Ingeniería de Seguridad

Una Guía para el Estudiante

Versión 0.9

Eliecer Cadenas Patrick O'Callaghan

7 de Mayo de 2002

Capítulo 1

Introducción

1.1. Un ejemplo de seguridad

Juan Precavido se levanta muy temprano en la mañana pues había prometido a su esposa que revisaría en la página web del banco si ya se había hecho efectiva la transferencia que realizaron el día anterior usando el nuevo servicio en línea del banco. Decide desayunar en la calle, camino de la oficina, así que solo se prepara un poco de café para despertarse, recoge su identificación personal, la cual le da acceso a su oficina, las llaves del carro, su celular y sale con el interés de resolver lo antes posible el tema de la transferencia.

Al llegar al estacionamiento desactiva la alarma del carro usando su control remoto, se introduce al carro, le retira el tranca-palanca y lo enciende. Siempre le ha llamado la atención esa llave luminosa que se enciende en el panel cuando introduce la llave para encenderlo. Sale del estacionamiento usando el otro control remoto para abrir la puerta del garaje. Afortunadamente hay poco tráfico y se dirige a su restaurante favorito.

Al llegar pide un café, un jugo y un sandwich. Desayuna revisando el periódico y al terminar pide al mesero la cuenta. Entrega al mesero su tarjeta de crédito. Este vuelve a los diez minutos con su voucher de pago. Lo firma y se levanta para continuar su camino a la oficina.

Una vez en la oficina acerca su identificación magnética al borde la puerta para poder entrar. Se dirige a su computador portátil, el cual dejó conectado en su oficina el día anterior y lo enciende. Como todas las mañanas, introduce su login y usa un software para generar un password aleatorio que le permite ingresar a la red de la oficina. Abre su navegador y coloca la dirección de la página web de su banco. La página web le solicita introduzca su identificación y su password, lo hace, y accede a la información personal de su cuenta. No, lamentablemente aun no se ha hecho efectiva la transferencia. Decide llamar al banco, levanta el teléfono y marca, lo atiende una señorita muy amable que le hace alguna preguntas de seguridad y luego le informa que la transferencia está en camino, que en 24 horas la tendrá reflejada en su saldo. Le agradece y tranca y el teléfono decidido a comenzar su jornada de trabajo. En ese momento suena el celular, es su esposa preguntándole por la transferencia. Le explica

la situación y tranca finalmente dispuesto a ponerse a trabajar.

1.2. ¿Qué es Ingeniería de Seguridad?

La historia anterior muestra cuan compleja y sofisticada se ha vuelto nuestra vida. En tan solo dos horas el personaje de nuestra historia a realizado al menos diez operaciones riesgosas. Veamos algunas de las cosas que pudieron haber sucedido:

1. Al usar el control remoto para desactivar la alarma un ladrón, oculto en el estacionamiento graba la señal electrónica enviada por el control para luego replicarla y poder abrir sin ningún problema el carro.

2. Una vez dentro del carro, la llave, al ser introducida en el switche de ignición envía una señal electrónica, recibe otra a la cual responde y finalmente el motor acepta la llave como autentica ya autorizada para encender el motor. (Mala cosa para el ladrón si no fuera por que Juan Precavido es lo suficientemente descuidado como para haber perdido una de las llaves de su carro hace un mes, sin haberse percatado de ello).

3. Al salir del estacionamiento usa el otro control para abrir la puerta, por supuesto nuestro ladrón también graba esta señal para luego abrir la puerta del estacionamiento, cuando pretenda sacar el carro.

4. Cuando paga en el restaurante entrega su tarjeta de crédito. El mesonero lleva la tarjeta adentro y sin que Juan lo vea, pasa la cinta magnética por un dispositivo que le permite grabar la información de la cinta. Por otro lado toma nota de los número y datos que aparecen en la misma.

5. Cuando ingresa en su oficina no se percata que al acercar la tarjeta magnética a la puerta un nuevo software instalado para controlar la productividad de sus empleados está grabando sus entradas y salidas de la oficina (este es mas un problema de privacidad que de seguridad).

6. Enciende su computadora y al dar el login y el password nota que extrañamente la pantalla de login apareció dos veces, así que introduce la información dos veces. Raro, las computadoras nunca fallan. En este caso un hacker logro ingresar a su computadora y sustituyo el login original por un falso login para poder ingresar a los sistemas de la compañía.

7. Aun cuando se hubiera percatado de ello, desafortunadamente la empresa envía la información del password y el login a través de la red sin encriptar. Nuestro hacker ya logró ingresar a otra cuenta e instaló un sniffer que recolecta login y passwords de toda la gente que hace login en la red local.

8. Al ingresar a la página del banco no se percata que la información no está siendo encriptada.

9. Al llamar a la señorita en el banco no se entera que el banco cambió el número de teléfono de atención al público y que un ex-empleado está aprovechando esta situación para recolectar información que luego le permita acceder a sus cuentas bancarias.

10. Cuando recibe la llamada de su esposa alguién escucha la conversación interfiriendo las señales enviadas entre el celular y la celda.

Este curso trata sobre Ingeniería de Seguridad entendida como los métodos para diseñar y construir sistemas que permanezcan confiables a pesar de posibles actos maliciosos o a errores de diverso tipo incluyendo las fallas de carácter fortuito.

Tradicionalmente se ha dado una particular atención a la criptografía como el centro de las actividades asociadas a la seguridad en sistemas de diversa naturaleza, como veremos a lo largo de este curso hay mucho mas que eso si queremos diseñar e implementar un entorno confiable.

Los sistemas se han vuelto cada vez mas complejos, y esa complejidad es la que permite que hayan mas y mas fallas de seguridad. En la medida en que los sistema están constituidos por múltiples componentes que interactúan permanentemente, cada espacio de interacción se convierte en una oportunidad para aquellos que quieren subvertir la seguridad establecida.

Incluso cuando se piensa en la criptografía como una herramienta para proteger nuestros sistemas, no solo es importante la fortaleza del algoritmo de encriptamiento, cuentan también la calidad de la clave, la seguridad con que se guarda, donde se guarda, que tipo de password se exige al usuario. Y nada de esto tiene ninguna utilidad si un hacker a logrado ingresar a una de las computadoras que están envueltas en la comunicación.

El interés por tanto de este curso está centrado en darle al estudiante una visión clara de los problemas de seguridad a los que hoy día nos enfrentamos y cuales son las medidas que pueden ayudar a reducir esos riesgos. Esta visión, por la naturaleza del problema tiene que ser necesariamente una visión sistémica, de no ser así, ninguna medida que implementemos tiene la esperanza de funcionar adecuadamente.

Cuando diseñamos un sistema de seguridad debemos tomar en cuenta que uno de nuestros principales enemigos es el usuario mismo. Su confianza casi ciega en las computadoras y lo que estas hacen le hace una persona vulnerable. Los sistemas informáticos se han vuelto cada vez mas y mas complejos, con lo que es prácticamente imposible que no fallen, y las fallas siempre pueden ser usadas por personas maliciosas para atacar el sistema.

El problema empeora notablemente gracias a la Internet. Un hacker exitoso es probable que decida publicar su ataque y compartir su habilidad técnica con otros. Esta es la razón por la que gente con pocos conocimientos y habilidad técnica pueden convertirse en atacantes exitosos de sistemas relativamente complejos. Multiplicando de esta forma el número de ataques realizados a diversos sistemas. Una medida útil de cuan vulnerables están siendo nuestros sistemas se puede conseguir poniendo “trampas” atractivas en Internet y midiendo cuanto demora en ser atacados. El tiempo es sorprendentemente breve. En el caso de sistemas basados en Linux, en promedio 72 horas desde que el sistema es puesto en funcionamiento hasta que un hacker tiene control de la cuneta del administrador

Algunos ejemplos de sistemas en los que la seguridad es un aspecto fundamental son los siguientes:

1. La contabilidad de un banco.
2. Los cajeros automáticos.
3. Los mecanismos de protección física, como alarmas y sensores en general, destinados a detectar intrusos no deseados.
4. Los servicios en línea, vía Internet.
5. Obviamente en aplicaciones militares.
6. La privacidad es un tema de importancia en el caso de información de salud, o la información obtenida a través de los censos. Los sistemas de seguridad tienen por objeto proteger esa privacidad impidiendo por diversos métodos el acceso a la información mientras se permite operar en forma general sobre la data (p.ejem. para obtener estadísticas).
7. La necesidad de manejar operaciones seguras en la Internet, espacio en el cual los ataques del tipo negación de servicios son comunes.
8. El clonado de teléfonos celulares.

1.3. Definiciones

A lo largo de este texto entenderemos por sistema:

Un producto o componente, como es el caso de un protocolo criptográfico, una smartcard o el hardware de un PC.

Una colección de componentes como los antes mencionados más un sistema operativo, de comunicaciones, y algunos otros componentes que constituyen la infraestructura de una organización.

Lo antes descrito mas una o mas aplicaciones (contabilidad, cuentas por pagar, etc). Lo anterior mas el personal de IT.

Lo anterior mas usuarios internos y gerentes. Lo anterior mas clientes y otros usuarios externos.

Lo anterior mas el entorno incluyendo la media, competidores, reguladores y políticos.

Por un *sujeto* entenderemos un persona física en cualquier rol, incluyendo el de operador, víctima, etc. Por una persona entenderemos tanto un sujeto como una persona legal (una compañía o un gobierno).

Un principal es una entidad que participa en un sistema de seguridad. Esa entidad puede ser un sujeto, una persona, un rol o una pieza de equipo (PC, SmartCard, etc). Un grupo es un conjunto de principales. Un rol es una función asumida por diferentes personas.

Un principal puede ser considerado a varios niveles de abstracción. Por ejemplo Bob en representación de Alice puede significar La smartcard de Bob representando a Bob quien actúa en representación de Alice en su ausencia.

Se entiende por *identidad* a la correspondencia entre los nombres de dos principales significando que ellos se refieren a la misma persona o equipo.

Un sistema *trusted* es uno, cuya falla puede romper la política de seguridad, mientras que un sistema trustworthy (digno de confianza) es uno que no falla.

Secreto es un término técnico que se refiere al efecto del mecanismo usado para limitar el número de principales que tienen acceso a la información. (Criptografía o controles de acceso al computador).

Confidencialidad envuelve una obligación de proteger el secreto de alguna otra persona u organización.

Privacidad es la habilidad y/o el derecho de una persona u organización de proteger sus secretos.

El término *autenticidad* se refiere a integridad y frescura.

Vulnerabilidad es una propiedad de un sistema o su ambiente, el cual en conjunción con una amenaza interna o externa puede conducir a una falla de seguridad, el cual es un estado de cosas contrario a la política de seguridad del sistema.

Una *política de seguridad* es una declaración sucinta de la estrategia de protección de un sistema.

Un *objetivo de seguridad* es una especificación mas detallada que define los medios mediante los cuales se implementa una política de seguridad en un producto particular.

Un *perfil de protección* es similar al objetivo de seguridad excepto que está escrito en una forma suficientemente genérica como para poder evaluar su efectividad con diversos productos.

1.4. Componentes de un sistema seguro

Tradicionalmente el tema de la seguridad en sistemas computarizados se ha asociado a la criptografía y sus técnicas. La inmensa complejidad que caracteriza a los sistema modernos hace que esta aproximación sea insuficiente. Hoy en día la seguridad esta asociada a la interacción de una multiplicidad de sistemas. La seguridad de un sistema en particular esta directamente relacionada a la seguridad de la mas débil de sus partes. Un sistema puede tener mecanismos criptográficos que sean considerados completamente seguros pero puede adolecer de debilidades que hagan que sea innecesario atacar al sistema criptográfico. Ejemplos de esto son:

1. Sistemas que fallan debido a problemas en el código, como en la mayoría de los ataques conocidos como de “buffer overflow”, en los cuales el no considerar aspectos de excepción asociados a los sistemas pueden representar que un atacante suficientemente hábil pueda asumir funciones del administrador. Si en un determinado sistema alguien asume las funciones del administrador puede tener muy fácil acceso a claves y sistemas criptográficos haciendo inútil la fortaleza del sistema criptográfico que lo protege.

2. Algunos ataques aprovechan fallas de diseño en los sistema. Si una determinada aplicación ejecuta sin evaluar un determinado comando, los atacantes del sistema pueden conseguir información vital sin necesidad de romper las claves criptográficas que lo protegen.

3. La existencia de sistemas de comunicación móvil, basados en transmisión inalámbrica facilitan actividades de fisgoneo en la señal.

4. Algunas veces lo que se conoce como ataques de ingeniería social pueden resultar muy efectivos a la hora de atacar sistema supuestamente seguros.

5. La poca comprensión que algunos implementadores de sistemas tienen sobre los mecanismos de seguridad implícitos también ocasionan que ciertos sistema sean mas vulnerables. Un ejemplo clásico es la forma en que se implementan algunos sistemas de contabilidad. El sistema de doble registro, ancestralmente usado para proteger sistemas de seguridad se ve violentado por sistemas "que obligan" al cuadre entre cuentas lo cual facilita el ataque a sistemas de esta naturaleza.

1.4.1. Las políticas de seguridad, los objetivos de seguridad y los perfiles de protección

En el establecimiento de las condiciones de seguridad de cualquier sistema juegan un papel fundamental las políticas de seguridad y los perfiles de protección. El modelado de riesgos es el primer paso a dar cuando uno piensa en el diseño de un sistema seguro. ¿Cuales son los riesgos reales que se afrontan con el sistema? ¿Con que frecuencia pueden ocurrir? Y sobre todo ¿Cual es su costo? y ¿Cuanto estamos dispuestos a invertir para garantizar que no ocurran? Esta actividad requiere de una visión sistémica de lo que pueden significar los riesgos de seguridad. No es suficiente con hacer un listado completo de los posibles riegos de seguridad. La idea básica es que debemos tomar en cuenta todas las vulnerabilidades o posibles ataques y estimar las perdidas asociadas a estos ataques desarrollando una expectativa anual de perdidas. Por ejemplo el riesgo de seguridad puede representar una perdida de 10.000.000 de Bs por cada ataque y se puede estimar que se realicen 10 ataques por año. Entonces la perdida estimada para ese modelo de seguridad es de 100.000.000,00 de Bs por año. Eso da cuenta de cuanto está dispuesta la empresa atacada a invertir para evitar estos ataques.

Algunos riesgos o tipos de ataque tienen una probabilidad muy baja de incidencia. Las industrias de seguro hacen este tipo de cálculo todo el tiempo y allí podemos conseguir múltiples ejemplos de como iniciar este trabajo. Para riesgos asociados al ataque de sistemas computacionales hay múltiples herramientas disponibles para su análisis, estas herramientas proveen mecanismos para la realización de este tipo de análisis. El foco debe mantenerse en los grandes riesgos y en todo caso despreciar aquellos hachos que tienen baja probabilidad o bajo costos. Por esto es muy importante poder determinar un "costo esperado" o una "esperanza de ataques exitosos" asociados a estos riesgos.

La ingeniería de seguridad procede secuencialmente desde los requerimientos de seguridad hasta la solución, no desde la tecnología mas novedosa. Esto significa que lo primero que hay que hacer es modelar el tipo de ataques al que se está sujeto, a partir

de esto crear una política de seguridad y luego a partir de esto escoger la tecnología a aplicar para evitar los riesgos antes modelados. Los riesgos determinan la política y esta define la tecnología a utilizar, los pasos a seguir serían los siguientes:

1. Comprender los riesgos reales del sistema y evaluar el las probabilidades de esos riesgos.
2. Describir la política de seguridad requerida para defenderse de esos ataques o riesgos.
3. Diseñar las medidas de seguridad destinadas a contrarrestar esos riesgos..

Una política de seguridad para un sistema define los objetivos. La política debe establecer:

1. Quien es responsable por que. (implementación, reforzamiento, auditoría y revisión).
2. Cuales son las políticas de seguridad básicas de la red, y
3. Por que ellas son implementadas en la manera en que lo son.

En pocas palabras la política de seguridad debe establecer el porque y no el como. Los comos son parte de la táctica. Las medidas de seguridad deben proteger no solo contra las amenazas sino también contra los problemas no previstos. La seguridad absoluta en los sistemas es algo imposible de lograr sin embargo algunos principios básicos pueden ayudar a fortalecer la seguridad de los sistemas, a continuación se listan algunas de estas medidas:

1. Compartimentación, o en otras palabras, nunca colocar todos los huevos en una sola canasta.
2. Proteger los componentes que puedan ser considerados mas débiles del sistema.
3. Usar “puntos de entrada” o de “acceso” bien definidos.
4. Usar técnicas de defensa combinadas para aumentar el nivel de resistencia de un sistema.
5. Manejo de fallas en un entorno seguro.
6. Sacar provecho de la impredicibilidad.
7. La simplicidad siempre es una buena premisa, mientras mas simple es un sistema mas sencillo es garantizar su seguridad.
8. Preparar a los usuarios para las potenciales fallas de seguridad.
9. Cuestionar permanentemente la seguridad del sistema y hacer esfuerzos permanentes desde nuestro lado para romper las medidas que nosotros mismos hemos diseñado.

10. Tener buenos mecanismos de detección, al menos debemos saber que hemos sido atacados o que estamos siendo atacados.

11. Tener buenos mecanismos para detectar los ataques esto incluye: Detección, localización, identificación y evaluación.

12. Tener respuestas preparadas a los potenciales ataques.

13. Estar vigilantes, en forma permanente.

14. Vigilar a los vigilantes, aunque suene paranoico.

15. Tener mecanismos de recuperación.

1.4.2. El acceso (físico o lógico) a los componentes del sistema

En todo sistema seguro es muy importante no descuidar los aspectos físicos de la seguridad. No hay que olvidar que los ataques ocurrirán siempre con mas frecuencia en las partes mas débiles del sistema. De nada sirve una buena escogencia de passwords si los dejamos sobre el escritorio frente a la máquina a la cual da acceso o si el atacante puede llevarse un laptop, muy protegido lógicamente pero sin ninguna protección física.

Capítulo 2

Protocolos

2.1. Rol de los protocolos en la seguridad de un sistema

Un sistema de seguridad típico consiste de un número de principales tales como individuos, compañías, computadoras, lectores de tarjetas magnéticas, los cuales se comunican usando una variedad de canales incluyendo teléfono, e-mail, radio, infrarrojos y a través del envío de información en dispositivos físicos tales como tarjetas bancarias y tickets de transporte.

Un protocolo de seguridad son las reglas que gobiernan esas comunicaciones. Son diseñadas típicamente para que el sistema pueda soportar ataques de carácter malicioso. Protegerse contra todos los ataques posibles es generalmente muy costoso por lo cual los protocolos son diseñados bajo ciertas premisas con respecto a los riesgos a los cuales está expuesto. La evaluación de un protocolo por lo tanto envuelve el responder a dos preguntas básicas: Es el modelo de riesgo realista? El protocolo puede controlar ese nivel de riesgo?

La base fundamental de una buen parte de los sistemas de seguridad en funcionamiento dependen de la confidencialidad de un password. Aun cuando hay muchos problemas de seguridad relacionados al uso de passwords la falla mas común es la factibilidad del atacante de interceptar el password en alguna forma para posteriormente usarlo rompiendo la seguridad del Sistema. Normalmente el password está relacionado a controles de autenticación y acceso.

Como un ejemplo inicial usemos el código usado en forma de una señal infrarroja para permitir a los usuarios de un estacionamiento la entrada y salida del mismo. Una forma de funcionamiento posible es que el dispositivo transmita un número de serie y luego transmita un bloque de autenticación que consiste del mismo serial seguido por un número aleatorio, todo encriptado usando una clave que es única para el dispositivo. En la notación particular que estaremos usando en este texto se describiría sucintamente de la siguiente forma:

$$T \rightarrow G : T, \{T, N\}_{KT}$$

El dispositivo ubicado dentro del carro envía su nombre seguido por el valor en-

criptado de T concatenado con N, donde N es un Número usado una vez o nonce. El propósito del nonce es asegurar al receptor que esta recibiendo un mensaje fresco, esto es , que no es el re-envío de otro mensaje previamente enviado que ha sido capturado previamente por un atacante. La verificación es simple, el receptor en el estacionamiento lee T, consigue la clave correspondiente KT, descifra el resto del mensaje, chequea que el texto descifrado contenga T y finalmente que el nonce N no haya sido usado antes.

El término nonce puede significar cualquier mecanismo que garantice la frescura del mensaje. Un nonce, puede, de acuerdo al contexto ser un número aleatorio, un número serial o un desafío al azar. Hay algunas diferencias sutiles entre esas tres aproximaciones en cuanto al nivel de resistencia que ofrecen a diversos tipos de ataque de reenvío.

El manejo de claves en este tipo de dispositivos suele ser simple. En algunos casos es el número serial encriptado bajo una clave maestra global, KM, conocida por el sistema central:

$$KT = \{T\}_{KM}$$

Algunos problemas típicos de este sistema son:

El uso de un sistema muy simple de verificación como una secuencia alternante del tipo ABABABABABABABA.., con esto un atacante solo tiene que replicar dos mensajes grabados o esperar intentos pares o impares. Cuando se usan número aleatorios el sistema debe recordar los últimos números utilizados incrementando las necesidades de memoria de este tipo de sistemas. Una opción es usar contadores, un problema típico con el uso de contadores es mantener la sincronización entre los principales del sistema. Esto muestra que incluso el diseño de un sistema de seguridad simple como una alarma de carro o una llave electrónica no es un problema trivial.

2.2. Definición de protocolo

Un protocolo es una serie de pasos, que involucra a dos o mas principales, diseñado para realizar una tarea particular. Es importante destacar que incluso en aquellos procesos donde apenas está envuelta una persona puede haber un protocolo que es vulnerable de ser atacado. Consideremos por ejemplo un login. Una persona accede a una ventana que le interroga sobre un nombre de usuario y un password, una vez introducido es autenticado ante el sistema y si los datos introducidos son correctos entonces tendrá acceso al sistema. Incluso en este caso tan simple estamos en presencia de un protocolo que puede ser atacado con el objeto por ejemplo de acceder al sistema con derechos de administrador. Una forma de hacerlo es por ejemplo creando un programa que presente un falso login y dejando la máquina sola ejecutando este programa. Un usuario desapercibido escribirá su login y su password, el programa graba entonces la información en un archivo de la cuenta del usuario tramposo y sale del shell con lo que aparecerá esta vez de nuevo el login (en esta ocasión el correcto). Nótese que en este protocolo de autenticación tan simple hay dos principales interactuando, el sistema y el usuario. La fallas que presenta este ataque explota el hecho que el sistema nunca necesita autenticarse ante el usuario.

Un caso en el que es mas evidente la necesidad de una autenticación mutua es cuando trabajamos en un ambiente distribuido. Algunas veces tendremos la necesidad de garantizar que realmente estamos haciendo login en el sistema que deseamos usar.

Algunas otras características importantes de los protocolos son las siguientes:

Todos los principales envueltos en el protocolo deben conocer el protocolo y todos los pasos a seguir previamente a su ejecución. (En efecto solo se requiere que conozca el subconjunto de partes asociadas a los pasos que ese principal debe seguir) Nota importante: en caso de protocolos que envuelven mas de dos principales, en efectos el que todos conozcan todo el protocolo puede constituirse en una debilidad y facilitar un ataque sobre el sistema. Todos los principales envueltos en el protocolo deben aceptar seguirlo. El protocolo no puede ser ambiguo, debe ser preciso y no debe haber ninguna oportunidad para malas interpretaciones. El protocolo debe ser completo, debe haber una acción especificada para cada situación posible.

Un protocolo criptográfico es un protocolo que usa criptografía. En esta sección mostraremos algunos de los protocolos criptográficos mas conocidos y haremos una breve introducción a la lógica BAN, una herramienta de abstracción que nos permite probar la seguridad de determinados protocolos.

En la literatura criptográfica se suelen usar algunos personajes para simplificar la explicación del funcionamiento de los protocolos:

Alice: Primer participante en todos los protocolos. Bob: Segundo participante en todos los protocolos. Carol: Participante en protocolos que envuelven tres o cuatro principales. Dave: Principal de los protocolos con cuatro participantes. Eve: Participante que escucha sin autorización en un canal. Atacante pasivo. Mallory: Atacante activo, modifica mensajes y actúa en el medio del protocolo. Trent: Arbitro confiable. Walter: Guardia, estará protegiendo a Alice y Bob en algunos protocolos. Victor: Verificador.

2.3. Protocolos para la autenticación de las partes

2.3.1. Sistemas de desafío y respuesta

Los sistemas de seguridad y alarmas para carros mas modernos usan un protocolo mas complejo con dos pasos llamado de desafío-respuesta. Cuando la llave es insertada, una unidad de control en el motor envía un desafío, el cual consiste de un número aleatorio de n-bits. Esto lo hace usando una señal de radio de onda corta. La llave calcula una respuesta encriptando el desafío. De esta forma si E es el controlador en el motor y T es el trasponder en la llave del carro, K la clave criptográfica y N el número aleatorio lanzado como desafío el protocolo puede representarse de la siguiente forma:

$$E \rightarrow T : N$$

$$T \rightarrow E : \{T, N\}_{KN}$$

Este es un esquema bastante mas seguro, sin embargo no es totalmente seguro. Los números aleatorios pueden ser predecibles y pueden ser usados para interrogar a la llave en el bolsillo del propietario. Una de las causas mas frecuentes para la ruptura en

la seguridad de este tipo de sistemas tiene que ver con que el número aleatorio no es realmente aleatorio sino seudo aleatorio. Generar un número verdaderamente aleatorio es un problema relativamente complejo y de esto depende en gran medida la seguridad de los sistemas criptográficos.

Estos sistemas de desafío-respuesta son frecuentemente usados también para ofrecer acceso a sistemas en diversas empresas mediante un mecanismo esencialmente idéntico al antes descrito, el cual puede ser representado de la siguiente forma (siendo S el servidor, P el generador de password, PIN la identificación personal del usuario que permite acceder al generador de passwords, U el usuario y N el número aleatorio que opera como desafío):

$$S \rightarrow U : N$$

$$U \rightarrow P : N, PIN$$

$$P \rightarrow U : \{N, PIN\}_K$$

$$U \rightarrow S : \{N, PIN\}_K$$

Este tipo de sistema ha sido usado en aplicaciones de guerra para identificar los aviones amigos o enemigos. Normalmente cuando un avión se acerca a un radar este le envía un desafío el cual debe recibir, encriptar y reenviar para que pueda ser identificado como un avión del mismo bando. Este sistema es vulnerable a un ataque del tipo hombre en el medio. De hecho durante la guerra entre Sur-africanos blancos (del apartheid) y Namibia y el Sur de Angola un ataque de este tipo ocurrió y le permitió a las fuerzas rusas y cubanas penetrar las defensas antiaéreas de los sur-africanos.

El ataque se realizó de la siguiente forma. Las tropas rusas mantuvieron un grupo de aviones MIGS justo al borde de las defensas antiaéreas de los sur-africanos y esperaron hasta que aviones sur-africanos atacaran un blanco angolanes. En ese momento volaron y atravesaron el espacio aéreo protegido. Los radares sur-africanos enviaron el desafío que fue recibido por los aviones rusos y reenviado a las base antiaérea Angolanes quienes lo reenviaron a los aviones Sur-africanos. Estos respondieron con su identificación encriptada y esta fue reenviada de vuelta hasta los aviones rusos y luego a los radares sur-africanos quienes entonces dejaron pasar al enemigo.

Este tipo de ataques es muy similar a los ataque de reflejo. Este tipo de ataques puede ser contrarrestado a través de la mutua autenticación de las partes. Para evitar este tipo de ataque solo debemos incluir la identificación encriptada de quien responde. De esta forma podemos estar seguros que no se está usando la respuesta reflejada de otro principal.

2.3.2. Protocolos arbitrados, adjudicados y auto-reforzados

Los protocolos pueden ser clasificados en base a su modo de funcionamiento en:

Protocolos arbitrados: Aquellos en los que es necesaria la participación permanente de un tercero para garantizar la seguridad del sistema.

