Autenticación de usuarios

Siguiente: Algunos sistemas Unix **Subir:** Seguridad del sistema **Anterior:** Copias de seguridad **Índice General**

Subsecciones

Autenticación de usuarios

Introducción y conceptos básicos

Ya sabemos que unos requerimientos primor diales de los sistemas informáticos que desempeñan tareas importantes son los mecanismo de seguridad adecuados a la información que se intenta proteger; el conjunto de tales mecanismos ha de incluir al menos un sistema que permita identificar a las entidades (elementos activos del sistema, generalmente usuarios) que intentan acceder a los objetos (elementos pasivos, como ficheros o capacidad de cómputo), mediante procesos tan simples como una contraseña o tan complejos como un dispositivo analizador de patrones retinales.

Los sistemas que habitualmente utilizamos los humanos para identificar a una persona, como el aspecto físico o la forma de hablar, son demasiado complejos para una computadora; el objetivo de los sistemas de identificación de usuarios no suele ser **identificar** a una persona, sino **autenticar** que esa persona es quien dice ser realmente. Aunque como humanos seguramente ambos términos nos parecerán equivalentes, para un ordenador existe una gran diferencia entre ellos: imaginemos un potencial sistema de identificación estrictamente hablando, por ejemplo uno biométrico basado en el reconocimiento de la retina; una persona miraría a través del dispositivo lector , y el sistema sería capaz de decidir si es un usuario válido, y en ese caso decir de quién se trata; esto es identificación. Sin embargo, lo que habitualmente hace el usuario es introducir su identidad (un número, un nombre de usuario...) además de mostrar sus retinas ante el lector; el sistema en este caso no tiene que identificar a esa persona, sino autenticarlo: comprobar los parámetros de la retina que está leyendo con los guardados en una base de datos para el usuario que la persona dice ser: estamos reduciendo el problema de una población potencialmente muy elevada a un grupo de usuarios más reducido, el grupo de usuarios del sistema que necesita autenticarlos.

Los métodos de autenticación se suelen dividir en tres grandes categorías ([DP84], [Eve92]), en función de lo que utilizan para la verificación de identidad: (a) algo que el usuario sabe, (b) algo que éste posee, y (c) una característica física del

usuario o un acto involuntario del mismo. Esta última categoría se conoce con el nombre de **autenticación biométrica**. Es fácil ver ejemplos de cada uno de estos tipos de autenticación: un *password* (Unix) o *passphrase* (PGP) es algo que el usuario conoce y el resto de personas no, una tarjeta de identidad es algo que el usuario lleva consigo, la huella dactilar es una característica física del usuario, y un acto involuntario podría considerarse que se produce al firmar (al rubricar la firma no se piensa en el diseño de cada trazo individualmente). Por supuesto, un sistema de autenticación puede (y debe, para incrementar su fiabilidad) combinar mecanismos de diferente tipo, como en el caso de una tarjeta de crédito junto al PIN a la hora de utilizar un cajero automático o en el de un dispositivo generador de claves para el uso de *One Time Passwords*.

Cualquier sistema de identificación (aunque les llamemos así, recordemos que realmente son sistemas de autenticación) ha de poseer unas determinadas características para ser viable; obviamente, ha de ser fiable con una probabilidad muy elevada (podemos hablar de tasas de fallo de en los sistemas menos seguros), económicamente factible para la organización (si su precio es superior al valor de lo que se intenta proteger , tenemos un sistema incorrecto) y ha de soportar con éxito cierto tipo de ataques (por ejemplo, imaginemos que cualquier usuario puede descifrar el *password* utilizado en el sistema de autenticación de Unix en tiempo polinomial; esto sería inaceptable). Aparte de estas características tenemos otra, no técnica sino humana, pero quizás la más importante: un sistema de autenticación ha de ser aceptable para los usuarios ([Tan91]), que serán al fin y al cabo quienes lo utilicen. Por ejemplo, imaginemos un potencial sistema de identificación para acceder a los recursos de la Universidad, consistente en un dispositivo que fuera capaz de realizar un análisis de sangre a un usuario y así comprobar que es quien dice ser; seguramente sería barato y altamente fiable, pero nadie aceptaría dar un poco de sangre cada vez que desee consultar su correo.

Sistemas basados en algo conocido: contraseñas

El modelo de autenticación más básico consiste en decidir si un usuario es quien dice ser simplemente basándonos en una prueba de conocimiento que *a priori* sólo ese usuario puede superar; y desde Alí Babá y su *`Ábrete, Sésamo'* hasta los más modernos sistemas Unix, esa prueba de conocimiento no es más que una contraseña que en principio es secreta. Evidentemente, esta aproximación es la más vulnerable a todo tipo de ataques, pero también la más barata, por lo que se convierte en la técnica más utilizada en entornos que no precisan de una alta seguridad, como es el caso de los sistemas Unix en redes normales (y en general en todos los sistemas operativos en redes de seguridad media-baja); otros entornos en los que se suele aplicar este modelo de autenticación son las aplicaciones que requieren de alguna identificación de usuarios, como el *software* de cifrado PGP o el escáner de seguridad NESSUS. También se utiliza como complemento a otros mecanismos de autenticación, por ejemplo en el caso del Número de Identificación Personal (PIN) a la hora de utilizar cajeros automáticos.

En todos los esquemas de autenticación basados en contraseñas se cumple el mismo protocolo: las entidades (generalmente dos) que participan en la autenticación acuerdan una clave, clave que han de mantener en secreto si desean que la autenticación sea fiable. Cuando una de las partes desea autenticarse ante otra se limita a mostrarle su conocimiento de esa clave común, y si ésta es correcta se otorga el acceso a un recurso. Lo habitual es que existan unos roles preestablecidos, con una entidad activa que desea autenticarse y otra pasiva que admite o rechaza a la anterior (en el modelo del acceso a sistemas Unix, tenemos al usuario y al sistema que le permite o niega la entrada).

Como hemos dicho, este esquema es muy frágil: basta con que una de las partes no mantenga la contraseña en secreto para que toda la seguridad del modelo se pierda; por ejemplo, si el usuario de una máquina Unix comparte su clave con un tercero, o si ese tercero consigue leerla y rompe su cifrado (por ejemplo, como veremos luego, mediante un ataque de diccionario), automáticamente esa persona puede autenticarse ante el sistema con éxito con la identidad de un usuario que no le corresponde.

En el punto 8.5 hablaremos con más detalle del uso de contraseñas para el caso de la autenticación de usuarios en Unix.

Sistemas basados en algo poseído: tarjetas inteligentes

Hace más de veinte años un periodista francés llamado Roland Moreno patentaba la integración de un procesador en una tarjeta de plástico; sin duda, no podía imaginar el abanico de aplicaciones de seguridad que ese nuevo dispositivo, denominado *chipcard*, estaba abriendo. Desde entonces, cientos de millones de esas tarjetas han sido fabricadas, y son utilizadas a diario para fines que varían desde las tarjetas monedero más sencillas hasta el control de accesos a instalaciones militares y agencias de inteligencia de todo el mundo; cuando a las *chipcards* se les incorporó un procesador inteligente nacieron las *smartcards*, una gran revolución en el ámbito de la autenticación de usuarios.

Desde un punto de vista formal ([GUQ92]), una tarjeta inteligente (o *smartcard*) es un dispositivo de seguridad del tamaño de una tarjeta de crédito, resistente a la adulteración, que ofrece funciones para un almacenamiento seguro de información y también para el procesamiento de la misma en base a tecnología VLSI. En la práctica, las tarjetas inteligentes poseen un chip empotrado en la propia tarjeta que puede implementar un sistema de ficheros cifrado y funciones criptográficas, y además puede detectar activamente intentos no válidos de acceso a la información almacenada ([MA94]); este chip inteligente es el que las diferencia de las simples tarjetas de crédito, que sólamente incorporan una banda magnética donde va almacenada cierta información del propietario de la tarjeta.

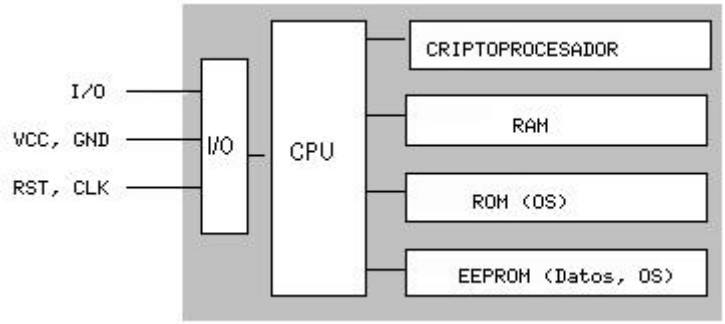

Figura 8.1: Estructura genérica de una *smartcard*.

