# Principles of Computer System Design

Pablo Umanzor A edited this page · 23 revisions

## Overview

https://booksite.elsevier.com/9780123749574/casestudies/05~11~chapter_11.pdf

Los sistemas informáticos seguros garantizan que la privacidad y las posesiones de los usuarios estén protegidas contra usuarios malintencionados e inquisitivos. La seguridad es un tema amplio, que va desde cuestiones como no permitir que un amigo lea tus archivos hasta proteger la infraestructura de una nación contra ataques. Defenderse contra un adversario es un objetivo negativo. El diseñador de un sistema informático debe asegurarse de que un adversario no pueda violar la seguridad del sistema de ninguna manera. Además, el diseñador debe dificultar que un adversario eluda el mecanismo de seguridad; una de las formas más simples para que un adversario robe información confidencial es sobornar a alguien desde dentro.

Dado que la seguridad es un objetivo negativo, requiere que los diseñadores sean cuidadosos y presten atención a los detalles. Cada detalle podría brindar una oportunidad para que un adversario viole la seguridad del sistema. Afortunadamente, muchos de los principios de diseño previamente encontrados también pueden guiar al diseñador de sistemas seguros. Por ejemplo, los principios del enfoque de la red de seguridad del Capítulo 8 [en línea], ser explícito (indicar tus suposiciones para que puedan ser revisadas) y diseñar para la iteración (asumir que cometerás errores), se aplican igualmente, o quizás incluso con más fuerza, a la seguridad.

El modelo conceptual para proteger los sistemas informáticos contra adversarios es que algún agente presenta a un sistema informático una identidad reclamada y solicita al sistema que realizar alguna acción especificada. Para lograr seguridad, el sistema debe obtener respuestas confiables a las siguientes tres preguntas antes de realizar la acción solicitada:

1. Autenticidad: ¿La identidad reclamada del agente es auténtica? (O, ¿alguien está suplantando al agente?)
2. Integridad: ¿Es esta solicitud realmente la que hizo el agente? (O, ¿alguien la manipuló?)
3. Autorización: ¿Ha otorgado una autoridad adecuada permiso a este agente para realizar esta acción?

El principal sustento de la seguridad de un sistema es el conjunto de mecanismos que aseguran que estas preguntas se respondan satisfactoriamente para cada acción que realiza el sistema. Esta idea se conoce como el principio de

`Mediación completa` `Para cada acción solicitada, verifica autenticidad, integridad y autorización.`

Para protegerse contra ataques internos (adversarios que en realidad son usuarios que tienen los permisos apropiados, pero los abusan) o adversarios que rompen exitosamente los mecanismos de

seguridad, el servicio también debe mantener registros de auditoría de quién utilizó el sistema, qué decisiones de autorización se han tomado, etc. Esta información puede ayudar a determinar quién fue el adversario después del ataque, cómo el adversario violó la seguridad del sistema y llevar al adversario ante la justicia. Al final, un instrumento primario para disuadir a los adversarios es aumentar la probabilidad de detección y castigo.

La siguiente sección proporciona una introducción general a la seguridad. Discute posibles amenazas (Sección 11.1.1), por qué la seguridad es un objetivo negativo (Sección 11.1.2), presenta el enfoque de la red de seguridad (Sección 11.1.3), establece principios para diseñar sistemas informáticos seguros (Sección 11.1.4), el modelo básico para estructurar sistemas informáticos seguros (Sección 11.1.6), una estrategia de implementación basada en minimizar la base de computación de confianza (Sección 11.1.7), y concluye con un mapa de ruta para el resto de este capítulo (Sección 11.1.8). El resto del capítulo trabaja las ideas introducidas en la siguiente sección con más detalle, pero de ninguna manera proporciona un tratamiento completo de la seguridad informática. La seguridad informática es un área activa de investigación con muchos problemas abiertos y se alienta al lector interesado a explorar la literatura de investigación para profundizar en el tema.

# Introducción a los Sistemas Seguros

En el Capítulo 4 vimos cómo dividir un sistema informático en módulos para que los errores no se propaguen de un módulo a otro. En la presentación, asumimos que los errores ocurren involuntariamente: los módulos no cumplen con sus contratos porque los usuarios cometen errores o el hardware falla accidentalmente. A medida que los sistemas informáticos se despliegan cada vez más para aplicaciones críticas para la misión, sin embargo, requerimos sistemas informáticos que puedan tolerar adversarios.

**Por adversario** nos referimos a una entidad que ingresa intencionalmente en sistemas, por ejemplo, para robar información de otros usuarios, extorsionar a una empresa, negar acceso a servicios a otros usuarios, hackear sistemas por diversión o fama, probar la seguridad de un sistema, etc. Un adversario abarca una amplia gama de personas malintencionadas, así como personas bien intencionadas (por ejemplo, personas contratadas por una organización para probar la seguridad de los sistemas informáticos de esa organización). Un adversario puede ser una sola persona o un grupo que colabora para romper la protección.

Casi todas las computadoras están conectadas a redes, lo que significa que pueden ser atacadas por un adversario desde cualquier lugar del mundo. No solo el mecanismo de seguridad debe resistir a los adversarios que tienen acceso físico al sistema, sino que también debe resistir a un mago de 16 años sentado detrás de una computadora personal en algún país del que nunca hemos oído hablar.

Dado que la mayoría de las computadoras están conectadas a través de redes públicas (por ejemplo, Internet), defenderse contra un adversario remoto es particularmente desafiante. Cualquier persona que tenga acceso a la red pública podría comprometer cualquier computadora o enrutador en la red.

Aunque, en la mayoría de los sistemas seguros, evitar que los adversarios hagan cosas malas es el objetivo principal, generalmente también hay necesidad de proporcionar a los usuarios diferentes

niveles de autoridad. Consideremos la banca electrónica. Ciertamente, un objetivo primordial debe ser asegurar que nadie pueda robar dinero de las cuentas, modificar transacciones realizadas a través de las redes públicas o hacer cualquier otra cosa mala. Pero además, un sistema bancario debe hacer cumplir otras restricciones de seguridad.

Por ejemplo, al propietario de una cuenta se le debería permitir retirar dinero de la cuenta, pero no se le debería permitir retirar dinero de otras cuentas. Sin embargo, al personal del banco (bajo ciertas condiciones) se le debería permitir transferir dinero entre cuentas de diferentes usuarios y ver cualquier cuenta. Se necesita algún esquema para hacer cumplir la estructura de autoridad deseada. En algunas aplicaciones, puede que no sea necesario ningún mecanismo de aplicación interna al sistema informático.