Protocolos adjudicados: Son protocolos que tienen dos partes, una parte no-arbitrada y una arbitrada. El subprotocolo arbitrado se activa solo en caso de disputa.

Protocolos auto-reforzados: No se requiere de una arbitro para garantizar el protocolo.

Por otra parte los ataques criptográficos pueden ser dirigidos contra los algoritmos criptográficos, contra las técnicas criptográficas usadas para implementar el algoritmo o contra los protocolos en si mismos.

Los ataques a este tipo de protocolos también pueden ser clasificados en pasivos, cuando el atacante se limita a escuchar en la línea con el objeto de obtener información o activos, cuando el atacante pretende modificar el contenido de la comunicación para engañar a las partes. También es posible que el atacante sea también uno de los principales en un protocolo y quiera trampear en el proceso, por ejemplo no reconociendo que el fue quien intervino en uno de los pasos de un determinado protocolo.

2.3.3. Intercambio de claves usando criptografía simétrica

Ahora veamos que ocurre en el caso mas simple en el cual Alice y Bob desean intercambiar un mensaje seguro:

Alice y Bob se ponen de acuerdo en cuanto al uso de un determinado sistema de criptografía (compuesto por un algoritmo criptográfico y una o varias claves). Alice y Bob se ponen de acuerdo en la clave. Alice toma su texto claro y lo en-cripta usando la clave, luego lo envía a Bob. Bob recibe el texto encriptado y usando la clave lo descrita.

Todo buen sistema criptográfico depende de la seguridad de clave y no del hecho que el algoritmo sea secreto. Esto implica que la clave debe ser distribuida en secreto.

Una buena parte de los sistemas criptográficos de clave pública dependen de la denominada funciones de un sentido, las cuales son funciones para las cuales dado x se puede calcular fácilmente f(x) mientras que dado f(x) resulta muy difícil o imposible calcular x.

Por otro lado una función de un sentido con puerta trasera es una para la cual es fácil calcular el inverso de f(x) si se conoce cierta información particular. Un sistema criptográfico de clave pública consiste generalmente de dos claves una pública y una privada. La clave pública es usada para encriptar y enviar mensajes. La clave privada es usada para des-encriptar. Si este tipo de sistemas es usado al revés, es decir encriptando con la clave privada y des-encriptando con la clave pública puede ser usado como un mecanismo de autenticación.

El siguiente protocolo asume que Alicia y Bob, usuarios en una red comparte cada uno una clave secreta con un centro de distribución de claves, Trent. (El protocolo asume que las claves han sido distribuidas a través de algún método seguro). El protocolo es el siguiente:

Alice llama a Trent y le solicita un clave de sesión para comunicarse con Bob. Trent genera una clave de sesión aleatoria y en-cripta dos copias de esta clave, una usando la clave que comparte con Alice y la otra usando la clave que comparte con Bob. Trent le envía ambas copias a Alice. Alice des-encripta su copia de la clave de sesión. Alice le envía a Bob su clave de sesión. Bob des-encripta el mensaje de Trent y recupera a

su vez su clave de sesión. Bob y Alice usan esta clave de sesión para comunicarse en forma segura.

2.3.4. Intercambio de claves usando criptografía de clave pública

Una variante del protocolo anterior usa criptografía pública.

1. Alice obtiene la clave pública de Bob usando un centro de distribución de claves.
2. Alice genera una clave de sesión aleatoria, la encripta usando la clave pública de Bob y la envía a Bob.
3. Bob des-encripta el mensaje de Alice usando su clave privada.
4. Posteriormente ambos se comunican usando la clave de sesión aleatoria generada por Alice.

2.3.5. El ataque del hombre en el medio

Estos protocolos pueden ser atacados con el ataque del hombre en el medio, el cual funciona de la siguiente forma:

1. Alice le manda a Bob su clave pública.
2. Mallory intercepta esta clave y envía a Bob su propia clave pública.
3. Bob le envía a Alice su clave pública, Mallory la intercepta y le envía a Alice su propia clave pública.
4. Cuando Alice envía a Bob un mensaje encriptado con su clave pública realmente usa la de Mallory, quien puede des-encriptar el mensaje y luego encriptarlo usando la clave pública de Bob.
5. Cuando Bob envía un mensaje a Alice, Mallory puede hacer lo mismo que con Alice.

2.3.6. Algunos protocolos conocidos:

Needham-Schroeder

Needham-Schroeder:

- Alice le envía un mensaje a Trent que consiste de su nombre, el nombre de Bob y un número aleatorio.

$$A, B, R_A$$

- Trent genera una clave de sesión aleatoria. Encripta el mensaje que consiste de una clave de sesión aleatoria y el nombre de Alice con la clave secreta que comparte con Bob. Luego encripta el valor aleatorio generado por Alice, el nombre de Bob, la clave, y el mensaje encriptado con la clave secreta que comparte con Alice. Finalmente le envía a Alice todos el mensaje.

$$E_A(R_A, B, K, E_B(K, A))$$

- Alice desencripta el mensaje y extrae K. Ella confirma que R_A es el mismo valor que envío a Trent en el paso (1). Luego envía a Bob el mensaje que Trent encripto con su clave.

$$E_B(K, A)$$

- Bob desencripta el mensaje y extrae K. El entonces genera otro valor aleatorio, R_B. El encripta el mensaje con la clave K y la envía a Alice.

$$E_K(R_B)$$

- Alice desencripta el mensaje con K. Ella calcula $R_B - 1$ y lo encripta usando K. Luego ella envía el mensaje de vuelta a Bob.

$$E_K(R_B - 1)$$

- Bob desencripta el mensaje usando K y verifica que es $R_B - 1$.

Kerberos

Kerberos es una variante del protocolo de Needham-Schroeder que se usa mucha mas extensamente. En la versión 5 del protocolo Alice y Bob comparte claves con Trent y Alice quiere generar una clave de sesión para la conversación con Bob:

- Alice envía un mensaje a Trent, con su identidad y la identidad de Bob.

$$A, B$$

- Trent genera un mensaje con un timestamp, un lifetime, L, una clave de sesión aleatoria y la identidad de Alice. El encripta todo esto con la clave que comparte con Bob. Luego toma el timestamp, el lifetime, la clave de sesión y la identidad de Bob y los encripta usando la clave que comparte con Alice. Envía ambos mensajes encriptados a Alice.

$$E_A(T, L, K, B), E_B(T, L, K, A)$$

- Alice genera un mensaje con su identidad y el timestamp, lo encripta usando K y lo envía a Bob. Alice también le envía a Bob el mensaje encriptado en la clave de Bob.

$$E_K(A, T), E_B(T, L, K, A)$$

- Bob crea un mensaje que consiste del timestamp mas uno, encripta esto usando K y envía esto a Alice.

$$E_K(T + 1)$$

Este protocolo asume que el reloj de todos los principales están sincronizados con el reloj de Trent.

2.3.7. Análisis formal de los protocolos:

Como puede observarse a través de los pocos ejemplos que hemos mostrado dependiendo de la complejidad del algoritmo resulta relativamente complicado demostrar que un determinado algoritmo funciona (en el sentido de proveer la seguridad que se aspira). Tampoco es evidente cuales son los aspectos débiles de los algoritmos hasta ahora presentados.

Para poder hacer un análisis adecuado requerimos de herramientas mas formales. Hay cuatro enfoques básicos para atender esta necesidad:

1. Modelar y verificar el protocolo usando lenguajes de especificación y herramientas de verificación que no han sido diseñadas específicamente para el análisis de protocolos criptográficos.

2. Sistemas expertos que puedan ser usados por el diseñador del protocolo para investigar diversos escenarios.

3. Modelar los requerimientos de una familia de protocolos usando análisis lógico del conocimiento y las creencias de la partes envueltas en el protocolo.

4. Desarrollo de un método formal basado en un álgebra diseñada específicamente para ese efecto.

De estas opciones disponibles la mas popular es la tercera. Fue impulsado principalmente por los aportes de Michael Burrows, Martin Abadi y Roger Needham. Ellos desarrollaron un modelo de lógica formal para el análisis del conocimiento y las creencias llamado lógica BAN. La lógica BAN es la herramienta de análisis lógico mas usada en algoritmos criptográficos.

Hay cuatro pasos en el análisis usando lógica BAN:

1. Convertir el protocolo a una forma idealizada, usando oraciones del tipo: Alice cree X, Alice ve X, Alice dijo X, X es un mensaje reciente, etc.

2. Agregar todas las premisas sobre el estado inicial del protocolo.

3. Agregar fórmulas lógicas a las oraciones, afirmaciones sobre el estado del sistema después de cada oración.

4. Aplicar los postulados lógicos a las aseveraciones y premisas para descubrir la creencias de las partes envueltas en el protocolo.

El cuarto enfoque usado hasta ahora (el uso de álgebra) no había recibido mucha atención hasta ahora, sin embargo esto ha venido cambiando recientemente gracias a un trabajo de Michael Merrit en el cual muestra que se puede usar un álgebra para analizar este tipo de protocolos.

La lógica BAN

Burrows, Abadi y Needham en una publicación titulada A Logic of Authentification generan un método sistemático y abstracto para analizar diversos protocolos con base a creencias. El método es bastante simple y se basa en definir cuales son las premisas válidas para las partes involucradas en el sistema antes de iniciarse el protocolo y luego a través del uso de una serie de derivaciones llegar, tras cada paso en el protocolo, a definir el estado en que se encuentran las partes del sistema. Al finalizar el protocolo se pueden dilucidar las creencias de cada uno de los principales involucrados en el mismo.

Este sistema es muy popular para el análisis de las fortalezas y debilidades de diversos protocolos. De hecho, fue usado por el propio Needham para demostrar una falla en un protocolo propuesto por el años atrás y que hasta el momento había pasado inadvertida (falla entendida en el sentido de debilidad contra determinados ataques).

Algunas de la preguntas que queremos poder responder sistemáticamente sobre los protocolos usados son:

1. ¿Qué logra este protocolo?

2. ¿Necesita este protocolo asumir mas premisas que algún otro?

3. ¿Hace este protocolo algo innecesario o que puede ser evitado sin afectar su nivel de seguridad?

4. ¿Encripta este protocolo algún texto que puede ser enviado en claro sin afectar la seguridad del mismo?

Los protocolos de autenticación son descritos indicando los mensajes que han sido enviados entre los principales en cada etapa del protocolo. Para poder analizarlos usando la lógica BAN deben ser transformados en expresiones lógicas aceptadas en este lenguaje.

La notación básica es la siguiente:

- A, B y S denotan principales específicos;

- K_{ab}, K_{as} y K_{bs} denotan claves compartidas específicas,

- K_a, K_b y K_s denotan claves públicas específicas, y $K_a^{-1}, K_b^{-1}, K_s^{-1}$ denotan las correspondientes claves privadas.
- N_a, N_b, N_c denotan un texto específico.
- Los símbolos P, Q y R son rango sobre los principales.
- X y Y son rango sobre las afirmaciones o textos.
- K son rangos sobre las claves de encriptamiento. Todas estas variables pueden ser usadas como metasímbolos o como variables independientes (en forma indistinta).

El único conector proposicional es la conjunción, denotado por la coma. Se asume que la conjunción cumple con las propiedades de asociatividad y conmutatividad. Adicionalmente a la conjunción se usan los siguientes constructores:

- $P \mid\equiv X$: P cree X, o le daría la autorización para creer X. En particular el principal P debe actuar como si X fuese verdad.
- $P \triangleleft X$: P mira X. Alguien ha enviado un mensaje que contiene X a P, quien a su vez puede leerlo y repetirlo.
- $P \mid\sim X$: P una vez dijo X. El principal P en algún momento envío un mensaje que incluía la afirmación X. En el momento actual no se sabe hace cuanto tiempo fue enviado, lo que se sabe es que P creía X cuando lo envío.
- $P \mid\Rightarrow X$: P tiene jurisdicción sobre X. El principal P es una autoridad sobre X y debe creerle en esta materia. Este constructor es usado cuando un principal tiene autoridad delegada sobre alguna afirmación. Por ejemplo una clave de encriptamiento necesita ser generada con cierto cuidado, y en algunos protocolos algunos servidores son considerados como confiables para realizar esta tarea. Esto puede se expresado a través de asumir que los principales creen que el servidor tiene jurisdicción sobre afirmaciones relacionadas con la calidad de la clave.
- $\#(X)$: X es reciente. Esto es, X no ha sido enviado en ningún mensaje en ningún momento antes de esta ejecución del protocolo.
- $P \leftrightarrow^K Q$: P y Q pueden usar la clave compartida K para comunicarse. La clave K es buena en el sentido que ella nunca sería descubierta por ningún principal excepto P y Q , o un principal en el cual ambos tengan confianza.
- $\rightarrow^K P$: P tiene K como clave pública. La clave secreta (el inverso de K) nunca será descubierta por ningún principal excepto P, o un principal en el cual P confíe.
- $P \rightleftharpoons^X Q$: La fórmula expresa que X es un secreto conocido solamente para P y Q y posiblemente para principales en los cuales ellos confían. Solo P y Q pueden usar X para probar su identidad al otro. Frecuentemente X es reciente y secreto. Un ejemplo de un secreto compartido es un password (compartido entre el usuario y el sistema).

- $\{X\}_K$: Esto representa a X encriptado usando la clave K . Formalmente $\{X\}_K$ es una abreviación para una expresión de la forma $\{X\}_K desde\ P$. Se asume que cada principal es capaz de reconocer e ignorar sus propios mensajes.

- $\langle X\rangle_Y$: Esto representa el mensaje X combinado con la fórmula Y, Y es un secreto y su presencia prueba la identidad de quien envío X.

En el estudio de la autenticación nos importa la distinción entre dos épocas, el presente y el pasado. Se considera presente a la actual ejecución del protocolo. Todo protocolo se cuida de no aceptar mensajes del pasado como si fueran pertenecientes a la ejecución actual. Todas las creencias del presente son estables a lo largo de esta ejecución del protocolo. Se asume también que cuando P dice X, también cree X.

Los postulados lógicos son los siguientes:

Las reglas de significado del mensaje, las cuales conciernen a la interpretación del mensaje. Para clave compartidas tenemos que:

$$\frac{P \mid\equiv Q \leftrightarrow^K P, P \triangleleft \{X\}_K}{P \mid\equiv Q \mid\sim X}$$

La cual se lee, si P cree que la clave K es compartida con Q y ve un mensaje X encriptado usando K, entonces P cree que Q una vez dijo X. La única cosa que debemos exigir en esta regla es que P no haya enviado X el mismo.

$$\frac{P \mid\equiv \rightarrow^K Q, P \triangleleft \{X\}_{K^{-1}}}{P \mid\equiv Q \mid\sim X}$$

y

$$\frac{P \mid\equiv Q \rightleftharpoons^Y P, P \triangleleft \langle X\rangle_Y}{P \mid\equiv Q \mid\sim X}$$

La regla de verificación de nonce chequea que el mensaje es reciente y que por tanto quien lo envío aun cree en el:

$$\frac{P \mid\equiv \#(X), P \mid\equiv Q \mid\sim X}{P \mid\equiv Q \mid\equiv X}$$

Esto significa que si P cree que X es reciente y que Q una vez dijo X, entonces P cree que Q cree X. Por simplicidad X debe ser texto en claro. (si no no podemos concluir que Q cree X).

La regla de jurisdicción establece que si P cree que Q tiene jurisdicción sobre X entonces P confía en Q sobre la verdad de X:

$$\frac{P \mid\equiv Q \Rightarrow X, P \mid\equiv Q \mid\equiv X}{P \mid\equiv X}$$

Una propiedad del operador de creencia es que P cree un conjunto de afirmaciones si y solo si P cree cada afirmación separadamente. Esto justifica las siguientes reglas:

$$\frac{P \mid\equiv X, P \mid\equiv Y}{P \mid\equiv (X, Y)}$$

$$\frac{P \mid\equiv (X, Y)}{P \mid\equiv X}$$

$$\frac{P \mid\equiv Q \mid\equiv (X, Y)}{P \mid\equiv Q \mid\equiv X}$$

Una regla similar aplica para el operador $\mid\sim$:

$$\frac{P \mid\equiv Q \mid\sim (X, Y)}{P \mid\equiv Q \mid\sim X}$$

Si un principal ve una fórmula, también ve sus componentes, suponiendo que el tenga las claves necesarias:

$$\frac{P \triangleleft (X, Y)}{P \triangleleft X}$$

$$\frac{P \triangleleft \langle X \rangle_Y}{P \triangleleft X}$$

$$\frac{P \mid\equiv Q \leftrightarrow^K P, P \triangleleft \{X\}_K}{P \triangleleft X}$$

$$\frac{P \mid\equiv \mapsto^K P, P \triangleleft \{X\}_K}{P \triangleleft X}$$

$$\frac{P \mid\equiv \mapsto^K Q, P \triangleleft \{X\}_{K^{-1}}}{P \triangleleft X}$$

Si el principal cree que una parte de la fórmula es reciente entonces cree que la fórmula entera es reciente:

$$\frac{P \mid\equiv \#(X)}{P \mid\equiv \#(X, Y)}$$

La misma clave es usada entre pares de principales en cualquier dirección. La siguientes dos reglas reflejan esa propiedad:

$$\frac{P \mid\equiv R \leftrightarrow^K R'}{P \mid\equiv R' \leftrightarrow^K R}$$

$$\frac{P \mid\equiv Q \mid\equiv R \leftrightarrow^{K} R'}{P \mid\equiv Q \mid\equiv R' \leftrightarrow^{K} R}$$

Similarmente un secreto puede ser usado entre un par de principales en cualquier dirección:

$$\frac{P \mid\equiv R \rightleftharpoons^{X} R'}{P \mid\equiv R' \rightleftharpoons^{X} R}$$

$$\frac{P \mid\equiv Q \mid\equiv R \rightleftharpoons^{X} R'}{P \mid\equiv Q \mid\equiv R' \rightleftharpoons^{X} R}$$

Con base a estas reglas o postulados básicos el análisis de protocolo se realiza a través de los siguientes pasos:

1. Se deriva un protocolo idealizado (usando la sintaxis de la lógica), se deriva del original.
2. Se escriben las premisas sobre el estado inicial.
3. Se añaden fórmulas lógicas a las afirmaciones del protocolo en la forma de afirmaciones sobre el estado del sistema después de cada paso.
4. Se aplican los postulados lógicos a las premisas y afirmaciones con el objeto de descubrir las creencias mantenidas por la partes en el protocolo.

En forma mas precisa se anota el protocolo idealizado con fórmulas y se manipulan esas fórmulas con los postulados. Un protocolo es una secuencia de mensajes enviados $S_1, ..., S_n$ cada uno de la forma $P \rightarrow Q : X$ con $P \neq A$. Una anotación para un protocolo consiste de una secuencia de afirmaciones que se insertan antes del primer mensaje y después de cada mensaje; las afirmaciones que se usan son conjunciones de fórmulas de las formas $P \mid\equiv Q$ y $P \triangleleft X$ .La Primera afirmación contiene las premisas o estado inicial mientras que la última contiene las conclusiones, de la siguiente forma:

$$[asumción]S_1[afirmación_1]...[afirmación_{n-1}]S_n[conclusiones]$$

En esta estructura se desea que las anotaciones o afirmaciones sean válidas en el siguiente sentido: Si las premisas son válidas entonces cada afirmación es válida después de la ejecución del protocolo hasta el paso anterior adonde ella aparece.

Las reglas para realizar anotaciones a los protocolos son las siguientes:

- Para cada paso singular del protocolo: La anotación $[Y](P \rightarrow Q : X)[Y, Q \triangleleft X]$ es legal.
- Para secuencias de protocolo si las anotaciones $[X]S_1...[Y]$ y $[Y]S'_1...[Z]$ son legales entonces $[X]S_1...[Y]S'_1...[Z]$ es legal.

- También valen los siguientes postulados lógicos:

1. Si X es una afirmación (no las premisas) en una anotación legal Δ, y si X' es demostrable a partir de X, y si Δ' es el resultado de substituir X' por X en Δ, entonces Δ' es una anotación legal.

2. Si X son las premisas de una anotación legal Δ, si X' es demostrable a partir de X y Δ' es el resultado de substituir (X, X') por X en Δ, entonces Δ' es una anotación legal.

Como un ejemplo de la aplicación de esta lógica al análisis de protocolos veamos como se usa para realizar un análisis de Kerberos:

Kerberos funciona de la siguiente forma:

1. Mensaje 1 $A \rightarrow S : A, B$ A S: A, B
2. Mensaje 2 $S \rightarrow A : \{T_s, L, K_{ab}, B, \{T_s, L, K_{ab}, A\}_{K_{bs}}\}_{K_{as}}$
3. Mensaje 3 $A \rightarrow B : \{T_s, L, K_{ab}, A\}_{K_{bs}}, \{A, T_a\}_{K_{ab}}$
4. Mensaje 4 $B \rightarrow A : \{T_a + 1\}_{K_{ab}}$

A partir de esto podemos escribir el protocolo idealizado de la siguiente forma:

1. Mensaje 2 $S \rightarrow A : \{T_S, (A \leftrightarrow^{K_{ab}} B), \{T_S, A \leftrightarrow^{K_{ab}} B\}_{K_{bs}}\}_{K_{as}}$
2. Mensaje 3 $A \rightarrow B : \{T_S, A \leftrightarrow^{K_{ab}} B\}_{K_{bs}}, \{T_a, A \leftrightarrow^{K_{ab}} B\}_{K_{ab}} fromA$
3. Mensaje 4 $B \rightarrow A : \{T_a, A \leftrightarrow^{K_{ab}} B\}_{K_{ab}} fromB$

El paso 1 es eliminado en el protocolo idealizado puesto que no aporta nada adicional a la seguridad del mismo. Igualmente se tratan el $timestamp$ y L como si fueran uno solo ya que ambos consiguen el efecto de garantizar que el mensaje es reciente.

Ahora se procede al análisis para lo cual lo primero es escribir las premisas:

$$A \mid\equiv A \leftrightarrow^{K_{as}} S$$

$$S \mid\equiv A \leftrightarrow^{K_{as}} S$$

$$S \mid\equiv A \leftrightarrow^{K_{ab}} B$$

$$A \mid\equiv (S \Rightarrow A \leftrightarrow^{K} B)$$

$$A \mid\equiv \#(T_S)$$

$$B \mid\equiv B \leftrightarrow^{K_{bs}} S$$

$$S \mid\equiv B \leftrightarrow^{K_{bs}} S$$

$$B \mid\equiv (S \Rightarrow A \leftrightarrow^{K} B)$$

$$B \mid\equiv \#(T_S)$$

$$B \mid\equiv \#(T_a)$$

A recibe el mensaje 2, y de las reglas de significado del mensaje derivamos:

$$A \mid\equiv A \leftrightarrow^{K_{ab}} B$$

$$A \triangleleft \{T_S, A \leftrightarrow^{K_{ab}} B\}_{K_{bs}}$$

En el mensaje 3 a pasa el mensaje encriptado que recibió del servidor de autenticación junto con otro mensaje que contiene un Timestamp. Inicialmente B puede desencriptar solo una parte, con lo que:

$$B \mid\equiv A \leftrightarrow^{K_{ab}} B$$

Y posteriormente,

$$B \mid\equiv A \mid\equiv A \leftrightarrow^{K_{ab}} B$$

El resultado final es:

$$A \mid\equiv A \leftrightarrow^{K_{ab}} B$$

$$A \mid\equiv B \mid\equiv A \leftrightarrow^{K_{ab}} B$$

$$B \mid\equiv A \leftrightarrow^{K_{ab}} B$$

$$B \mid\equiv A \mid\equiv A \leftrightarrow^{K_{ab}} B$$

Capítulo 3

La autenticación y el acceso

En este capítulo trataremos el tema general de autenticación y acceso. La autenticación es el procedimiento mediante el cual demostramos que somos quienes decimos ser. El acceso se refiere a los recursos de diversa naturaleza que puedo usar en base a mi identidad. El acceso está directamente vinculado a la autorización, que es el procedimiento mediante el cual se me da acceso a un determinado recurso. La autenticación puede realizarse de múltiples formas que veremos mas adelante y puede representar el medio a través del cual se nos da acceso a un recurso. En este contexto el término recurso puede representar un sistema, un espacio físico etc.

3.1. La seguridad física

En algunas ocasiones lo que pretendemos lograr con un determinado sistema de seguridad es evitar el acceso físico de personas no autorizadas a un área protegida. Ese área protegida puede ser la bóveda de un banco, un museo o incluso los circuitos protegidos de una tarjeta de encriptamiento en un determinado sistema. Para lograr este objetivo se requiere de un mecanismo de protección física (un medio infranqueable o pretendidamente infranqueable), de un medio para autentificar y a luego autorizar el acceso a determinados individuos o sistemas y de mecanismos de detección de intrusos.

Un aspecto importante a considerar al diseñar sistemas seguros es la existencia de puertas traseras, es decir mecanismos muy simples de acceso que no fueron considerados cuando se realizó el diseño del sistema. Cuando hablamos de seguridad física esto se traduce en la revisión exhaustiva de planos de acceso, y cuando hablamos de sistema computarizados en la revisión permanente de las interacciones de los diversos componentes del sistema. Un sistema de seguridad físico está constituido por varios componentes:

1. Un mecanismo para evitar el acceso.
2. Un mecanismo para detectar el acceso.
3. Una alarma.

4. Un mecanismo para impedir o retrasar la huida.

5. Un mecanismo para responder al ataque.

3.1.1. Modelo de riesgo

Cuando se diseña un sistema para manejar la seguridad de un espacio físico se debe considerar el tipo de atacante al que pudiéramos enfrentarnos. Una posible clasificación, que usaremos acá es la realizada por RL Barnard [**?**]:

1. Un atacante sin habilidad.

2. Un atacante hábil.

3. Un atacante muy hábil que además puede contar con ayuda interna.

4. Un atacante muy hábil, que puede contar con ayuda interna y además tiene una cantidad sustancial de recursos que puede usar para atacar el sistema.

El diseño de un determinado sistema de seguridad física tiene relación directa con el tipo de atacante que esperamos tener. Para defendernos de un atacante del tipo 4 quizás requiramos demasiados recursos. En todo caso hay un compromiso permanente entre el valor de lo que protegemos y el tipo de diseño de seguridad que se requiere.

Por otra parte también existe una relación directa entre el “perímetro” protegido, falsas alarmas y el tiempo que puede tener el personal de seguridad para atrapar al atacante. Un error frecuente es el de dejarse deslumbrar por sistemas de alarma muy sofisticados que pueden ser burlados con relativa facilidad. De nada sirve un sistema de alarma ultra-sensible si el atacante puede quedarse dentro del aŕea protegida en momento en que el acceso está permitido. Múltiples dispositivos le pueden permitir generar una señal interpretada como “normal” y puede tratar de afectar sistemas de detección. Para evitar ataques al sistema de comunicación entre el detector y la alarma, o entre la alarma y los vigilantes o la policía se usan mecanismos de autenticación entre las partes y encriptamiento.

Otra forma de atacar es crear múltiples falsas alarmas, esto generalmente tiene el efecto de reducir la respuesta de quienes están encargados de vigilar. La energía eléctrica también es un punto de falla. Finalmente disparar múltiples alarmas en un zona también puede servir como método de ataque toda vez que los organismos de seguridad tienen demasiados sitios adonde dirigirse.

3.2. Gestión de Passwords

Como pudimos ver en la sección de este curso dedicada a los protocolos de autenticación, autentificar dos dispositivos electrónicos (mutuamente) es una tarea relativamente fácil de realizar. Un sistema típico requiere que un usuario se autentifique ante un dispositivo cliente (cajero electrónico, teléfono celular, etc) y luego este dispositivo cliente se autentifica con el servidor (sistema informático del banco o la empresa de

telefonía). La proliferación de este tipo de servicios hace del problema de gestión de password uno de los mas relevantes en el tema de seguridad.

