establecer sesión y

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

HolaServidor

Certificado

PideCertificado

HolaServidorHecho

Cliente Servidor Certificado

VerificaCertificado

CambiaEspecCifrado

Final

CambiaEspecCifrado

Final

características canal

Enviar certificado para autentifcar servidor y pedir Opcionalmente el del cliente

Enviar certificado del cliente si es el caso

Cambia conjunto de cifrado y finaliza handshake

*54*

**

**SSL**

 **Protocolo de handshake**

– **1ª fase**: Establecer sesión y características de canal seguro

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

 No Cifrado. Vulnerable al ataque del “hombre entre medias”

– Puede evitarse utilizando clave pública con certificados obtenidos por un canal seguro (CD-ROM) de empresas certificadoras o incluyendo el nombre del dominio o la IP en certificados

 Ofrece una variedad de funciones criptográficas: **catálogo de cifrado** que incluye una elección única para

– Método de intercambio de clave de sesión: Ej. RSA con certificados de clave públlica – Cifrado para la transferencia de datos: Ej. IDEA

– Función de resumen de mensajes: Ej. SHA

*55*

**

**SSL**

 **Protocolo de handshake** – **2ª fase**: autenticación

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

 Se utiliza criptografía de clave pública para autenticar a las partes

– Certificados X.509 de una autoridad o generados de modo temporal

 Un participante genera una *clave pre-maestro* y lo envía encriptada con clave pública al otro participante.

– Clave pre-maestro: valor aleatorio con el que ambos partipantes generan las claves de sesión (claves de escritura) para encriptar los datos en cada dirección.

– **3ª fase**: sesión segura

 Intercambio de mensajes *CambiaEspecCifrado* y *Final*

– Los participantes inician la **transferencia de datos** intercambian mensajes utilizando las claves preacordadas.

*56*

**

**SSL**

 **Protocolo de Registro SSL**

– Protocolo de nivel de sesión que

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

proporciona un canal seguro a los protocolos de nivel de aplicación (http, ftp, smtp, telnet, ...) de modo transparente garantizando integridad y autenticidad.

– Cada sesión tiene un identificador que se puede almacenar en una *cache* para su uso subsiguiente, evitando la sobrecarga de establecer una nueva sesión segura con un mismo compañero.

**fragmentar cifrar**

**transmitir**

abcdefghi

abc def ghi **resumir**

abc

abc

Paquete TCP/IP

*57*

**

**SSL**

**Implementaciónmediante técnicade*tunneling***

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

– Inicialmente, los programas que querían utilizar SSL tenían que modificar el código para utilizar las funciones de la biblioteca SSL. Para evitarlo se propuso realizar esto mediante ***tunneling:***

EsposibleañadirlacapaSSL utilizandounprocesoexternoque hagadeproxy(wrapper)entrela redyunservidornoseguro.

Suformadeoperaressimilaral port－forwardingdeSSH.

cifrado

cliente**443**

claro

**80**

stunnel cert

www

*58*

**

**SSL**

 **stunnel**

– Diseñado para ofrecer la seguridad SSL a servidores (demonios) o clientes

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

que no han sido programados con soporte SSL. – Puede actuar en:

 La parte del cliente

 La parte del servidor

*59*

**

**SSL**

 **stunnel como cliente**

– Al igual que con socket podemos conectarnos a cualquier puerto de cualquier

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

ordenador conectado a internet:

 $ socket www.upv.es 80

 GET / HTTP/1.0

 nos retorna la página principal de la UPV.

– Con la siguiente orden se consigue el mismo resultado, pero utilizando una conexión segura.

 $ /usr/sbin/stunnel conf-cli

 GET / HTTP/1.0

– Fichero conf.cli

 client = yes

 connect = www.upv.es:443

 # el puerto estándar de http seguro es el 443 (ver /etc/services).

*60*

**

**SSL**

 **stunnel como servidor**

– La siguiente orden deja en segundo plano (background) stunnel:

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

 # stunnel –conf-ser2

– Fichero conf-ser2

 cert = /home/usuarios/jvila/ssl/server.pem  #key = /home/usuarios/jvila/ssl/server.pem  #debug = 1

 foreground = yes

 pid =

 #outfile = /tmp/error

 #local = futura.disca.upv.es

 [lsd]

 accept = 9500

 exec = /bin/ls

 execargs = ls -l

 #pty = yes

*61*

**

**SSL**

 **stunnel como servidor**

– **Nota importante**: la clave privada a partir de la cual se crea el certificado no

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

debe estar protegida con password para que el servidor (web) pueda hacer uso de ella para encriptar la información.

 **1.-** el fichero con el certificado debe contener primero la clave privada (sin password), seguido del certificado.

 **2.-** Tras la clave privada y el certificado debe aparecer una línea en blanco.  **3.-** Por tanto, al fichero creado por la orden:

$ openssl req －new －x509 －nodes －keyout server.pem －out server.pem se le han de han de añadir un retorno de carro después de la clave privada y otro al final (después del certificado).

– new: Generate a new key

– x509: Generate an X509 certificate (self sign)

– nodes: Don't put a password on this key.

– out stunnel.pem where to put the SSL certificate

– keyout stunnel.pem put the key in this file

*62*

**

**SSL**

 **Conversión de certificados de Java a certificados pem** – Los certificados de java son DER codificados en X.509 y no en ASN.1

**Diseño Y Aplicaciones deSistemas Distribuidos DYA**

 openssl x509 -inform DER -outform PEM -in JoanVila.cer -out JoanVila.pem

*63*

**

**Certificados**

 **Formato de los certificados autofirmados** – { ID, Ke, Kd, {D(ID, Kd)}Ke} siendo Ke secreta y Kd pública

 No publicarlo!!!! Contiene Ke en claro!!

**Diseño Y Aplicaciones deSistemas Distribuidos**

 Pero el servidor web lo necesita así

– Al exportarlo (pide Ke), a partir del anterior se genera:

{ID, Kd, {D(ID, Kd)}Ke} y probablemente Ke en un fichero aparte

 Lo exportan los clientes de correo y los servidores web. Es lo que se envía en los correos y en las transferencias https

 **Formato de los certificados firmados por entidad certificadora** – { ID, Kd, {D(ID, Kd)}Ks}

**DYA**

*64*