En la figura 8.1 se muestra la estructura más generalista de una tarjeta inteligente; en ella podemos observar que el acceso a las áreas de memoria sólamente es posible a través de la unidad de entrada/salida y de una CPU (típicamente de 8 bits), lo que evidentemente aumenta la seguridad del dispositivo. Existe también un sistema operativo empotrado en la tarjeta - generalmente en ROM, aunque también se puede extender con funciones en la EEPROM - cuya función es realizar tareas criptográficas (algoritmos de cifrado como RSA o T riple DES, funciones re sumen...); el criptoprocesador apoya estas tareas ofreciendo operaciones RSA con claves de 512 a 1024 bits. Un ejemplo de implementación real de este esquema lo constituye la tarjeta inteligente CERES, de la Fábrica Nacional de Moneda y T imbre española ([Pit00]); en ella se incluye además un generador de números aleatorios junto a los mecanismos de protección internos de la tarjeta.

Cuando el usuario poseedor de una *smartcard* desea autenticarse necesita introducir la tarjeta en un *hardware* lector; los dos dispositivos se identifican entre sí con un protocolo a dos bandas en el que es necesario que ambos conozcan la misma clave (CK o CCK, *Company Key* o *Chipcard Communication Key*), lo que elimina la posibilidad de utilizar tarjetas de terceros para autenticarse ante el lector de una determinada compañía; además esta clave puede utilizarse para asegurar la comunicación entre la tarjeta y el dispositivo lector . Tras identificarse las dos p artes, se lee la identificación personal (PID) de la tarjeta, y el usuario teclea su PIN; se inicia entonces un protocolo desafío-respuesta: se envía el PID a la máquina y ésta desafía a la tarjeta, que responde al desafío utilizando una clave personal del usuario (PK, *Personal Key*). Si la respuesta es correcta, el *host* ha identificado la tarjeta y el usuario obtiene acceso al recurso pretendido.

Las ventajas de utilizar tarjetas inteligentes como medio para autenticar usuarios son muchas frente a las desventajas; se trata de un modelo ampliamente aceptado entre los usuarios, rápido, y que incorpora *hardware* de alta seguridad tanto para almacenar datos como para realizar funciones de cifrado. Además, su uso es factible tanto para controles de acceso físico como para controles de acceso lógico a los *hosts*, y se integra fácilmente con otros mecanismos de autenticación como las contraseñas; y en caso de desear bloquear el acceso de un usuario, no tenemos más que retener su tarjeta cuando la introduzca en el lector o marcarla como inválida en una base de datos (por ejemplo, si se equivoca varias veces al teclar su PIN, igual que sucede con una tarjeta de crédito normal). Como principal inconveniente de las *smartcards* podemos citar el coste adicional que supone para una organización el comprar y configurar la infraestructura de dispositivos lectores y las propias tarjetas; aparte, que un usuario pierda su tarjeta es bastante fácil, y durante el tiempo que no disponga de ella o no puede acceder al sistema, o hemos de establecer reglas especiales que pueden comprometer nuestra seguridad (y por supuesto se ha de marcar como tarjeta inválida en una base de datos central, para que un potencial atacante no pueda utilizarla). T ambién la distancia lógica entre la *smartcard* y su poseedor - simplemente nos podemos fijar en que la tarjeta no tiene un interfaz para el usuario - puede ser fuente de varios problemas de seguridad ([BF99], [GSTY96]).

Aparte de los problemas que puede implicar el uso de *smartcards* en sí, contra la lógica de una tarjeta inteligente existen diversos métodos de ataque, como realizar ingeniería inversa - destructiva - contra el circuito de silicio (y los contenidos de la ROM), adulterar la información guardada en la tarjeta o determinar por diferentes métodos el contenido de la memoria EEPROM. Sin duda una de las personas que más ha contribuido a incrementar la seguridad de las *smartcards* gracias a sus estudios y ataques es el experto británico Ross J. Anderson ([And97], [AK96]...); en su página *web* personal, `http://www.cl.cam.ac.uk/users/rja14`, podemos encontrar todos sus artículos sobre este tema [9.1], demasiados como para citarlos aquí uno a uno.

Sistemas de autenticación biométrica

A pesar de la importancia de la criptología en cualquiera de los sistemas de identificación de usuarios vistos, existen otra clase de sistemas en los que no se aplica esta ciencia, o al menos su aplicación es secundaria. Es más, parece que en un futuro no muy lejano estos serán los sistemas que se van a imponer en la mayoría de situaciones en las que se haga necesario autenticar un usuario: son más amigables para el usuario (no va a necesitar recordar *passwords* o números de identificación complejos, y , como se suele decir, el usuario puede olvidar una tarje ta de identificación en casa, pero nunca se olvidará de su mano o su ojo) y son mucho más difíciles de falsificar que una simple contraseña o una tarjeta magnética; las principales razones por la que no se han impuesto ya en nuestros dias es su elevado precio, fuera del alcance de muchas organizaciones, y su dificultad de mantenimiento ([GKK97]).

Estos sistemas son los denominados **biométricos**, basados en características físicas del usuario a identificar . El reconocimiento de formas, la inteligencia artificial y el aprendizaje son las ramas de la informática que desempeñan el papel más importante en los sistemas de identificación biométricos; la criptología se limita aquí a un uso secundario, como el cifrado de una base de datos de patrones retinales, o la transmisión de una huella dactilar entre un dispositivo analizador y una base de datos. La autenticación basada en características físicas existe desde que existe el hombre y,

sin darnos cuenta, es la que más utiliza cualquiera de nosotros en su vida cotidiana: a diario identificamos a personas por los rasgos de su cara o por su voz. Obviamente aquí el agente reconocedor lo tiene fácil porque es una persona, pero en el modelo aplicable a redes o sistemas Unix el agente ha de ser un dispositivo que, basándose en características del sujeto a identificar, le permita o deniegue acceso a un determinado recurso.

Tabla 8.1: Comparación de métodos biométricos.

	Ojo - Iris	Ojo - Retina	Huellas dactilares	Geometría de la mano	Escritura - Firma	Voz
Fiabilidad	Muy alta	Muy alta	Alta	Alta	Alta	Alta
Facilidad de uso	Media	Baja	Alta	Alta	Alta	Alta
Prevención de ataques	Muy Alta	Muy alta	Alta	Alta	Media	Media
Aceptación	Media	Media	Media	Alta	Muy alta	Alta
Estabilidad	Alta	Alta	Alta	Media	Media	Media
Identificación y autenticación	Ambas	Ambas	Ambas	Autenticación	Ambas	Autenticación
Estándars	-	-	ANSI/NIST, FBI	-	-	SVAPI
Interferencias	Gafas	Irritaciones	Suciedad, heridas, asperezas ...	Artritis, reumatismo ...	Firmas fáciles o cambiantes	Ruido, resfriados ...
Utilización	Instalaciones nucleares, servicios médicos, centros penitenciarios	Instalaciones nucleares, servicios médicos, centros penitenciarios	Policía, industrial	General	Industrial	Accesos remotos en bancos o bases de datos
Precio por nodo en 1997 (USD)	5000	5000	1200	2100	1000	1200

Aunque la autenticación de usuarios mediante métodos biométricos es posible utilizando cualquier característica única y mesurable del individuo (esto incluye desde la forma de teclear ante un ordenador hasta los patrones de ciertas venas, pasando por el olor corporal), tradicionalmente ha estado basada en cinco grandes grupos ([Eve92]). En la tabla 8.1 ([Huo98], [Phi97]) se muestra una comparativa de sus rasgos más generales, que vamos a ver con más detalle en los puntos siguientes.

Los dispositivos biométricos tienen tres partes principales; por un lado, disponen de un mecanismo automático que lee y captura una imagen digital o analógica de la característica a analizar . Además disponen de una entidad para manejar aspectos como la compresión, almacenamiento o comparación de los datos capturados con los guardados en una base de datos (que son considerados válidos), y también ofrecen una interfaz para las aplicaciones que los utilizan. El proceso general de autenticación sigue unos pasos comunes a todos los modelos de autenticación biométrica: **captura** o lectura de los datos que el usuario a validar presenta, **extracción** de ciertas características de la muestra (por ejemplo, las minucias de una huella dactilar), **comparación** de tales características con las guardadas en una base de datos, y **decisión** de si el usuario es válido o no. Es en esta decisión donde principalmente entran en juego las dos características básicas de la fiabilidad de todo sistema biométrico (en general, de todo sistema de autenticación): las tasas de falso rechazo y de falsa aceptación. Por tasa de **falso rechazo** (*False Rejection Rate* , FRR) se entiende la probabilidad de que el sistema de autenticación rechaze a un usuario legítimo porque no es capaz de identificarlo correctamente, y por tasa de **falsa aceptación** (*False Acceptance Rate*, FAR) la probabilidad de que el sistema autentique correctamente a un usuario ilegítimo; evidentemente, una FRR alta provoca descontento entre los usuarios del sistema, pero una FAR elevada genera un grave problema de seguridad: estamos proporcionando acceso a un recurso a personal no autorizado a acceder a él.