Cuando se diseñan los mecanismos de acceso a un sistema y el método de autenticación del usuario se usa algo que el usuario es (biometría), o algo que el usuario conoce (passwords) o algo que el usuario tiene (un smartcard o un generador aleatorio de tokens). El sistema mas comúnmente usado es el que se basa en algo que el usuario conoce.

En muchas ocasiones las fallas de seguridad que están asociadas al manejo de passwords están relacionadas a la falta de entrenamiento al usuario en cuanto a como seleccionar un password y de que manera administrarlo. Los usuarios suelen seleccionar passwords que son fáciles de recordar para lo cual con mucha frecuencia usan combinaciones de datos que les son familiares (fecha de nacimiento, placa del carro, nombre de la esposa o esposo, etc). Esto reduce notablemente el espacio de posibilidades que un atacante tiene que explorar para atacar con éxito una determinada cuenta.

En aquellos casos en los que por alguna razón o se les ha suministrado un password o deben escoger uno con reglas mas complicadas entonces puede ser que lo escriban en un papel comprometiéndolo al dejarlo en el lugar de acceso.

Por ejemplo, es mucho mejor si se establece una política clara de como debe manejarse el password. Sin embargo, la seguridad del sistema no puede depender exclusivamente de la capacidad de entrenar al usuario. Un estudio realizado sobre un grupo de estudiantes de computación generó datos interesantes al respecto [**?**]En ese estudio se usaron tres grupos de control a quienes se les indicó como deberían seleccionar sus passwords:

A los primeros (grupo rojo) se les usó como grupo de control y se les sugirió que escogieran un password cualquiera de al menos 6 caracteres.

Al grupo verde se le pidió que pensara en una frase y escogiera letras de allí para formar el password. (por ejemplo: “esta mañana estuve en clases sentado con un perro desdentado”)

Al grupo amarillo se le pidió que seleccionara 8 caracteres (alfabéticos o numéricos) al azar de una tabla que se les entregó, que escribieran el password en alguna parte y que luego destruyeran el papel luego de haberlo memorizado (con el uso).

Como resultado de este estudio se identificó que el 30 % de los passwords del grupo de control podían ser hallados usando passwords de crackeo, mientras que en los otros casos solo se tenía éxito con 10 % de los casos. Por otro lado para el grupo amarillo fue mucho mas difícil recordar los passwords mientras que era similar la capacidad de recordarlos en los grupos rojo y verde.

Por otro lado la creciente proliferación de sistemas antes los cuales un usuario se tiene que autentificar introduciendo un password han crecido llevando al usuario a seleccionar con frecuencia el mismo password para diversos sistemas, aumentando así su vulnerabilidad.

Otra fuente común de fallas es el diseño pobre de algunos sistemas o el cambio en las condiciones de entorno. Un sistema puede proteger de la intrusión de usuarios externos al sistema y de la intrusión de usuarios internos al sistema con diferentes niveles de seguridad. El manejo de passwords en Unix se hace de tal forma que es mas fácil obtener la cuenta del administrador si se es un usuario del sistema que si se es alguien externo.

En redes locales los passwords puede que sean enviado “en claro” por el cable. Un usuario medianamente hábil usando un sniffer puede identificar passwords de otros usuarios.

Todos estos ejemplo muestran aspectos diversos del problema. Revisemos algunos aspectos importantes a considerar en este tema.

3.2.1. Lo que nadie piensa: Ingeniería Social y falso login

Una forma común de ataque es la ingeniería social. No importa que tan bien esté diseñado un determinado sistema si un atacante puede llamar telefónicamente al administrador de sistemas de una compañía haciéndose pasar por un alto ejecutivo y luego de algunos días haciendo esto llamarle para pedirle una cuenta con privilegios basándose en alguna excusa creíble. En un estudio sistemático hecho al respecto [**?**] se le envió un e-mail a 336 estudiantes de ciencias de la computación en la Universidad de Sidney pidiéndoles que enviaran de vuelta su password con el pretexto de que era requerido para validar la base de datos de passwords. Sorprendentemente 138 de ellos devolvieron un password válido, 30 enviaron un password parecido mientras que cerca de 200 cambiaron su password.

El ataque del falso login es otro ejemplo en el cual el atacante se encuentra camuflado de forma inesperada. Un usuario malicioso puede dejar corriendo un programa en un terminal que simule el login de algún sistema operativo. Un usuario desprevenido intenta ingresar y provee el login y luego el password. El programa almacena la información en la cuenta del usuario y luego sale. Normalmente el usuario desprevenido verá un comportamiento extraño del terminal pero es difícil que detecte el ataque.

Se usan sistemas mas sofisticados para realizar ataques de este tipo. La colocación de falsos cajeros electrónicos o dispositivos de venta automática usando ATM es otro método para realizar el mismo tipo de ataque.

La moraleja de esto es que no importa que tan sofisticado sea el sistema usado ni la complejidad del password, los aspectos siempre van a ocurrir con éxito en los aspectos mas débiles del sistema y el factor humano suele ser una fuente común de debilidad en sistemas que se piensan que son seguros.

3.2.2. Seguridad vs Capacidad de uso

Al diseñar sistemas de autenticación usando passwords es importante conseguir un balance adecuado entre seguridad y capacidad de uso. En la medida en que el password es mas corto es mas fácil de recordar y de introducir, con lo que se hacen mucho menos probables los intentos fallidos de acceso al sistema. Sin embargo de esta forma lo hacemos mas vulnerable al ataque externo y es mas fácil de adivinar. Los bancos usan típicamente 4 números en sus cajeros electrónicos (combinándolo con un máximo de reintentos lo cual evita que se pueda hacer una prueba exhaustiva sobre una cuenta) mientras que el sistema de “armado” de misiles nucleares (en el caso de los Estados Unidos) usa 12 caracteres.

3.2.3. Tipos de ataque

Al analizar el diseño que tiene un determinado sistema de seguridad y el modo en que usa los passwords como medio de identificación y acceso de los usuarios se debe tomar en cuenta contra que tipo de ataque estamos protegiendo el sistema. Los niveles de seguridad están relacionados con el tipo de ataque que es factible realizar en un momento determinado (con cierto éxito). Es responsabilidad del diseñador prever los distintos modos posibles en que su sistema puede ser vulnerado.

Los objetivos del ataque pueden variar. Un atacante puede querer ganar acceso a una cuenta, a cualquier cuenta o a cualquier cuenta en cualquier sistema. Incluso este tipo de ataques no tiene porque estar desconectado. El atacante puede querer tener acceso a una cuenta cualquiera para luego tener acceso a una cuenta en particular (por ejemplo el administrador). Por otro lado debido a las frecuencia de uso de malas prácticas en la selección de passwords suele ser más fácil ingresar a cualquier cuenta que a una en particular.

El atacante también puede querer evitar que un determinado usuario tenga acceso a un servicio (ataque de bloqueo de servicio). Este último puede ser muy sencillo de ejecutar si un sistema pone restricciones sobre el número de veces que se intenta acceder al sistema, lo único que el atacante tiene que hacer es superar esta barrera y el atacado estará incapacitado para usar el servicio.

En esta guía revisaremos dos tipos de ataques, los que se realizan en el momento del ingreso del password y los que se realizan sobre información almacenada.

3.2.4. Ataques en el ingreso del password:

Son todos aquellos ataques que se realizan aprovechando vulnerabilidades de la forma en que el usuario debe ingresar su password. Veamos algunos ejemplos:

Interfase de acceso

En muchos casos la interfase de acceso está mal diseñada facilitando que el atacante pueda obtener el password mirando cuando el usuario ingresa la información. Un ejemplo de esto es la forma en que se realiza el pago usando tarjetas de débito en automercados o restaurantes. La interfase en la que se tipea suele ser relativamente fácil de mirar desde diversos ángulos en la mayoría de los casos. Cuando uno frecuenta un supermercado y suele pagar de esta forma aumentan los riesgos de comprometer el password.

Un posible ataque de este tipo es el siguiente: El atacante (que en este caso debe ser el mismo cajero) podría pasar la tarjeta por un dispositivo falso (muy parecido al verdadero) que primero grabara la información magnética y luego almacenara la información que el usuario tipea. Desde el punto de vista del usuario fue un intento fallido de uso de su sistema de pago. El cajero una vez almacenada la clave pasa la tarjeta por un punto de venta verdadero, el cual por supuesto en esta ocasión funciona.

El diseño de algunos cajeros también facilita que las personas que están en la cola puedan observar el número introducido por el usuario, debido a la altura a la que se

encuentra el teclado y en no pocas ocasiones debido a la poca precaución del usuario de la tarjeta.

Fisgoneo

Una forma en que se puede comprometer el password en sistemas con un pobre diseño es cuando el mismo se envía en claro entre las partes. Una versión del ataque anterior podría ser "pinchar" el cable que conecta el teclado con el dispositivo principal y recuperar todos los datos enviados. Igualmente en una red local es posible realizar este ataque usando un "sniffer". Una herramienta que permite monitorear todo el tráfico entre máquinas. Si ese tráfico no es encriptado es muy sencillo para el atacante, si tiene acceso a una máquina en la red interna conseguir el password de cualquier usuario del sistema.

Mutua autenticación

En ocasiones no es suficiente que el usuario autentifique su identidad a través del uso de un password sino que se requiere que el sistema, a su vez, se identifique ante el usuario. Un ejemplo clásico de ataques que se pueden realizar por falta de autenticación del sistema es el del falso login. Un atacante puede dejar ejecutando un programa que simula la interfase de login del sistema y almacenar en esa forma la información de login y password del próximo usuario del sistema. Por eso algunos sistema han introducido trucos como el uso de la secuencia Ctrl-Alt-Del en sistemas como NT, con la particularidad que esa combinación tiene comportamientos diferentes si se ejecuta al momento del login que si se ejecuta desde la cuenta de un usuario.

El problema de mutua autenticación se agudiza cuando hablamos de usar sistema remotos. Un atacante puede a través de algún medio hacerse pasar por una página web conocida para un determinado usuario y de esta manera conseguir información confidencial. Esto requiere que el sistema en el cual se accede también sea autentificado.

Errores de diseño

En algunas ocasiones determinado errores de diseño pueden facilitar el acceso a un password. Veamos un ejemplo en que se facilita la tarea de adivinar el password. Si el password que se usa en un sistema tiene n caracteres y el número de caracteres aceptado es A en teoría adivinar el password le tomaría a un atacante $\frac{A^n}{2}$ intentos en promedio. En algunos casos el sistema chequea el password en forma secuencial y se detiene tan pronto encuentra un carácter erróneo. Esto permite serializar la búsqueda del password. Se busca el primer caracter, luego el segundo y así sucesivamente. Se sabe que se ha tenido éxito por el tiempo que tarde el sistema en responder esto hace que el número de intentos en promedio que tenga que hacer se reduzca a $\frac{A^{\frac{n}{2}}}{2}$

3.2.5. Ataques al almacenamiento del password:

La seguridad del sistema depende en gran medida de la forma en que el password es almacenado y de quienes tienen acceso a esta información. En general los pass-

words deben ser almacenados encriptados para evitar que una persona con acceso al sistema pueda conseguirlos. Por otro lado se debe restringir el acceso a ese archivo de tal forma que nadie que no sea el sistema pueda revisarlo. La modalidad de passwords encriptados utiliza una función Hash. En lugar de almacenarse el password de la forma en que lo introduce del usuario lo que se almacena es el resultado de aplicarle una función hash. Cuando el usuario va a hacer login se aplica la transformación sobre el password y se compara el resultado con el valor almacenado, si son iguales el usuario tiene acceso.

El archivo de auditoría

Otra potencial debilidad de los sistema tiene que ver con los registros de auditoría. El sistema puede almacenar los intentos fallidos de acceso al sistema facilitando la tarea de adivinar un password. Si mi password es 7654 y escribí 7653 y eso queda registrado el atacante lo que requiere es conseguir acceso a este archivo y romper cualquier password de lo que se encuentra allí registrado puede ser muy fácil.

Ataques de diccionario

Los ataques de diccionario se basan en que la mayoría de los usuarios escogen passwords que son combinaciones de palabras con sentido en algún idioma. Esto reduce la cantidad de intentos que el atacante tiene que realizar. La forma en que funciona en forma simplificada es la siguiente: El atacante recupera de alguna forma el archivo de passwords encriptados, luego usa un programa al cual provee de un diccionario de palabras claves, a estas palabras claves añade la información que ha podido reunir de la víctima (fecha de nacimiento, nombre de los hijos, placa del carro, etc, etc) y ejecuta el programa. Este programa realiza combinaciones de las palabras en el diccionario y las encripta para luego compararlas con las que están en el archivo de passwords.

Si el atacante no desea acceder a una cuenta en particular sino a cualquier cuenta en un sistema el ataque es mas sencillo pues se puede invertir. Se comparan palabras encriptadas con todo el archivo hasta que se encuentra una coincidencia. Este ataque tiene altas probabilidades de éxito si el número de usuarios es grande.

3.3. Certificados

Los certificados son la contraparte electrónica de la cédula de identidad, el pasaporte o la licencia. Una persona o empresa puede presentar su certificado electrónicamente para probar su identidad o su derecho al acceso a cierta información o servicios en línea. Los certificados vinculan una identidad a un par de claves electrónicas que pueden ser usadas para encriptar y firmar información digital. Un certificado hace posible el verificar la afirmación hecha por alguien de que tiene derecho a usar una clave determinada, lo que ayuda a prevenir el uso de claves “inventadas” o modificadas para suplantar a otras personas. Si se usa conjuntamente con encriptamiento provee una solución de seguridad bastante completa.

Un certificado es emitido por una Autoridad Certificadora y es firmado con la clave privada de esa autoridad. Un certificado normalmente contiene los siguientes datos:

1. La clave pública del propietario del certificado.
2. La fecha de expiración de la clave pública.
3. El nombre de la autoridad certificadora que lo expidió.
4. Número serial del certificado.
5. La firma digital de la autoridad certificadora.

El formato generalmente aceptado para certificados es la norma X.509 de la CCITT.

Los certificados pueden ser usados para una variedad de transacciones electrónicas incluyendo envío y recepción de e-mails, comercio electrónico, transferencia electrónica de fondos y otros. Algunos browsers como Netscape requieren de un certificado para cada servidor seguro. Como un ejemplo, un cliente que esta comprando en un mall electrónico exige que este tenga un certificado del servidor para autentificar la identidad del operador del mall y el contenido provisto por el comerciante. Si no se autentifica el servidor el comprador no provee información delicada como el número de la tarjeta de crédito.

Los certificados usan técnicas de encriptamiento de clave pública, en el cual se usan dos claves vinculadas, una clave pública y una clave privada. La clave pública se publica para que pueda se accedida por cualquiera que quiera comunicarse con el dueño del par de claves. La clave pública puede ser usada para verificar que un mensaje fué firmado por el dueño del par, es decir con la clave privada, o también puede ser usado para encriptar mensajes con la clave pública de tal forma que solo puedan ser leidos si se desencriptan con la clave privada.

En los certificados el par de claves son vinculadas o relacionadas a la información del dueño de las claves. Cuando se instalan en un browser el certificado funciona como una credencial electrónica que el sitio puede chequear. Esto habilita a los certificados la posibilidad de sustituir los diálogos de passwords.

Múltiples certificados pueden ser anexados al mensaje o transacción formando una cadena en la que cada certificado certifica la validez del anterior.

3.3.1. La norma X.509

El estándar X.509 define que información va en el certificado y describe el formato de esos datos. Todos los certificados X.509 tienen los siguientes datos, adicionalmente a la firma:

1. Versión: Identifica que versión del estándar X.509 aplica para este certificado, lo cual afecta que información contiente. Hasta ahora solo tres versiones han sido definidas.
2. Número Serial: La entidad que creo el certificado es responsable por asignarle un número serial que lo distinga de otros certificados que ha emitido. Esta información es usada en diversas formas, por ejemplo cuando un certificado es

revocado su número serial es publicado en una lista de certificados revocados (CRL: Certificate Revocation List)

3. Identificador del Algoritmo de Firma: Esto identifica el algoritmo usado por la autoridad certificadora (CA) para firmar el certificado.

4. Nombre del expedidor: Es el nombre en formato X.500 de la entidad que firmó el certificado. Normalmente es una autoridad certificadora. El uso del certificado significa que se confía en la autoridad certificadora. Nótese que en algunos casos como cuando estamos en el nivel máximo de la jerarquís certificadora la CA firma su propio certificado.

5. Período de validez: Cada certificado es válido por una cantidad limitada de tiempo. El período es descrito por una fecha y hora de inicio y por una fecha y hora de finalización. El tiempo puede ser tan corto como unos pocos segundos o tan largo como un siglo. El período de validez seleccionado depende de varios factores tales como la fortaleza de la clave privada que se usó para firmar el certificado o la cantidad que uno está dispuesto a pagar por el certificado. Este es el período durante el cual terceros pueden confiar en el certificado si la clave privada asociada no es comprometida.

6. Nombre del sujeto: Es el nombre de la entidad cuya clave pública identifica el certificado. Ese nombre usa el estándar X.500, con lo cual se supone que es único a través de Internet. Este es el nombre distintivo de la entidad, por ejemplo: CN=Java Duke, OU=Java Software Division, O=Sun Microsystems Inc, C=US (estos campos se refieren al nombre común, unidad dentro de la organización, organización y país)

7. Información de la clave pública del sujeto: Esta es la clave pública de la organización nombrada en el certificado junto con un identificador de algoritmo que especifica cual sistema de criptografía pública se está usando así como otros parámetros claves asociados.

X.509 Versión 1 está disponible desde 1988, es de amplio uso y es el más genérico. X.509 Version 2 introduce el concepto de identificadores únicos para el expedidor y para el propietario para manejar la posibilidad de reuso tanto del propietario como del expedidor en el tiempo. La mayoría de las políticas asociadas a la gestión de certificados recomiendan que no se reusen y que los certificados no deben hacer uso de identificadores únicos. La Versión 2 casi no es usada. X.509 versión 3 es la más reciente (1996) y soporta la noción de extensiones mediante la cual cualquiera puede definir una extensión e incluirla en el certificado. Algunas extensiones comunes en uso hoy día son: KeyUsage (limita el uso de las claves para propósitos específicos como por ejemplo “solo para firmar”) y AlternativeNames (permite a otras identidades el ser asociadas con esa clave pública, por ejemplo, nombres de DNS, direcciones de Email, direcciones IP). Las extensiones pueden marcarse como críticas para indicar que la extensión debe ser chequeada y usada. Por ejemplo si un certificado tiene la extensión KeyUsage marcada como crítica y con el valor “keyCertSign” entonces si el certificado

es presentado durante una comunicación SSL debe ser rechazado ya que la extensión del certificado indica que debe ser usada solo para firma.

Toda la data en un certificado es codificada usando dos estándares relacionados ASN.1/DER. Abstract Syntax Notation describe la data. El Definite Encoding Rule describe un forma de almacenamiento y envío de la data.

El grupo de trabajo PKIX del Internet Engeenering Task Force está en proceso de definir los estándares para PKI (La infraestructura de clave pública para Internet)

3.4. Biometría

Otro método que permite autentificar la identidad de alguién ante un sistema es la biometría. La biometría basa su principio de funcionamiento en el uso de rasgos distintivos de carácter individual que se supone son únicos (o prácticamente en el sentido probabilístico). Un concepto relevante al evaluar un sistema de autentificación basado en el uso de biometría es el de “falsa aceptación” y el de “falso rechazo”, tecnicamente conocidos como errores tipo 1 y errores tipo 2 respectivamente. Ocurre una falsa aceptación cuando el sistema acepta una identidad que es falsa. Y hay un falso rechazo o error tipo 2 cuando rechazo a alguién que debía ser aceptado. Existe un compromiso obvio entre ambos, para un mismo sistema y una determinada tecnología a menor tasa de error en uno las tasa de error en el otro aumenta. Generalmente se mide la calidad de un sistema en función de la tasa de error en que ambos valores se igualan.

3.4.1. Reconocimiento de firmas

El reconocimiento de firmas ha sido uno de los métodos tradicionales para comprometer o verificar la aceptación de un particular sobre una determinada operación. La firma de contratos y la emisión y pago de cheques son casos clásicos de sistemas que dependen de una correcta revisión de las características de la firma. Generalmente esta revisión es realizada manualmente. En los bancos se suele usar una cifra límite a partir de la cual es necesario revisar en registros del banco la firma original. Hacerlo de otro modo haría el proceso demasiado engorroso y costoso comparativamente con el número de casos de fraude y el valor de esos fraudes. En un estudio realizado usando a 105 sujetos expertos en revisión de firmas a los cuales se les dió a cada uno 144 pares a comparar realizaron una mala asignación en el 6,5 % de los casos.[**?**]Por otro lado se uso un grupo de control que lo hicieron incorrectamente en el 38.3 % de los casos.

El reconocimiento de firmas en forma digital implica técnicas de procesamiento de imágenes relativamente complejos por lo que una solución mas común es el uso de tabletas para firmas en las cuales se registra no solo la forma sino la velocidad, los movimientos y otros factores característicos que pueden ayudar a identificar al sujeto. Los métodos comunes de reconocimiento de firmas tiene una tasa de error igual (en el valor que se igualan los errores de tipo 1 y tipo 2) de 1 %. Es útil dependiendo del tipo de aplicación, para reconocimiento y acceso a un sistema puede estar bien pero es demasiado alto para autorizar transacciones electrónicas de pago.

3.4.2. Reconocimiento de rasgos faciales

Otro método tradicional de identificación es el reconocimiento de rasgos faciales. El ser humano desarrolló a lo largo de su evolución una elevada capacidad de reconocimiento de rasgos faciales. El hecho de que hoy en día una buena cantidad de sistemas de seguridad dependen de fotos en carnets o de algún método de reconocimiento facial demuestra este hecho. Sin embargo existe una falsa percepción con respecto a que tan buenos somos los seres humanos comparando rasgos faciales en el caso de desconocidos. Un estudio realizado con la ayuda de una cadena de supermercados y de un banco permitieron a investigadores de la Universidad de Westminter[**?**]establecer algunos datos al respecto. Reclutaron 44 estudiantes y les proveyeron de identificaciones, de la siguiente forma:

1. Identificaciones del tipo “bueno, bueno”, es decir fotos recientes pertenecientes al individuo.
2. Identificaciones “bueno,malo”, una identificación verdadera pero con una foto desactualizada.
3. Identificaciones “malo, bueno”, una identificación falsa de gran parecido con quien la presenta.
4. Identificaciones del tipo “malo, malo”, una identificación falsa sin ningún parecido.

El resultado fué sorprendente, a pesar de que se usaron cajeros experimentados casi niguno podía identificar entre identificaciones “bueno, malo” de identificaciones “malo, bueno”, incluso no podían hacerlo entre identificaciones del tipo “bueno, bueno” y “malo,malo”. Esta dificultad, reconocida en este estudio, en una buena razón para intentar automatizar el proceso. Sin embargo hacer un reconocimiento automatizado de rasgos faciales resulta ser una taréa bastante compleja. Su tasa de error igual es cercana al 20 % cosa que lo hace inaceptable por los momentos como método de reconocimiento de los individuo.

3.4.3. Reconocimiento de huellas digitales

El reconocimiento de huellas digitales ha sido un método ampliamente utilizado para la identificación de personas, especialmente en el contexto de crímenes y actos ilegales. Se basa en el reconocimiento de patrones característicos en las huellas digitales. Galton desarrollo un esquema para clasificar las figuras que aparecen y Edward Henry, un comisionado de la policía metropolitana de Londres desarrollo un mecanismo para clasificarlas. Para ello organizó un esquema que se basa en si un individuo tiene o no una forma particular (whorl) en sus diez dedos. Esto permitió crear un sistema de clasificación que parte en 1024 los registros y reduce en varios órdenes de magnitud la longitud de la busqueda.

Para reducir costos y la dificultad de compararlos se han desarrollado múltiples sistemas automatizados. Si le interesa usted puede encontrar algunos algoritmos orientados al procesamiento de imágenes en [**?**], hay un tutorial y la descripción de us

sistema en [**?**]. Algunos sistemas también usan la lectura de la huella digital para autentificar a las personas en tiempo real [**?**]. Por otro lado la tasa en que se iguala los errores tipo 1 y 2 es un poco inferior al 1 %. La huella digital tiene el problema que puede ser "falsificada" ya que puede ser reproducida para ser usada posteriormente.

3.4.4. Reconocimiento del Iris

El último método basado en biometría que revisaremos es el reconocimiento del Iris. Existen algunos otros como el reconocimiento de la voz, la forma de la mano, de la oreja, los patrones de calor en la cara que pueden ser revisados en bibliografía especializada si es de interés para el lector. El reconocimiento del iris es uno de los métodos mas confiables que existe. No solo se estima que es un rasgo verdaderamente único que tiene poca influencia genotípica con lo que incluso en gemelos idénticos es diferente sino que es diferente entre un ojo y otro. Además es el que muestra una tasa de error inferior bejo condiciones de laboratorio. Para su uso se aplica una técnica de procesamiento de señales (filtros Gabor) la cual extrae la información del Iris en un código de iris de 256 bytes. La técnica envuelve la aplicación de una transformada sobre los datos que se obtienen de la lectura concéntrica del iris entre la pupila y el lado externo del iris. Entre las propiedades interesantes que tiene es que dos códigos tomados del mismo iris coinciden en el 90 % de sus bits. Provee el mínimo nivel de los errores tipo 1 (cero en prubeas de laboratorio).

3.5. Gestión de Recursos en sistemas computarizados

Una buena parte de los problemas de seguridad que enfrentamos tienen que ver con la gestión de recursos. Quien, cuando y como alguién puede hacer uso de un determinado recurso del sistema y por supuesto el poder evitar que alguién no autorizado logre gestionar recursos del sistema para los cuales no está autorizado. Este ha sido historicamente uno de los centros de gravedad de la seguridad de sistemas informáticos. Logicamente este problema se ha hecho cada vez mas complejo por una parte y por otra parte cada vez se ocupa un espacio menor en el vaso territorio de la seguridad. Sin embargo es fundamental que entendamos las opciones básicas que tenemos para proteger la gestión y el acceso de nuestros sistemas

3.5.1. Control de acceso a diferentes niveles del sistema

El control de acceso funciona a diferentes niveles del sistema y con diversos niveles de complejidad:

1. El primer nivel son los mecanismos de control de acceso que los usuarios ven a nivel de las aplicaciones, el cual puede expresar una política de seguridad compleja y variada. Las organizaciones modernas y sus sistemas manejan una gran multiplicidad de roles y funciones que tienen que verse expresados en una u otra manera en el sistema.

2. El segundo nivel son aplicaciones como bases de datos, sistemas de gestión y otros, que pueden reforzar de una u otra manera la seguridad que se establece a nivel de aplicación. Por ejemplo una aplicación de base de datos atiende al problema de integridad de los datos y una de contabilidad garantiza que un debito coincida con un cargo a otra cuenta en el sistema.

3. Un tercer nivel es el control ejercido por bases de datos, sistemas de contabilidad, aplicaciones cliente-servidor sobre recursos del sistema que tienen que ser administrados en diversas formas y con arreglo a determinadas políticas de seguridad. (Acceso a puertos de servicio en servidores, conecciones TCP, etc, etc)

4. Un cuarto nivel es el control ejercido por el sistema operativo propiamente dicho el cual descansa en características del hardware que le permiten controlar el acceso a diversos recursos del sistema tales como archivos, impresoras, memoria principal, tiempo de ejecución, etc.

En la medida en que ascendemos en esta jerarquía, desde los controles del sistema operativo hacia los controles establecidos a través de las aplicaciones la complejidad e interacción de diversos componentes del sistema la hacen menos seguro y cada vez menos confiable.