Por ejemplo, un código de ética administrado externamente u otros mecanismos fuera del sistema informático pueden proteger adecuadamente el sistema. Por otro lado, con la creciente importancia de las computadoras y Internet, muchos sistemas requieren algún plan de seguridad. Ejemplos incluyen servicios de archivos que almacenan información privada, tiendas en línea, sistemas de información policial, distribución electrónica de software propietario, sistemas de información médica en línea y sistemas de procesamiento de datos de servicios sociales del gobierno. Estos ejemplos abarcan una amplia gama de necesidades de privacidad organizativa y personal.

No todos los campos de estudio utilizan los términos "privacidad", "seguridad" y "protección" de la misma manera. Este capítulo adopta definiciones que se encuentran comúnmente en la literatura de ciencias de la computación. El significado tradicional del término privacidad es la capacidad de un individuo para determinar si, cuándo y a quién se debe divulgar la información personal (ver Recuadro 11.1). El término seguridad describe técnicas que protegen la información y los sistemas de información contra el acceso no autorizado o la modificación de la información, ya sea en almacenamiento, procesamiento o tránsito, y contra la denegación de servicio a los usuarios autorizados. En este capítulo, el término protección se utiliza como sinónimo de seguridad.

Privacidad La definición de privacidad (la capacidad de un individuo para determinar si, cuándo y a quién se debe divulgar la información personal) proviene del libro de 1967 "Privacidad y Libertad" de Alan Westin [Sugerencias para Lecturas Adicionales 1.1.6]. Algunos defensores de la privacidad (ver, por ejemplo,

Sugerencias para Lecturas Adicionales 11.1.2) sugieren que, con la creciente interconectividad proporcionada por la tecnología cambiante, la definición de Westin ahora abarca solo un subconjunto de la privacidad, y necesita actualizarse.

Sugieren esta definición más amplia: la capacidad de un individuo para decidir cómo y en qué medida la información personal puede ser utilizada por otros. Esta definición más amplia incluye el concepto original, pero también abarca el control sobre el uso de información que el individuo ha acordado divulgar, pero que posteriormente puede

ser acumulada sistemáticamente de diversas fuentes, como registros públicos, tarjetas de comprador frecuente de supermercados, registros de navegación web, registros en línea de vendedores de libros sobre qué libros le interesan a esa persona, etc.

El razonamiento es que la tecnología moderna de redes y minería de datos agrega una nueva dimensión a las actividades que pueden constituir una invasión de la privacidad. La definición tradicional implicaba que la privacidad puede protegerse mediante mecanismos de confidencialidad y control de acceso; la definición más amplia implica agregar responsabilidad por el uso de información que el individuo ha acordado divulgar.

Un objetivo común en un sistema seguro es hacer cumplir alguna política de privacidad. Un ejemplo de política en el sistema bancario es que solo el propietario y el personal bancario seleccionado deberían tener acceso a la cuenta de ese propietario. La naturaleza de una política de privacidad no es una cuestión técnica, sino una cuestión social y política.

Para avanzar sin tener que resolver el problema de cuál es una política aceptable, nos enfocamos en los mecanismos para hacer cumplir las políticas. En particular, nos interesan los mecanismos que pueden respaldar una amplia variedad de políticas. Por lo tanto, el principio de separar el mecanismo de la política es especialmente importante en el diseño de sistemas seguros.

# Clasificación de Amenazas

El diseño de cualquier sistema de seguridad comienza con la identificación de las amenazas que el sistema debe resistir. Las amenazas son violaciones potenciales de seguridad causadas tanto por un ataque planeado por un adversario como por errores no intencionados de usuarios legítimos del sistema. El diseñador de un sistema informático seguro debe considerar ambos.

Existen tres amplias categorías de amenazas:

**Divulgación no autorizada de información:** una persona no autorizada puede leer y aprovechar la información almacenada en la computadora o que se está transmitiendo a través de redes. Esta categoría de preocupación a veces se extiende al "análisis de tráfico", en el que el adversario observa solo los patrones de uso de la información y, a partir de esos patrones, puede inferir cierto contenido de información.

**Modificación no autorizada de información:** una persona no autorizada puede realizar cambios en la información almacenada o modificar mensajes que cruzan una red. Un adversario podría participar en este comportamiento para sabotear el sistema o engañar al receptor de un mensaje para divulgar información útil o tomar acciones no deseadas. Este tipo de violación no necesariamente requiere que el adversario pueda ver la información que ha cambiado.

**Denegación no autorizada de uso:** un adversario puede evitar que un usuario autorizado lea o modifique información, aunque el adversario puede no poder leer o modificar la información. Provocar

un "crash" del sistema, inundar un servicio con mensajes o disparar una bala a una computadora son ejemplos de denegación de uso. Este ataque es otra forma de sabotaje.

En general, el término "no autorizado" significa que la divulgación, modificación o denegación de uso ocurren en contra de la intención de la persona que controla la información, posiblemente incluso en contra de las restricciones supuestamente impuestas por el sistema.

Como se mencionó en la descripción general, una complicación en la defensa contra estas amenazas es que el adversario puede explotar el comportamiento de los usuarios que están legítimamente autorizados para usar el sistema pero son descuidados en cuanto a la seguridad. Por ejemplo, muchos usuarios no son expertos en seguridad y ponen sus computadoras en riesgo al navegar por Internet y descargar programas de terceros no confiables voluntaria o incluso sin darse cuenta. Algunos usuarios traen sus propios dispositivos y gadgets personales a su lugar de trabajo; estos dispositivos pueden contener software malicioso. Sin embargo, otros usuarios permiten que amigos y familiares usen computadoras en instituciones para fines personales (por ejemplo, almacenar contenido personal o jugar juegos). Algunos empleados pueden estar descontentos con su empresa y pueden estar dispuestos a colaborar con un adversario.

Es difícil defenderse contra un usuario legítimo que actúa como un adversario porque las acciones del adversario parecerán legítimas. Debido a esta dificultad, esta amenaza tiene su propio término, la amenaza interna.

Debido a que existen muchas amenazas posibles, existe un amplio conjunto de técnicas de seguridad. La siguiente lista solo proporciona algunos ejemplos (consulte Sugerencias para lecturas adicionales 1.1.7 para obtener una gama más amplia de muchos más ejemplos):

- haciendo que la información de la tarjeta de crédito enviada a través de Internet sea ilegible para cualquier persona que no sea el destinatario previsto,
- verificando la identidad declarada de un usuario, ya sea local o a través de una red,
- etiquetando archivos con listas de usuarios autorizados,
- ejecutando protocolos seguros para votación electrónica o subastas,
- instalando un enrutador (en la jerga de seguridad se le llama firewall) que filtra el tráfico entre una red privada y una red pública para hacer más difícil que los extraños ataquen la red privada,
- protegiendo la computadora para evitar la interceptación y la interpretación posterior de la radiación electromagnética,
- bloqueando la habitación que contiene la computadora,
- certificando que el hardware y el software se implementen realmente según lo previsto.
- proporcionando a los usuarios perfiles de configuración para simplificar las decisiones de configuración con valores predeterminados seguros,
- alentar a los usuarios legítimos a seguir buenas prácticas de seguridad,
- monitorear el sistema informático, mantener registros para proporcionar trazas de auditoría y proteger los registros contra manipulaciones.