Por último, y antes de entrar más a fondo con los esquemas de autenticación biométrica clásicos, quizás es conveniente desmentir uno de los grandes mitos de estos modelos: la vulnerabilidad a ataques de simulación. En cualquier película o libro de espías que se precie, siempre se consigue `engañar' a autenticadores biométricos para conseguir acceso a determinadas instalaciones mediante estos ataques: se simula la parte del cuerpo a analizar mediante un modelo o incluso utilizando órganos amputados a un cadáver o al propio usuario vivo (crudamente, se le corta una mano o un dedo, se le saca un ojo...para conseguir que el sistema permita la entrada). Evidentemente, esto sólo sucede en la ficción: hoy en día cualquier sistema biométrico - con excepción, quizás, de algunos modelos basados en voz de los que

hablaremos luego - son altamente inmunes a estos ataques. Los analizadores de retina, de iris, de huellas o de la geometría de la mano son capaces, aparte de decidir si el miembro pertenece al usuario legítimo, de determinar si éste está vivo o se trata de un cadáver.

Verificación de voz

En los sistemas de reconocimiento de voz no se intenta, como mucha gente piensa, reconocer lo que el usuario dice, sino identificar una serie de sonidos y sus características para decidir si el usuario es quien dice ser. Para autenticar a un usuario utilizando un reconocedor de voz se debe disponer de ciertas condiciones para el correcto registro de los datos, como ausencia de ruidos, reverberaciones o ecos; idealmente, estas condiciones han de ser las mismas siempre que se necesite la autenticación.

Cuando un usuario desea acceder al sistema pronunciará unas frases en las cuales reside gran parte de la seguridad del protocolo; en algunos modelos, los denominados de texto dependiente, el sistema tiene almacenadas un conjunto muy limitado de frases que es capaz de reconocer: por ejemplo, imaginemos que el usuario se limita a pronunciar su nombre, de forma que el reconocedor lo entienda y lo autentique. Como veremos a continuación, estos modelos proporcionan poca seguridad en comparación con los de texto independiente, donde el sistema va `proponiendo' a la persona la pronunciación de ciertas palabras extraídas de un conjunto bastante grande. De cualquier forma, sea cual sea el modelo, lo habitual es que las frases o palabras sean características para maximizar la cantidad de datos que se pueden analizar (por ejemplo, frases con una cierta entonación, pronunciación de los diptongos, palabras con muchas vocales...). Conforme va hablando el usuario, el sistema registra toda la información que le es útil; cuando termina la frase, ya ha de estar en disposición de facilitar o denegar el acceso, en función de la información analizada y contrastada con la de la base de datos.

El principal problema del reconocimiento de voz es la inmunidad frente a *replay attacks*, un modelo de ataques de simulación en los que un atacante reproduce (por ejemplo, por medio de un magnetófono) las frases o palabras que el usuario legítimo pronuncia para acceder al sistema. Este problema es especialmente grave en los sistemas que se basan en textos preestablecidos: volviendo al ejemplo anterior, el del nombre de cada usuario, un atacante no tendría más que grabar a una persona que pronuncia su nombre ante el autenticador y luego reproducir ese sonido para conseguir el acceso; casi la única solución consiste en utilizar otro sistema de autenticación junto al reconocimiento de voz. Por contra, en modelos de texto independiente, más interactivos, este ataque no es tan sencillo porque la autenticación se produce realmente por una especie de desafío-respuesta entre el usuario y la máquina, de forma que la cantidad de texto grabado habría de ser mucho mayor - y la velocidad para localizar la parte del texto que el sistema propone habría de ser elevada -. Otro grave problema de los sistemas basados en reconocimiento de voz es el tiempo que el usuario emplea hablando delante del analizador, al que se añade el que éste necesita para extraer la información y contrastarla con la de su base de datos; aunque actualmente en la mayoría de sistemas basta con una sola frase, es habitual que el usuario se vea obligado a repetirla porque el sistema le deniega el acceso (una simple congestión hace variar el tono de voz, aunque sea levemente, y el sistema no es capaz de decidir si el acceso ha de ser autorizado o no; incluso el estado anímico de una persona varía su timbre...). A su favor, el reconocimiento de voz posee la cualidad de una excelente acogida entre los usuarios, siempre y cuando su funcionamiento sea correcto y éstos no se vean obligados a repetir lo mismo varias veces, o se les niegue un acceso porque no se les reconoce correctamente.

Verificación de escritura

Aunque la escritura (generalmente la firma) no es una característica estrictamente biométrica, como hemos comentado en la introducción se suele agrupar dentro de esta categoría; de la misma forma que sucedía en la verificación de la voz, el objetivo aquí no es interpretar o entender lo que el usuario escribe en el lector, sino autenticarlo basándose en ciertos rasgos tanto de la firma como de su rúbrica.

La verificación en base a firmas es algo que todos utilizamos y aceptamos día a día en documentos o cheques; no obstante, existe una diferencia fundamental entre el uso de las firmas que hacemos en nuestra vida cotidiana y los sistemas biométricos; mientras que habitualmente la verificación de la firma consiste en un simple análisis visual sobre una impresión en papel, estática, en los sistemas automáticos no es posible autenticar usuarios en base a la representación de los trazos de su firma. En los modelos biométricos se utiliza además la forma de firmar, las características dinámicas (por eso se les suele denominar *Dynamic Signature Verification*, DSV): el tiempo utilizado para rubricar, las veces que se separa el bolígrafo del papel, el ángulo con que se realiza cada trazo...

Para utilizar un sistema de autenticación basado en firmas se solicita en primer lugar a los futuros usuarios un número determinado de firmas ejemplo, de las cuales el sistema extrae y almacena ciertas características; esta etapa se denomina de *aprendizaje*, y el principal obstáculo a su correcta ejecución son los usuarios que no suelen firmar uniformemente. Contra este problema la única solución (aparte de una concienciación de tales usuarios) es relajar las restricciones del sistema a la hora de *aprender* firmas, con lo que se decrementa su seguridad.

Una vez que el sistema conoce las firmas de sus usuarios, cuando estos desean acceder a él se les solicita tal firma, con un número limitado de intentos (generalmente más que los sistemas que autentican mediante contraseñas, ya que la firma puede variar en un individuo por múltiples factores). La firma introducida es capturada por un lápiz óptico o por una lectora sensible (o por ambos), y el acceso al sistema se produce una vez que el usuario ha introducido una firma que el verificador es capaz de distinguir como auténtica.

Verificación de huellas

Típicamente la huella dactilar de un individuo ha sido un patrón bastante bueno para determinar su identidad de forma inequívoca, ya que está aceptado que dos dedos nunca poseen huellas similares, ni siquiera entre gemelos o entre dedos de la misma persona. Por tanto, parece obvio que las huellas se convertirían antes o después en un modelo de autenticación biométrico: desde el siglo pasado hasta nuestros días se vienen realizando con éxito clasificaciones sistemáticas de huellas dactilares en entornos policiales, y el uso de estos patrones fué uno de los primeros en establecerse como modelo de autenticación biométrica.

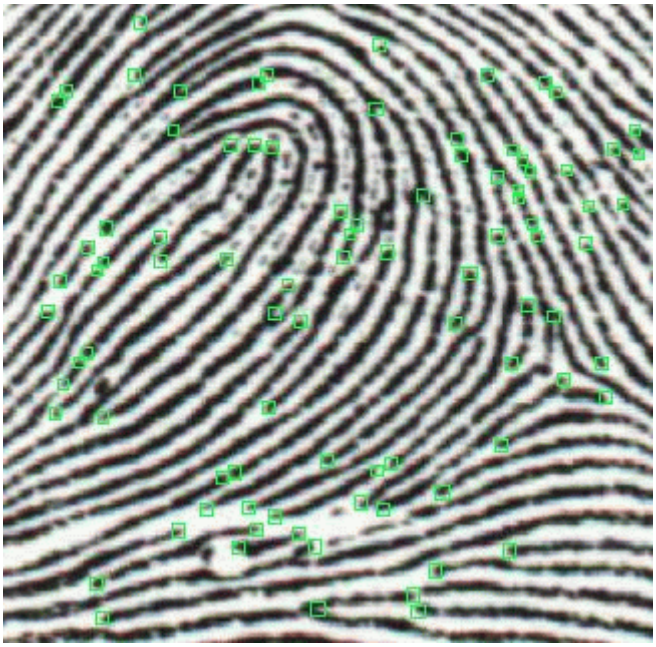

Figura 8.2: Huella dactilar con sus minucias extraídas.
©1998 Idex AS, http://www.idex.no/

Cuando un usuario desea autenticarse ante el sistema situa su dedo en un área determinada (área de lectura, no se necesita en ningún momento una impresión en tinta). Aquí se toma una imagen que posteriormente se normaliza mediante un sistema de finos espejos [9.2] para corregir ángulos, y es de esta imagen normalizada de la que el sistema extrae las minucias (ciertos arcos, bucles o remolinos de la huella) que va a comparar contra las que tiene en su base de datos; es importante resaltar que lo que el sistema es capaz de analizar no es la huella en sí sino que son estas minucias, concretamente la posición relativa de cada una de ellas. Está demostrado que dos dedos nunca pueden poseer más de ocho minucias comunes, y cada uno tiene al menos 30 o 40 de éstas (en la figura 8.2 podemos ver una imagen de una huella digitalizada con sus minucias). Si la comparación de las posiciones relativas de las minucias leídas con las almacenadas en la base de datos es correcta, se permite el acceso al usuario, denegándosele obviamente en caso contrario.