En este capítulo nos concentraremos en los controles provistos directamente por el sistema operativo y que son implementados a través de listas de acceso, grupos y roles. Un sistema operativo usa tipicamente algún mecanismo de identificación (como el uso de passwords, kerberos, tokens físicos como smartcards) para identificar aun determinado usuario y luego vincularlo a los permisos que tiene en el sistema. Existen múltiples problemas relacionados al manejo adecuado y seguro de ese proceso de autentificación que no serán tratados en este punto pero que son vitales para la seguridad general del sistema.

3.5.2. Grupos y roles

Cuando analizamos como implementar mecanismos de control de acceso existen múltiples alternativas. Una manera directa y obvia es a través del uso de matrices de control de acceso, a través de las cuales podemos definir los permisos que un determinado usuario tiene sobre diversos recursos del sistema.

Aun cuando parece una solución relativamente sencilla no escala bien. En un sistema con miles de usuario el tamaño de la matriz puede resultar excesivo, no solo para el sistema, sino para el administrador para quien la taréa de asignar o denegar permisos se puede hacer extremadamente compleja. Y de nuevo, en la medida en que la complejidad crece, los puntos frágiles del sistema también aumentan.

Debido a esta dificultad surge la idéa de manejar esto a través de grupos y roles. En una organización a pesar de que puede haber miles de personas trabajando, es muy factible que el número de roles sea efectivamente limitado. Algunas personas se encargan de atender quejas al cliente por ejemplo, otras de administar recursos del sistema, otras de auditar, otras tienen la responsabilidad de manejar despachos, etc, etc. Si el staff que realiza alguna de estas funciones está constituido por 20 personas, es probable

que en lugar de tener que crear 20 perfiles diferentes podamos simplificar el problema creando tan solo 1 rol o grupo vinculado a esa tarea.

Existe una sutil diferencia en los términos grupo y rol. Un grupo es una lista de "principales". Por ejemplo, el personal de atención al público puede constituir un grupo. Un rol es un conjunto fijo de permisos de acceso que uno o mas principales pueden asumir en un momento determinaod y por un período de tiempo.

Hasta hace poco no existía mucho soporte al uso de grupos y roles. Normalmente los programadores usaban middleware para poder implementar estas estrategias de control de acceso. Recientemente Windows 2000 ha incorporado soporte para el manejo de grupos.

3.5.3. Listas de control de acceso

Otra forma de implementar controles de acceso es tener una lista de que usuarios tienen acceso a que recurso. Es decir una lista por cada recurso con todos los usuarios autorizados para usarlo. Las listas de control de acceso son ampliamente usadas en ambientes donde los usuarios manejan su propia seguridad (por ejemplo Unix). Cuando la política de control de acceso es establecida centralmente resulta de utilidad para ambientes en los cuales los mecanismos de protección son orientados a los datos. Por otra parte es menos útil cuando la población usuaria es grande y cambia constantemente, o donde los usuarios quieren tener la posibilidad de delegar su autoridad de ejecutar un programa a otro usuario para que lo pueda hacer durante un período limitado de tiempo.

Las listas de control de acceso son fáciles de implementar pero no son eficientes a la hora de chequear los permisos de acceso a tiempo real. Si un usuario va a acceder a un recurso el sistema operativo debe buscar en toda la lista para identificar si el usuario tiene permiso y esto debe ser hecho cada vez que se intenta accesar algún recurso. También la distribución de las reglas de acceso en la forma de listas de control de acceso puede hacer tedioso el identificar a cuales recursos tiene acceso un determinado usuario.

3.5.4. Capacidades

Otra forma de manejar el acceso de los usuarios a recursos del sistema son las capacidades. Las capacidades son listas que indican para cada usuario que permisos de acceso tiene para cada recurso del sistema. Es el simétrico de las listas de control de acceso con lo que se simplifica enormemente lo que en el caso de las Listas de control de acceso podía ser considerado engorroso. Ahora el usuario puede delegar facilmente sus capacidades a otro usuario y el chequeo del sistema es bastante mas sencillo. Quitar las capacidades de un usuario es bastante sencillo. Pero resulta mucho mas difícil saber que usuarios tienen acceso a un determinado recurso con lo que se complican las actividades de auditoría del sistema.

Las capacidades han comenzado a usarse recientemente en combinación con los certificados. Un certificado es un documento firmado por una autoridad certificadora (un tercero en que el sistema confía) autentificando la identidad de una persona. Esto resulta de gran utilidad en sistemas distribuidos, donde un determinado usuario debe

demostrar ante diversos sistemas su identidad, y esta identidad debe vincularse a una lista de capacidades en cada caso. Esta es la forma en que implementa el proyecto Globus sus permisos de acceso otorgados a los usuarios.

3.5.5. Granularidad y sandboxes

La granularidad es un tema directamente vinculado con los controles de acceso. Los sistemas operativos en general manejan los controles de acceso a nivel de archivos, con lo que se dificulta manejar accesos a niveles de granularidad mas finos, por ejemplo, al nivel de registro en bases de datos. Esto obliga a que hayan dos niveles de seguridad, uno manejado por el sistema operativo y otro implementado por la base de datos o por la aplicación.

Esto generalmente obliga a la existencia de múltiples puntos de autentificación, a la necesidad de introducir varios login y passwords para cada usuario complicando asi la administración de los sistemas.

Otro método para controlar el acceso a recursos del sistema es el uso de sandboxes (“cajas de arena”). Este es el método usado por Java. Cuando un usuario accede a una aplicación Java esta es ejecutada en un entorno virtual con un estricto control sobre operaciones consideradas riesgosas, como escribir a un archivo, leer de un archivo o establecer una conexión en la red.

3.6. Seguridad en Sistemas Distribuidos

Una parte muy importante del desarrollo informático reciente se encuentra en el área de los sistemas distribuidos. Los sistemas distribuidos son sistemas complejos que enfrentan múltiples problemas y riesgos de seguridad. En general, su modo de funcionamiento depende de un compromiso entre costo/efectividad y seguridad. Un ejemplo práctico es el modo de funcionamiento del sistema de tarjetas de crédito. Cuando un usuario usa su tarjeta de crédito en un local comercial esta accediendo a un sistema distribuido con presencia a nivel mundial. La tarjetas de crédito tienen reglas bien definidas de como manejar las transacciones clasificándolas en función de su monto, de esta forma una persona puede hacer pequeñas compras y el mecanismo de chequeo será solo local, si la compra pasa un determinado monto el chequeo entonces debe hacerse a un sistema centralizado para corroborar que exista el crédito necesario y que el dueño de la tarjeta no la ha reportado extraviada. En este sistema aparecen los problemas típicos en sistemas distribuidos, los cuales pasaremos a revisar en detalle a continuación.

3.6.1. Concurrencia

Decimos que dos o mas procesos son concurrentes si se ejecutan simultaneamente. Cuando dos o mas procesos se ejecutan concurrentemente pueden presentarse los siguientes problemas:

1. Los procesos pueden estar haciendo uso de datos desactualizados.
2. Los procesos pueden realizar actualizaciones inconsistentes.

3. El sistema puede bloquearse (deadlock).

4. Los datos en diferentes sistemas ejecutándose concurrentemente puede que nunca converjan a un valor consistente o con sentido.

5. Puede ser difícil el problema de sincronización, es decir, el acuerdo entre distintos procesos en relación a una hora común.

Datos desactualizados

El uso de datos desactualizados puede conducir a varios problemas de seguridad. Por ejemplo, cuando se trata de la autentificación entre dos sistemas que usan un certificado, un atacante malicioso puede ustar usando un certificado viejo para acceder a un sistema. También se presentan condiciones de borde cuando una operación está compuesta de varias fases, unas de ellas para determinar permisos de acceso y otras para efectivamente realizar el acceso. En el medio el usuario puede arreglar las cosas para tener acceso a determinados recursos para los que no está autorizado.

Estos dos son ejemplos de problemas del tipo “tiempo de chequeo-a-tiempo de uso”. La solución general tiene que ver con los tiempos de propagación de los datos a todo el sistema. Supongamos que un certificado es revocado. Un certificado viejo puede ser usado en un determinado sistema hasta tanto llegue la notificación de revocación. Si la notificación fuese inmediata no habría ningún problema pero esto generalmente está vinculado al costo total del sistema. Por esta razón las tarjetas de crédito imponen un límite al monto que puede ser autorizado localmente ya que sería antieconómico actualizar instantaneamente (cosa de hecho imposible lo cual siempre deja abierta la posibilidad de un ataque al sistema) todo el sistema distribuido de autorizaciones.

Por otro lado siempre hay una desición de carácter económico en el sistema. Si el total de fraudes que se realizan a nivel mundial en el sistema de tarjetas de crédito monta a N dólares, una compañía estará dispuesta a gastar solo hasta N dólares en evitarlo, por encima de ese valor es preferible que deje que ocurran y devolver el dinero a los agraviados.

Actualizaciones inconsistentes

Cuando varios procesos realizan actualizaciones sobre un conjunto común de datos la forma normal de atender y controlar el problema de incosistencia es a través del bloqueo. El sistema debe garantizar que solo un proceso a la vez puede actualizar la data. Sin embargo, no siempre es posible hacer esto. Cuando esto no es posible el sistema debe tener adicionalmente un mecanismo de callback, a través del cual el sistema puede avisar a los diversos procesos que la actualización no se realizó para evitar inconsistencias de tal forma que los procesos puedan actualizar la situación real de los datos.

Para la consistencia de los datos también es relevante el orden en que se realizan las actualizaciones. Un sistema bancario de créditos y cargos funciona diferente si primero registra débitos y luego créditos (lo cual seguramente conduciría a un mayor número de cheques rechazados) que si las actualizaciones se realizan en orden contrario. Por

norma general los bancos actualizan cada noche primero todos los crédito y luego todos los débitos.

Por otro lado cuando varios sistemas interactuan es posible que no atiendan con la misma relevancia diversos tipos de datos lo cual puede conducir a inconsistencias.

Deadlocks

Este problema surge cuando varios procesos se bloquean mutuamente debido a que requieren recursos que otros tienen ocupados para poder continuar. El ejemplo clásico de este tipo de problemas es la cena de los filósofos, su solución fué analizada por Dijkstra. El problema de bloqueo puede hacerse terriblemente complejo en sistemas con diversos niveles de bloque de recursos y con altos grados de complejidad.

Estados no convergentes

Cuando se diseñan protocolos que actualizan el estado de sistemas distribuidos se aspira que las operaciones sean atómicas, consistentes, aisladas y durables. Estás condiciones hacen que sea mas fácil recuperar el estado inicial del sistema despues de una falla. Las transacciones son atómicas si el sistema puede garantizar que se realizan en una sola operación. Es decir que se ejecutan como si fuera una sola instrucción, o no se ejecutan. Son consistentes cuando respetan alguna determinada regla predefinida (como por ejemplo débitos y créditos en un sistema de contabilidad), son aisladas si pueden ser vistas en forma independiente una de otras y son durables si una vez realizadas no pueden invertirse. La existencia en unos sistema de todas estas condiciones lo hacen mas resistentes a diversos tipos de ataques. Por ejemplo, el ataque descrito en una de las secciones anteriores que aprovecha la ejecución en dos pasos de la instrucción mkdir ocurre por la falta de atomicidad en la misma. Si el orden o la forma en que se realizan determinadas operaciones pueden producir inconsistencias entonces esa debilidad puede ser aprovechada por un atacante malicioso que quiera explotar esa inconsistencia. Si una operación puede ser reversada sin dejar huella entonces alguién podría entrar a un sistema bancario, extraer una suma de dinero y reversar una operación para evitar dejar huellas de su fechoría.

El tiempo en sistemas distribuidos

Algunos sistemas distribuidos dependen del tiempo para implementar mecanismos de seguridad. Kerberos puede ser atacado si se puede engañar al sistema en cuanto a la hora en que se encuentra. El problema del tiempo en sistemas distribuidos puede ser manejado de diversas maneras:

1. El sistema puede instalarse con un reloj en hardware preciso que recibe la hora de algún transmisor confiable externo.

2. El sistema puede basarse en algún protocolo de sincronización, en el cual N sistemas votan para determinar la hora actual. Esto dificulta que un atacante pueda infuenciar en la hora del sistema.

3. En lugar de usar tiempo absoluto se puede usar el tiempo de Lamport que se basa en que lo único que requiere un sistema es saber cuando un evento A a ocurrido antes que un evento B. La posibilidad de hacer esto dependerá del diseño del sistema.

En una gran cantidad de aplicaciones se está usando NTP: Network Time Protocol, el cual usa un mecanismo de votación mezclado con un mecanismo de autentificación de los servidores de tiempo.

3.6.2. Tolerancia a fallas y recuperación

Una buena parte del trabajo de investigación y desarrollo que se ha hecho en seguridad está directamente relacionado con problemas de autentificación, consistencia y privacidad. Poco trabajo se ha realizado en el tema de tolerancia a fallas y recuperación. A los efectos de los aspectos que estudiaremos a continuación entenderemos por una falta un suceso en el sistema que puede conducir a un error, el cual puede conducir a una falla la cual es una desviación de la conducta especificada para el sistema. La resistencia a fallas que incorporamos a los sistemas que diseñamos e implementamos permiten tolerar las faltas, recuperarnos de errores y si es necesario también de fallas.

A pesar de que el tema de tolerancia a fallas y recuperación ha sido obviado en gran medida en la literatura y los trabajos hasta ahora desarrollados por la comunidad académica sobre seguridad, resulta tener una importancia central. Una buena parte de los ataques que se han registrado tiene que ver con la resistencia de estos a fallas y el tipo de error que pueden manejar. Muchos de los ataques de negación de servicio o de stack overflow que permiten asumir control del sistema en modo privilegiado surgen de la conducta impropia que se le puede obligar a cometer cuando se inducen errores intencionadamente.

Fallas bizantinas

Las fallas en que incurre un sistema podemos clasificarlas en dos tipos, fallas normales o fallas bizantinas. El término de falla bizantina surge del ejemplo en el cual hay N generales defendiendo bizancio. Si los turcos han comprado a t generales queremos saber hasta cuanto puede llegar el valor de t que el sistema puede soportar (sin caer en manos de los turcos), es decir sin que los t generales puedan engañar al sistema.

El análisis de este problema se lo debemos a Lamport, Shostack y Peace [**?**]en el cual se prueba que el problema tiene una solución si y solo si $N > 3t + 1$. Por otro lado si los generales pueden firmar los mensajes ninguno puede engañar al sistema. Esta es una de las razones para implementar el uso de certificados y firmas en sistemas complejos donde múltiples componentes deben interactuar en forma confiable.

Interacciones con la tolerancia a fallas

Hay varias formas para limitar la frecuencia de fallas. Las dos formas mas obvias son a través del uso de procesadores “fail-stop” y a través de redundancia. Los sistemas del primer tipo son menos tolerantes a ataques de negación del servicio, pero a la vez

son mejores desde el punto de vista de la confidencialidad. A través de redundancia podemos garantizar que los datos siempre esten disponibles con lo que resulta mas difícil un ataque de negación de servicio pero la confidencialidad puede verse comprometida con tan solo el compromiso de uno de los sistemas donde se almacena la información.

Cuando se diseñan sistemas tolerantes a fallas debe entenderse claramente cual es el modelo de riesgo al cual nos estamos enfrentando. Un sistema puede distribuir cálculos entre diversos servidores para hacer mas complejo y difícil de atacar el mecanismo de administración de claves, o puede cambiar periodicamente todas las calves tomando en cuenta el tiempo promedio en que un atacante que ha comprometido un recurso puede tener acceso a todos los recursos del sistema. También puede estar diseñado, como la Internet, para que ningún punto en particular pueda desconectar las comunicaciones.

En efecto los protocolos TCP/IP son un claro ejemplo de un sistema diseñado con un objetivo claro en mente, el que ningún ataque puntual sobre la red pudiera interrumpir en forma permanente las comunicaciones. Sin embargo nunca se tomaron en cosideración muchos otros tipos de ataque que en la práctica eran posibles. Estas fallas no son errores de diseño, al momento de diseñar un sistema resistente a fallas se debe tener muy claro el tipo de fallas ante el cual el sistema debe ser resistente.

Niveles de Redundancia

Los sistemas pueden ser diseñados para ser resistentes a errores, ataques y fallas del equipo a diversos niveles. Algunas computadoras han sido diseñadas con niveles de redundancia internos a nivel de hardware, pudiendo contener múltiples CPUś, o arreglos de discos (RAID). En un siguiente nivel está la redundancia a nivel de grupo de procesos, en el cual podemos tener varias copias de un sistema corriendo en computadores diferentes. El resultado del cómputo puede ser obtenido por consenso. Este diseño puede defender al sistema contra los ataques en que alguién toma control del sistema y modifica los resultados, en este caso tendrá que conseguir que $(N/2) + 1$ de los servidores o computadores que operan el sistema voten a favor de los resultados modificados.

Otro nivel de protección es la realización de respaldos en momentos determinado en el tiempo, de esta forma si en un momento determinado ocurre un falla o detectamos una intrusión podemos devolver el sistema a la situación anterior al ataque. En términos generales los sistemas con respaldo son resistentes a fallas gracias a un sistema general de registro de los cambios en el sistema a intervalos regulares.

También suelen utilizarse los sistemas denominados “fallback”, los cuales son aquellos a los que revierte todo el sistema cuando los sistemas principales fallan (como por ejemplo cuando no hay disponibilidad en línea para conseguir la autorización de una tarjeta de crédito). Los sistemas de fallback son un ejemplo de seguridad a nivel de la capa de aplicación.

Es importante destacar que cada uno de estos mecanismos tiene finalidades diferentes y que debe ser usados con atención al tipo de falla que pretenden prevenir.

3.6.3. Nombres

El manejo de nombres es un tópico extremadamente importante para el manejo de la seguridad en sistemas distribuidos. Como un ejemplo sencillo piense en el nombre que debe colocarse en un certificado, tomando en cuenta que eso estará vinculado a derechos y permisos otorgados a esa persona en determinados sistemas. IPV6 fué diseñado pensando en la posibilidad de que cada objeto individual existente en el universo pudiese vincularse a una dirección IP y por tanto pudiera ser identificado (aun quedaría el problema de como manejar la resolución de nombres, con lo que estamos de vuelta en el problema anterior). Needham escribió un artículo en donde analiza los principales problemas asociados al manejo de nombres en sistemas distribuidos [**?**]:

1. Los nombres cumplen la función de facilitar la compartición de recursos.
2. La información sobre nombres puede no estar toda en un solo lugar con lo que el problema de resolución de nombres y los problemas vinculados a esto son un aspecto central de los sistemas distribuidos.
3. Es malo asumir que solo se requeriran un determinado número de nombres.
4. Los nombres globales son menos útiles de lo que parecen (ejemplo IPV6, al final hay que vincularlo a un nombre local).
5. Los nombres implican compromisos, por lo tanto hay que mantener los sistemas lo suficientemente flexibles para que puedan lidiar con cambios organizacionales.
6. Los nombres pueden convertirse en medios de acceso y manejo de capacidades.
7. Las cosas son mucho mas simples si un nombre incorrecto es obvio.
8. La consistencia de nombres es difícil de conseguir y esto puede tener implicaciones en permisos, accesos, autentificaciones y consultas a bases de datos.
9. Los nombres deben ser vinculados al objeto nombrado tan tarde como sea posible.

Hay algunos otros problemas que podemos mencionar vinculados al manejo de nombres y su vinculación con la seguridad de los sistemas. En primer lugar está la diferencia entre nombre e identidad. Un nombre es una referencia simbólica a un objeto y hay identidad cuando por alguna razón podemos vincular dos referencias simbólicas a un mismo objeto. Esto tiene relevancia pues Pedro Pérez no representa un objeto único y determinar su identidad es de gran relevancia en la seguridad de los sistemas en general.

Por otro lado está el problema de las asunciones culturales. En algunas culturas la identidad de una persona es establecida a través de su nombre materno, en otras es a través de su nombre paterno. Asumir en un sistema que el nombre “primario” de una persona es su apellido paterno puede generar problemas cuando se trata de sistema que involucran a múltiples culturas. Algunos nombres también pueden tener significado solo en un determinado contexto, otros pueden cambiar a lo largo del tiempo, o

pueden haber sido diseñado para comunidades pequeñas y perder relevancia a la hora de integrarlos a comunidades mas amplias.

Capítulo 4

Seguridad Multilateral

En algunas ocasiones requerimos manejar la información y el acceso a determinados datos en forma tal que determinados usuarios tengan acceso a "compartimentos" de información. Esta situación se presenta en las agencias de inteligencia en la forma de información confidencial por teatros de operaciones. De esta forma la información asociada a agentes de seguridad y a sus operaciones en determinados paises está aislada de la información en otras regiones respondiendo a un criterio de carácter organizacional. Hay otros casos en los cuales es indispensable separar el acceso a la información de clientes de una consultora (firma de abogados, auditores, etc) de tal manera que quienes tienen acceso a la información nunca puedan inferir mas información de la permitida por la buena praxis de negocios. Finalmente algunas leyes pueden exigir el respeto a la privacidad como es el caso de los servicios de salud en algunos paises con lo que es relevante la manera en que se maneja el acceso a la información para evitar que se vea comprometida la vida privada de un paciente.

Todos estos ejemplos son casos en los cuales aplican las diversas técnicas que veremos a continuación. El objeto general de estos modelos es limitar el flujo de información horizontal. Estas técnicas son conocidas bajo el término genérico de seguridad multilateral.

Existen al menos tres modelos diferentes de como implementar controles de acceso y de flujo de información en un esquema de seguridad multilateral:

1. Compartimentación: Usada por agencias de inteligencia y seguridad.
2. La Muralla China: Que describe los mecanismos usados para prevenir conflictos de interés en la práctica profesional.
3. El modelo BMA: Desarrollado por la British Medical Association para describir los flujos de información permitidos por las reglas de ética médica.

4.1. Compartimentación

La compartimentación es usada principalmente en ambientes de inteligencia con el objeto de controlar el flujo de información entre áreas de información que es preferible

mantener separadas. Normalmente se combina con el uso de seguridad multinivel. Su funcionamiento es relativamente sencillo, la información almacenda en compartimento es etiquetada con palabras claves como por ejemplo Lobo Rojo o Enigma. Adicionalmente existen etiquetas asociadas a diversos niveles de seguridad (por ej. Confidencial, Secreto, Ultra Secreto). Si un documento es creado usando información de un compartimento etiquetado (Secreto, Lobo Rojo) y de otro etiquetado (Ultra Secreto, Enigma) esto crea automaticamente un nuevo compartimento, a nivel Ultra Secreto y con el código Lobo Rojo y Enigma. De esta forma solo alguien que tenga todos esos permisos podrá acceder al documento (con seguridad al menos quien lo creo). Esta estructura constituye lo que en matemáticas se conoce como un *lattice,* el cual basicamente para dos elementos A y B cumple la siguiente propiedad:

1. $A > B$ o
2. $B > A$ o
3. A, B son incomparables pero tienen una cota superior C y una cota inferior D tal que $A \Lambda B < C$ y $A \wedge B > D$

Si ambas etiquetas son incomparables entonces no puede haber flujo de información entre ambos compartimentos. Mientras que en el caso 1 A puede ver todo lo contenido en B y en el caso 2 B puede ver todos lo contenido en A.

4.2. La muralla china

Otro modelo de seguridad multilateral, la muralla china, recibe su nombre de la terminologia usada por algunas empresas financieras y bancos de inversión para nombrar el mecanismo a través del cual este tipo de organizaciones intenta proteger los conflictos de intereses entre sus clientes actuales y potenciales. Este tipo de problema se da frecuentemente en empresas de servicio que pueden estar atendiendo simultaneamente a clientes que compiten en el mercado. No es deseable que un consultor que ha trabajado asesorando en una determinada materia a una determinada empresa asesore a una empresa de la competencia, al menos por un tiempo determinado. Este modelo fué desarrollado por Brewer y Nash y puede ser caracterizado a través de dos simples propiedades:

1. **La propiedad de seguridad simple:** Un sujeto S tiene acceso a $c \Leftrightarrow$ para todo c^1 que S puede leer, $(y(c) \notin x(c^1)) \vee (y(c) = y(c^1))$
2. **La propiedad *:** Un sujeto S puede escribir a $c \Leftrightarrow S$ no puede leer ningún c^1 con $(x(c^1) \neq \oslash) \wedge (y(c) = x(c^1))$

4.3. El modelo BMA

Finalmente el modelo BMA fue diseñado por la asociación de medicina británica para atender a los requerimientos de privacidad de los pacientes y a las exigencias

legales respecto a esta privacidad. Simultaneamente el modelo de manejo de información debe permitir a médicos y enfermeras el poder acceder a toda la información crítica del paciente, o necesaria simplemente para poder realizar un diagnóstico. El primer intento de atender a este problema fue el de un esquema de seguridad multinivel. A primera vista es un modelo que no puede funcionar toda vez que es demasiado complejo definir que pedazo de información puede usarse para deducir aspectos de la historia médica del paciente. Por ejemplo,¿con que nivel de seguridad se almacena un recipe médico para VIH? Y si se tiene que almacenar con carácter confidencial entonces como se maneja la necesidad de información de todo el sistema. Con miras en este problema se desarrollo un modelo cuya principal innovación fue la manera en que se maneja el "Registro electrónico del Paciente" el cual fue pensado originalmente como un solo registro de información médica, con toda la información médica del paciente, desde su nacimiento hasta su muerte. La innovación consiste en redefinir el EPR como el máximo conjunto de datos relacionados a un paciente y que son accesibles por un determinado staff médico. De esta forma un paciente tendrá varios EPR, simplificándose el manejo de la seguridad en este caso. Con base a esta nueva definición la BMA diseño una política de seguridad basada en 9 principios que se describen a continuación:

1. **Control de Acceso:** Cada registro clínico identificable debe ser marcado con una lista de control de acceso que nombre las personas o grupos de personas que pueden leer y modificar los datos. El sistema debe prevenir que cualquiera que este fuera de la lista de control de acceso pueda accesar el registro en cualquier manera.

2. **Apertura de Registros:** Un médico puede abrir un registro con él y el paciente en la lista de control de acceso. Si el paciente ha sido referido debe abrir la lista de control de acceso conteniéndolo a él, al paciente y al médico que lo refirió.

3. **Control:** Uno de los médicos de la lista de control de acceso debe ser identificado como el responsable. Solamente él puede modificar la lista de control de acceso, y solo él puede agregar otro nombre en la lista de control de acceso.

4. **Consentimiento y notificación:** El médico responsable debe notificar al paciente de los nombres que aparecen en la lista de control de acceso de su registro cuando esta es creada, con cualquier adición, y cuando sea que la responsabilidad sea transferida. Se requiere también de su consentimiento, excepto en emergencias o cuando las reglas señalen una excepción.

5. **Persistencia:** Nadie puede borrar información clínica hasta que el tiempo apropiado haya transcurrido.

6. **Atribución:** Todos los accesos a los registros clínicos deben ser marcados en el registro indicando el nombre del sujeto, fecha y hora. Ademas se debe mantener un registro de auditoría de todo lo eliminado.

7. **Flujo de Información:** La información derivada de un registro A puede ser agregada a un registro B si y solo si la lista de control de acceso de B está contenida en A.

8. **Control de Agregación:** Deben tenerse medidas efectivas para prevenir la agregación de información de los pacientes. En particular los pacientes deben recibir una notificación particular si una persona que va a ser agregada en su lista de control de acceso ya tiene acceso a un número grande deregistros con información médica.

9. **Base de Computación confiable:** Los sistemas computarizados que manejan información personal de salud deben contar con subsistemas que refuerzen los principios antes descritos en una forma efectiva. Su efectividad debe estar sujeta a evaluación por diferentes expertos.

4.4. Control de Inferencia

Un tema importante y por demás muy interesante es el del control de inferencia. Lo problemas de control de inferencia están directamente vinculados a la protección de la privacidad ya que a través de la realización de consultas a una base de datos uno podría llegar a identificar información personal. La teoría de control de inferencia fue desarrollada por Dorothy Denning a finales de los 70 y comienzos de los 80 en respuesta a los problemas de privacidad impuestos por el censo.