# La seguridad es un objetivo negativo

Tener una visión estrecha de la seguridad es peligroso porque el objetivo de un sistema seguro es prevenir todas las acciones no autorizadas. Este requisito es un tipo de requisito negativo. Es difícil demostrar que este requisito negativo se ha cumplido, ya que se debe demostrar que se ha anticipado cada posible amenaza. Por lo tanto, un diseñador debe tener una visión amplia de la seguridad y considerar cualquier método en el que el esquema de seguridad pueda ser penetrado o eludido.

Para ilustrar la dificultad, consideremos el objetivo positivo, "Alice puede leer el archivo x". Es fácil probar si un diseñador ha logrado el objetivo (pedimos a Alice que intente leer el archivo). Además, si el diseñador falla, es probable que Alice proporcione comentarios directos enviando un mensaje al diseñador "¡No puedo leer x!" En contraste, con un objetivo negativo, como "Lucifer no puede leer el archivo x", el diseñador debe verificar que todas las formas en que el adversario Lucifer podría leer x estén bloqueadas, y es probable que el diseñador no reciba ningún comentario directo si comete un error. Lucifer no le dirá al diseñador porque Lucifer no tiene motivos para hacerlo y puede que ni siquiera esté en el interés de Lucifer.

Un ejemplo del campo de la biología ilustra bien la diferencia entre probar un positivo y probar un negativo. Consideremos la pregunta "¿Está extinta una especie (por ejemplo, el pájaro carpintero de marfil)?". Generalmente es fácil probar que una especie existe; simplemente exhibir un ejemplo vivo. Pero demostrar que está extinta requiere buscar exhaustivamente en todo el mundo. Dado que esto último suele ser difícil, la respuesta más común para probar un negativo es "no estamos seguros".

La pregunta "¿Es seguro un sistema?" tiene estos mismos tres resultados posibles: inseguro, seguro o no lo sabemos. Para probar que un sistema es inseguro, se debe encontrar solo un ejemplo de un agujero de seguridad. Encontrar el agujero generalmente es difícil y típicamente requiere una experiencia sustancial, pero una vez que se encuentra un agujero, queda claro que el sistema es inseguro. En contraste, para demostrar que un sistema es seguro, se debe mostrar que no hay ningún agujero de seguridad en absoluto. Dado que esto último es tan difícil, el resultado típico es "no conocemos ningún agujero de seguridad restante, pero estamos seguros de que los hay".

Otra forma de apreciar la dificultad de lograr un objetivo negativo es modelar un sistema informático como una máquina de estados con estados para todas las configuraciones posibles en las que puede estar el sistema y con enlaces entre estados para transiciones entre configuraciones. Como se muestra en la Figura 11.1, los estados y enlaces posibles forman un grafo, con los stados como nodos y transiciones posibles como aristas. Supongamos que el sistema está en algún estado actual.

---

El objetivo de un adversario es llevar al sistema desde el estado actual a un estado, etiquetado como "Malo" en la figura, que le dé al adversario acceso no autorizado. Para defenderse contra el adversario, los diseñadores de seguridad deben identificar y bloquear cada camino que conduzca al estado malo. Pero al adversario solo le basta con encontrar un camino desde el estado actual hasta el estado malo.

# El Enfoque de la Red de Seguridad

Para diseñar sistemas que satisfagan objetivos negativos de manera exitosa, este capítulo adopta el enfoque de la red de seguridad del Capítulo 8 [en línea], que en esencia guía al diseñador a ser paranoico: nunca asumir que el diseño es correcto. En el contexto de la seguridad, los dos principios de la red de seguridad, ser explícito y diseñar para la iteración, refuerzan esta actitud paranoica:

1. Ser explícito: Hacer explícitas todas las suposiciones para que puedan ser revisadas. Puede requerir solo un agujero en la seguridad del sistema para penetrarlo. Por lo tanto, el diseñador debe considerar cualquier amenaza que tenga implicaciones de seguridad y hacer explícita la suposición en la que se basa el diseño de seguridad. Además, asegurarse de que todas las suposiciones en las que se basa la seguridad del sistema sean evidentes en todo momento para todos los participantes. Por ejemplo, en el contexto de los protocolos, el significado de cada mensaje debería depender solo del contenido del mensaje en sí mismo y no debe depender del contexto de la conversación. Si el contenido de un mensaje depende de su contexto, un adversario podría ser capaz de romper la seguridad de un protocolo engañando a un receptor para interpretar el mensaje en un contexto diferente.
2. Diseño para la iteración: Suponer que cometerás errores. Dado que el diseñador debe asumir que el diseño en sí contendrá fallas, debe estar preparado para iterar el diseño. Cuando se descubre un agujero de seguridad, el diseñador debe revisar las suposiciones, ajustarlas si es necesario y reparar el diseño. Cuando un diseñador descubre un error en el sistema, debe reiterar todo el proceso de diseño e implementación.

El enfoque de la red de seguridad implica varios requisitos para el diseño de un sistema seguro:

- Certificar la seguridad del sistema. La certificación implica verificar que el diseño coincida con la política de seguridad prevista, que la implementación coincida con el diseño y que el sistema en funcionamiento coincida con la implementación, seguido de pruebas de extremo a extremo realizadas por especialistas en seguridad en busca de errores que puedan comprometer la seguridad. La certificación proporciona un enfoque sistemático para revisar la seguridad de un sistema frente a las suposiciones. Idealmente, la certificación la realizan revisores independientes y, si es posible, utilizando herramientas formales. Una forma de hacer que la certificación sea manejable es identificar aquellos componentes en los que se debe confiar para garantizar la seguridad, minimizar su número y construir un muro alrededor de ellos. La sección 11.1.7 discute esta idea, conocida como la base de confianza informática, con más detalle.
- Mantener registros de auditoría de todas las decisiones de autorización. Dado que el diseñador debe asumir que los usuarios legítimos pueden abusar de sus permisos o que un adversario puede estar haciéndose pasar por un usuario legítimo, el sistema debe mantener un registro a prueba de manipulaciones (para que un adversario no pueda borrar registros) de todas las decisiones de autorización tomadas. Si, a pesar de todos los mecanismos de seguridad, un adversario (ya sea desde el interior o desde el exterior) logra romper la seguridad del sistema, el registro podría ayudar en la investigación forense. Un experto forense puede usar el registro para recopilar evidencia que sea válida en el tribunal y ayudar a establecer la identidad del adversario para que pueda ser procesado después del hecho. El registro también se puede

utilizar como fuente de retroalimentación que revele una suposición incorrecta, diseño o implementación.