Los sistemas basados en reconocimiento de huellas son relativamente baratos (en comparación con otros biométricos, como los basados en patrones retinales); sin embargo, tienen en su contra la incapacidad temporal de autenticar usuarios que se hayan podido herir en el dedo a reconocer (un pequeño corte o una quemadura que afecte a varias minucias pueden hacer inútil al sistema). T ambién elementos como la suciedad del dedo, la presión ejercida sobre el lector o el estado de la piel pueden ocasionar lecturas erróneas. Otro factor a tener muy en cuenta contra estos sistemas es psicológico, no técnico: hemos dicho en la introducción que un sistema de autenticación de usuarios ha de ser aceptable por los mismos, y generalmente el reconocimiento de huellas se asocia a los criminales, por lo que muchos usuarios recelan del reconocedor y de su uso ([vKPG97]).

Verificación de patrones oculares

Los modelos de autenticación biométrica basados en patrones oculares se dividen en dos tecnologías diferentes: o bien analizan patrones retinales, o bien analizan el iris. Estos métodos se suelen considerar los más efectivos: para una población de 200 millones de potenciales usuarios la probabilidad de coincidencia es casi 0, y además una vez muerto el individuo los tejidos oculares degeneran rápidamente, lo que dificulta la falsa aceptación de atacantes que puedan robar este órgano de un cadáver.

La principal desventaja de los métodos basados en el análisis de patrones oculares es su escasa aceptación; el hecho de mirar a través de un binocular (o monocular), necesario en ambos modelos, no es cómodo para los usuarios, ni aceptable para muchos de ellos: por un lado, los usuarios *no se fían* de un haz de rayos analizando su ojo [9.3], y por otro un examen de este órgano puede revelar enfermedades o características médicas que a muchas personas les puede interesar mantener en secreto, como el consumo de alcohol o de ciertas drogas. Aunque los fabricantes de dispositivos lectores aseguran que sólo se analiza el ojo para obtener patrones relacionados con la autenticación, y en ningún caso se viola la privacidad de los usuarios, mucha gente no cree esta postura oficial (aparte del hecho de que la información es procesada vía *software*, lo que facilita introducir modificaciones sobre lo que nos han vendido para que un lector realice otras tareas de forma enmascarada). Por si esto fuera poco, se trata de sistemas demasiado caros para la mayoría de organizaciones, y el proceso de autenticación no es todo lo rápido que debiera en poblaciones de usuarios elevadas. De esta forma, su uso se ve reducido casi sólo a la identificación en sistemas de alta seguridad, como el control de acceso a instalaciones militares.

Retina

La vasculatura retinal (forma de los vasos sanguíneos de la retina humana) es un elemento característico de cada individuo, por lo que numerosos estudios en el campo de la autenticación de usuarios se basan en el reconocimiento de esta vasculatura.

En los sistemas de autenticación basados en patrones retinales el usuario a identificar ha de mirar a través de unos binoculares, ajustar la distancia interocular y el movimiento de la cabeza, mirar a un punto determinado y por último pulsar un botón para indicar al dispositivo que se encuentra listo para el análisis. En ese momento se escanea la retina con una radiación infrarroja de baja intensidad en forma de espiral, detectando los nodos y ramas del área retinal para compararlos con los almacenados en una base de datos; si la muestra coincide con la almacenada para el usuario que el individuo dice ser, se permite el acceso.

La compañía EyeDentify posee la patente mundial para analizadores de vasculatura retinal, por lo que es la principal desarrolladora de esta tecnología; su página *web* se puede encontrar en `http://www.eyedentify.com/`

Iris

El iris humano (el anillo que rodea la pupila, que a simple vista diferencia el color de ojos de cada persona) es igual que la vasculatura retinal una estructura única por individuo que forma un sistema muy complejo - de hasta 266 grados de libertad - , inalterable durante toda la vida de la persona. El uso por parte de un atacante de órganos replicados o simulados para conseguir una falsa aceptación es casi imposible con análisis infrarrojo, capaz de detectar con una alta probabilidad si el iris es natural o no.

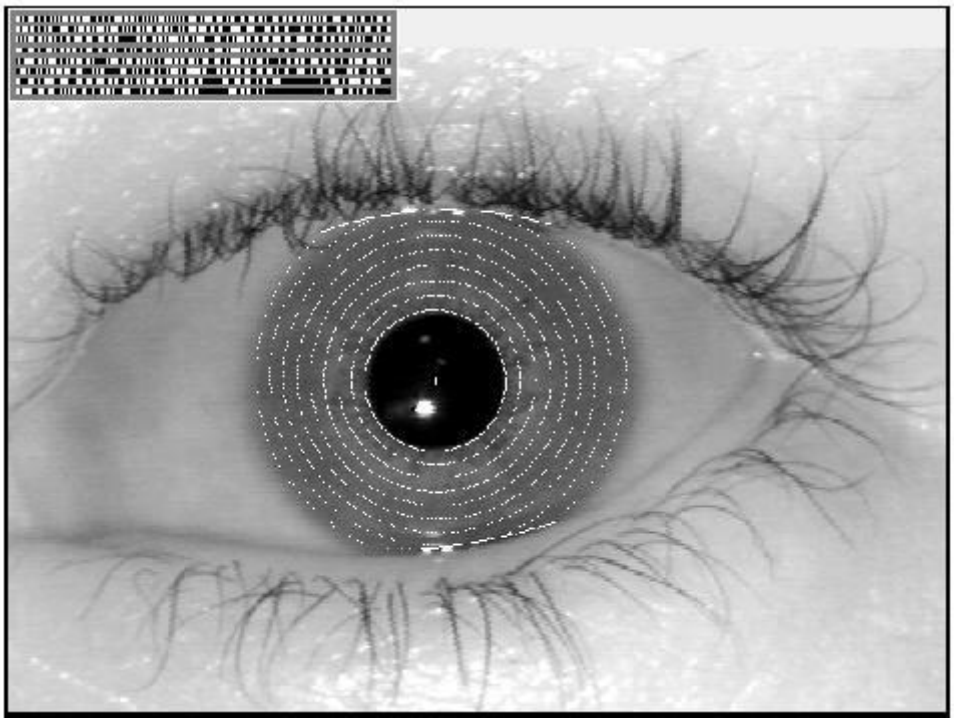

Figura 8.3: Iris humano con la extracción de su *iriscode*.

La identificación basada en el reconocimiento de iris es más moderna que la basada en patrones retinales; desde hace unos años el iris humano se viene utilizando para la autenticación de usuarios ([BAW96], [Dau97]). Para ello, se captura una imagen del iris en blanco y negro, en un entorno correctamente iluminado; esta imagen se somete a deformaciones pupilares (el tamaño de la pupila varía enormemente en función de factores externos, como la luz) y de ella se extraen patrones, que a su vez son sometidos a transformaciones matemáticas ([McM97]) hasta obtener una cantidad de datos (típicamente 256 *KBytes*) suficiente para los propósitos de autenticación. Esa muestra, denominada *iriscode* (en la figura 8.3 se muestra una imagen de un iris humano con su *iriscode* asociado) es comparada con otra tomada con anterioridad y almacenada en la base de datos del sistema, de forma que si ambas coinciden el usuario se considera autenticado con éxito; la probabilidad de una falsa aceptación es la menor de todos los modelos biométricos ([Dau98]).

La empresa estadounidense *IriScan* es la principal desarrolladora de tecnología (y de investigaciones) basada en reconocimiento de iris que existe actualmente, ya que posee la patente sobre esta tecnología; su página *web*, con interesantes artículos sobre este modelo de autenticación (a diferencia de la página de EyeDentify), se puede consultar en `http://www.iriscan.com/`