Una fórmula característica es la expresión (en algún lenguaje de consulta a bases de datos) que selecciona un **conjunto de registros,** conocido como **conjunto de consulta** El conjunto de consulta mas pequeño que es obtenido a través de conjnciones lógicas de lo atributos o de su negación es conocido como **conjunto elemental** . Las estadísticas correspondientes al conjunto de consultas puede ser **estadísticas sensibles**. El objetivo del control de inferencia es prevenir el acceso a estadísticas sensibles.

Si dejamos que D sea el conjunto de estadísticas que son obtenidas, y P el conjunto de estadísticas sensibles que deben ser protegidas, entonces para garantizar la privacidad necesitamos que $D \subseteq P^1$donde P^1 es el complemento de P . Si $D = P^1$ entonces decimos que la protección es precisa.

Para lograr esto existen varias estrategias que se mencionan brevemente a continuación:

1. *Control sobre el tamaño de la consulta:* Los resultados de la consultas deben ser limitados. Si hay N registros se define un valor t tal que solo se permiten resultados de consulta cuyo tamaño este entre t y $N - t$.

2. *Trackers:* El esquema de control sobre el tamaño de la consulta provee un mecanismo simple para controlar el acceso a información individual. Desafortunadamente este mecanismo de seguridad puede ser facilmente violado con un concepto relativamente simple. Se agregan registros falso hasta conseguir un tamaño de muestra apropiado, se hace la consulta, se retiran los registros falsos y se obtiene información confidencial. Un resultado importante obtenido por dening en su investigación general sobre *trackers* es que si n es el tamaño permitido de la consulta y $n < N/4$ siempre se puede crear un tracker que acceda a registros individuales. Si $n >= N/4$ entonces no es factible. Esto nos deja con un tamaño de consulta bastante amplio.

3. *La regla de "n-respuestas, k % de dominancia":* Este método no ofrece resultados a consultas en la cuales el $k\,\%$ o mas es producido por n o menos valores.

4. *Supresión de celdas:* Otro método general de control de inferencia es la supresión de celdas. Por ejemplo la eliminación de datos ubicados claramente en los extremos. Si se pueden consultar edades por ejemplo quizás sea relativamente fácil conseguir información de alguién que tenga 106 años. Es poco probable que hayan muchos. Por tanto eliminando los extremos se logra controlar también el acceso a información privada.

5. *Control de máximo orden:* Este mecanismo se basa en limitar el número de atributos que se pueden consultar simultaneamente. Esto genera una estructura matemática del tipo reticulado con una cota máxima y una mínima. Dependiendo del número de registros se puede identificar el número de atríbutos que impiden la identificación de información personal.

6. *Control basado en auditoría:* Otro método de control frecuentemente usado el control de auditoría. Este método permite rechazar la consulta de una persona de la cual sabemos que posee suficiente información para derivar información privada con una nueva consulta.

7. *Randomization:* Son diversos métodos para generar ruido en una muestra de tal forma que manteniendo los resultados estadísticos se haga mas difícil el acceso a información individualizada.

Capítulo 5

Ataques y defensas en redes

5.1. Introducción

Este capítulo revisa los tipos de ataques mas comunes que se realizan en redes. Existe la falsa idéa de que la solución a este tipo de ataques es el encriptamiento cuando en realidad la mayoría de los ataques que han sido realizado con éxito tienen que ver con otro tipo de vulnerabilidades del sistema. Algunos ejemplos de ataques (y vulnerabilidades) son los siguientes:

1. Un ataque de overflow de la pila en el programa BIND, usado por muchos servidores DNS que usan UNIX o Linux como sistemas operativos el cual da acceso inmediato a una cuenta.

2. Programas CGI, Java o ActiveX mal diseñados, con un código poco seguro que pueden ser usados para atacar el sistema.

3. Un ataque de overflow de la pila sobre el mecanismo RPC (Remote Procedure Call) de sistemas Unix y Linux.

4. Un bug existente en el Internet Information Server de Microsoft que permite acceso inmediato a la cuenta del administardor en el servidor.

5. Un bug en sendmail que permite instruir a sendmail que envié el archivo de passwords a una determinada dirección de correo electrónico.

6. Un ataque de overflow de la pila en el sistema operativo Solaris de Sun, que le da al atacante acceso inmediato a la cuenta root.

7. Ataques sobre NFS y sobre NT.

8. Ataques al nombre de usuario y password.

9. Ataques sobre los protocolos IMAP y POP.

En general la proliferación de páginas web que implementan cookies, programas en CGI, ActiveX, Applets de Java permiten a un atacante malicioso hacer llegar un software malintencionado a la máquina del usuario y son una de las fuentes mas importantes de debilidad en sistemas de redes.

5.2. Vulnerabilidades en protocolos de redes

La vulnerabilidad inicial y mas fuerte de los protocolos de redes estan asociados a los valores por defecto que tienen los sistema operativos mas comunes como NT, Windows 2000 o Unix. Estos valores por defecto suelen habilitar servicios que pueden ser eventualmente usados para atacar el sistema. El desarrollo de los protocolos de Internet se hizo en un momento en que el acceso a la tecnologia era muy limitada y solo habian servidores confiables, y la gente con acceso era la comunidad cientifica y algunas organizaciones militares. Eso hizo que se le dedicara poco esfuerzo en ese momento a las actividades de autentificacion y seguridad. Se estan haciendo esfuerzo para implementar mejores mecanismo de seguridad en la version IPV6 de los protocolos.

5.2.1. Ataques en redes locales

Si una persona tiene acceso a una máquina conectada a una red local en la que quiere llevar a cabo una ataque basta con que instale un software de sniffer para inspeccionar los passwords y tener acceso por ejemplo a la cuenta del administrador. Para contrarestar esto el sistema puede usar un esquema de “reto-respuesta” que haga mas difícil el acceso a esta información. La otra forma es que el administrador no haga login sino en la máquina en la cual esta abierta su cuenta. Esto puede funcionar para él y para la protección de su cuenta, sin duda muy importante, pero no protege a otros usuarios de este tipo de ataques. Otro método es usar mecanismos de autentificación que encripten la data, como por ejemplo SSH, el cual puede ser usado en combinación con claves públicas y privadas para proteger este acceso.

Un ataque mas sofisticado es que el atacante se disfraze como una máquina en la cual la víctima ya ha ingresado. El protocolo ARP es una opción para este tipo de ataque. Ejecutando un programa adecuadamente diseñado el atacante puede dar respuestas erradas a mensajes ARP y proclamarse como la víctima, de esta manera puede pasar a ocupar su identidad. Si existe software que permite detectar este tipo de ataque lo único que necesita el atacante es esperar hasta que esta máquina esta apagada para poder hacerse pasar por ella.

En el caso en que la organización use el sistema operativo Unix un posible ataque se puede desarrollar contra el NFS (Network File System). Este sistema tiene múltiples vulnerabilidades reconocidas. Por ejemplo cuando un volumen es montado la primera vez, el cliente requiere del servidor un gestor de archivo raiz (root filehandle) el cual se refiere al directorio raiz del sistema recién montado. Este valor no depende del tiempo, o de la versión del servidor, y no puede ser revocado. Esto permite que un usuario pueda tomar la identidad del administrador una vez que ha ingresado al sistema que tiene NFS funcionando.

5.2.2. Ataque usando protocolos y mecanismos de Internet

Cuando se trata de los protocolos TCP/IP el problema fundamental es similar, no hay mecanismo reales de autentificación o confidencialidad en la mayoría de los mecanismos. Esto es particularmente cierto en los niveles mas bajos del protocolo.

Consideremos por ejemplo el triple “handshake” usado por Alice para iniciar una conexion TCP con Bob. Este protocolo puede ser explotado en múltiples formas. El ataque mas simple es la inundacion SYN.

SYN Flooding

El ataque de “inundación” consiste en enviar un gran número de paquetes SYN y nunca responder con el acknowledge ninguna de las respuestas que se reciban. Esto hace que el receptor acumule mas registros de paquetes SYN que los que su software puede manejar. La solución a este ataque es el SYNcookie. Con esto la máquina en lugar de mantener una copia del paquete SYN, B simplemente envía como su número de secuencia una versión encriptada del número de secuencia enviado por el atacante.

Smurfing

Este ataque explota el protocolo de control de mensajes de Internet (ICMP) el cual permite al usuario enviar un paquete a un host remoto para chequear si esta “vivo”. El problema reside en el uso de direcciones de broadcast. Algunas implementaciones de los protocolos de Internet responden a un ping a la dirección de broadcast y a la dirección local. La idea de esto es poder chequear si una red esta viva. Una colección de servidores que respondan a la direccion de broadcast de esta forma se conocen como un amplificador de smurf.

El ataque consiste en construir un paquete con una dirección de origen forjada igual a la de la víctima y enviar esta a un grupo de amplificadores de smurf. Todas las máquinas responderán enviando una respuesta a la direccion IP de la víctima. Esto hace que la víctima reciba mas paquetes de los que puede manejar.

Parte de la respuesta a este ataque es técnica, un cambio en los protocolos en Agosto de 1999 de tal manera que estos ping enviados a la dirección de broadcast ya no son respondidos. La otra es socioeconómica, se han creado lista de amplificadores de smurf de tal forma que los administradores puedan deshabilitar esa función en sus respectivas redes.

Ataque de bloqueo del servicio (distribuido)

El ataque de bloqueo distribuido consiste en el desarrollo de una ataque simultáneo (de paquetes tipo ping) pero que se lleva a cabo desde múltiples maquinas simultaneamente, luego que el atacante ha logrado tomar contro de un numero suficiente de máquinas. Para controlarlo se ha desarrollado una innovación en los protocolos ICMP para que cada vez que se envía un paquete, con una probabilidad de 1/20000 a la dirección de destino con información del paquete que se envió previamente. Este paquete ICMP contiene información del salto anterior, del siguiente y tanta información sobre

el paquete como sea posible. En teoría esto permitiría identificar la máquina desde la cual se esta haciendo el ataque.

Spam y forjamiento de direcciones

Servicios como el e-mail y el Web (SMTP y HTTP) asumen que los niveles mas bajos son seguros. Lo que normalmente se hace es chequear el nombre del host contra una dirección IP usando el DNS. Por lo tanto alguién que pueda forjar una dirección IP puede realizar un ataque. El ejemplo mas común es el forjamiento de email por spammers. Por ejemplo si un atacante puede dar al DNS información incorrecta sobre la ubicación de una pagina web, por ejemplo embasurando el cache del DNS con direcciones falsas entonces las conexiones pueden ser hechas a una pagina web fraudulenta.

Ataques de spoofing

Podemos combinar algunos de los tipos de ataque antes descritos en ataques de spoofing que funcionan en amplios rango. Supongamos por ejemplo que Charlie sabe que Alice y Bob son servidores en una red de área local y quiere disfrazarse como Alice ante Bob. Puede lanzar un ataque de bloqueo de servicio sobre Alice y luego iniciar una nueva conexión con Bob. Esto implica adivinar el numero de secuencia que Bob establece para la sesión. Una forma simple de adivinar este numero de secuencia es establecer una conexión con Alice un poco antes de realizar el ataque debido a que el valor puede cambiar en una forma predecible. Actualmente el software usa una forma aleatoria de generar los números de secuencia para evitar que Charlie pueda adivinar facilmente el número siguiente que se usará.

Si es posible adivinar el número de secuencia entonces Charlie puede ser capaz de enviar mensajes a Bob, haciéndole creer que vienen de Alice.

Ataques de enrutamiento

El ataque básico consiste en que Charlie le dice a Bob y Alice que hay una mejor ruta, que pasa a través de él, para alcanzar una determinado destino. Este protocolo fue establecido inicialmente para recuperarse de caidas de determinados enrutadores. Sin embargo en ese momento se asumía que los hosts eran honestos. La única manera de resolver esto es bloquear la aplicación de este tipo de protocolo en los enrutadores que constituyen la red.

Otro tipo de ataque implica el redireccionamiento de los mensajes, diciendo por ejemplo “debiste haber enviado el mensaje a esta dirección” y en esencia funciona de la misma forma que el ataque de ruta de origen.

5.3. Defensas contra ataques en la red

Normalmente los ataques de red antes descritos pueden ser contrarestados por una administrador eficiente que revise permanentemente los boletines de actualizacion sobre correciones en seguridad.

5.3.1. Gestión de configuración

El aspecto mas crítico en el manejo de la seguridad de un sistema es un manejo cuidadoso de la configuración. Si uno puede asegurarse que todas las máquinas en una determinada organización están corriendo versiones actualizadas del sistema operativo y que se aplican las correcciones en el sistema que son requeridas por el fabricante, que los archivos de servicio y de configuración no tienen ninguna falla importante, que los passwords por defecto de algunos productos son removidos y que todos los contenidos son respaldados con suficiente cuidado y diligencia entonces uno puede estar seguro que va a estar protegido contra la mayoria de los ataques. Uno tiene que tener cuidado ademas con los scripts de CGI, ActiveX, Cookkies y otros programas que puedan funcionar en el sistema como una consecuencia del funcionamiento del sistema.

La gestión de configuración es tan importante como el tener en funcionamiento un buen firewall, en efecto, si uno tiene que escoger entre ambas opciones, el firewall es menos importante. Hay algunas herramientas disponibles a los administradores de sistemas para mantener las cosas bajo control. Algunas permiten un manejo centralizado para el control de versiones. Algunos otros como Satán intentan acceder al sistema usando algunas vulnerabilidades conocidas permitiendo así evaluar la seguridad del sistema.

5.3.2. Cortafuegos

La solución mas popular a los problemas de la seguridad en Internet es el uso de cortafuegos. Este es un dispositivo que se coloca entre la red local y el Internet y que filtra el tráfico que puede ser dañino. Los firewall vienen básicamente como uno de los siguientes tres tipos:

Cortafuegos a nivel de paquetes

El tipo mas simple de firewall filtra paquetes basado en direcciones y en números de puertos de servicio. Esta funcionalidad está disponible en enrutadores y en algunos sistemas operativos. Este tipo de protección puede bloquear algunos de los ataques descritos anteriormente, evitando que ningún paquete que parezca tener la dirección de una máquina local ingrese desde afuera.

Cortafuegos a nivel de circuitos

Otro tipo de cortafuegos mas complejo actúa a nivel de circuito, re-ensamblando y examinando todos los paquetes en cada circuito TCP. Esto es más costoso en términos de procesamiento que el filtrado de paquetes y puede proveer servicios adicionales como el de redes privadas virtuales sobre la Internet usando encriptamiento de cortafuegos a cortafuegos. Sin embargo este tipo de protección no protege de ataques al nivel de aplicación.

Cortafuegos a nivel de aplicación

El tercer tipo de firewall funciona a nivel de aplicación, actuando como un proxy para uno o mas servicios. A este nivel se pueden establecer reglas como la extracción de los archivos que se anexan a los e-mails, o macros de word o removiendo contenido activo. Como contraparte este tipo de control puede convertirse en un cuello de botella para las aplicaciones en la organización.

DMZ: Zona Demilitarizada

Los cortafuegos pueden usarse para definir una zona demilitarizada. Se puede construir con un cortafuegos a nivel de paquetes en el lado externo de la zona que permita el ingreso de determinados paquetes filtrando otros y que permita atender requerimientos de información sobre proxies colocados en esta área. Luego puede haber proxies internos mas severos que mantengan una barrera mucho mas alta y estricta a los paquetes que ingresen desde el exterior.

5.4. Troyanos, virus y gusanos

Troyanos virus y gusanos son conocidos por el nombre genérico de *malware* La diferencia es sútil, un *caballo de troya* es un programa que hace algo malicioso (como capturar el password) cuando es ejecutado por un usuario desprevenido, un *gusano* es un algo que se replica y un *virus* es un gusano que se replica copiándose el mismo a otros programas.

*Los **Virus** y **gusanos*** tienen normalmente dos componentes, un mecanismo de replicación y un *payload* . Un gusano hace una copia de si mismo en otra parte cuando es ejecutado por ejemplo enviándose el mismo por e-mail como un anexo a todas las direcciones en la lista de direcciones del sistema infectado. Historicamente los virus se “pegan” a programas del usuario y modifican el hilo de ejecución para que el programa comience a ejecutarse y les pase el control. Estos son los mecanismos de replicación y dependen de características particulares del sistema operativo para el que fueron diseñados. El otro componente es el *payload* que se activa en base a algún evento particular, como una fecha, una acción del usuario, y luego de activado puede hacer una o varias cosas dañinas, como por ejemplo:

1. Hacer cambios aleatorios o selectivos a los estados de protección o seguridad del sistema.
2. Hacer cambios aleatorios o selectivos a datos del usuario.
3. Bloquear la red (por medio de un ataque distribuido de negación del servicio).
4. Robar recursos (de Cpu o memoria).
5. Usar el modem para hacer una llamada costosa al otro lado del mundo.
6. Robar los datos, incluyendo las claves criptográficas.

7. Crear una puerta trasera en el sistema para entrar cada vez que lo desea.

Para contrarestar a los virus y gusanos han surgido los antivirus coon diversas estrategias. estas estrategias siempre han sido respondidas con contraestrategias por parte de los fabricantes de virus. Al comienzo los antivirus *inmunizaban* los archivos. El método usaba el hecho de que los virus cuentan con un mecanismo para identificar cuando ya han infectado un archivo. El antivirus agregaba entonces pedazos del virus para engañar al antivirus. Esto fué útil cuando la población de virus no era muy alta. La segunda generación de antivirus son los *scanners* que identifican código del virus y lo remueven. Los virus mas avanzados usan encriptamiento para no ser reconocidos y cada vez que infectan un archivo encriptan parte de la información bajo una clave diferente haciendo mas difícil el trabajo de detección del antivirus.

Otro método para contrarrestar a los virus es el uso de un método conocido como *checksummer* . Con este método el sistema calcula un checksum con las longitudes de los archivos ejecutables en el sistema, cada vez que se realiza un chequeo de virus se computa el checksum y se compara con el valor almacenado. Por otro lado el virus puede ser lo suficientemente habilidoso para coultarse en el momento del checksum y luego reaparecer.

La fuente principal de virus hoy día esta relacionada con la aparición y diseminación de lenguajes interpretados, como Java o los Macros de Word. Por otro lado para que un virus se convierta en una epidemia debe pasar un determinado umbral, a partir del cual la velocidad a la que se reproduce es mayor que la velocidad a la que es removido. En el análisis y atención a este tipo de problemas puede ser de interés el uso de modelos epidemiológicos.

5.5. Detección de intrusos

El último aspecto que trataremos en este tema de *Ataques y defensas en redes* es el de la detección de intrusos. Un virus es un ejemplo particular de un intruso, pero en general puede ser útil tener mecanismos para detectar cuando se produce una actividad inusual en el sistema y producir una alarma con el objeto de evitar potenciales ataques. Esto puede atender a la necesidad de detectar clonado de teléfonos celulares, fisgoneo, o usuario que han ingresado a un determinado sistema a través de cualquiera de los tipos de ataques hasta ahora estudiados. El método de detección de intrusos mas simple se basa en disparar una señal de alarma tan pronto como un determinado umbral de un valor específico es alcanzado. Por ejemplo 3 o mas intentos de login, gastos en una tarjeta de crédito por mas de dos veces el gasto promedio en los últimos meses, el uso de un teléfono celular desde una localidad en la que nunca antes se había utilizado, llamadas con tarjetas prepagadas desde paises inusuales para ese cliente en particular. Los sistemas de detección de intrusos en general caen en una de las siguientes dos categorias, *sistemas de detección de mala utilización o mal uso,* y *sistemas de detección de anomalías*. Un ejemplo del primero puede ser una alarma que se dispara cuando un usuario de un sistema UNIX trata de acceder al archivo de logins, el segundo intenta preveer ataques que no han sido vistos previamente y su aproximación es a través del uso de inteligencia artificial.

Algunos tipos de intrusión son verdaderamente obvios, sin embargo en general la detección de intrusos en sistemas complejos es una tarea verdaderamente difícil. La limitación viene de la existencia de dos tipos de fallas de seguridad en este caso, aquellas que producen un error o un mal funcionamiento del sistema (con lo que eventualmente puede ser detectado) y otros que no. Es por tanto bueno diseñar sistemas tratando que las intrusiones en la mayoría de los casos generen un error. Sin embargo se debe ser cuidadoso con la posibilidad de generar demasiadas falsas alrmas convirtiendo al sistema en algo inusable.

En el caso del problema particular de detección de intrusos en ataques en redes, este es mucho mas difícil que la detección de ataques por ejemplo de clonación de celulares, las razones para esto son las siguientes:

1. La Internet es un ambiente con mucho ruido, no solo a nivel de aplicaciones y contenidos sino a nivel de paquetes.

2. Hay muy pocos ataques en comparación con conexiones que no son ataques de ningún tipo, esto hace difícil establecer un umbral razonable para el mecanismo de detección.

3. Muchos de los ataques en redes son específicos a versiones particulares de software.

Capítulo 6

Protección de sistemas de Comercio Electrónico

La aparición del World Wide Web y el desarrollo acelerado de Internet ha abierto un nuevo espacio de comercio que permite a millones de usuarios del web comprar en línea usando tarjetas de crédito. Una de las limitantes para el desarrollo del comercio electrónico es la seguridad. Los usuarios, bancos y comerciantes quieren estar seguros que pueden proveer servicios de comercio electrónico sin posibilidades de fraude por parte de ninguno de los participantes en el sistema o por parte de agentes externos. Mucha gente piensa que la panacea para atender este tipo problemas es la criptografía, cuando en realidad esta solo protege ciertos tipos de ataque y esos ataques no son los mas comunes en la red.

Podemos decir que una computadora es segura si uno puede confiar en que se comportará como uno espera, tanto a nivel de hardware como de software. La seguridad en el web tiene tres facetas principales:

1. Seguridad del servidor web y de los datos almacenados en él.
2. Seguridad de la información que viaja entre el servidor web y el usuario.
3. Seguridad de la computadora del usuario final y de otros dispositivos que la gente usa para accesar la Internet.

Atender a estos requerimientos de seguridad puede significar una gran cantidad de actividades entre las cuales se puede mencionar a título de ejemplo:

1. Implementación de sistemas para garantizar la identidad de los usuarios que se conectan al servicio, esto puede implicar el uso de login y password y el desarrollo de políticas y sistemas para distribuir los password y para sustituirlos en forma segura si se encuentran comprometidos. Tambien pueden requerirse mecanismo de autentificación mas sofisticados como el uso de certificados y ssh para tener acceso a un servicio.

2. Análisis de programas y scripts que operan en el web site con la idea de buscar errores o debilidades que puedan ser explotados para tener acceso al sistema.

3. Implementación de un sistema eficiente y seguro de respaldo.

4. Creación de un mecanismo seguro de login e intercambio de datos.

5. Balanceo de la carga entre múltiples servidores para garantizar un servicio confiable y eficiente.

6. Proveer de un segundo centro de datos que pueda funcionar en caso de desastre (natural o intencionado).

7. Proveer seguridad al DNS de tal manera que un usuario malicioso no pueda cambiar los datos para apuntar un dominio a otra dirección IP.

8. Protección de los registros de transacciones, tanto por privacidad de los usuarios como por requerimiento de auditabilidad.

9. Provisión de seguridad física al site y los servidores.

6.1. Introducción al comercio electrónico

El comercio electrónico surgió mucho antes que el boom de Internet. Se remonta a la época de la aparición del telégrafo, alrededor de 1760. Los bancos fueron los primeros en usar el telégrafo para hacer transacciones comerciales y comenzaron a usar determinadas técnicas para evitar que algún operario intermedio pudiera cometer un fraude. Muchos de los problemas típicos del comercio electrónico ya eran conocidos para estos comerciantes Victorianos.

En los 60's los bancos en muchos países informatizaron sus sistemas de contabilidad e introdujeron sistemas interbancarios nacionales para poder manejar directamente los pagos a los clientes. Esto le permitió a los bancos habilitar nuevos servicios. La siguiente gran expansión del comercio electrónico ocurrió entre el final de los 70's y mediados de los 80's con la expansión de los sistemas EDI (Electronic Data Interchange). Compañías como General Motors construyeron sistemas que le permitían ligar sus sistemas a los de sus clientes. Agentes de viajes crearon sistemas similares para poder manejar reservaciones y luego pagos en linea.

En 1985 apareció el primer servicio al detal de banca electrónica, ofrecido por el Banco de Escocia, cuyos clientes debían usar Prestel, un sistema de email propietario operado por British Telecom para hacer pagos. Finales de los 80's y los 90's fueron testigos de un rapidísimo crecimiento de los call centers y el comienzo del uso del Internet como medio para realizar transacciones electrónicas. A pesar del inmenso interés en los problemas de seguridad de Internet aun los call center representan un parte importante del comercio electrónico y aun Internet no es el medio a través del cual se realiza el grueso de las transacciones electrónicas.

6.2. Tarjetas de crédito

Por muchos años después de su invención en 1950s las tarjetas de crédito fueron tratadas por la mayoría de los bancos como un mecanismo de atraer clientes de alto valor. En un momento determinado en muchos países el numero de comerciantes y clientes alcanzo un nivel de masa crítica y el volumen de transacciones comenzó a crecer exponencialmente. En Gran Bretaña por ejemplo tomo casi 20 años que el negocio se hiciera rentable, de pronto se hizo extremadamente rentable. Hoy día las tarjetas de crédito se han convertido en el mecanismo formal de pago en la red.

Cuando usted usa una tarjeta de crédito para pagar en una tienda la transacción fluye desde el comerciante a su banco, el cual paga una vez que descuenta una tasa por servicio. Si la tarjeta es de otro banco la transacción fluye a otro centro administrado por la marca de la tarjeta (por ejemplo Master) quien toma una comisión y pasa la transacción al banco que expidió la tarjeta.

6.2.1. Fraude

El riesgo de fraude sobre las tarjetas de crédito ha sido tradicionalmente manejado a través de listas negras y a través de limites de autorización para el comerciante. Cada comerciante recibe una lista de tarjetas locales que están en mora (anteriormente lo recibía en papel, ahora esta información es accedida en linea) adicionalmente a un limite definido por el banco al cual debe llamar cada vez que requiere una autorización. El servicio en linea tiene acceso a una lista negra a nivel nacional y por encima de un determinado nivel debe ser autorizado por la casa matriz de la tarjeta a nivel internacional.

La aparición del comercio electrónico y la posibilidad de pedir por teléfono determinados bienes significa que el comerciante ya no tiene enfrente suyo a la persona que realiza la transacción, con lo que no puede corroborar si esta persona realmente tiene esa tarjeta. Por otro lado la aparición de la internet ha empeorado notablemente esta situación. El usuario puede estar en cualquier parte del mundo, lo que significa que para transacciones muy pequeñas no hay forma de determinar si la tarjeta se encuentra en la lista negra a nivel local o no.

La forma en que los bancos han manejado esto es usando la fecha de expiración como un password, bajando aun mas los limites para compras, aumentando lo que descuentan al comerciante y a través de un proceso mas cuidadoso de envío de las renovaciones.

Si una persona reniega de una transacción hecha en linea o por teléfono la cantidad completa es debitada al comerciante además de un costo por manejo. Por supuesto que el hecho de tener al tarjeta habiente en persona no garantiza nada con respecto a la seguridad de la transacción.

6.2.2. Falsificación

A comienzos de 1980 fueron introducidos terminales electrónicos a través de los cuales un vendedor podía deslizar una tarjeta de crédito y obtener una autorización automáticamente. El primer método de falsificación se basaba en recuperar los recibos

de compra para luego recodificar la banda magnética de una tarjeta robada con el numero de cuenta y la fecha de expiración de una tarjeta valida. Una tarjeta recodificada funcionaba perfectamente. Cuando el comerciante enviaba el recibo para el pago era rechazado por el banco ya que no coincidían los números de la cuenta con la información de la tarjeta.