- Diseñar el sistema para recibir retroalimentación. Es poco probable que un adversario proporcione retroalimentación al comprometer el sistema, por lo que depende del diseñador crear formas de obtener retroalimentación. Obtener retroalimentación comienza con la declaración explícita de las suposiciones, para que el diseñador pueda verificar el sistema diseñado, implementado y operativo contra las suposiciones cuando se identifica un error. Este método por sí solo no identifica debilidades de seguridad, por lo que el diseñador debe buscar activamente posibles problemas.

  Los métodos incluyen revisar los registros de auditoría y ejecutar programas que alerten a los administradores del sistema sobre comportamientos inesperados, como tráfico de red inusual (por ejemplo, muchas solicitudes a una máquina que normalmente no recibe muchas solicitudes), fallos repetidos en el inicio de sesión, etc.

  El diseñador también debería crear un ambiente en el que el personal y los clientes no sean culpados por los compromisos del sistema, sino que en cambio sean recompensados por informarlos, para que se sientan alentados a informar problemas en lugar de ocultarlos. Diseñar para obtener retroalimentación reduce la posibilidad de que se pasen por alto agujeros de seguridad. Anderson ilustra bien, a través de varios ejemplos del mundo real, lo importante que es diseñar para obtener retroalimentación.

Como parte del enfoque de la red de seguridad, un diseñador debe considerar el entorno en el que se ejecuta el sistema. El diseñador debe asegurar todas las conexiones de comunicación (por ejemplo, líneas de módem de marcado que de lo contrario podrían evitar el firewall que filtra el tráfico entre una red privada y una red pública), prepararse para el mal funcionamiento del equipo, encontrar y eliminar las puertas traseras que crean problemas de seguridad, proporcionar configuraciones seguras por defecto para los usuarios, y determinar quién es lo suficientemente confiable para tener una llave de la habitación que protege la parte más segura del sistema. Además, el diseñador debe protegerse contra sobornos y preocuparse por los empleados descontentos. La literatura de seguridad está llena de historias de fracasos porque los diseñadores no tuvieron en cuenta alguno de estos problemas.

Como otra parte del enfoque de la red de seguridad, el diseñador debe considerar la dinámica de uso. Este término se refiere a cómo se establece y cambia la especificación de quién puede obtener acceso a qué. Por ejemplo, Alice podría revocar el permiso de Bob para leer el archivo "x". Para obtener una idea de la complejidad introducida por los cambios en la autorización de acceso, considere nuevamente la pregunta, "¿Hay alguna manera de que Lucifer pueda obtener acceso al archivo x?" Uno debe verificar no solo si Lucifer tiene acceso al archivo x, sino también si Lucifer puede cambiar la especificación de la accesibilidad del archivo x. El siguiente paso es ver si Lucifer puede cambiar la especificación de quién puede cambiar la especificación de la accesibilidad del archivo x, etc.

Otro problema de dinámica surge cuando el propietario revoca el acceso de un usuario a un archivo mientras se está utilizando ese archivo. Permitir que el usuario previamente autorizado continúe hasta que el usuario haya "terminado" con la información puede ser inaceptable si el propietario de repente se da cuenta de que el archivo contiene datos sensibles. Por otro lado, la revocación inmediata de la

autorización puede interrumpir gravemente al usuario o dejar datos inconsistentes si el usuario estaba en medio de una acción atómica. Las disposiciones para la dinámica de uso son al menos tan importantes como aquellas para la especificación estática de seguridad.

Finalmente, el enfoque de la red de seguridad sugiere que un diseñador nunca debe creer que un sistema es completamente seguro. En su lugar, uno debe diseñar sistemas que se defiendan en profundidad utilizando defensas redundantes, una estrategia que el ejército ruso ha desplegado con éxito durante siglos para defender a Rusia. Por ejemplo, un diseñador podría haber diseñado un sistema que proporciona seguridad de extremo a extremo sobre redes no confiables.

Además, el diseñador también podría incluir un firewall entre la red confiable y la no confiable para seguridad a nivel de red. En principio, el firewall es completamente redundante con los mecanismos de seguridad de extremo a extremo; si el mecanismo de seguridad de extremo a extremo funciona correctamente, no hay necesidad de seguridad a nivel de red. Sin embargo, para que un adversario rompa la seguridad del sistema, el adversario tiene que encontrar fallas tanto en el firewall como en los mecanismos de seguridad de extremo a extremo, y tener la suerte de que la primera falla permita la explotación de la segunda.

La estrategia de diseño en profundidad no ofrece garantías, pero parece ser efectiva en la práctica. La razón es que conceptualmente, la estrategia de defensa en profundidad corta más bordes en el gráfico de todos los posibles caminos desde un estado actual hasta algún estado no deseado. Como resultado, un adversario tiene menos caminos disponibles para llegar a y explotar el estado no deseado.

# Principios de diseño

En la práctica, debido a que la seguridad es un objetivo negativo, producir un sistema que realmente evite todas las acciones no autorizadas ha demostrado ser extremadamente difícil. Ejercicios de penetración que involucran muchos sistemas diferentes han demostrado que los usuarios pueden obtener acceso no autorizado a estos sistemas. Incluso si los diseñadores siguen cuidadosamente el enfoque de la red de seguridad, los fallos de diseño e implementación proporcionan vías que evitan las restricciones de acceso previstas.

Además, debido a que los sistemas informáticos cambian rápidamente o se implementan en nuevos entornos para los que no fueron diseñados originalmente, surgen nuevas oportunidades para comprometer la seguridad. La Sección 11.11 proporciona varias historias de guerra sobre violaciones de seguridad.

Las técnicas de diseño y construcción que excluyen sistemáticamente fallos son el tema de mucha actividad de investigación, pero aún no existe un método completo aplicable al diseño de sistemas informáticos. Esta dificultad está relacionada con la calidad negativa del requisito de prevenir todas las acciones no autorizadas. En ausencia de técnicas metodológicas, la experiencia ha proporcionado varios principios de seguridad para orientar el diseño hacia la minimización del número de fallos de seguridad en una implementación. Discutimos estos principios a continuación.

**El diseño no debe ser secreto:**

## Principio de diseño abierto

Permite que cualquiera comente sobre el diseño. Necesitas toda la ayuda que puedas conseguir.

La violación del principio de diseño abierto históricamente ha demostrado casi siempre conducir a diseños defectuosos. Los mecanismos no deben depender de la ignorancia de posibles adversarios, sino más bien de la posesión de claves secretas o contraseñas específicas, que son más fáciles de proteger. Este desacoplamiento de los mecanismos de seguridad de las claves de seguridad permite que los mecanismos sean examinados por muchos revisores sin preocupación de que la revisión en sí misma comprometa las salvaguardias.