Verificación de la geometría de la mano

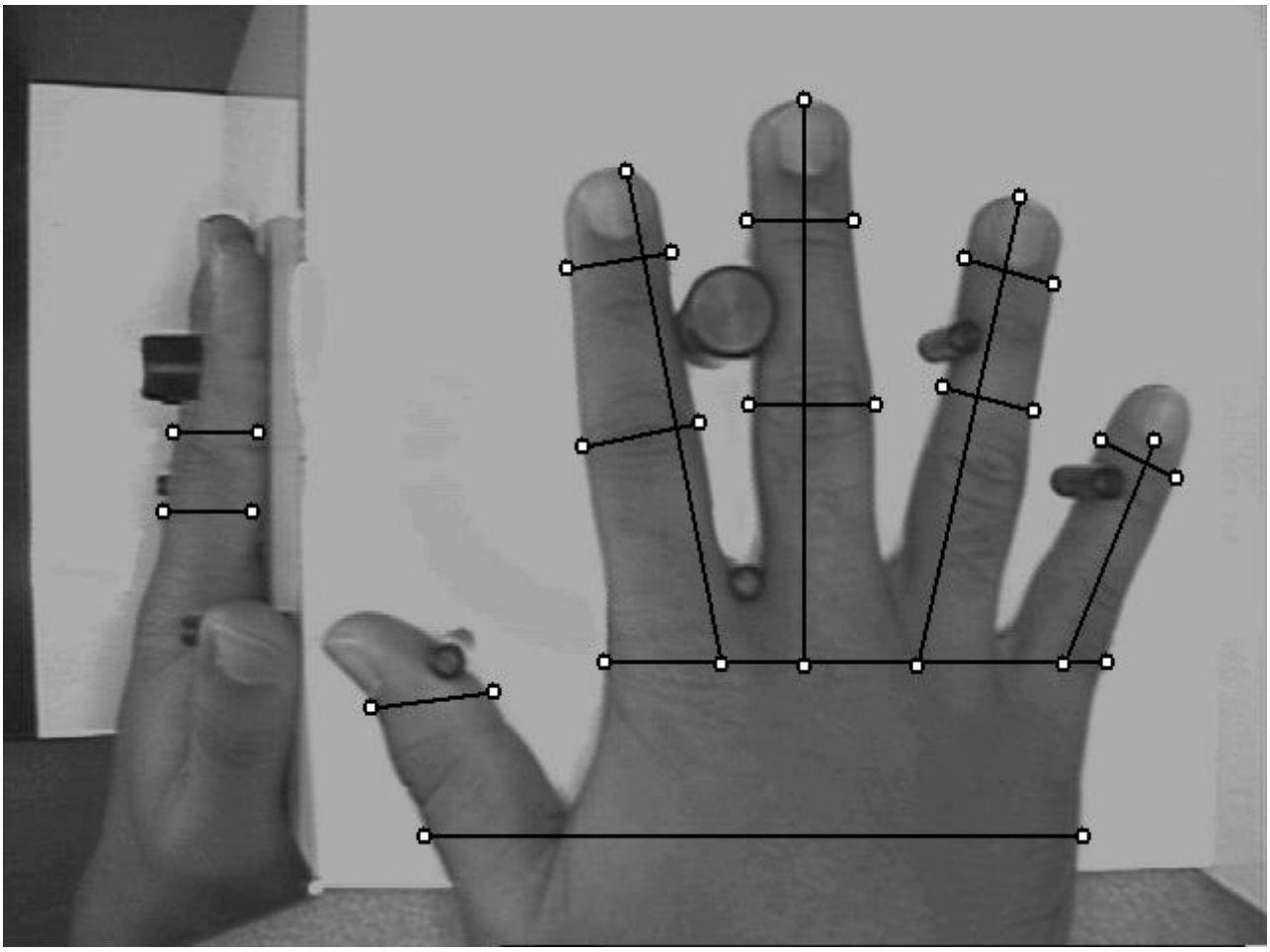

Figura 8.4: Geometría de una mano con ciertos parámetros extraídos.

Los sistemas de autenticación basados en el análisis de la geometría de la mano son sin duda los más rápidos dentro de los biométricos: con una probabilidad de error aceptable en la mayoría de ocasiones, en aproximadamente un segundo son capaces de determinar si una persona es quien dice ser .

Cuando un usuario necesita ser autenticado situa su mano sobre un dispositivo lector con unas guías que marcan la posición correcta para la lectura (figura 8.4). Una vez la mano está correctamente situada, unas cámaras toman una imagen superior y otra lateral, de las que se extraen ciertos datos (anchura, longitud, área, determinadas distancias...) en un formato de tres dimensiones. T ransformando estos datos en un modelo matemático que se contrasta contra una base de patrones, el sistema es capaz de permitir o denegar acceso a cada usuario.

Quizás uno de los elementos más importantes del reconocimiento mediante analizadores de geometría de la mano es que éstos son capaces de aprender: a la vez que autentican a un usuario, actualizan su base de datos con los cambios que se puedan producir en la muestra (un pequeño crecimiento, adelgazamiento, el proceso de cicatrizado de una herida...); de esta forma son capaces de identificar correctamente a un usuario cuya muestra se tomó hace años, pero que ha ido accediendo al sistema con regularidad. Este hecho, junto a su rapidez y su buena aceptación entre los usuarios, hace que los autenticadores basados en la geometría de la mano sean los más extendidos dentro de los biométricos a pesar de que su tasa de falsa aceptación se podría considerar inaceptable en algunas situaciones: no es normal, pero sí posible, que dos personas tengan la mano lo suficientemente parecida como para que el sistema las confunda. Para minimizar este problema se recurre a la identificación basada en la geometría de uno o dos dedos, que además puede usar dispositivos lectores más baratos y proporciona incluso más rapidez.

Autenticación de usuarios en Unix

Autenticación clásica

En un sistema Unix habitual cada usuario posee un nombre de entrada al sistema o *login* y una clave o *password*; ambos datos se almacenan generalmente en el fichero `/etc/passwd` Este archivo contiene una línea por usuario (aunque hay entradas que no corresponden a usuarios reales, como veremos a continuación) donde se indica la información necesaria para que los usuarios puedan conectar al sistema y trabajar en él, separando los diferentes campos mediante `:'. Por ejemplo, podemos encontrar entradas parecidas a la siguiente:

```
toni:LEgPN8jqSCHCg:1000:100:Antonio Villalon,,,:/export/home/toni:/bin/sh
```

En primer lugar aparecen el *login* del usuario y su clave cifrada; a continuación tenemos dos números que serán el identificador de usuario y el de grupo respectivamente. El quinto campo, denominado `GECOS` es simplemente información

administrativa sobre la identidad real del usuario, como su nombre, teléfono o número de despacho. Finalmente, los dos últimos campos corresponden al directorio del usuario (su *$HOME* inicial) y al *shell* que le ha sido asignado.

Al contrario de lo que mucha gente cree, Unix no es capaz de distinguir a sus usuarios por su nombre de entrada al sistema. Para el sistema operativo lo que realmente distingue a una persona de otra (o al menos a un usuario de otro) es el UID del usuario en cuestión; el *login* es algo que se utiliza principalmente para comodidad de las personas (obviamente es más fácil acordarse de un nombre de entrada como *toni* que de un UID como 2643, sobre todo si se tienen cuentas en varias máquinas, cada una con un UID diferente). Por tanto, si en `/etc/passwd` existen dos entradas con un mismo UID, para Unix se tratará del mismo usuario, aunque tengan un *login* y un *password* diferente: así, si dos usuarios tienen asignado el UID 0, ambos tendrán privilegios de superusuario, sin importar el *login* que utilicen. Esto es especialmente aprovechado por atacantes que han conseguido privilegios de administrador en una máquina: pueden añadir una línea a `/etc/passwd` mezclada entre todas las demás, con un nombre de usuario normal pero con el UID 0; así garantizan su entrada al sistema como administradores en caso de ser descubiertos, por ejemplo para borrar huellas. Como a simple vista puede resultar difícil localizar la línea insertada, especialmente en sistemas con un gran número de usuarios, para detectar las cuentas con privilegios en la máquina podemos utilizar la siguiente orden:

```
anita:~# awk -F: '$3==0 {print $1}' /etc/passwd
root
anita:~#
```

En el fichero de claves van a existir entradas que no corresponden a usuarios reales, sino que son utilizadas por ciertos programas o se trata de cuentas mantenidas por motivos de compatibilidad con otros sistemas; típicos ejemplos de este tipo de entradas son `lp`, `uucp` o `postmaster`. Estas cuentas han de estar bloqueadas en la mayoría de casos, para evitar que alguien pueda utilizarlas para acceder a nuestro sistema: sólo han de ser accesibles para el *root* mediante la orden `su`. Aunque en su mayoría cumplen esta condición, en algunos sistemas estas cuentas tienen claves por defecto o, peor , no tienen claves, lo que las convierte en una puerta completamente abierta a los intrusos; es conveniente que, una vez instalado el sistema operativo, y antes de poner a trabajar la máquina, comprobemos que están bloqueadas, o en su defecto que tienen claves no triviales. Algunos ejemplos de cuentas sobre los que hay que prestar una especial atención son[9.4] `root`, `guest`, `lp`, `demos`, `4DGifts`, `tour`, `uucp`, `nuucp`, `games` o `postmaster`; es muy recomendable consultar los manuales de cada sistema concreto, y chequear periódicamente la existencia de cuentas sin clave o cuentas que deberían permanecer bloqueadas y no lo están.

Para cifrar las claves de acceso de sus usuarios, el sistema operativo Unix emplea un criptosistema irreversible que utiliza la función estándar de C `crypt(3)`, basada en el algoritmo DES. Para una descripción exhaustiva del funcionamiento de `crypt(3)` se puede consultar [MT79], [FK90] o [GS96]. Esta función toma como clave los ocho primeros caracteres de la contraseña elegida por el usuario (si la longitud de ésta es menor , se completa con ceros) para cifrar un bloque de texto en claro de 64 bits puestos a cero; para evitar que dos *passwords* iguales resulten en un mismo texto cifrado, se realiza una permutación durante el proceso de cifrado elegida de forma automática y aleatoria para cada usuario, basada en un campo formado por un número de 12 bits (con lo que conseguimos 4096 permutaciones diferentes) llamado *salt*. El cifrado resultante se vuelve a cifrar utilizando la contraseña del usuario de nuevo como clave, y permutando con el mismo *salt*, repitiéndose el proceso 25 veces. El bloque cifrado final, de 64 bits, se concatena con dos bits cero, obteniendo 66 bits que se hacen representables en 11 caracteres de 6 bits cada uno y que, junto con el *salt*, pasan a constituir el campo *password* del fichero de contraseñas, usualmente `/etc/passwd` Así, los dos primeros caracteres de este campo estarán constituidos por el *salt* y los 11 restantes por la contraseña cifrada:

```
toni:LEgPN8jqSCHCg:1000:100:Antonio Villalon,,,:/export/home/toni:/bin/sh
```

SALT: PASSWORD CIFRADO:

Como hemos dicho antes, este criptosistema es irreversible. Entonces, >cómo puede un usuario conectarse a una máquina Unix? El proceso es sencillo: el usuario introduce su contraseña, que se utiliza como clave para cifrar 64 bits a 0 basándose en el *salt*, leído en `/etc/passwd`, de dicho usuario. Si tras aplicar el algoritmo de cifrado el resultado se corresponde con lo almacenado en los últimos 11 caracteres del campo *password* del fichero de contraseñas, la clave del usuario se considera válida y se permite el acceso. En caso contrario se le deniega y se almacena en un fichero el intento de conexión fallido.