Para evitar esto los bancos modificaron los terminales que capturaban las transacciones para que imprimieran la información de la banda magnética. A partir de allí los fraudes fueron realizados con tarjetas completamente forjadas. Para resolver esto, VISA introdujo unos valores de verificación de tarjeta (CVV por sus siglas en ingles) que básicamente era un MAC (Message Authentification Code) de tres dígitos calculado sobre la data de la cinta magnética (numero de cuenta, numero de versión y fecha de expiración) y se añade al final de la cinta.

A partir de acá el método de fraude a consistido en mecanismos (sistemas) para grabar la información de la banda magnética para luego reproducirla.

6.3. Fraude en linea

Esta historia de fraudes genero una gran preocupación en la banca y los comercios una vez que se inicio el uso de la Internet como mecanismo a través del cual los usuarios de tarjetas de crédito podían realizar compras. Sin embargo los fraudes que se han producido en la red, a pesar de ser posibles, no son tan frecuentes como uno podría suponer. Existen numerosos tipos de ataque que pueden ser realizados sobre la pagina web de un proveedor de servicio, desde buscar en su base de datos de clientes hasta atacar el DNS para sustituir un servicio real por uno fraudulento. Sin embargo estos ataques requieren de una cierta sofisticación técnica y es poco probable que sean los responsables de la mayor parte de los fraudes que ocurren con el uso de tarjetas de crédito.

Estos temores llevaron al desarrollo y uso de mecanismos de encriptamiento y autenticación sofisticados que han sido usados para mejorar la confiabilidad de estos sistemas. SSL y SET han sido sistemas desarrollados con ese fin.

6.4. Mecanismos de protección criptográficos

Aun cuando muchos de los problemas de seguridad asociados al comercio electrónico no están vinculados al uso de la criptografía es indudable que su uso es una necesidad para garantizar un cierto grado de seguridad en las transacciones. Los requerimientos básicos de un sistema de comercio electrónico son autenticidad, certificación y confidencialidad. Un comprador en Internet querría:

1. Tener un mecanismo para autentificar el sitio web que esta visitando y así estar seguro que no es una pagina web fraudulenta que fue colocada para estafarle.

2. Tener un mecanismo seguro para autenticar su identidad como usuario del sistema.

3. Poder enviar datos sobre la red tales como login y password y posteriormente datos de su tarjeta de crédito y pagos en forma confidencial.

Sin embargo no esta de mas insistir en que hay otros problemas mas relevantes cuando hablamos de la seguridad de un sistema. Un usuario también estará preocupado por su privacidad y por los datos que quedan registrados en la máquina del comerciante una vez que una transacción ha sido realizada.

6.4.1. SSL

SSL es el acrónimo de Secure Socket Layer, un protocolo de propósito general para enviar información encriptada sobre la Internet. SSL fue popularizado inicialmente por Netscape y la idea era estimular las ventas de las compañías que usaran el web server de Netscape. En 1996 el Internet fue creado el grupo de trabajo del Engineering Task Force (IETF) para establecer un estándar abierto de encriptamiento. El grupo comenzó con SSL 3.0 y en 1999 publico el RFC 2246 “TLS Protocol Version 1.0”, el RFC 2712 añade autenticación usando Kerberos y el RFC 2817 y 2818 aplica a TLS usando HTTP/1.1.

SSL es una capa que existe entre el protocolo TCP/IP y la capa de aplicación. Mientras que la comunicación de TCP/IP es en “claro” SSL añade numerosas características a esa aplicación, incluyendo:

1. Autenticación del servidor usando firmas digitales.
2. Autenticación del cliente usando firmas digitales.
3. Confidencialidad de los datos a través del uso de encriptamiento.
4. Integridad de los datos a través del uso de códigos de autenticación de mensajes.

La criptografía avanza rápidamente y por eso SSL implementa un mecanismo flexible que le permite a las partes acordar el protocolo criptográfico que ambos puedan usar. A través de un protocolo conocido como el SSL Hello cuando dos partes quieren comunicarse escogen el protocolo criptográfico mas fuerte que tienen en común. Cuando SSL fue creado aun existían muchas restricciones del gobierno norteamericano a la exportación de tecnología criptográfica por lo cual SSL incorporo mecanismos para poder adaptarse a un mercado mundial con diferencias en cuanto a las restricciones de uso de la tecnología. La versión en uso de SSL es la 3.0 y esta presente en Netscape Navigator, Microsoft Windows y la librería de código libre OpenSSL.

Las características principales de SSL/TLS son las siguientes:

1. SSL/TLS usa algoritmos separados para encriptamiento autenticación e integridad de los datos: La ventaja principal de esto es que se pueden usar longitudes de claves diferentes para autenticar y encriptar.
2. Eficiencia: El encriptamiento y desencriptamiento de clave publica es un procedimiento que consume tiempo. En lugar de repetir el proceso para cada comunicación entre un cliente y un servidor SSL puede almacenar en cache una

clave maestra que es preservada entre conexiones. Esto le permite a las nuevas conexiones comenzar una conexión segura muy rápidamente.

3. Autenticación basada en certificados: SSL y TLS provee autenticación para el servidor y el cliente a través del uso de certificados digitales y de "retos" firmados usando los certificados. SSLv3 y TLS usan certificados X.509v3.

4. Independiente del protocolo: Aun cuando fue diseñado para funcionar sobre TCP/IP en efecto puede funcionar con cualquier otro protocolo de red orientado a conexión.

5. Provee protección contra el ataque del hombre en el medio.

6. Da soporte para compresión, previa al encriptamiento.

Cual es la verdadera protección que ofrece SSL?

SSL realmente hace muy poco para proteger contra los ataques reales que los consumidores y comerciantes han experimentado en la Internet. En principio SSL no intenta resolver los principales problemas del comercio electrónico. En la siguiente lista se enumeran esos problemas en orden de importancia:

1. Ataques al cliente.

2. Ataques al comerciante o vendedor.

3. Impostura de paginas web.

4. Ataques al DNS.

5. Fisgoneo a la navegación.

6. Ataques a la autoridad certificadora.

7. Criptoanálisis de las claves.

La protección real del consumidor viene de las regulaciones E y Z y de las políticas de compañías como VISA, Master Card, American Express y otras. Si usted compra en Internet usando su tarjeta de crédito y descubre un cargo fraudulento, usted puede llamar a la compañía de tarjeta de crédito y rechazar el cargo. Las tarjetas de débito por otro lado tienen menor protección pues requiere que el banco reponga el dinero extraído.

Certificados Digitales

SSL/TLS hace un uso extensivo de certificados de clave publica para autentificar ambos, cliente y servidor en cada transacción. Usa certificados X.509 y Diffie-Hellman. Soporta los siguientes tipos de certificados:

1. Certificados de clave publica RSA, con claves publicas de longitud arbitraria.

2. Certificados de clave publica RSA limitados a 512 bits.
3. Certificados RSA solo para firma.
4. Certificados DSS.
5. Certificados Diffie-Hellman.

Como funciona realmente

En la capa mas baja del protocolo TLS se encuentra la capa de registro TLS. La capa de registro TLS envía bloques de datos, llamados registros, entre el cliente y el servidor. Cada bloque puede contener hasta 16.383 bytes de datos. Los bordes o limites del mensaje del cliente no son preservados en la capa de registro. Si el mensaje es muy largo puede ser dividido en varios registros. Cada registro TLS contiene la siguiente información:

1. Tipo de Contenido.
2. Numero de versión del protocolo.
3. Longitud.
4. Contenedor de datos (opcionalmente comprimido y encriptado).
5. Código de autenticación de Mensaje (MAC).

Cada registro es comprimido y encriptado usando el algoritmo negociado entra las partes. Al comienzo de la conexión la función de compresión es definida como CompressionMethod.null y el método de encriptamiento es TLS_NULL_WITH_NULL_NULL, lo que significa que no hay compresión ni encriptamiento. La compresión y el encriptamiento pueden ser inicializado durante el "SSL Hello".

El MAC es calculado usando la formula:

$$HMAC\_hash(MAC\_write\_secret, seq\_num + TLSCompressed.type$$

$$+TLSCompressed.version+TLSCompressed.length+TLSCompressed.fragment)$$

donde:

seq_num es el numero de secuencia del mensaje.

$HMAC\_hash()$ es el algoritmo de hashing. Puede ser usado con una variedad de algoritmos pero los mas usados son MD5.

MAC_write_secret es un secreto compartido entre el cliente SSL y el servidor SSL que es usado para validar la transmisión.

$TLSCompressed.type$ es el tipo del registro.

$TLSCompressed.version$ es el numero de versión de TLS.

$TLSCompressed.length$ es la longitud del fragmento de datos.

$TLSCompressed.fragment$ es la data en si misma.

La capa de registro provee integridad. El uso del MAC previene un ataque de reenvío a lo largo de una sesión, debido a que cada mensaje tiene un numero de secuencia único. Y la capa de registro provee también compresión lo cual es importante pues una vez que la data es encriptada ya no puede ser comprimida.

Protocolos SSL/TLS

Los protocolos SSL y TLS son tipos de mensajes específicos que son enviados usando el registro de capa. La versión 3.0 define tres protocolos:

1. El protocolo de Handshake, el cual realiza la negociación inicial de la clave.
2. El protocolo de Alerta, el cual envía mensajes sobre el estado de la conexión SSL/TLS de un lado al otro.
3. El protocolo ChangeCipherSec, el cual cambia el sistema de encriptamiento en uso actualmente.
4. El protocolo de aplicación de data, el cual envía la data del usuario.

El protocolo de handshake

El protocolo de handshake es usado para autentificar al servidor SSL ante el cliente (y opcionalmente el cliente con el servidor) y para acordar sobre el algoritmo inicial de encriptamiento y sobre las claves. El handshake establece el protocolo que sera usado durante la comunicación, selecciona los algoritmos criptográficos, autentifica la partes y usa criptografía de clave publica para crear un "secreto maestro" del cual se derivan la claves de encriptamiento y autenticación.

El secreto maestro para la sesión es creado por el servidor usando un "pre-secreto maestro" enviado por el cliente. El secreto maestro es usado para generar cuatro claves secretas adicionales:

1. Una clave de encriptamiento usada para enviar datos desde el cliente hasta el servidor.
2. Una clave de encriptamiento usada para enviar datos desde el servidor al cliente.
3. Una clave de autenticación usada para enviar datos desde el cliente hasta el servidor.
4. Una clave de autenticación usada para enviar datos desde el servidor hacia el cliente.

Secuencia de Eventos

El protocolo de handshake implica una serie de mensajes e intercambios entre cliente y servidor, en la siguiente secuencia:

1. El cliente abre una conexión y envía un $ClientHello$
2. El servidor envía un $ServerHello$
3. [El servidor envía un certificado]
4. [El servidor envía un $ServerKeyExchange$]

5. [El servidor envía un $CertificateRequest$]
6. El servidor envía un $ServerHelloDone$ (solo TLS)
7. [El cliente envía su certificado]
8. El cliente envía un $ClientKeyExchange$
9. [El cliente envía un $CertificateVerify$]
10. El cliente y el servidor envían ambos un $ChangeCipherSec$
11. El cliente y el servidor envían ambos mensajes de finalización.
12. Comienzan a intercambiar los datos de la aplicación.

Protocolo de alerta

Los alertas son tipos específicos de mensaje que pueden ser transmitidos por la capa de registro SSL/TLS. La alerta consiste de dos partes: un nivel de alerta y una descripción de la alerta. Ambos son codificados como un numero de 8 bits.

Protocolo de ChangeCipherSpec

Este protocolo es usado para cambiar de un algoritmo de encriptamiento a otro. Para cambiar el algoritmo de encriptamiento. Para cambiar el algoritmo de encriptamiento el cliente y el servidor primero negocian un nuevo CypherSpec y claves. Cada uno de ellos entonces envían un mensaje de ChangeCipherSpec lo cual causa que el proceso receptor comience a usar el nuevo protocolo.

6.4.2. SET

SET es el acrónimo de “Secure Electronic Transaction”, un protocolo que fue diseñado para pagos en línea usando tarjetas de crédito en Internet. La motivación fundamental detrás de SET es acelerar las transacciones y reducir los fraudes. Para acelerar las transacciones el protocolo automatiza el proceso de compra permitiendo que la computadora provea automáticamente el número de tarjeta de crédito y alguna otra información sobre el pago. Con el objeto de reducir los fraudes SET fue diseñado de tal forma que el comerciante nunca tiene acceso al número de tarjeta de crédito del comprador. En lugar de esto el comerciante recibe un número de tarjeta de crédito encriptado que solo puede ser desencriptado por el banco.

SET esta constituido por tres partes: una “cartera electrónica” que reside en la computadora del comprador, un servidor que funciona en la computadora del comerciante y el servidor de pagos que funciona en el banco en el cual el comerciante tiene sus cuentas.

SET usa encriptamiento para proveer de confidencialidad y firmas digitales para autenticación. A los comerciantes se les exige tener certificados digitales que sean expedidos por sus respectivos bancos. Lo clientes pueden o no tener certificados digitales. Un mensaje de compra de SET esta constituido por dos grupos de información, una que

es privada entre el comerciante y el cliente y otra que es privada entre el cliente y el banco. El campo con información para el comerciante es encriptado usando la clave publica del comerciante mientras que la información de tarjeta de credito es encriptada con la clave publica del banco. Adicionalmente a estos bloques encriptados el mensaje de requerimiento de compra contiene un "message digest" para cada bloque y una firma para todo el mensaje. La firma se obtiene concatenando los dos "message digest", obteniendo el "message digest" de esa concatenación y firmando el "message digest" resultante. La firma dual permite tanto al comerciante como al banco confirmar su parte sin necesidad de desencriptar la otra parte del mensaje.

6.4.3. PKI

La infraestructura de claves publicas (Public Key Infraestructure) es el sistema de certificados digitales, autoridades certificadoras, herramientas, sistemas y hardware que son usados para desarrollar tecnología de clave publica. La idea de sus proponentes iniciales era la posibilidad de que todas las personas (en USA) podrían recibir un certificado que les serviría para autenticación, firmas digitales y para el manejo de confidencialidad.

La historia de su desarrollo esta vinculado con la iniciativa de Netscape de facilitar el desarrollo de transacciones seguras en Internet. En su momento Netscape desarrollo la primera versión de SSL que permitía al usuario del navegador confirmar, a través de un ícono en la pantalla (una llave rota o entera) si el servidor web al cual estaba conectándose era o no seguro. Con el objeto de evitar acusaciones sobre las condiciones de competencia que esta tecnología podía producir Netscape decidió contratar a RSA Data Security quienes habían suplido la tecnología de clave pública sobre la que Netscape funcionaba para que proveiera de certificados.

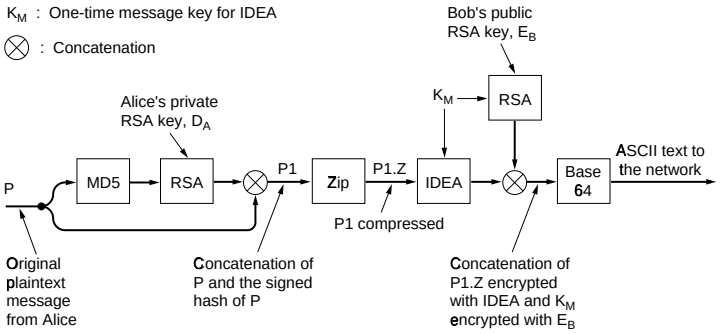
Posteriormente la aparición de IIS (Internet Information Server) de Microsoft y de la versión de código libre producida por Erick Young, llamada SSLeay. Esto hizo perder parte del mercado a Netscape. En 1995 RSA creó una empresa que manejara su área de CA llamada Verisign, los potenciales clientes se preocuparon por la posibilidad de verse forzados a comprarle certificados a una sola compañía de tal forma que los principales navegadores proveieron de otras autoridades certificador preinstaladas en sus sistemas. A pesar de que han surgido algunas otras autoridades certificadoras hoy día Verisign sigue siendo dominante en ese mercado.

6.4.4. PGP

Pretty Good Privacy de Phil Zimmermann (http://www.pgp.net) y http://www.pgp.com:

- Libre distribución: el autor quería que el público tuviera acceso a criptografía fuerte ("Si la privacidad es ilegal, sólo los ilegales tendrán privacidad").
- Disponible para Unix, Mac, Windows . . .
- Orientado a correo electrónico privado y/o firmado, pero no limitado a ello.
- *Web of trust* en vez de CA's.

- Utiliza RSA (para autenticidad), MD5 (para integridad) e IDEA (para privacidad). Versión 5 también ofrece otras alternativas.
- Tiene facilidades para administración personal de claves.
- Versión "internacional" disponible fuera de EE.UU. es idéntica en funcionalidad a la versión "nacional", e interoperable con la misma.



- Alicia quiere mandar el mensaje P a Bob. PGP calcula un hash de P usando MD5.
- Si Alicia quiere firmar el mensaje, PGP utiliza RSA con la clave privada de Alicia para firmar el resultado del MD5.
- Si Alicia quiere privacidad:

1. La firma, si existe, se concatena con P . Se comprime usando el algoritmo de Ziv y Lempel (conocido como ZIP).
2. Se genera una clave aleatoria de "sesión", usando el teclado como fuente de entropía. Tanto los tiempos como el texto se utilizan para generar 128 bits,K_M
3. El algoritmo IDEA, en modo CFB con un IV de 0 , encripta el mensaje comprimido con la clave K_M. Además, K_M es encriptada con la clave pública de Bob.
4. El resultado final se codifica en ASCII usando **base64**. Este paso es opcional.

Manejo de Claves:

PGP soporta 3 tamaños de clave RSA: **casual** (384 bits), **comercial** (512 bits), y **militar** (1024 bits).

> This whole business of protecting public keys from tampering is the single most difficult problem in practical public key applications.} – Phil Zimmermann

PGP mantiene dos estructuras locales en la máquina de Alicia:

- El **llavero privado** contiene uno o más pares de claves públicas y privadas. Alicia puede tener varios pares, cada uno con un identificador (que consiste de los 64 bits más a la derecha de la clave pública). Todo esto se guarda en disco, encriptado a su vez usando una contraseña de longitud arbitrario.
- El **llavero público** contiene las claves públicas de los corresponsales de Alicia, además de sus identificadores respectivos y una indicación del nivel de confianza que Alicia tiene en cada una. Este último dato se usa porque las claves vienen de distintos fuentes. Cada clave obtenida de otra persona está firmada con la clave pública de las personas que la certifican.

6.4.5. Kerberos

Desarrollado como parte del Proyecto *Athena* en MIT, con el propósito de autentificar clientes a servidores y vice versa. Se considera que las siguientes amenazas existen:

- Manuel puede accesar físicamente a la estación de trabajo de Alicia (en su ausencia) y aparentar ser ella.
- Manuel puede alterar la dirección de red de una estación para que parezca a otra.
- Manuel puede escuchar el tráfico entre cliente y servidor y usar un ataque de *replay*.
- La filosofía de Kerberos es que un servidor central provee servicios de autentificación para usuarios y otros servidores. Se asume que el servidor central está físicamente protegido.

Hay dos versiones de Kerberos: Versión 4 (ahora considerada muerta) y Versión 5. Ambas usan criptografía simétrica. V4 especifica DES, mientras V5 permite distintos sistemas.

Requisitos

Los siguientes fueron metas de Kerberos a la hora de diseño del sistema:

- Que fuera suficientemente fuerte para no constituir el punto débil del sistema Athena.
- Todo servicio que usa Kerberos para autentificación depende totalmente de su confiabilidad, por lo tanto esta debe ser alta. Debe permitir que un servidor respalde a otro.

- Con excepción de pedirle una contraseña, el sistema debe ser transparente para el usuario.
- Debe ser capaz de soportar grandes números de clientes, usuarios y servidores. Esto sugiere una arquitectura modular y distribuida.

La técnica de autentificación está basada en el protocolo de Needham y Schroeder. Sin embargo su implementación concreta es mucho más compleja. Describimos el sistema en una serie de versiones cada vez más refinadas. La última de estas corresponde a V4.

Primera Versión

Sea C un cliente (un proceso corriendo en una estación de trabajo, ej. para correo electrónico), A un servidor de autentificación, S otro servidor (ej. de correo electrónico), I_C el identificador de un usuario de C , P_C la contraseña de dicho usuario, D_C la dirección de red de la estación de C, y K_S una clave compartida entre A y S

$$C \rightarrow A : [I_C, P_C, S]$$

$$A \rightarrow C : [ticket]$$

$$C \rightarrow S : [I_C, ticket]$$

donde **Ticket** $= E_{k_S}(I_C, D_C, I_S)$ Se incluye D_C para garantizar que el *ticket* sólo puede ser usado desde el mismo punto que lo solicitó.

Hay dos problemas principales con éste método:

- Para no guardar una copia de P_C en la estación, hay que pedírselo al usuario cada vez que C necesita contactar a S . Esto puede mejorar si los *tickets* son reutilizables, pero todavía se necesita uno para cada tipo de servicio.
- El primer mensaje ($C \rightarrow A$) requiere transmitir P_C sobre la red, lo cual es vulnerable.

Segunda Versión

Introducimos un nuevo servidor T , cuya función es emitir *tickets* (se llama un **TGS** o *Ticket Granting Server*). Ahora tenemos:

Una vez por sesión del usuario:

$$C \rightarrow A : [I_C, I_T]$$

$$A \rightarrow C : [E_{K_C}, (\mathbf{Ticket}_T)]$$

Una vez por tipo de servicio:

$$C \rightarrow T : [I_C, I_S, \mathbf{Ticket}_T]$$

$$T \rightarrow C : [\mathbf{Ticket}_S]$$

Una vez por uso del servicio:

$$C \rightarrow S : [I_C, \mathbf{Ticket}_S]$$

donde $\mathbf{Ticket}_T = E_{K_T}(I_C, D_C, I_T, X_1, V_1)$ (o sea, un "*ticket* para pedir *tickets*") y $\mathbf{Ticket}_S = E_{K_S}(I_C, D_C, I_S, X_2, V_2)$ (o sea, un "*ticket* para pedir el servicio S").

$\mathbf{Ticket}_T$ es enviado por A utilizando una clave K_C , derivada de P_C, que A conoce. Por lo tanto C puede decriptar la respuesta de A y guardar el *ticket*. $\mathbf{Ticket}_T$ es utilizado cuando C necesita pedir un *ticket* para algún servicio nuevo. Las siguientes veces, utiliza $\mathbf{Ticket}_S$.

Dado que los *tickets* son reutilizables, es importante evitar que Manuel intente un ataque de *replay*. Para reducir sus posibilidades, cada *ticket* incluye un **timestamp**, X_i, y un período de validez V_i.

Todavía quedan dos problemas:

- ¿Qué valores se deben usar para los V_i? Si son cortos, habrá que pedirle la contraseña al usuario con mucha frecuencia.
- Si son largos, aumenta la posibilidad de un ataque *replay*. En efecto, se tiene un requerimiento adicional: T o S debe poder comprobar que el usuario de un *ticket* es aquello a quien el mismo fue entregado.

La manera de lograr esto es que A le entregue un secreto a C y a T que sólo ellos conocen. Kerberos usa una clave de sesión para esto.

- Manuel podría simular S (sólo tiene que recibir *tickets* y "aprobarlos"). Existe una necesidad para la autentificación de servidores a usuarios.

Kerberos Versión 4

Para obtener Tickets para pedir un Ticket:

$$C \rightarrow A : [I_c, I_T, X_1]$$

$$A \rightarrow C : [E_{K_C}(K_{C,T}, I_T, X_2, V_2, \mathbf{Ticket}_T)]$$

Para obtener un Ticket de servicio:

$$C \rightarrow T : [I_S, \mathbf{Ticket}_T, \mathbf{Aut}_C]$$

$$T \rightarrow C : [E_{K_{C,T}}(K_{C,S}, I_S, X_4, \mathbf{Ticket}_S)]$$

Para obtener un servicio:

$$C \rightarrow S : [\mathbf{Ticket}_S, \mathbf{Aut}'_C]$$

$$\mathrm{S} \rightarrow C : [E_{K_{C,S}}(X_5 + 1)$$

Donde $\mathbf{Ticket}_T = E_{K_T}(K_{\{C,T\}}, I_C, D_C, I_T, X_2, V_2)$, ($\mathbf{Ticket}_S = E_{K_S}(K_{\{C,S\}}, I_C, D_C, I_S, X_4, V_4)$), $\mathbf{Aut}_C = E_{K_{\{C,T\}}}(I_C, D_C, X_3)$ y $\mathbf{Aut}'_C = E_{K_{\{C,S\}}}(I_C, D_C, X_5)$
Estos últimos se llaman **autentificadores** (*authenticators*).

Los autentificadores comprueban la identidad de quienes los generan, ej. ($\mathbf{Aut}_C$ significa "*A la hora X_3 yo conozco la clave $K_{\{C,T\}}$*". Cada autentificador se usa una sola vez y tiene vigencia muy corta. El último paso del protocolo es para la mutua autenticación. X_5 es usado como un *nonce*, y S demuestra que lo conoce. Además, al final C y S comparten una clave secreta que pueden usar para privacidad de la sesión. Esto no forma parte de Kerberos. Un servidor A más un grupo de clientes y usuarios constituyen un **reino** (*realm*) de Kerberos. Existe un protocolo adicional que permite a un reino autentificar a los usuarios de otro, bajo mutuo acuerdo.

Kerberos Versión 5

V4 tiene varias deficiencias que V5 trata de corregir:

- V4 depende de DES, lo cual es un problema por las leyes exportación de EE.UU. En V5 esto se puede parametrizar.
- V4 requiere de direcciones IPv4. V5 permite otros tipos de dirección (OSI, IPv6, ...).
- V4 permite dos ordenamientos de byte. V5 expresa los mensajes en ASN.1.
- En V4, las vigencias se expresan como un valor de 8 bits, en unidades de 5 minutos. En V5, se da la hora de comienzo y terminación del período de validez.
- En V4, un servidor no puede pedir servicio a otro por parte de un cliente. V5 permite pasar los credenciales de un cliente entre servidores.
- V4 encripta los *tickets* dos veces (segundo y cuarto pasos). El segundo encriptamiento no hace falta y V5 lo quita.
- V4 usa un modo de encriptamiento DES no-estándar, que resultó ser vulnerable. V5 usa modo CBC.

Un problema que existe en ambas versiones es que son vulnerables de un ataque a la contraseña del usuario. Hay interés en el uso de tarjetas físicas para reducir esta posibilidad.

Capítulo 7

Criptografía

7.1. Repaso de conceptos básicos

Un mensaje en su estado original consiste de **texto en claro (plaintext, cleartext)**. Se disfraza por un proceso de **encriptamiento (encryption)**, el cual produce texto **encriptado (cyphertext**). El proceso inverso es **decriptamiento (decryption)** (también encifrado, descifrado)).

Criptografía es la ciencia de la seguridad de mensajes, practicado por **criptógrafos**. La ciencia (o arte) de violar la seguridad de mensajes es **criptanálisis**, practicado por **criptanalistas**. **Criptología** es una rama de las matemáticas que estudia ambos aspectos. Lo practican los **criptólogos**.

M representa el texto en claro (no necesariamente es texto), C el texto encriptado. C puede ser mas largo que M. Encriptamiento es una función: $E(M) = C$, y decriptamiento: $D(C) = M$ Además, se requiere que: $D(E(M)) = M$.

7.1.1. Algoritmos y Claves

Las funciones E y D son implementadas por un algoritmo criptográfico, o cifrado (cypher). El algoritmo no debe depender por su seguridad, del hecho que sea desconocido por el adversario. Por lo tanto, todo algoritmo moderno depende de una clave (key), escogido al azar de un espacio grande (el keyspace):

$$E_K(M) = C$$

$$D_K(C) = M$$

$$D_K(E_K(M)) = M$$

(Ojo: no necesariamente $E_K(D_K(X)) = X$)
A veces se usan dos claves:

$$E_{K_1}(M) = C$$

$$D_{K_2}(C) = M$$

$$D_{K_2}(E_{K_1}(M)) = M$$

Un **criptosistema** consiste de un algoritmo mas todos los posibles textos (en claro y encriptados) y claves.