Además, cualquier usuario escéptico debe poder revisar que el sistema sea adecuado para el propósito del usuario. Finalmente, simplemente no es realista mantener en secreto cualquier sistema que reciba una amplia distribución. Sin embargo, el principio de diseño abierto puede entrar en conflicto con otros objetivos, lo que ha dado lugar a numerosos debates; la Nota al margen 11.2 resume algunos de los argumentos. Las personas adecuadas deben realizar la revisión porque detectar fallas de seguridad es difícil. Incluso si el diseño y la implementación son públicos, eso no es una condición suficiente para detectar problemas de seguridad.

Por ejemplo, los comités de estándares suelen ser abiertos en principio, pero su apertura a veces tiene barreras que hacen que el estándar propuesto no sea revisado por las personas adecuadas. Para participar en el diseño del estándar de Privacidad Equivalente con Cable de WiFi se requería que los miembros del comité pagaran una tarifa sustancial, lo que aparentemente desalentó la participación de investigadores en seguridad.

---

¿Deberían ser públicos los diseños y las vulnerabilidades? El debate entre diseños cerrados y abiertos ha estado en marcha literalmente desde hace siglos, y no es único de la seguridad informática. Los defensores de los diseños cerrados argumentan que hacer públicos los diseños ayuda a los adversarios, ¿por qué hacerlo entonces?

Los defensores de los diseños abiertos argumentan que los diseños cerrados en realidad no proporcionan seguridad porque a la larga es imposible mantener en secreto un diseño. El resultado práctico del intento de secreto suele ser que los malos conocen las fallas pero los buenos no. Los defensores del diseño abierto desacreditan los diseños cerrados al describirlos como "seguridad a través de la oscuridad".

Por otro lado, el principio de diseño abierto puede entrar en conflicto con el deseo de mantener un diseño y su implementación de forma privada por razones comerciales o de seguridad nacional. Por ejemplo, las empresas de software a menudo no quieren que un competidor revise su software por miedo a que el competidor pueda aprender o copiar fácilmente ideas.

Muchas empresas intentan resolver este conflicto organizando revisiones, pero restringiendo quién puede participar en las revisiones. Este enfoque tiene el peligro de que las personas adecuadas no estén realizando las revisiones. Estrechamente relacionada con la pregunta de si los diseños deben ser públicos o no está la pregunta de si las vulnerabilidades deben ser publicadas o no.

Una vez más, el debate sobre la respuesta correcta a esta pregunta ha estado en marcha durante siglos, y quizás está mejor ilustrado por la siguiente cita de un libro de 1853 sobre cerraduras antiguas: "En los últimos años ha surgido una duda comercial, y en ciertos aspectos social, sobre si es correcto o no discutir tan abiertamente la seguridad o inseguridad de las cerraduras.

Muchas personas bien intencionadas suponen que la discusión sobre los medios para frustrar la supuesta seguridad de las cerraduras ofrece un incentivo para la deshonestidad, al mostrar a otros cómo ser deshonestos. Esto es una falacia. Los ladrones son muy hábiles en su profesión y ya saben mucho más de lo que podemos enseñarles sobre sus diversos tipos de picardía.

Los ladrones sabían bastante sobre la apertura de cerraduras mucho antes de que los cerrajeros lo discutieran entre ellos, como lo han hecho últimamente. Si una cerradura, sea cual sea su país de origen o su fabricante, no es tan inviolable como hasta ahora se ha considerado, seguramente es del interés de las personas honestas conocer este hecho, porque los deshonestos tienen bastante certeza de aplicar el conocimiento en la práctica; y la difusión del conocimiento es necesaria para dar juego limpio a aquellos que podrían sufrir por ignorancia. No puede enfatizarse lo suficiente que el conocimiento de los hechos reales será, en última instancia, mejor para todas las partes".

Los expertos en seguridad informática generalmente creen que se deben publicar las vulnerabilidades por las razones expuestas por Hobbs y que los usuarios deben saber si el sistema que están utilizando tiene un problema para que puedan decidir si les importa o no. Sin embargo, las empresas suelen ser reacias a divulgar vulnerabilidades. Por ejemplo, un banco tiene poco incentivo para publicitar compromisos exitosos porque puede ahuyentar a los clientes.

Para manejar esta tensión, muchos gobiernos han creado leyes y organizaciones que hacen públicas las vulnerabilidades. En California, las empresas deben informar a sus clientes si un adversario podría haber tenido éxito en robar información privada del cliente (por ejemplo, un número de seguro social).

El gobierno federal de los Estados Unidos ha creado el Equipo de Respuesta a Emergencias Informáticas (CERT) para documentar vulnerabilidades en sistemas de software y ayudar con la respuesta a estas vulnerabilidades (ver www.cert.org). Cuando CERT se entera de una nueva vulnerabilidad, primero notifica al proveedor, luego espera un tiempo para que el proveedor desarrolle un parche, y luego hace pública la vulnerabilidad y el parche.

cuando se finalizó y los investigadores de seguridad comenzaron a examinar el estándar, encontraron inmediatamente varios problemas, uno de los cuales se describe en la página 11-51.

Dado que es difícil mantener un secreto:

**Minimizar los secretos**

**porque probablemente no permanecerán en secreto por mucho tiempo**

Siguiendo este principio tiene la siguiente ventaja adicional. Si el secreto se ve comprometido, debe ser reemplazado; si el secreto es mínimo, entonces reemplazarlo es más fácil. Un diseño abierto que minimiza los secretos no proporciona seguridad por sí mismo. El fundamento principal de la seguridad de un sistema es, como se mencionó en la página 11–5, el principio de mediación completa. Este principio obliga a autenticar y autorizar explícitamente cada acceso, incluidos los de inicialización, recuperación, apagado y mantenimiento.

Implica que debe idearse un método infalible para verificar la autenticidad del origen y los datos de cada solicitud. Este principio se aplica a un servicio que medie las solicitudes, así como a un kernel que medie llamadas de supervisión y a un administrador de memoria virtual que medie una solicitud de lectura de un byte en la memoria. También implica que las propuestas para almacenar en caché resultados de una verificación de autoridad deben ser examinadas con escepticismo; si ocurre un cambio en la autoridad, los resultados almacenados en caché deben actualizarse.

El principio de ingeniería humana de la menor sorpresa se aplica especialmente a la mediación. El mecanismo de autorización debe ser lo suficientemente transparente para el usuario como para que tenga una buena comprensión intuitiva de cómo se relacionan los objetivos de seguridad con el mecanismo de seguridad proporcionado. Es esencial que la interfaz humana esté diseñada para facilitar su uso, de modo que los usuarios apliquen rutinaria y automáticamente los mecanismos de seguridad correctamente.