Mejora de la seguridad

Problemas del modelo clásico

Los ataques de texto cifrado escogido constituyen la principal amenaza al sistema de autenticación de Unix; a diferencia de lo que mucha gente cree, no es posible descifrar una contraseña, pero es muy fácil cifrar una palabra junto a un determinado *salt*, y comparar el resultado con la cadena almacenada en el fichero de claves. De esta forma, un atacante leerá el fichero `/etc/passwd`(este fichero ha de tener permiso de lectura para todos los usuarios si queremos que el sistema funcione correctamente), y mediante un programa adivinador (o *crackeador*) como `Crack` o `John the Ripper` cifrará todas las palabras de un fichero denominado *diccionario* (un fichero ASCII con un gran número de palabras de cualquier idioma o campo de la sociedad - historia clásica, deporte, cantantes de *rock*...), comparando el resultado obtenido en este proceso con la clave cifrada del fichero de contraseñas; si ambos coinciden, ya ha obtenido una clave para acceder al sistema de forma no autorizada. Este proceso se puede pero no se suele hacer en la máquina local, ya que en este caso hay bastantes posibilidades de detectar el ataque: desde modificar en código de la función `crypt(3)` para que alerte al administrador cuando es invocada repetidamente (cada vez que el adivinador cifra una palabra utiliza esta función) hasta simplemente darse cuenta de una carga de CPU excesiva (los programas adivinadores suelen

consumir un tiempo de procesador considerable). Lo habitual es que el atacante transfiera una copia del archivo a otro ordenador y realice el proceso en esta otra máquina; ni siquiera se tiene que tratar de un servidor Unix con gran capacidad de cómputo: existen muchos programas adivinadores que se ejecutan en un PC normal, bajo MS-DOS o Windows. Obviamente, este segundo caso es mucho más difícil de detectar, ya que se necesita una auditoría de los programas que ejecuta cada usuario (y utilidades como `cp` o `ftp` no suelen llamar la atención del administrador). Esta auditoría la ofrecen muchos sistemas Unix (generalmente en los ficheros de *log* `/var/adm/pacct` o `/var/adm/acct`), pero no se suele utilizar por los excesivos recursos que puede consumir, incluso en sistemas pequeños; obviamente, no debemos fiarnos nunca de los archivos históricos de órdenes del usuario (como `$HOME/.sh_history` o `$HOME/.bash_history`), ya que el atacante los puede modificar para ocultar sus actividades, sin necesidad de ningún privilegio especial.

Contraseñas aceptables

La principal forma de evitar este tipo de ataque es utilizar *passwords* que no sean palabras de los ficheros *diccionario* típicos: combinaciones de minúsculas y mayúsculas, números mezclados con texto, símbolos como &, $ o %, etc. Por supuesto, hemos de huir de claves simples como *internet* o *beatles*, nombres propios, combinaciones débiles como *Pepito1* o *qwerty*, nombres de lugares, actores, personajes de libros, deportistas...Se han realizado numerosos estudios sobre cómo evitar este tipo de *passwords* en los usuarios ([dA88], [Kle90], [Spa91b], [Bel93a], [Bis91], [BK95]...), y también se han diseñado potentes herramientas para lograrlo, como `Npasswd` o `Passwd+` ([Spa91b], [Bis92], [CHN$^+$92]...). Es bastante recomendable instalar alguna de ellas para `obligar' a los usuarios a utilizar contraseñas aceptables (muchos Unices ya las traen incorporadas), pero no conviene confiar toda la seguridad de nuestro sistema a estos programas[9.5]. Como norma, cualquier administrador debería ejecutar con cierta periodicidad algún programa adivinador, tipo *Crack*, para comprobar que sus usuarios no han elegidos contraseñas débiles (a pesar del uso de `Npasswd` o `Passwd+`): se puede tratar de claves generadas antes de instalar estas utilidades o incluso de claves asignadas por el propio *root* que no han pasado por el control de estos programas.

Por último es necesario recordar que para que una contraseña sea aceptable obligatoriamente ha de cumplir el principio **KISS**, que hablando de *passwords* está claro que no puede significar *`Keep it simple, stupid!'* sino **`Keep it SECRET, stupid!'**. La contraseña más larga, la más difícil de recordar, la que combina más caracteres no alfabéticos...pierde toda su robustez si su propietario la comparte con otras personas [9.6].

Shadow Password

Otro método cada día más utilizado para proteger las contraseñas de los usuarios el denominado *Shadow Password* u oscurecimiento de contraseñas. La idea básica de este mecanismo es impedir que los usuarios sin privilegios puedan leer el fichero donde se almacenan las claves cifradas; en el punto anterior hemos comentado que el fichero `/etc/passwd` tiene que tener permiso de lectura para todo el mundo si queremos que el sistema funcione correctamente. En equipos con oscurecimiento de contraseñas este fichero sigue siendo legible para todos los usuarios, pero a diferencia del mecanismo tradicional, las claves cifradas no se guardan en él, sino en el archivo `/etc/shadow`, que sólo el *root* puede leer. En el campo correspondiente a la clave cifrada de `/etc/passwd` no aparece ésta, sino un símbolo que indica a determinados programas (como `/bin/login`) que han de buscar las claves en `/etc/shadow`, generalmente una `x`:

```
toni:x:1000:100:Antonio Villalon,,,:/export/home/toni:/bin/sh
```

El aspecto de `/etc/shadow` es en cierta forma similar al de `/etc/passwd` que ya hemos comentado: existe una línea por cada usuario del sistema, en la que se almacena su *login* y su clave cifrada. Sin embargo, el resto de campos de este fichero son diferentes; corresponden a información que permite implementar otro mecanismo para proteger las claves de los usuarios, el envejecimiento de contraseñas o *Aging Password*, del que hablaremos a continuación:

```
toni:LEgPN8jqSCHCg:10322:0:99999:7:::
```

Desde hace un par de años, la gran mayoría de Unices del mercado incorporan este mecanismo; si al instalar el sistema operativo las claves aparecen almacenadas en `/etc/passwd` podemos comprobar si existe la orden `pwconv`, que convierte un sistema clásico a uno oscurecido. Si no es así, o si utilizamos un Unix antiguo que no posee el mecanismo de *Shadow Password*, es muy conveniente que consigamos el paquete que lo implementa (seguramente se tratará de un fichero `shadow.tar.gz` que podemos encontar en multitud de servidores, adecuado a nuestro clon de Unix) y lo instalemos en el equipo. Permitir que todos los usuarios lean las claves cifradas ha representado durante años, y sigue representando, uno de los mayores problemas de seguridad de Unix; además, una de las actividades preferidas de piratas novatos es intercambiar ficheros de claves de los sistemas a los que acceden y *crackearlos*, con lo que es suficiente una persona que lea nuestro fichero para tener en poco tiempo una colonia de intrusos en nuestro sistema.

Envejecimiento de contraseñas

En casi todas las implementaciones de *Shadow Password* actuales[9.7] se suele incluir la implementación para otro mecanismo de protección de las claves denominado envejecimiento de contraseñas (*Aging Password*). La idea básica de este mecanismo es proteger los *passwords* de los usuarios dándoles un determinado periodo de vida: una contraseña sólo va a ser válida durante un cierto tiempo, pasado el cual expirará y el usuario deberá cambiarla.

Realmente, el envejecimiento previene más que problemas con las claves problemas con la transmisión de éstas por la red: cuando conectamos mediante mecanismos como `telnet`, `ftp` o `rlogin` a un sistema Unix, cualquier equipo entre el nuestro y el servidor puede leer los paquetes que enviamos por la red, incluyendo aquellos que contienen nuestro nombre de usuario y nuestra contraseña (hablaremos de esto más a fondo en los capítulos dedicados a la seguridad del sistema de red y a la criptografía); de esta forma, un atacante situado en un ordenador intermedio puede obtener muy

fácilmente nuestro *login* y nuestro *password*. Si la clave capturada es válida indefinidamente, esa persona tiene un acceso asegurado al servidor en el momento que quiera; sin embargo, si la clave tiene un periodo de vida, el atacante sólo podrá utilizarla antes de que el sistema nos obligue a cambiarla.