Los algoritmos se dividen en dos clases:

1. Simétricos, también de una clave, de clave secreta. Hay dos categorías: stream y block.
2. Asimétricos, también de dos claves, de clave pública/privada. La clave pública es para encriptar, la privada para decriptar. A veces se usan al revés para obtener autenticidad (firmas digitales).

Criptanálisis Es la ciencia de recuperar el mensaje original sin tener acceso a la clave (si la clave se revela por otros medios se llama un compromiso). Un intento de criptanálisis es un ataque. Hay 4 tipos principales:

1. Cyphertext-only solo se dispone del texto encriptado.
2. Known plaintext se conoce parte del texto original (ej. login:)
3. Chosen plaintext se puede insertar texto para ser encriptado, ej. en una base de datos
4. Adaptive chosen plaintext igual que el anterior pero en forma iterativa
5. Chosen cyphertext en el caso de sistemas de clave pública
6. Chosen key se conoce algo sobre el método de escoger claves
7. A paliza . . .

7.1.2. Seguridad de Algoritmos

Distintos algoritmos ofrecen distintos grados de seguridad, pero los estimados siempre son probabilísticos el criptanalista podría tener suerte. Categorías de éxito que puede tener el adversario:

1. **Total**: se encuentra K tal que $D_K(C) = M$
2. **Deducción Global**: se encuentra un algoritmo alternativo, A, que permite calcular $D_K(C)$ sin conocer K
3. **Deducción Local**: se encuentra M (de alguna manera)

4. **Deducción de Información**: se obtiene alguna información sobre K o M algunos bits de K, algunas palabras de M.

Un algoritmo es totalmente seguro **(unconditionally secure)** si ninguna cantidad de texto encriptado provee suficiente información para hallar el texto en claro (o la clave), sin límite de recursos computacionales. Tal sistema existe.

En la práctica, es mas importante considerar los algoritmos condicionalmente seguros (también llamados fuertes), que resisten ataques con una cantidad razonable de recursos. Importante: si los datos mantienen su valor por tiempo prolongado, hay que prever los recursos computacionales del futuro.

7.1.3. Teoría de Números

$a \equiv b(mod\ n)$ si $a = b + kn$ para algún entero k . b se llama el residuo de a, módulo n. a es **congruente a** b , módulo n. Los enteros $0...n-1$ forman **el conjunto completo de residuos** módulo n. (Nota: la función de librería mod( ), o el operador % en el lenguaje C, no siempre siguen la definición precisa.)

Aritmética modular es conmutativa, asociativa y distributiva.

Además:

$(a + b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$

$(a - b) \bmod n = ((a \bmod n) - (b \bmod n)) \bmod n$

$a * b \bmod n = ((a \bmod n) * (b \bmod n)) \bmod n$

$(a * (b + c)) \bmod n = ((a * b \bmod n) + (a * c \bmod n)) \bmod n$

Una operación criptográfica comunes la exponenciación modular, $a^x mod\ n$, ej.:

$a^9 mod\ n = (((a^2) mod\ n)^2 mod\ n)^2 mod\ n) * a\ mod\ n$

7.1.4. Números Primos

Un número primo es un entero mayor que 1 cuyos únicos factores son 1 y el mismo. Hay una cantidad infinita de primos. Los de interés para la criptografía son números grandes (512 bits o más). a y n son relativamente primos si no tienen factores en común sino 1 , o sea que $\gcd(a, n) = 1$. Calcular $\gcd(a, n)$ es fácil usando el conocido Algoritmo de Euclides. a y b son inversos multiplicativos si $a * b\ mod\ n = 1$ El inverso de a se escribe a^{-1} y no siempre existe.

En general: si a y n no son relativamente primos, entonces $a^{-1} \equiv x(mod\ n)$ no tiene solución. Si lo son, puede haber mas de una solución. Para garantizar una solución única, basta que n sea primo. En general, encontrar un inverso es mas costoso que gcd, pero todavía es tratable.

Los números $n\ mod\ p$, cuando p es primo o una potencia de un primo forman un cuerpo finito, también llamado un cuerpo de Galois, y denotado $GF(p)$. Están definidos los operadores de suma, resta, multiplicación, y división por un factor distinto a 0. Existe una identidad aditiva (0) y otra multiplicativa (1). Cada número distinto a 0 tiene inverso único. Muchos criptosistemas se basan en $GF(p)$, donde p es un primo grande.

Teorema: si m es primo, y no es un factor de a, entonces $a^{m-1} \equiv 1(mod\ n)$, eso se conoce como el pequeño teorema de FERMAT.

Definición: el conjunto reducido de residuos $mod\ n$ es el subconjunto de residuos que sean relativamente primos a n. Si n es primo, es el conjunto $1,2....n-1$. La **función indicatriz de Euler** (Euler´s totient function) $\phi(n)$ es el número de elementos en ese conjunto.

Si $\gcd(a,n)=1$, $a^{\phi(n)} mod\ n = 1$

Si n es **primo**, $\phi(n) = n-1$. Si $n = pq$ donde p y q son primos, entonces $\phi(n) = (p-1)(q-1)$.

7.1.5. Factorización

El problema de la factorización es el de hallar los factores primos de un entero n. Un resultado básico es que tal factorización es única para cada n. Esto se conoce como El Teorema Fundamental de la Aritmética.

La complejidad de la factorización es mucho mayor que la de la multiplicación. El método ingenuo es probar todos los primos menores que $\sqrt{n}$, pero el número de posibles candidatos es aproximadamente $\sqrt{n}/\ln\sqrt{n}$ Sin embargo, el área es extremadamente activo, y varios algoritmos novedosos se han inventado en los últimos años:

1. Criba Cuadrática (Quadratic Sieve o QS): el mejor método para números de menos que 110 dígitos (decimales). Las mejores versiones tienen complejidad temporal asintótica de $e^{(1+O(1))(\ln(n)^{1/2}(\ln(n))^{1/2}}$

2. Criba de Cuerpo Numérico (Number Field Sieve o NFS): el mejor conocido para mas de 110 dígitos. La complejidad asintótica se estima en $e^{1,923+O(1))(ln(n))^{1/3}(ln(ln(n)))^{2/3}}$

Los algoritmos asimétricos importantes usan números primos grandes.

Algunas preguntas:

1. Si todo el mundo necesita un número primo diferente, hay suficientes? Hay aproximadamente 10^{151} primos de $<= 512$ bits. Si cada átomo en el universo necesitara 10^9 primos nuevos cada segundo desde el comienzo del universo hasta ahora, solo se necesitarían 10^{109} primos casi 10^{151} sobran . . .

2. Qué pasa si dos usuarios escogen el mismo primo, por accidente? No va a pasar al menos que el método de generación sea defectuoso.

3. Si alguien crea una base de datos de todos los primos, no podría usarla para atacar al criptosistema? Sí, pero si pudiéramos guardar 1 Gb en un disco de 1 gramo de peso, la lista de primos de solo 512 bits sería tan pesado que la base de datos se convertiría en un hueco negro . . .

7.1.6. Información y Complejidad

El papel del criptanalista es utilizar información redundante en C para deducir M o K, quizás con la ayuda de mensajes o claves anteriores, o información externa sobre los participantes.

Shannon (1948) define la **entropía de un mensaje** como el número de bits necesarios para representarlo en una codificación óptima:

$$H(x) = \sum_{i=1}^{n} p(x_i) log_2 \frac{1}{p(x_i)}$$

donde los x_i son los posibles mensajes y $p(x_i)$ es la probabilidad de que el mensaje sea x_i. Entropía es una medida de incertidumbre.

Ejemplo: sean A, B, y C mensajes con probabilidades 0.5, 0.25 y 0.25 respectivamente. Entonces

$$H(x) = 0{,}5 log_2 2 + 2 log_2 4 = 0{,}5 + 1{,}0 = 1{,}5$$

Podemos asignar 1 bit a A y 2 bits a cada uno de B y C: A=0, B=10, C=11. Nótese que si las $p(x_i)$ son todas iguales, $H(x) = log_2 n$

7.1.7. Distancia de Unicidad

Una pregunta básica es: *cuantos bits de C necesito para que M sea unívocamente determinable?* Es pertinente considerar algunas propiedades de los lenguajes:

La **tasa** (*rate*) de un lenguaje para todo mensaje de longitud M caracteres se define como $r = \frac{H(x)}{n}$ (o sea, el número promedio de bits por caracter). Para N grande, $1 < r_{inglés} < 1{,}5$

La **tasa absoluta** es la máxima cantidad de información que pudiera tener cada caracter bajo una codificación óptima: $R = log_2 L$, donde L es el número de caracteres en el lenguaje. Para inglés, $L = 26$, entonces $R = 4{,}7$ (bits por letra).

La **redundancia** del lenguaje se define como $D = R - r$, aunque es usual representarlo como porcentaje: $\frac{D}{R} 100$ Para el inglés está en el orden de $70 - 80\,\%$ (ignorando puntuación, espacios, números etc.) Es por esto que es conveniente **comprimir** los mensajes en idiomas naturales antes de encriptarlos (pero *Ojo:* los programas normales de compresión incluyen varios bytes conocidos al principio de la salida).

La **entropía** de un criptosistema $H(K)$ mide el tamaño del espacio de claves. Es aproximadamente el número de bits que tiene cada clave.

En un criptosistema simétrico, la **equivocación de clave**, $H_C(K)$ es el grado de incertidumbre en K dado C. Para la mayoría de los sistemas, decrece con la longitud de C. La longitud en que $H_C(K) \approx 0$ se llama la **distancia de unicidad**, U.

En sentido probabilístico, obtener más de U bits de C implica que hay un sólo candidato válido para K, y por lo tanto para M. En sistemas reales, U puede ser difícil calcular. Dado un mensaje de longitud n, existe un cierto número de candidatos para la clave. Si el espacio de claves es pequeño relativo al espacio de mensajes, este número es

$$N_k \approx 2^{H(K) - nD} - 1$$

$n = U$ cuando $N_k = 1$, que sucede cuando $n \approx \frac{H(K)}{D}$

Para un sistema con clave de 56 bits (como DES), y mensajes en inglés, $U \approx 66$ (unos 8 bytes de ASCII). Para una clave de 128 bits, $U \approx 297$

Algunas consideraciones adicionales:

- Para sistemas en que el tamaño del espacio de claves se acerca al tamaño del espacio de mensajes, esta derivación no es válida, y en el límite (espacios de tamaños iguales) la distancia de unicidad es infinita ($H_C(K)$ nunca se acerca a 0) , lo cual implica que se puede tener un sistema **impenetrable**.
- La teoría tampoco dice nada sobre la dificultad *computacional* de hallar M.
- La teoría no es aplicable a los sistemas asimétricos? porqué no?

7.1.8. Complejidad

La Teoría de Complejidad distingue entre **complejidad de algoritmos** y **complejidad de problemas**.

Complejidad de Algoritmos :Nos interesa el costo de ejecución de los algoritmos, en términos de *tiempo* (T) y *espacio* S, expresados como funciones del tamaño de la entrada n. El valor preciso de T o S depende de detalles de implementación, que no interesan, así que usamos una notación de "ordenes de magnitud": $f(n) = O(g(n))$, que significa: $\exists c, n_0$ constantes, tales que $f(n) \leq c \mid g(n) \mid \forall n \geq n_0$

Ejemplo: si $f(n) = 17n + 10$, entonces $f(n) = O(n)$, porque $f(n) \leq 18n$, $\forall n \geq 10$ entonces $g(n) = n, n_0 = 10, c = 18$.

Si $f(n)$ es un polinomio $a_t n^t + a_{t-1} n^{t-1} + ...$, entonces $f(n) = O(n^t)$. Si $T = O(n^t)$ para alguna constante t, se dice que el algoritmo es de complejidad *polinomial*. Si $t = 0$, es *constante*; si $t = 1$, es *lineal*; si $t = 2$, es *cuadrático*, etc.

Si $T = O(t^{h(n)})$, para t constante y $h(n)$ polinomio, el algoritmo es *exponencial*.

La notación $O(n)$ es *relativa*. Es posible que, para el rango de n que nos interesa, un algoritmo cuadrático sea mejor que uno lineal, dependiendo de los constantes, por ejemplo $f(n) = 500n + 5000000$ vs. $f(n) = n^2$, para $n < 1000$.

7.1.9. Complejidad de Problemas

La complejidad de un problema se clasifica en términos del costo de solucionar las instancias más difíciles de ello (usando una máquina de Turing o equivalente) con el mejor algoritmo posible.

Si el algoritmo es polinomial, el problema se define como *tratable* (*tractable*). La clase de problemas tratables se llama **P**. A los no-tratables también se les llaman **difíciles** o **duros** (*hard*}¡). La clase de problemas duros que son solucionables con una máquina de Turing *no-determinística* se llama **NP**. Muchos criptosistemas simétricos, y todo sistema asimétrico práctico, están en **NP**.

(*Nota*: Alan Turing demostró que algunos problemas son **no-decidibles** (*undecidable*), o sea *no puede existir* un algoritmo para resolverlos. No parecen tener aplicaciones criptográficas.)

Está claro que $P \subseteq NP$, pero no se sabe si la inclusión es estricta. Es decir $P = NP$ Este es quizás el problema teórico más importante en Ciencias de la Computación.

Algunos problemas en NP son tan difíciles como cualquier miembro de NP. Ellos forman una subclase llamado **NP-Completos**. Si se logra obtener un algoritmo polinomial para cualquiera de ellos, la equivalencia de P y NP se resolverá positivamente. El Problema del Viajero es uno muy conocido.

7.1.10. Problemas Difíciles y Criptografía

No todo problema en NP es apropiado como base para un sistema criptográfico: La Teoría de la Complejidad trata sobre problemas aislados. Generalmente el criptanalista dispone de varios problemas interrelacionados. Por ejemplo, Lempel (1979) menciona un posible criptosistema para el cual la posesión de un sólo mensaje encriptado implica un problema NP-Completo para el adversario, pero k mensajes (usando una clave de k bits) permite construir un sistema de k ecuaciones simultáneas en k incógnitas. La complejidad de un problema se mide por el peor caso (o el caso promedio). Para ser útil criptográficamente, debe ser difícil en todos o casi todos los casos.

7.2. Algoritmos criptográficos

7.2.1. Algoritmos Criptográficos Clásicos

Los sistemas de *transposición* permutan los caracteres del mensaje. La clave es el mapa de permutación. Más usual es simplemente una permutación de período fijo:

M=SIMONBOLIVAR

K=4213...

C=OISMLBNORVIA

Nótese que las permutaciones de un período dado definen un grupo finito.

¿Cuántos caracteres de C se necesitan para penetrar un sistema de transposición? $U = \frac{H(K)}{D} = log_2 \frac{d!}{D}$ donde d es el período y todas las permutaciones son igualmente probables.

$$U \approx \frac{d \log_2(d/e)}{D} \approx ,3d \log_2(d/e)$$

(para el inglés)

En la práctica, todos estos sistemas son susceptibles a una análisis de frecuencia de caracteres.

7.2.2. Sistemas de Sustitución

Sea A_m el *alfabeto* de los mensajes y A_c el del texto encriptado. Pueden ser iguales o no. Los sistemas de *sustitución* cambian cada símbolo (o grupo de símbolos) en A_m por un símbolo (o grupo) de A_c. Hay cuatro tipos: Los sistemas **Simples** o **monoalfabéticos**: ej.\ sustituir cada 'A' por 'L', 'B' por 'Q', 'C' por 'A', etc.

Un ejemplo clásico es el sistema de César. Es muy fácil romper.

El número de claves posibles sobre un alfabeto de tamaño n es $n!$. Si son equiprobables tenemos:

$$U \approx \frac{H(K)}{D} = \frac{\log_2 n!}{D} \approx 27{,}6$$

(para el inglés)

Hoy día estos sistemas sólo sirven cuando no buscamos la seguridad, como el sistema ROT13 de Usenet. Algunos son de interés histórico, ej. el compositor J.S.Bach

codificó su propio apellido como notas musicales en “La Ofrenda Musical” y “El Arte de la Fuga” (BACH = *Si~bemol, La, Do, Si*).

Los sistemas **Homofónicos** cambian cada caracter del mensaje a uno de varias posibilidades, ej. “A”→ “5”, ’17’, ’23’ o ’64’. Cada vez que se encripta un caracter, el símbolo usado se escoge al azar entre las opciones. Los primeros sistemas fueron usados por el Duque de Mantua en 1401.

Nótese que los subconjuntos de A_c son disjuntos, de modo que no hay ambigüedad para decriptar. Además, no necesariamente son del mismo tamaño – se asignan más posibilidades a los caracteres más frecuentes para hacer más difícil el análisis estadístico.

Sin embargo, con suficiente cantidad de texto encriptado, sólo un candidato para K decripta C en un M razonable, y las propiedades estadísticas no se esconden totalmente. Penetración es casi trivial con un ataque “texto conocido”, y fácil con “sólo encriptado” si hay suficientes datos y disponemos de un computador.

Los sistemas **Polialfabéticos** usan múltiples sustituciones simples en rotación, según la posición del caracter en el mensaje. El primer sistema fue inventado por Leon Battista Alberti en 1568, en la forma de dos discos alfabéticos concéntricos que podían girar independientemente. Esto equivale a múltiples sistemas de sustitución sencilla. La contribución de Alberti era la idea que los discos podían girar arbitrariamente *durante* la encripción, por ejemplo para cada caracter del mensaje.

Ejemplos famosos son los sistemas de **Blaise de Vigenére** y **Sir Francis Beaufort**, (ambos realmente inventados por otra gente).

7.2.3. Criptografía Perfecta

Los sistemas polialfabéticos de período más largo son más difíciles para penetrar. El mejor caso sería que el período fuera más largo que el mensaje (o de un conjunto de mensajes consecutivos).

Los sistemas de **clave corrida** (*running key*) utilizan una fuente no-periódica para la clave, por ejemplo el texto de un libro.

Sin embargo, todavía hay efectos de la distribución no-uniforme de letras (y de “N-grams”) en la fuente de la clave.

El mejor caso posible es si la clave es *aleatoria*, y no repetida durante toda la historia de la comunicación. Esto se llama un **one-time pad** y es el único criptosistema que es totalmente seguro. Es decir, su distancia de unicidad es infinita.

7.2.4. Máquinas de Rotor

En la práctica, la longitud de la clave hace difícil el manejo de un “one-time pad”, así que se suele usar una fuente periódica y/o pseudo-aleatoria.

Durante los años 20 se inventaron sistemas electromecánicos de tipo polialfabético. Varios **rotors** tienen el alfabeto representado en contactos eléctricos por las dos caras, implementando una serie de mapas de sustitución. Los rotores están montados sobre un eje común, y conectados a un motor eléctrico, un teclado y un arreglo de luces.

Cada letra de la entrada gira el primer rotor una posición. Después de una vuelta, la segunda gira una posición, etc. La clave consiste del conjunto de rotores a utilizar (son

sustituibles), más la configuración inicial. Para un sistema de r rotores, cada uno con N posiciones ($N = A_m$), el período de repetición de la clave es N^r.

El ejemplo más conocido es **Enigma**, usada por los alemanes en WWII. Usaba 3 rotores de un conjunto de 5 (después 3 de 8). La historia de como los británicos lograron romperlo tiene mucha relación con los comienzos de la computación.

7.2.5. El Sistema de Vernam

En 1917 Gilbert Vernam de AT&T diseño un dispositivo sencillo y efectivo para encriptar mensajes de telégrafo (usando el código Baudot de 5 bits). Dada una secuencia de bits $K = k_1 k_2 k_3 \ldots k_i$, encriptar y decriptar se hacen así:

$$c_i = m_i \oplus k_i$$

$$m_i = c_i \oplus k_i$$

Si K es aperiódica y aleatoria, tenemos nuevamente el "one-time pad".

Si K se repite con período d , el criptanalista puede tomar dos secuencias de texto encriptado, de longitud d cada una:

$$c_i = m_i \oplus k_i$$

$$c'_i = m'_i \oplus k'_i$$

$$c''_i = c_i \oplus c'_i = m_i \oplus k_i \oplus m'_i \oplus k_i = m_i \oplus m'_i$$

que es equivalente a un sistema de clave corrida con M' (que no es aleatorio) como la clave, o sea ya no es impenetrable.

Importante: para obtener la seguridad total, no basta usar secuencias pseudo-aleatorias. Además, la clave *nunca* puede ser reutilizada.

El último tipo de criptosistema de sustitución lo constituyen los sistemas de **Polígramos**, que cambian bloques de texto a la vez, ej.

$$'ABA' \rightarrow' XYZ'$$

para esconder las estadísticas de letras sencillas. Un ejemplo es el sistema **Playfair** (en realidad inventado por el físico Wheatstone en 1854), usado por los británicos en la primera guerra mundial:

1	2	3	4	5
C	S	Y	Q	I
V	G	O	L	N
T	B	H	D	A
P	K	W	R	Z

('J' no se usa)

M= SI MO NB OL IV AR

C= YC WH GA LN CN DZ

7.2.6. DES

DES fue derivado de una propuesta de IBM, llamada **Lucifer**, y adoptado por NBS (**National Bureau of Standards**) como Estándar Federal de EE.UU. en 1976, para uso en comunicaciones gubernamentales *no-clasificadas*. También ha sido aprobado por ANSI, y varias asociaciones bancarias.

Su introducción en 1976 fue acompañada por bastante controversia, sobre todo respecto al papel de la NSA. Por ejemplo, el diseño original de Lucifer tenía claves de 112 bits. Por recomendación de NSA, NBS redujo la clave a 56 bits. Nunca se ha explicado por qué . . .

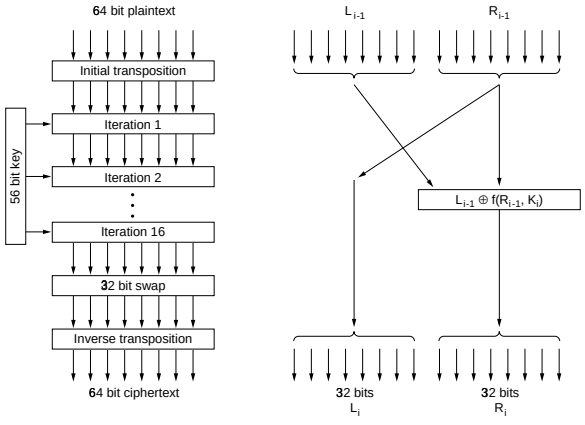
Hoy día DES es el criptosistema más usado en el mundo, a pesar de los esfuerzos de NIST y NSA para retirar su certificación por razones de edad.

DES Por Encima

DES es un **criptosistema de producto** (combina técnicas de sustitución y transposición), y está diseñado para ser muy eficiente en hardware. Encripta grupos de 64 bits de texto con una clave de 56 bits, para producir 64 bits de salida. Decriptamiento es esencialmente igual. Internamente, DES consiste de:

- Una **permutación inicial** del texto, seguida por la separación en dos grupos de 32 bits cada uno.

- 16 **rondas** que combinan permutación, sustitución, y XOR con partes de la clave:



- Una **permutación final** que invierte la inicial.

Las Rondas

Sean I_i y D_i las dos mitades de los datos en la ronda i . Entonces las siguientes relaciones se cumplen:

$$I_i = D_{i-1}$$

$$D_i = I_{i-1} \oplus f(D_{i-1}, K_i)$$

La excepción es la última ronda, en que las dos mitades no se intercambian.

La Función $f()$:

Cada ronda utiliza un subconjunto distinto, K_i, de 48 de los 56 bits de la clave original. Este subconjunto también sufre una permutación.

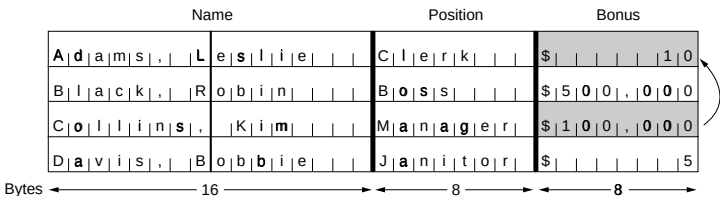
Luego D_i es expandido de 32 a 48 bits para poder realizar el $\oplus$ con K_i Esto aumenta rápidamente la dependencia entre bits de salida y de entrada, para que cada bit de salida sea una función de cada bit de entrada y cada bit de la clave.

Ahora el resultado del $\oplus$ pasa por operaciones de sustitución, realizadas por 8 **S-boxes**. Cada S-box tiene una entrada de 6 bits y una salida de 4 (lo cual vuelve a reducir los datos a 32 bits). En el diseño de los S-boxes está la seguridad de DES.

Modos de Operación

DES es un criptosistema simétrico de bloque, por lo tanto es posible usarlo en varias maneras (modos) distintas:

ECB (*Electronic Code Book*): Cada bloque de texto es encriptado independientemente. Tiene una debilidad teórica en que el mismo texto siempre se encripta de la misma manera (con la misma clave), pero tiene la ventaja que el encriptamiento no tiene que ser secuencial, ej. puede usarse para registros de una base de datos:

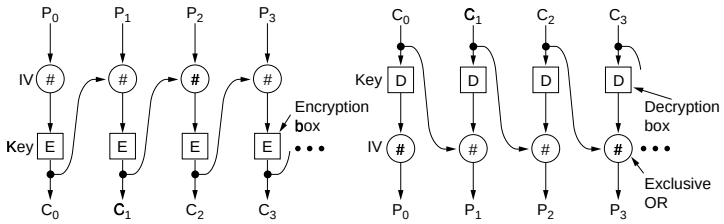


Un problema es que mismo texto (ej. el comienzo de un mensaje) siempre se encripta igual, lo cual permite un ataque "**known plaintext**". Además, si Manuel tiene acceso físico al texto encifrado, podría hacer un **replay** de algunos registros.

CBC (*Cipher Block Chaining*): Antes de encriptar cada bloque, se le hace un $\oplus$ con el bloque anterior ya encriptado. Para el primer bloque, se usa un **vector de inicialización** (IV) aleatorio (puede ser distinto cada vez pero no es esencial). No es necesario que el IV sea secreto. Para encriptar o decriptar cada bloque, se hace:

$$C_i = E_K(P_i \oplus C_{i-1})$$

$$P_i = C_{i-1} \oplus D_K(C_i)$$



Ahora Manuel no puede realizar el ataque anterior sin ser detectado. También dificulta el criptanálisis. La desventaja es que no puede empezarse el encriptamiento antes que haya llegado el primer bloque de texto, lo cual no sería recomendable para un terminal, por ejemplo.

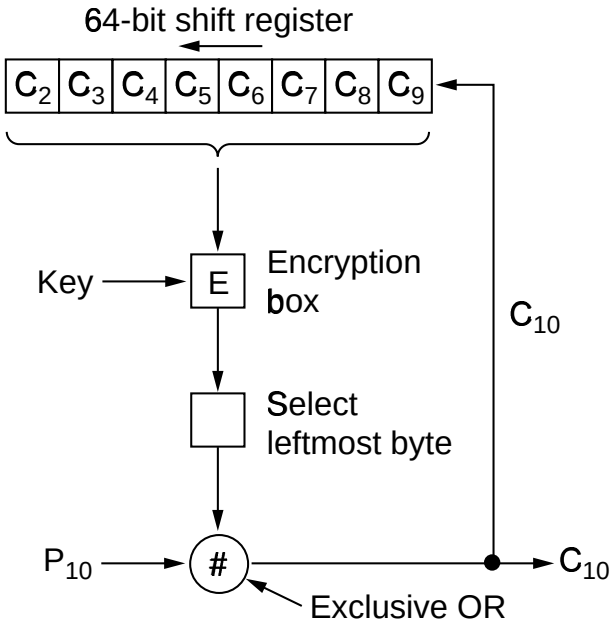
Un error de transmisión (o memoria) en un bit del texto encifrado convierte un bloque entero en basura, y daña un bit del bloque siguiente. Un error de sincronización (se inserta o se quita un bit) echa a perder el resto de la comunicación.