Por ejemplo, un sistema debería proporcionar configuraciones predeterminadas intuitivas para los mecanismos de seguridad para que solo se autoricen las operaciones apropiadas. Si un administrador de sistema o usuario debe configurar primero o pasar por dificultades para usar un mecanismo de seguridad, el usuario no lo utilizará. Además, en la medida en que la imagen mental del usuario de los objetivos de seguridad coincida con los mecanismos de seguridad, se minimizarán los errores.

Si un usuario debe traducir objetivos de seguridad intuitivos a un lenguaje de especificación radicalmente diferente, los errores son inevitables. Idealmente, los mecanismos de seguridad deberían mejorar la experiencia informática del usuario en lugar de empeorarla. Otro principio ampliamente aplicable, adoptar simplificaciones generales, también se aplica a la seguridad. Cuantos menos mecanismos deben estar correctos para garantizar la protección, más probable será que el diseño sea correcto.

**Economía de mecanismo. Cuanto menos haya, más probable será que lo hagas bien.**

Diseñar un sistema seguro es difícil porque se debe considerar cada camino de acceso para garantizar una mediación completa, incluidos aquellos que no se utilizan durante la operación normal. Como resultado, pueden ser necesarias técnicas como la inspección línea por línea del software y el examen físico del hardware que implementa los mecanismos de seguridad. Para que estas técnicas tengan éxito, es esencial un diseño pequeño y simple.

Reducir el número de mecanismos necesarios ayuda a verificar la seguridad de un sistema informático. Para los que quedan, sería ideal si solo unos pocos son comunes a más de un usuario y dependen de todos los usuarios, porque cada mecanismo compartido podría proporcionar caminos de comunicación no deseados entre los usuarios. Además, cualquier mecanismo que sirva a todos los usuarios debe ser certificado para la satisfacción de cada usuario, un trabajo presumiblemente más difícil que satisfacer solo a uno o unos pocos usuarios. Estas observaciones llevan al siguiente principio de seguridad:

**Minimiza el mecanismo común**

*Los mecanismos compartidos proporcionan rutas de comunicación no deseadas*

Este principio ayuda a reducir el número de rutas de comunicación no deseadas y disminuye la cantidad de hardware y software en los que todos los usuarios dependen, lo que facilita verificar si existen implicaciones de seguridad no deseadas. Por ejemplo, dado la opción de implementar una nueva función como un procedimiento del kernel compartido por todos los usuarios o como un procedimiento de biblioteca que pueda manejarse como si fuera propio del usuario, elige el último curso. Entonces, si uno o unos pocos usuarios no están satisfechos con el nivel de certificación de la función, pueden proporcionar un sustituto o no usarla en absoluto. De cualquier manera, pueden evitar ser perjudicados por un error en ella. Este principio es un argumento de extremo a extremo.

La mediación completa requiere que cada solicitud sea verificada para autorización y solo las solicitudes autorizadas sean aprobadas. Es importante que las solicitudes no sean autorizadas accidentalmente. El siguiente principio de seguridad ayuda a reducir tales errores:

**Configuraciones predeterminadas a prueba de fallos.** La mayoría de los usuarios no las cambiarán, así que asegúrate de que las configuraciones predeterminadas realicen alguna acción segura.

Las decisiones de acceso deben basarse en permisos en lugar de exclusiones. Este principio significa que la falta de acceso debería ser la configuración predeterminada, y el esquema de seguridad enumera las condiciones bajo las cuales se permite el acceso. Este enfoque muestra un mejor modo de falla que el enfoque alternativo, donde la configuración predeterminada es permitir el acceso. Un error de diseño o implementación en un mecanismo que otorga permiso explícito tiende a fallar al negar el permiso, una situación segura que puede detectarse rápidamente. Por otro lado, un error de diseño o implementación en un mecanismo que excluye explícitamente el acceso tiende a fallar al permitir el acceso, un fallo que puede pasar desapercibido durante mucho tiempo en el uso normal.

Para asegurar que la mediación completa y los valores predeterminados a prueba de fallos funcionen bien en la práctica, es importante que los programas y los usuarios tengan privilegios solo cuando sea necesario. Por ejemplo, los programas del sistema o los administradores que tienen privilegios especiales deberían tener esos privilegios solo cuando sea necesario; cuando están realizando actividades ordinarias, los privilegios deberían ser retirados. Dejarlos en su lugar simplemente abre la puerta a accidentes. Estas observaciones sugieren el siguiente principio de seguridad:

**Principio del menor privilegio**, No guardes el almuerzo en la caja fuerte con las joyas.

Este principio limita el daño que puede resultar de un accidente o un error. Además, si menos programas tienen privilegios especiales, se debe auditar menos código para verificar la seguridad de un sistema. La regla de seguridad militar del "necesidad de saber" es un ejemplo de este principio. A veces, los expertos en seguridad utilizan formulaciones alternativas que combinan aspectos de varios principios.

Por ejemplo, la formulación "minimizar la superficie de ataque" combina aspectos de la economía del mecanismo (una interfaz estrecha con una implementación simple proporciona menos oportunidades para errores del diseñador y, por lo tanto, ofrece menos posibilidades de ataque), minimizar los secretos (pocas oportunidades para descifrar secretos), menor privilegio (ejecutar la mayor parte del código con pocos privilegios para que un ataque exitoso haga poco daño) y minimizar el mecanismo común (reducir el número de oportunidades de rutas de comunicación no deseadas).

**A High d(technology)/dt Poses Challenges For Security**

Gran parte del software en Internet y en computadoras personales no sigue estos principios, aunque la mayoría de estos principios fueron entendidos y articulados en la década de 1970, antes de que las computadoras personales y Internet entraran en existencia. Las razones por las que no se siguieron son diferentes para Internet y las computadoras personales, pero ilustran lo difícil que es lograr la seguridad cuando la tasa de innovación es alta.

Cuando Internet se implementó por primera vez, las implementaciones de software de las técnicas criptográficas necesarias para autenticar y proteger mensajes (ver Sección 11.2 y Sección 11.1) se consideraron, pero habrían aumentado la latencia a niveles inaceptables. Las implementaciones de hardware de operaciones criptográficas en ese momento eran demasiado caras y no exportables porque el gobierno de EE. UU. aplicaba reglas para limitar el uso de la criptografia.

Dado que inicialmente Internet era utilizada principalmente por académicos, una comunidad mayormente cooperativa, la falta resultante de seguridad inicialmente no fue un defecto grave. En 1994, Internet se abrió a actividades comerciales. Surgieron tiendas electrónicas y muchas más computadoras que almacenaban información valiosa se conectaron en línea. Este desarrollo atrajo a muchos más adversarios.