A primera vista, puede parecer que la utilidad del envejecimiento de contraseñas no es muy grande; al fin y al cabo, la lectura de paquetes destinados a otros equipos (*sniffing*) no se hace por casualidad: el atacante que lea la red en busca de claves y nombres de usuario lo va a hacer porque quiere utilizar estos datos contra un sistema. Sin embargo, una práctica habitual es dejar programas escuchando durante días y grabando la información leída en ficheros; cada cierto tiempo el pirata consultará los resultados de tales programas, y si la clave leída ya ha expirado y su propietario la ha cambiado por otra, el haberla capturado no le servirá de nada a ese atacante.

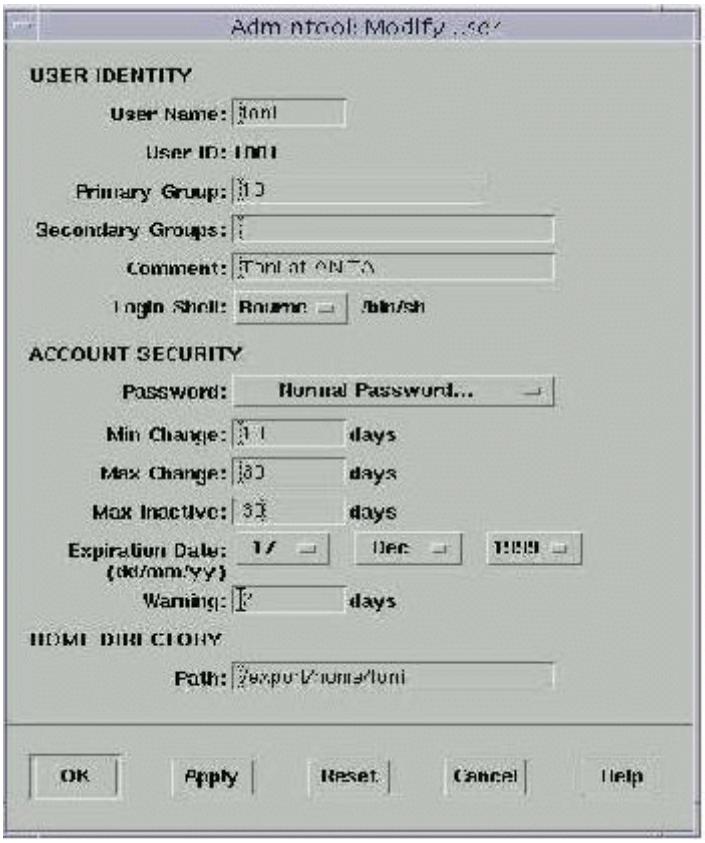

Figura 8.5: La herramienta de administración `admintool` (Solaris), con opciones para envejecimiento de claves.

Los periodos de expiración de las claves se suelen definir a la hora de crear a los usuarios con las herramientas que cada sistema ofrece para ello (por ejemplo, Solaris y su `admintool`, mostrado en la figura 8.5). Si queremos modificar alguno de estos periodos una vez establecidos, desde esas mismas herramientas de administración podremos hacerlo, y también desde línea de órdenes mediante órdenes como `chage` o `usermod`. Como antes hemos dicho, en el archivo `/etc/shadow` se almacena, junto a la clave cifrada de cada usuario, la información necesaria para implementar el envejecimiento de contraseñas; una entrada de este archivo es de la forma

```
toni:LEgPN8jqSCHCg:10322:0:99999:7:::
```

Tras el *login* y el *password* de cada usuario se guardan los campos siguientes:

- Días transcurridos desde el 1 de enero de 1970 hasta que la clave se cambió por última vez.
- Días que han de transcurrir antes de que el usuario pueda volver a cambiar su contraseña.
- Días tras los cuales se ha de cambiar la clave.
- Días durante los que el usuario será avisado de que su clave va a expirar antes de que ésta lo haga.
- Días que la cuenta estará habilitada tras la expiración de la clave.
- Días desde el 1 de enero de 1970 hasta que la cuenta se deshabilite.
- Campo reservado.

Como podemos ver, cuando un usuario cambia su clave el sistema le impide volverla a cambiar durante un periodo de tiempo; con esto se consigue que cuando el sistema obligue a cambiar la contraseña el usuario no restaure inmediatamente su clave antigua (en este caso el esquema no serviría de nada). Cuando este periodo finaliza, suele existir un intervalo de cambio voluntario: está permitido el cambio de contraseña, aunque no es obligatorio; al finalizar este nuevo periodo, el *password* ha expirado y ya es obligatorio cambiar la clave. Si el número máximo de días en los que el usuario no puede cambiar su contraseña es mayor que el número de días tras los cuales es obligatorio el cambio, el usuario **no puede cambiar nunca su clave** . Si tras el periodo de cambio obligatorio el *password* permanece inalterado, la cuenta se bloquea.

En los sistemas Unix más antiguos (hasta *System V Release 3.2*), sin *shadow password*, toda la información de envejecimiento se almacena en `/etc/passwd`, junto al campo correspondiente a la clave cifrada de cada usuario pero separada de éste por una coma:

```
root:cp5zOHITeZLWM,A.B8:0:0:El Spiritu Santo,,,:/root:/bin/bash
```

Tabla: Códigos de caracteres para el envejecimiento de contraseñas.

Carácter	.	/	0	1	2	3	4	5	6	7	8	9	A	B	C
Valor (semanas)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Carácter	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
Valor (semanas)	16	17	18	19	20	21	21	22	23	24	25	26	27	28	29
Carácter	S	T	U	V	W	X	Y	Z	a	b	c	d	e	f	g
Valor (semanas)	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
Carácter	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
Valor (semanas)	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
Carácter	w	x	y	z											
Valor (semanas)	60	61	62	63											

En este caso el primer carácter tras la coma es el número máximo de semanas antes de que el *password* expire; el siguiente carácter es el número mínimo de semanas antes de que el usuario pueda cambiar su clave, y el tercer y cuarto carácter indican el tiempo transcurrido desde el 1 de enero de 1970 hasta el último cambio de contraseña. T odos estos tiempos se indican mediante determinados caracteres con un significado especial, mostrados en la tabla 8.2. También se contemplan en este esquema tres casos especiales: si los dos primeros caracteres son `..' el usuario será obligado a cambiar su clave la siguiente vez que conecte al sistema; el programa `passwd` modificará entonces su entrada en el archivo para que el usuario no se vuelva a ver afectado por el envejecimiento. Otro caso especial ocurre cuando los dos últimos caracteres también son `..', situación en la cual el usuario igualmente se verá obligado a cambiar su clave la próxima vez que conecte al sistema pero el envejecimiento seguirá definido por los dos primeros caracteres. Por último, si el primer carácter tras la coma es menor que el siguiente, el usuario no puede cambiar su *password* nunca, y sólo puede ser modificado a través de la cuenta *root*.

Claves de un solo uso

El envejecimiento de contraseñas tiene dos casos extremos. Por un lado, tenemos el esquema clásico: una clave es válida hasta que el usuario voluntariamente decida cambiarla (es decir , no hay caducidad de la contraseña). El extremo contrario del *Aging Password* es otorgar un tiempo de vida mínimo a cada clave, de forma que sólo sirva para una conexión: es lo que se denomina clave de un solo uso, *One Time Password* ([Lam81]).

>Cómo utilizar contraseñas de un sólo uso? Para conseguirlo existen diferentes aproximaciones; la más simplista consiste en asignar al usuario una lista en papel con la secuencia de claves a utilizar , de forma que cada vez que éste conecte al sistema elimina de la lista la contraseña que acaba de utilizar. Por su parte, el sistema avanza en su registro para que la próxima vez que el usuario conecte pueda utilizar la siguiente clave. Otra aproximación consiste en utilizar un pequeño dispositivo que el usuario debe llevar consigo, como una tarjeta o una calculadora especial, de forma que cuando desee conectar el sistema le indicará una secuencia de caracteres a teclear en tal dispositivo; el resultado obtenido será lo que se ha de utilizar como *password*. Para incrementar la seguridad ante un robo de la tarjeta, antes de teclear el número recibido desde la máquina suele ser necesario utilizar un P .I.N. que el usuario debe mantener en secreto ([GS96]).

Una de las implementaciones del *One Time Password* más extendida entre los diferentes clones de Unix es S/KEY ([Hal94]), disponible también para clientes Windows y MacOS. Utilizando este *software*, la clave de los usuarios no viaja nunca por la red, ni siquiera al ejecutar órdenes como `su` o `passwd`, ni tampoco se almacena información comprometedora (como las claves en claro) en la máquina servidora. Cuando el cliente desea conectar contra un sistema genera una contraseña de un solo uso, que se verifica en el servidor; en ambas tareas se utilizan las funciones resumen MD4 ([Riv90]) o MD5 ([Riv92]). Para realizar la autenticación, la máquina servidora guarda una copia del *password* que recibe del cliente y le aplica la función resumen; si el resultado no coincide con la copia guardada en el fichero de contraseñas, se deniega el acceso. Si por el contrario la verificación es correcta se actualiza la entrada del usuario en el archivo de claves con el *one time password* que se ha recibido (antes de aplicarle la función), avanzando así en la secuencia de contraseñas. Este avance decrementa en uno el número de iteraciones de la función ejecutadas, por lo que ha de llegar un momento en el que el usuario debe reiniciar el contador o en caso contrario se le negará el acceso al sistema; para ello ejecuta una versión modificada de la orden `passwd`.