CFB (*Cipher FeedBack*): Esta técnica convierte un sistema de bloque en uno de "stream". El sistema de bloque se usa en modo "encriptar" tanto para la encripción como la decripción:

$$C_i = P_i \oplus E_K(C_{i-1})$$

$$P_i = C_i \oplus E_K(C_{i-1})$$

Inicialmente se carga un registro de acarreo con un IV único (no repetido). Esto se encripta, y se combina con el primer byte del texto:



Un bit de error en el texto encriptado causa un bit de error en el byte correspondiente del texto recuperado, seguido por 8 bytes de basura (porque el bit incorrecto se introduce en el registro). Un error de sincronización (de un byte) también afecta 8 ciclos. CFB es un **self-synchronizing stream cipher**.

OFB (*Output FeedBack*): OFB es un método similar a CFB, excepto que los bits de la clave se generan independientemente del texto, por lo tanto se llama un **synchronous stream cipher**. Esto es razonable si la secuencia de clave tiene período suficientemente largo (mucho más largo que el mensaje más largo). Usa un IV, que tiene que ser único pero no necesariamente secreto.

$$S_i = E_K(S_{i-1})$$

$$C_i = P_i \oplus S_i$$

$$P_i = C_i \oplus S_i$$

La ventaja es que los bits de la clave (el **keystream**) pueden ser generados previamente (sólo dependen de la clave). Un error de transmisión afecta únicamente al bit correspondiente, pero un error de sincronización destruye el resto del mensaje.

Debilidades de OFB

Si Manuel tiene la posibilidad de insertar bits en el texto original, puede atacar un sistema OFB así:

1. El texto original consiste de los bits $p_1p_2p_3p_4p_5\ldots$, el cual se encripta con la secuencia de clave $k_1k_2k_3k_4k_5\ldots$ para producir $c_1c_2c_3c_4c_5\ldots$
2. Manuel graba los c_i de un paquete de red, y destruye el paquete.
3. Manuel inserta un bit p' en la secuencia de texto.
4. Alicia retransmite: $p_1p_2p'p_3p_4p_5\ldots$ encriptado como $c_1c_2c'_3c'_4c'_5\ldots$
5. Ahora Manuel puede hacer: $k_3 = c'_3 \oplus p'$, $p_3 = k_3 \oplus c_3$,$k_4 = c'_4 \oplus p_3$, $p_4 = k_4 \oplus c_4$, etc.

Para protegerse, Alicia nunca debe reutilizar la misma secuencia de clave para dos mensajes (paquetes), aún cuando se trata de una retransmisión.

La solidez de OFB también depende de la periodicidad de la secuencia de clave. Para 64 bits, el período es aproximadamente 2^{64}. *si toda la salida se reintroduce como entrada*. Sin embargo, si sólo un byte de salida se vuelve a "enchufar", la periodicidad cae a $\approx 2^{32}$, que no es mucho.

Comentarios sobre DES

- DES tiene cierto número de *claves débiles*, ej. todos 0's , todos $1'$s , 28 0 's más 28 1 's etc. Es fácil para Alicia verificar que no está usando una de estas.
- Recientemente algunos investigadores han desarrollado una técnica llamada **criptanálisis diferencial**, que puede reducir la dificultad de buscar en el espacio de claves de $O(2^{56})$ a $O(2^{30})$
- Por mucho tiempo, no se sabía si DES era un *grupo*, es decir dadas las claves K_1 y K_2, siempre existe una K_3 tal que $E_{K_3}(M) = E_{K_1}(E_{K_2}(M))$. En 1992 se demostró que no. En otras palabras, vale la pena encriptar con más de una clave. La mejor formar se llama **Triple DES** y usa dos claves. Encriptamiento es: $C = E_{K_1}(D_{K_2}(E_{K_1}(M))$ Esto es equivalente a usar una clave de 112 bits. (La etapa D_{K_2} es para mantener compatibilidad con sistemas normales, con $K_1 = K_2$.)
- Se considera que Triple DES es bastante seguro, mientras DES normal ya no lo es, al menos contra un adversario con grandes recursos. Por ejemplo, en julio de 1998 una máquina con miles de chips especiales llamados "**Deep Crack**", construida por el *Electronic Frontier Foundation* por un costo de $250.000, logró romper una clave DES en menos de 3 días.

7.2.7. Algoritmos Asimétricos

El artículo original que introduce el concepto de algoritmo asimétrico también presentó un algoritmo que permite a Alicia y Bob intercambiar una clave de sesión. Se conoce como **Diffie-Hellman key exchange**.

Su seguridad radica en la dificultad de computar **logaritmos discretos**. Sea p un primo, y a un entero tal que $a \bmod p\ a^2 \bmod p\ldots$ sean todos distintos y consisten de los enteros de $1\ldots p-1$ en alguna permutación. a *raíz primitiva* de p . Entonces, para

cualquier b , existe una i única, tal que $b = a^i \bmod p$, $(0 \leq i \leq (p-1))$. i es el *logaritmo discreto* de b para la base $a \pmod p$. Calcular un logaritmo discreto es por lo menos tan difícil como la factorización (el converso no se ha demostrado). Un algoritmo eficiente tiene complejidad temporal $e^{((\ln p)^{\frac{1}{3}} \ln(\ln p))^{\frac{2}{3}}}$

El algoritmo de Diffie y Hellman:

- Sea q un primo, y α una raíz primitiva de q. Ambos son públicos.
- Alicia escoge un entero aleatorio $X_A < q$ y calcula $Y_A = \alpha^{X_A} \bmod q$
- Bob escoge $X_B < q$ y calcula $Y_B = \alpha^{X_B} \bmod q$
- Alicia manda Y_A a Bob, y Bob manda Y_B a Alicia.
- Alicia calcula $K_A = (Y_B)^{X_A} \bmod q$, y Bob calcula $K_B = (Y_A)^{X_B} \bmod q$. Las dos K son iguales:

$$\begin{aligned} K_A &= (Y_B)^{X_A} \bmod q \\ &= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\ &= (\alpha^{X_B})^{X_A} \bmod q \\ &= \alpha^{X_A X_B} \bmod q \\ &= \ldots \\ &= (Y_A)^{X_B} \bmod q \\ &= K_B \end{aligned}$$

Eva sólo dispone de $q, \alpha, Y_A,$ yY_B. Para encontrar X_B, por ejemplo, tendría que calcular el logaritmo discreto de Y_B para la base $\alpha \pmod q$.

Este método es usado en varias aplicaciones prácticas, incluyendo *Secure RPC* y *Secure Session Layer* (SSL).

7.2.8. El Sistema RSA

En 1978 Rivest, Shamir y Adleman publicaron su algoritmo, conocido como RSA. Sigue siendo el algoritmo de encriptamiento con claves públicas de mayor aceptación.

RSA es un esquema de bloque, en que los mensajes de representan como enteros en el rango $0 \ldots n$. Encriptar y decriptar son:

$$C = M^e \bmod n$$

$$M = C^d \bmod n = M^{ed} \bmod n$$

La clave pública consiste del par (e, n) . La clave privada es (d, n) . Para que funcione el sistema, se quiere:

- Que sea fácil hallar e, d, n tal que $M^{ed} = M \pmod{n} \; \forall \; M < n$
- Que sea difícil hallar d a partir de e y n

Recordamos el Teorema de Euler:

$$a^{\phi(n)} \equiv 1 \text{ mód } n$$

para a y n relativamente primos, y además para $n = pq$, p y q primos, $\phi(n) = (p-1)(q-1)$, esto implica que $m^{(k(p-1)(q-1)+1} \equiv m \text{ mód } n$ para cualquier k .

Entonces una posibilidad es escoger e y d tal que $ed = k\phi(n) + 1$, es decir que $ed \equiv 1 \text{ mód } \phi(n)$, o $e \equiv d^{-1} \text{ mód } \phi(n)$ O sea, e y d son inversos multiplicativos

$$\text{mód} \sim \phi(n)$$

Ahora el sistema RSA consiste de:

- Bob escoge p, q primos, y calcula $n = pq$, ej. $p = 7, q = 17, n = 119$
- Bob calcula $\phi(n) = (p-1)(q-1) = 96$
- Bob escoge $e < \phi(n)$ tal que $\gcd(e, \phi(n)) = 1$, ej. $e = 5$ (en la práctica e puede ser pequeño sin ningún problema).
- Bob calcula $d = e^{-1} \text{ mód } \phi(n) = 77$ (porque $77 \times 5 = 385 = 4 \times (96 + 1)$
- La clave pública de Bob es $5, 119$ y su clave privada es $77, 119$
- Alicia quiere mandar el mensaje $M = 19$. Encripta: $C = 19^5 \text{ mód } 119 = 66$
- Bob decripta:$M = 66^{77} \text{ mód } 119 = 19$

Seguridad de RSA

Para atacar a un sistema RSA, Manuel tiene varias opciones:

- Probar todas las claves posibles. *Pero*: Se supone que esto no es práctico (p y q tienen en el orden de cientos de dígitos c/u).
- Tratar de factorizar n . Esto le permite hallar p y q , y por lo tanto $\phi(n)$. Como ya conoce e , puede repetir el cálculo de Bob para determinar d *Pero*: Factorización *parece ser* difícil.
- Determinar $\phi(n)$ directamente, sin p y q , o determinar d directamente, sin $\phi(n)$.*Pero*: No se conoce ningún método.

Los esfuerzos para atacar a RSA se concentran en la factorización.

Lecciones Aprendidas Respecto a RSA

- Conocimiento de un par (e, d) para un n determinado permite al adversario factorizar n .
- El mismo conocimiento permite atacar a otros pares, sin factorizar n .
- Por lo tanto, no se debe usar un n común entre un grupo de usuarios.
- Es razonable usar valores pequeños para e , ej. 3, 17, 65537 (todos tienen sólo dos bits en 1, lo cual facilita la exponenciación). Sin embargo, RSA es vulnerable si $m^e \bmod n = m$. En estos casos los mensajes deben tener "relleno" (*padding*) con valores aleatorios.
- Escoger un valor grande para d .
- Algunos pares (p, q) pueden dar claves para las cuales $((\ldots(M^e \bmod n)^e \bmod n)\ldots)^e \bmod n = M$ después de pocas iteraciones. De hecho, para cualquier par (p, q) ¡existen al menos 9 mensajes tales que $M = M^e \bmod n$

7.3. Funciones de Hashing

Una función unidireccional de hash, $H(M)$ opera sobre un mensaje como preimagen y que tiene una longitud arbitraria M , retornando un valor de longitud fija o hash, h .

$$h = H(M)$$

donde h es de longitud m.

Las funciones de hash deben además tener las siguientes características:

1. Dado M, es fácil calcular h.
2. Dado h, es difícil calcular M tal que $H(M) = h$
3. Dado M es difícil encontrar otro mensaje, M^1 tal que $H(M) = H\left(M^1\right)$

En algunas ocasiones se requiere que la función de hash tenga una propiedad adicional llamada "resistencia a colisiones". Esta propiedad implica que es difícil encontrar dos mensajes aleatorios, M y $M^1 \ni H(M) = H\left(M^1\right)$

La aplicación fundamental de las funciones unidireccionales de hash es garantizar la autenticidad de un mensaje. Cada vez que Alice y Bob intercambian mensajes generan una "huella" a través de la función de hash. El receptor toma el mensaje y aplica la función de hash sobre el mensaje, si es idéntico al hash enviado entonces puede confiar en la autenticidad del mismo.

La mayoría de las funciones de hash producen un hash de 128 bits. Un hash de 128 bits requiere en promedio 2^{64} intentos para generar aleatoriamente dos mensajes diferentes que produzcan el mismo hash. En el mundo real las funciones de hash son construidas en base al concepto de funciones de compresión que son capaces de tomar

un mensaje de longitud m y reducirlo a un mensaje de longitud n . La entrada para la función de compresión son un bloque de mensaje de tamaño m y la salida producida del bloque de texto previo. El hash del bloque M_{i} es:

$$h_i = f\left(M_i, h_{i-1}\right)$$

7.3.1. SHA-I (Secure Hash Algorithm)

SHA produce un hash de 160 bits. Como primer paso el mensaje es rellenado para hacerlo múltiplo de 512 bits. El relleno se hace primero añadiendo un 1 y luego tantos ceros como sea necesario para hacerlo 64 bits mas corto que un múltiplo de 512, finalmente se le agrega una representación en 64 bits de la longitud del mensaje. Luego 5 variables de 32 bits de longitud son inicializadas de la siguiente forma:

$$A = 0x67452301$$

$$B = 0xEFCDAB89$$

$$C = 0x98BADCFE$$

$$D = 0x10325476$$

$$E = 0xC3D2E1F0$$

Luego comienza el lazo principal del algoritmo. Procesa 512 bits del mensaje cada vez y continua con todos los bloques de 512 bits que constituyan el mensaje. En primer lugar las cinco variables son copiadas en variables diferentes: A en a, B en b , C en c , D en d , E en e . El lazo principal tiene 4 rondas de 20 operaciones cada uno. Cada operación aplica una función no lineal sobre tres de las cinco variables, y luego se hacen sumas y rotaciones. El conjunto de funciones no lineales en SHA son:

$$ft\left(X, Y, Z\right) = \left(X \wedge Y\right) \vee \left(\left(\neg X\right) \wedge Z\right), t = 0.,19$$

$$f_t\left(X, Y, Z\right) = X \oplus Y \oplus Z, t = 20.,39$$

$$f_t\left(X, Y, Z\right) = \left(X \wedge Y\right) \vee \left(X \wedge Z\right) \vee \left(Y \wedge Z\right), t = 40.,59$$

$$f_t\left(X, Y, Z\right) = X \oplus Y \oplus Z, t = 60.,79$$

Cuatro constantes son usadas en el algoritmo:

$$K_t = 0x5A827999, t = 0.,19$$

$$K_t = 0x6ED9EBA1, t = 20.,39$$

$$K_t = 0x8F1BBCDC, t = 40.,59$$

$$K_t = 0xCA62C1D6, t = 60.,79$$

El bloque de mensaje es transformado de 16 palabras de 32 bits $(M_0..M_{15})$ a 80 palabras de 32 bits $(W_0..W_{79})$ usando el siguiente algoritmo:

$$W_t = M_t, t = 0.,15$$

$$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) <<< 1, t = 16.,79$$

Si t es el numero de la operación $(t = 0.,79)$, W_t representa el t th sub-bloque del mensaje expandido, y $<<<\ s$ representa un shift circular de s bits, entonces el lazo principal luce de la siguiente forma:
$FOR(t = 0.,79)$
$TEMP = (a <<< 5) + f_t\,(b, c, d) + e + W_t + K_t$
$e = d$
$d = c$
$c = b <<< 30$
$b = a$
$a = TEMP$
Al final de este proceso a, b, c, d, e son añadidos a A, B, C, D, E respectivamente y el algoritmo continua con el siguiente bloque de datos. La salida final es la concatenación de A, B, C, D, E

7.3.2. MD5

MD5 procesa el mensaje de entrada en bloques de 512 bits, dividiéndolos en 16 sub-bloques de 32 bits cada uno. La salida del algoritmo son 4 bloques de 32 bits que concatenados forman un hash de 128 bits. Como primer paso el mensaje es rellenado de tal forma que solo le falten 64 bits para ser un múltiplo de 512. Este relleno se hace añadiendo un 1 seguido de tantos 0's como sean necesarios. Al final se añade una representación en 64 bits de la longitud del mensaje. Luego se inicializan 4 variables de 32 bits cada una:

$$A = 0x01234567$$

$$B = 0x89ABCDEF$$

$$C = 0xFEDCBA98$$

$$D = 0x76543210$$

Se comienza entonces el lazo principal del algoritmo. El lazo se repite para cada bloque de 512 bits del texto que se deban procesar. La 4 variables se copian en 4 variables de trabajo $a = A, b = B, c = C, d = D$. El lazo principal tiene 4 rondas, todas muy similares. Cada ronda usa una operación diferente 16 veces. Cada operación aplica una función no lineal sobre tres de las 4 variables a, b, c, d y luego añade el resultado a la cuarta variable, un sub-bloque del texto y una constante. Luego se rota el resultado a la derecha un numero variable de posiciones y se añade el resultado a alguna de las variables a, b, c, d.

Hay 4 funciones no lineales que se aplican en cada ronda (una diferente en cada ronda):

$$F(X, Y, Z) = (X \wedge Y) \vee ((\neg X) \wedge Z)$$

$$G(X, Y, Z) = (X \wedge Z) \vee (Y \wedge (\neg Z))$$

$$H(X, Y, Z) = X \oplus Y \oplus Z$$

$$I(X, Y, Z) = Y \oplus (X \vee (\neg Z))$$

Estas funciones han sido diseñadas de tal forma que los bits correspondientes de X, Y y Z son independientes. Si M_j representa el jth sub-bloque del mensaje ($j = 0.,15$) y $<<< s$ una rotación circular de s bits, las cuatro operaciones son:

$$FF(a, b, c, d, M_j, s, t_i) \Leftrightarrow a = b + ((a + F(b, c, d) + M_j + t_i) <<< s)$$

$$GG(a, b, c, d, M_j, s, t_i) \Leftrightarrow a = b + ((a + G(b, c, d) + M_j + t_i) <<< s)$$

$$HH(a, b, c, d, M_j, s, t_i) \Leftrightarrow a = b + ((a + H(b, c, d) + M_j + t_i) <<< s)$$

$$II(a, b, c, d, M_j, s, t_i) \Leftrightarrow a = b + ((a + I(b, c, d) + M_j + t_i) <<< s)$$

Las constantes t_i son escogidas de la siguiente manera, en el paso i , t_i es la parte entera de $2^{32} * abs(\sin(i))$, donde i esta en radianes.

Después de todos estos pasos a, b, c, d son añadidos a A, B, C, D respectivamente. Se continua con el siguiente bloque y el resultado final es la concatenación de A, B, C, D.

7.3.3. Esquemas en los cuales la longitud del hash es igual al tamaño del bloque

El esquema general en este caso es como sigue:

$$H_0 = I_{H,}$$

$$H_i = E_A(B) \oplus C$$

siendo I_H un valor inicial aleatorio. Donde A, B y C pueden ser o M_i, H_{i-1}, $(M_i \oplus H_{i-1})$ o una constante. El mensaje es dividido en pedazos del tamaño del bloque a usar, M_i, y es procesado individualmente. Se usa generalmente un mecanismo para rellenar el bloque similar a los aplicados en MD5 o en SHA.

Debido al numero de variables y a que cada una solo puede tomar uno de 4 valores hay 64 esquemas de este tipo. De ellos 15 son débiles porque el resultado no depende de una de las entradas y otros 37 son triviales por razones mas sutiles. En total, luego de analizados, quedan solo 12 que son seguros. De esos 12 los primeros 4 son seguros contra todo tipo de ataque. Los 4 esquemas mas seguros son los que se muestran a continuación:

1. $H_i = E_{H_{i-1}}(M_i) \oplus M_i$
2. $H_i = E_{H_{i-1}}(M_i \oplus H_{i-1}) \oplus M_i \oplus H_{i-1}$
3. $H_i = E_{H_{i-1}}(M_i) \oplus H_{i-1} \oplus M_i$
4. $H_i = E_{H_{i-1}}(M_i \oplus H_{i-1}) \oplus M_i$

7.4. La Vida Real

7.4.1. Problemas con Sistemas de Clave Pública

Los sistemas asimétricos tienen varios problemas:

- Son lentos. Los simétricos son 3 ordenes de magnitud más rápidos.
- Son vulnerables a ataques “texto escogido”. En casos extremos el adversario pueden encriptar todo posible texto original y comparar. Ej. si el texto consiste de un número entre 1 y 5000, esto puede ser muy efectivo. ¿Qué podemos hacer para aumentar la dificultad?

Debido a la lentitud, usualmente se usan claves públicas para acordar una **clave de sesión**, la cual se utiliza con un sistema simétrico.

7.4.2. Longitud de Claves

La seguridad de un criptosistema simétrico depende de:

- El algoritmo
- La longitud de la clave

Un criterio para evaluar el algoritmo es: *no hay mejor manera de romper el sistema que el método de fuerza bruta*.

Un ataque de fuerza bruta es de la clase de ataques "texto conocido". Ej. el criptosistema es de bloque y se ataca un bloque que contiene el encriptamiento de *login:*. Si el algoritmo es fuerte, y la clave es de N bits, el adversario necesitará en promedio 2^{N-1} operaciones para encontrarla, donde cada operación es una encripción.

Si una supercomputadora puede realizar 10^6 intentos por segundo, una clave de 64 bits implica 585000 años de trabajo. Si la clave es de 128 bits, son 10^{25} años (la edad del universo es de 10^{10} años.

Sin embargo, en la práctica muchas aplicaciones usan el sistema DES, con claves de 56 bits. Se ha estimado que en 1996 una máquina especializada podría romper una clave DES en 3.5 horas. La máquina costaría 1millón $ hoy, pero en 10 años costaría quizás $100.000.

7.4.3. Sistemas Basados en Problemas Matemáticos

Muchos sistemas modernos basan su seguridad en la supuesta dificultad de un problema matemático, ej. factorización para RSA, logaritmos discretos para Diffie-Hellman, etc. Es importante notar que hasta ahora ninguno de estos problemas tiene dificultad *conocida*.

> '*La factorización de un número de 125 dígitos tardaría 40 quadrillones de años.*" – Ron Rivest, 1977

En 1994, se factorizó un número de 129 dígitos (el RSA-129), usando 1600 máquinas en el Internet. Esto representaba alrededor de 0.03 % del poder computacional del Internet en su momento.

(Curiosamente, cada avance en factorización es acompañado por uno similar en el cálculo de logaritmos discretos; no está claro por qué.)

El método utilizado para RSA-129 se llama la *Criba Cuadrática*. Ha sido superado por la *Criba de Cuerpo Numérico* (NFS), el cual es un método sub-exponencial.

NFS consiste de dos fases: la primera subdivide el problema en muchas partes que pueden ser atacados en paralelo; la segunda procesa una matriz grande cuyos elementos son los resultados de la primera fase, y es difícil paralelizar. En particular, esta fase es muy sensible a la velocidad de la memoria, y menos a la velocidad básica del CPU, sobre todo porque los patrones de acceso son irregulares y los *caches* no ayudan mucho.

Actualmente el *NFS* puede factorizar un número de 80 dígitos en un computador portátil, en un día de trabajo. Un número de 130 dígitos necesitaría unos 700 MIPS-años para la primera fase, y 68 horas de un Cray C-90 para la segunda.

Para un número de 512 bits se estima 20.000 MIPS-años para Fase 1, y 3 meses del Cray para procesar una matriz de 4 Gb de memoria.

Para 1024 bits, serian 10^{11} MIPS-años para Fase 1, y Fase 2 sería totalmente imposible.

También hay trabajo especulativo sobre el uso de moléculas de ADN, o sobre una máquina basada en la mecánica cuántica.

7.4.4. Números Aleatorios

En muchas aplicaciones de la criptografía, la seguridad depende de que ciertos elementos (ej. claves) sean *aleatorias*. Para que un número o secuencia de bits sea aleatoria, debe generarse a partir de una fuente natural de entropía:

- Un proceso radiactivo
- El voltaje a través de un diodo invertido (ej. el Pentium III)
- #cat /dev/audio | gzip -dc >random#

En la práctica, puede ser difícil acceder a estos fuentes, y tenemos que utilizar procesos determinísticos llamados **PRNG** (*Pseudo-Random Number Generators*) o **PRBG** (*Pseudo-Random Bitsream Generators*).

Las PRNG o PRBG tradicionales fueron desarrollados para el área de la simulación de procesos, donde lo que importa es el comportamiento estadístico. En aplicaciones criptográficas, se agrega el requisito que el producto del proceso no sea sujeto a extrapolación. Por ejemplo, si usamos un PRBG tradicional llamado el *Linear Congruential Generator*, y Manuel logra encontrar parte de la secuencia de bits que produce, puede fácilmente "sincronizarse" con el generador y seguir produciendo los mismos bits que Alicia.

El Generador BBS

Uno de los PRBG más populares y seguros es el de Blum, Blum y Shub, llamado **BBS**:

Sean p y q dos primos tal que $p \equiv q \equiv 3 \bmod 4$, y $n = pq$. Escoger como *semilla* s_0, cualquier entero relativamente primo a n. Para $i \geq 0$, definamos: $s_{i+1} = s_i^2 \bmod n$ y $z_i = s_i \bmod 2 = (s_o^{2^i} \bmod n) \bmod n$ La secuencia de bits z_i (derivada de los bits menos significativos de los s_i) es pseudo-aleatoria con alto grado de seguridad. Es más, se puede mejorar la eficiencia sin menoscabo a la seguridad, tomando m bits menos significativos a la vez, donde $m \leq \log_2 \log_2 n$.

También se puede mostrar que $s_i = s_0^{2^i \bmod ((p-1)(q-1))}$. Esto significa que se puede calcular s_i sin tener que calcular todos los anteriores, lo cual es útil para aplicaciones en las cuales se necesita acceso no-secuencial a los datos, ej. en un archivo o base de datos. (Nótese que sólo necesitamos algunos de los bits menos significativos de s_i.)

Bibliografía

[1] JX Yan, S Early, R Anderson, A Grant, “The Memorability and Security of Passwords-Some Empirical Results”, University of Cambridge Computer Laboratory Technical Report No 500; at http://www.cl.cam.ac.uk/ftp/users/rja14/tr500.pdf

[1] T Greening, “Ask and Ye Shall Receive: A Study in Social Engeenering”, in SIGSAC Review, v14 N 2 (Apr 1996), pp 173-182

[2] RL Barnard, Intrusion Detection Systems, Butterworths (1988)

[3] M Kam, G Fielding, R Conn, “Writer Identification by Professional Document Examiners”, in Journal of Foresinc Sciences, v 42 (1997), pp 778-786

[4] R Kemp, N Towell, G Pike, “When Seeing Should not be believing: Photopraphs, Credit Cards and Frauds”, in Applied Cognitive Psychology, v 11 Nro 3 (1997) pp 211-222

[5] D Maio, D Maltoni, “Direct Gray-Scale Minutiae Detection in Fingerprints,” in IEEE Transactions on Pattern Analysis and Machine Intelligence, v 19 no 1 (Jan 1997), pp 27-40

[6] AK Jain, L Hong, S Pankanti, R Bolle, “An Identity-Authentification System Using Fingerprints”, in Proceedings of the IEEE, v 85, n 9 (Sept 1997), pp 1365-1388

[7] I Drury, “Pointing the Finger”, in Security Surveyor, v 27 no 5 (Jan 1997), pp 15-17.

[8] L Lamport, R Shostack, M Peace, “The Byzantine Generals-Problem,” in ACM Transactions on Programming Languages and Systems, v4 no 3 (1982), pp 382-401

[9] RM Needham, “Naming”, in Distributed Systems, Addison-Wesley (1993), pp 318-327

[10] Bruce Schneier, “Applied Criptography”, Second Edition. John Wiley and Sons.

[11] Ross Anderson, “Security Engineering: A guide to building dependable distributed systems”, (2001), John Wiley and Sons.

[12] Simson Garfinkel, “Web Security, Privacy and Commerce”, (1997), O’reilly.

[13] Bruce Schneier, “Secret and Lies”.

[14] M Burrows, M Abadi, RM Needham, “A Logic of authetification,” in Proceeding of the Royal Society of London A, v 426 (1989), pp 233-271.

[15] Bugtraq: Lista de correo con temas de seguridad. http://www.securityfocus.com