De repente, los diseñadores de Internet se vieron obligados a proporcionar seguridad. Debido a que la seguridad no era parte del plan de diseño inicial, los mecanismos de seguridad de hoy en día han sido diseñados como adiciones posteriores y se han proporcionado de manera ad hoc en lugar de seguir un plan general basado en principios de seguridad establecidos.

Por razones históricas diferentes, la mayoría de las computadoras personales venían con poca seguridad interna y solo intentos limitados de seguridad en red. Sin embargo, hoy en día, las computadoras personales están casi siempre conectadas a redes donde son vulnerables.

Originalmente, las computadoras personales fueron diseñadas como dispositivos independientes para ser utilizadas por una sola persona (por eso se llaman computadoras personales). Para mantener los costos bajos, prácticamente no tenían mecanismos de seguridad, pero como se usaban de manera independiente, la situación era aceptable. Con la llegada de Internet, el deseo de conectarse en línea expuso sus problemas de seguridad previamente benignos.

Además, debido a las rápidas mejoras en la tecnología, las computadoras personales son ahora la plataforma principal para todo tipo de cómputo, incluido la mayoría de los cómputos relacionados con los negocios. Como las computadoras personales ahora almacenan información valiosa, están conectadas a redes y tienen protección mínima, se han convertido en un objetivo principal para los adversarios.

Los diseñadores de la computadora personal no previeron originalmente que el acceso a la red se convertiría rápidamente en un requisito universal. Cuando más tarde respondieron a las preocupaciones de seguridad, los diseñadores intentaron agregar rápidamente mecanismos de seguridad. Sin embargo, simplemente lograr que los mecanismos de hardware fueran correctos tomó múltiples iteraciones, tanto debido a errores como porque se agregaron después del hecho.

Hoy en día, los diseñadores todavía están tratando de descubrir cómo adaptar el software existente de las computadoras personales y configurar los ajustes predeterminados correctamente para mejorar la seguridad, mientras también se enfrentan a requisitos de seguridad mejorada para manejar ataques de denegación de servicio, ataques de phishing*, virus, gusanos, malware y adversarios que intentan tomar el control de las máquinas sin ser notados para crear botnets (consulte la Nota al margen 11.3). Como

consecuencia, existen muchos mecanismos ad hoc que se encuentran en el campo y que no siguen los modelos o principios sugeridos en este capítulo.

# Security Model

Aunque hay muchas formas de comprometer la seguridad de un sistema, el modelo conceptual para asegurar un sistema es sorprendentemente simple. Para ser seguro, un sistema requiere mediación completa: el sistema debe mediar cada acción solicitada, incluidas aquellas para configurar y administrar el sistema. El plan de seguridad básico entonces es que para cada acción solicitada el El agente que solicita la operación debe probar su identidad al sistema y luego el sistema decide si se permite al agente realizar esa operación.

Este modelo simple abarca una amplia gama de instancias de sistemas. Por ejemplo, el agente puede ser un cliente en una aplicación cliente/servicio, en cuyo caso la solicitud está en forma de un mensaje a un servicio. Para otro ejemplo, el agente puede ser un hilo que hace referencia a la memoria virtual, en cuyo caso la solicitud está en forma de una operación de CARGA o ALMACENAMIENTO en una celda de memoria nombrada.

En cada uno de estos casos, el sistema debe establecer la identidad del agente y decidir si realizar o no la solicitud. Si todas las solicitudes se medían correctamente, entonces el trabajo del adversario se vuelve mucho más difícil. El adversario debe comprometer el sistema de mediación, lanzar un ataque interno o limitarse a ataques de denegación de servicio.

El resto de esta sección desarrolla el modelo de mediación con más detalle e ilustra con varios ejemplos. Por supuesto, un modelo conceptual simple no puede cubrir todos los ataques y todos los detalles. Y, desafortunadamente, en seguridad, el diablo a menudo está en los detalles de la implementación: ¿el sistema que se pretende seguro implementa el modelo para todas sus operaciones y es la implementación correcta? Sin embargo, el modelo es útil para enmarcar muchos problemas de seguridad y luego abordarlos.

Los agentes actúan en nombre de alguna entidad que corresponde a una persona fuera del sistema informático; llamamos a la representación de tal entidad dentro del sistema informático un principal. El principal es la unidad de autorización en un sistema informático, y por lo tanto también la unidad de responsabilidad y rendición de cuentas. Usando estos términos, mediar una acción implica hacer la pregunta: "¿Está autorizado el principal que solicitó la acción para realizar la acción?"

El enfoque básico para mediar cada acción solicitada es asegurarse de que realmente solo haya una forma de solicitar una acción. Conceptualmente, queremos construir un muro alrededor del sistema con una pequeña abertura a través de la cual pasen todas las acciones solicitadas. Entonces, para cada acción solicitada, el sistema debe responder "¿Debería realizar la acción?".

Para hacerlo, un sistema típicamente se descompone en dos partes: una parte, llamada guardia, que se especializa en decidir la respuesta a la pregunta y una segunda parte que realiza la acción. (En la literatura, una guardia que proporciona mediación completa generalmente se llama monitor de referencia).

La guardia puede aclarar la pregunta, "¿Está autorizado el principal que originó la acción solicitada para realizar la acción?" obteniendo respuestas a las tres subpreguntas de la mediación completa (ver Figura 11.2).

La guardia verifica que el mensaje que contiene la solicitud sea auténtico (es decir, que la solicitud no haya sido modificada y que el principal sea realmente la fuente de la solicitud) y que el principal esté permitido para realizar la acción solicitada sobre el objeto (autorización). Si es así, la guardia permite la acción; de lo contrario, deniega la solicitud. La guardia también registra todas las decisiones para auditorías posteriores.

Las dos primeras preguntas de mediación (¿se ha modificado la solicitud y cuál es la fuente de la solicitud?) caen en la provincia de la autenticación de la solicitud. Usando un servicio de autenticación, la guardia verifica la identidad del principal. Usando información adicional, a veces parte de la solicitud pero a veces comunicada por separado, la guardia verifica la integridad de la solicitud. Después de responder a las preguntas de autenticidad, la guardia sabe quién es el principal asociado con la solicitud y que ningún adversario ha modificado la solicitud.

La tercera y última pregunta cae en la provincia de la autorización. Un servicio de autorización permite a los principales especificar qué objetos comparten con quién. Una vez que la guardia ha establecido de manera segura la identidad del principal asociado con la solicitud utilizando el servicio de autenticación, la guardia verifica con el servicio de autorización que el principal tiene la autorización adecuada y, si es así, permite que el servicio solicitado realice la acción solicitada.