Otros métodos

Algo por lo que se ha criticado el esquema de autenticación de usuarios de Unix es la longitud - para propósitos de alta seguridad, demasiado corta - de sus claves; lo que hace años era poco más que un planteamiento teórico ([DH77]), actualmente es algo factible: sin ni siquiera entrar en temas de *hardware* dedicado, seguramente demasiado caro para la mayoría de atacantes, con un supercomputador es posible romper claves de Unix en menos de dos días ([KI99]).

Un método que aumenta la seguridad de nuestras claves frente a ataques de intrusos es el cifrado mediante la función

conocida como `bigcrypt()` o `crypt16()`, que permite longitudes para las claves y los *salts* más largas que `crypt(3)`; sin embargo, aunque se aumenta la seguridad de las claves, el problema que se presenta aquí es la incompatibilidad con las claves del resto de Unices que sigan utilizando `crypt(3)`; este es un problema común con otras aproximaciones ([Man96], [KI99]...) que también se basan en modificar el algoritmo de cifrado, cuando no en utilizar uno nuevo.

PAM

PAM (*Pluggable Authentication Module*) no es un modelo de autenticación en sí, sino que se trata de un mecanismo que proporciona una interfaz entre las aplicaciones de usuario y diferentes métodos de autenticación, trantado de esta forma de solucionar uno de los problemas clásicos de la autenticación de usuarios: el hecho de que una vez que se ha definido e implantado cierto mecanismo en un entorno, es difícil cambiarlo. Mediante PAM podemos comunicar a nuestra aplicaciones con los métodos de autenticación que deseemos de una forma transparente, lo que permite integrar las utilidades de un sistema Unix clásico (`login`, `ftp`, `telnet`...) con esquemas diferentes del habitual *password*: claves de un solo uso, biométricos, tarjetas inteligentes...

PAM viene `de serie' en diferentes sistemas Unix, tanto libres como comerciales (Solaris, FreeBSD, casi todas las distribuciones de Linux...), y el nivel de abstracción que proporciona permite cosas tan interesantes como *kerberizar* nuestra autenticación (al menos la parte servidora) sin más que cambiar la configuración de PAM, que se encuentra bien en el fichero `/etc/pam.conf` o bien en diferentes archivos dentro del directorio `/etc/pam.d/`, en el primero de los dos casos, por ejemplo en Solaris, el fichero de configuración de PAM está formado por líneas de la siguiente forma:

servicio tipo control ruta_módulo argumentos_módulo

El campo `` `servicio' `` indica obviamente el nombre del servicio sobre el que se va a aplicar la autenticación (`ftp`, `telnet`, `dtlogin`, `passwd`...), y el campo `` `tipo' `` define el tipo de servicio sobre el que se aplica el módulo; PAM define cuatro posibles valores para este campo ([Her00]):

- `account`
  Determina si el usuario está autorizado a acceder al servicio indicado, por ejemplo si su clave ha caducado, si el acceso se produce desde una línea determinada o si se supera el número máximo de conexiones simultáneas.
- `auth`
  Determina si el usuario es autenticado correctamente, por ejemplo mediante una clave clásica de Unix o mediante un método biométrico.
- `password`
  Proporciona mecanismos para que el usuario actualice su elemento de autenticación, por ejemplo para cambiar su contraseña de acceso al sistema.
- `session`
  Define procesos a ejecutar antes o después de autenticar al usuario, como registrar el evento o activar un mecanismo de monitorización concreto.

Por su parte, el campo al que hemos etiquetado como `` `control' `` marca qué hacer ante el éxito o el fracaso del módulo al que afecten; los módulos PAM son apilables, esto es, podemos combinar un número indeterminado de ellos (del mismo tipo) para un único servicio de forma que si uno de ellos falla la autenticación es incorrecta, si uno de ellos es correcto no nos preocupamos del resto, si algunos son necesarios pero otros no para una correcta autenticación, etc. Se definen cuatro tipos de control:

- `required`
  El resultado del módulo ha de ser exitoso para que se proporcione acceso al servicio; si falla, el resto de módulos de la pila se ejecutan, pero sin importar su resultado el acceso será denegado.
- `requisite`
  De nuevo, el resultado del módulo ha de ser exitoso para que se proporcione acceso al servicio; en caso contrario, no se ejecutan más módulos y el acceso se deniega inmediatamente.
- `sufficient`
  Si la ejecución el módulo correspondiente tiene éxito el acceso se permite inmediatamente (sin ejecutar el resto de módulos para el mismo servicio) siempre y cuando no haya fallado antes un módulo cuyo tipo de control sea `` `required' ``; si la ejecución es incorrecta, no se implica necesariamente una negación de acceso.
- `optional`
  El resultado de su ejecución no es crítico para determinar el acceso al servicio requerido: si falla, pero otro módulo del mismo tipo para el servicio es exitoso, se permite el acceso. Sólo es significativo si se trata del único módulo de su tipo para un cierto servicio, en cuyo caso el acceso al servicio se permite o deniega en función de si la ejecución del módulo tiene éxito.

Finalmente, el campo `` `ruta_modulo' `` marca el nombre del módulo o la ruta donde está úbicado el fichero, mientras que `` `argumentos_modulo `` define los argumentos que se le han de pasar cuando se invoca; este último es el único campo opcional del fichero.

En el caso de que la configuración de PAM se distribuya en diferentes ficheros dentro del directorio `/etc/pam.d/` (generalmente implementaciones más modernas, como las de Linux), el nombre de cada fichero marca el servicio al que afecta la autenticación (es decir, encontraremos un archivo llamado `` `telnet' ``, otro llamado `` `ftp' ``, etc.), de forma que las líneas de cada fichero únicamente tienen los cuatro últimos campos de los comentados aquí.

Veamos un ejemplo: la autenticación definida para el servicio `` `login' `` en un sistema Solaris; el fichero `/etc/pam.conf` contendrá líneas como las siguientes:

```
anita:/# grep ^login /etc/pam.conf
login   auth required   /usr/lib/security/$ISA/pam_unix.so.1
login   auth required   /usr/lib/security/$ISA/pam_dial_auth.so.1
login   account requisite       /usr/lib/security/$ISA/pam_roles.so.1
login   account required        /usr/lib/security/$ISA/pam_unix.so.1
anita:/#
```

La primera línea indica que cuando un usuario desee autenticarse contra el servicio de `` `login' ``, ha de ejecutar correctamente el módulo `pam_unix`, el principal de Solaris, que proporciona funcionalidad para los cuatro tipos de servicio de los que hemos hablado; como en este caso el tipo es `` `auth' ``, lo que hace el módulo es comparar la clave introducida por el usuario con la que existe en el archivo de contraseñas de la máquina, autenticándolo si coinciden. Evidentemente, el control es de tipo `` `required' ``, lo que viene a decir que el *password* tecleado ha de ser el correcto para poder autenticarse contra el sistema; algo parecido sucede con la segunda línea, que invoca al módulo `pam_dial_auth`, encargado de validar la línea de conexión y las claves de *dialup* en Solaris, si los archivos `/etc/dialups` y `/etc/d_passwd` existen. Si cualquiera de los módulos devolviera un código de ejecución incorrecta, el acceso al servicio de `login` - el acceso a la máquina - se denegaría.

Las dos líneas siguientes se utilizan para la gestión de las claves de usuario, también para el control de acceso al servicio `` `login' ``; el módulo `pam_roles` comprueba que el usuario que ejecuta el proceso está autorizado a asumir el rol del usuario que quiere autenticarse, mientras que `pam_unix`, del que ya hemos hablado, lo que hace ahora que el tipo de servicio es `` `account' `` es simplemente verificar que el *password* del usuario no ha caducado. El tipo de control en el primer caso es `` `requisite' ``, lo que implica que si el módulo falla directamente se niega el acceso y no se ejecuta el módulo `pam_unix`; si el primero no falla, sí que se ejecuta este último, y su resultado ha de ser correcto para permitir el acceso (algo por otra parte evidente).

La arquitectura PAM ha venido a solucionar diferentes problemas históricos de la autenticación de usuarios en entornos Unix - especialmente en entornos complejos, como sistemas distribuidos o reinos Kerberos; proporciona una independencia entre los servicios del sistema y los mecanismos de autenticación utilizados, beneficiando tanto al usuario como a los administradores Unix. Desde 1995, año en que se adoptó la solución propuesta por Sun Microsystems hasta la actualidad, cada vez más plataformas integran PAM por defecto, con lo que se ha convertido en el estándar *de facto* en la autenticación dentro de entornos Unix.

Siguiente: Algunos sistemas Unix **Subir:** Seguridad del sistema **Anterior:** Copias de seguridad **Índice General**
2002-07-15