El enfoque de la guardia de mediación completa se aplica ampliamente a los sistemas informáticos. Ya sea que los mensajes sean solicitudes web para una tienda en línea, operaciones de CARGA y ALMACENAMIENTO en memoria o llamadas de supervisor para el núcleo, en todos los casos las mismas tres preguntas deben ser respondidas por el servicio web, el administrador de memoria virtual o el núcleo, respectivamente. Sin embargo, la implementación de los mecanismos de mediación podría ser bastante diferente para cada caso.

Consideremos un periódico en línea. El servicio de periódico puede restringir ciertos artículos a suscriptores pagos y, por lo tanto, debe autenticar a los usuarios y autorizar las solicitudes, que a menudo funcionan de la siguiente manera. El navegador web envía solicitudes en nombre de un usuario de Internet al servidor web del periódico. La guardia utiliza el número de suscriptor del principal y un autenticador (por ejemplo, una contraseña) incluido en las solicitudes para autenticar al principal asociado con las solicitudes.

Si el principal es un suscriptor legítimo y tiene autorización para leer el artículo solicitado, la guardia permite la solicitud y el servidor responde con el artículo. Dado que Internet no es de confianza, las comunicaciones entre el navegador web y el servidor deben estar protegidas; de lo contrario, un adversario puede, por ejemplo obtener la contraseña del suscriptor. Usando la criptografía, uno puede crear un canal seguro que proteja las comunicaciones sobre una red no confiable. La criptografía es una rama de la informática que diseña primitivas como cifrados, generadores de números pseudoaleatorios

y hashes, que pueden ser utilizados para proteger mensajes contra una amplia gama de ataques. Como otro ejemplo, consideremos un sistema de memoria virtual con un dominio por hilo.

En este caso, el procesador emite instrucciones de CARGA y ALMACENAMIENTO en nombre de un hilo a un administrador de memoria virtual, que verifica si las direcciones en las instrucciones caen en el dominio del hilo. Conceptualmente, el procesador envía un mensaje a través de un bus, que contiene la operación (CARGA o ALMACENAMIENTO) y la dirección solicitada. Este mensaje está acompañado de un identificador de principal que nombra al hilo.

Si el bus es un enlace de comunicación de confianza, entonces el mensaje no tiene que estar protegido. Si el bus no es un canal seguro (por ejemplo, una aplicación de gestión de derechos digitales puede querer protegerse contra un propietario que espía en el bus para robar el contenido con derechos de autor), entonces el mensaje entre el procesador y la memoria podría protegerse usando técnicas criptográficas.

El administrador de memoria virtual desempeña el papel de una guardia. Utiliza el identificador de hilo para verificar si la dirección cae en el dominio del hilo y si el hilo está autorizado para realizar la operación. Si es así, la guardia permite la operación solicitada, y el administrador de memoria virtual responde leyendo y escribiendo en la ubicación de memoria solicitada. Aunque los mecanismos para una mediación completa se implementen perfectamente (es decir, no hay errores de diseño e implementación en la criptografía, el verificador de contraseñas, el administrador de memoria virtual, el núcleo, etc.), un sistema aún puede dejar oportunidades para que un adversario rompa la seguridad del sistema.

El adversario puede ser capaz de eludir la guardia o lanzar un ataque interno o sobrecargar el sistema con solicitudes de acciones, retrasando o incluso negando el acceso legítimo a los principales. Un diseñador debe estar preparado para estos casos, un ejemplo de la actitud de diseño paranoico. Discutimos estos casos con más detalle.

Para eludir la guardia, el adversario podría crear o encontrar otra abertura en el sistema. Una abertura simple para un adversario podría ser una línea de módem de marcado que no está mediada. Si el adversario encuentra el número de teléfono (y tal vez la contraseña para marcar), el adversario puede tomar el control del servicio.

Una forma más sofisticada de crear una abertura es un ataque de desbordamiento de búfer en servicios escritos en el lenguaje de programación C (ver el cuadro 11.4), que hace que el servicio ejecute un programa bajo el control del adversario, que luego crea una interfaz para el adversario que no es verificada por el sistema.

Como ejemplos de ataques internos, el adversario puede ser capaz de adivinar la contraseña de un principal, puede ser capaz de sobornar a un principal para que actúe en nombre del adversario, o puede ser capaz de engañar al principal para que ejecute el programa del adversario en la computadora del principal con los privilegios del principal (por ejemplo, el principal abre un archivo adjunto de correo electrónico ejecutable enviado por el adversario). O bien, el adversario puede ser un principal legítimo que está descontento.

Las medidas contra los principales que se comportan mal también son la última línea de defensa contra los adversarios que logran romper con éxito la seguridad del sistema, apareciendo así como usuarios legítimos. Las medidas incluyen (1) ejecutar cada operación solicitada con el menor privilegio posible porque eso minimiza el daño que un principal legítimo puede hacer mantener un registro de auditoría, de las decisiones de mediación realizadas para cada operación, (3) hacer copias y archivar datos en lugares seguros, y (4) revisar periódicamente manualmente qué principales deberían seguir teniendo acceso y con qué privilegios.

Por supuesto, los datos archivados y el registro de auditoría deben mantenerse de forma segura; un adversario no debe poder modificar los datos archivados o el registro de auditoría. Las medidas para asegurar los archivos y los registros de auditoría incluyen diseñarlos para que sean de escritura única y solo se puedan añadir.

Los archivos y el registro de auditoría se pueden utilizar para recuperarse de una brecha de seguridad. Si una inspección del servicio revela que algo malo ha sucedido, las copias archivadas se pueden utilizar para restaurar los datos. El registro de auditoría puede ayudar a averiguar qué sucedió (por ejemplo, qué datos han sido dañados) y qué principal lo hizo. Como se mencionó anteriormente, el registro de auditoría también podría ser útil como prueba en la corte para castigar a los adversarios. Estas medidas pueden verse como un ejemplo de defensa en profundidad: si la primera línea de defensa falla, se espera que la próxima medida ayude.

El objetivo de un adversario puede ser simplemente denegar el servicio a otros usuarios. Para lograr este objetivo, un adversario podría inundar un enlace de comunicación con solicitudes que tomen suficiente tiempo del servicio como para que no esté disponible para otros usuarios. El desafío al manejar un ataque de denegación de servicio es que los mensajes enviados por el adversario pueden ser solicitudes legítimas y el adversario puede utilizar muchos equipos para enviar estas solicitudes legítimas (ver Sugerencias para lecturas adicionales 11.6.4 para un ejemplo). No hay una técnica única que pueda abordar Los ataques de denegación de servicio.

Las soluciones típicamente implican varias ideas: auditar los mensajes para poder detectar y filtrar el tráfico malicioso antes de que llegue al servicio, diseñar cuidadosamente los servicios para controlar los recursos dedicados a una solicitud y para devolver el trabajo a los clientes, y replicar servicios (ver Sección 10.3 en línea) para mantener el servicio disponible durante un ataque. Al replicar el servicio, un adversario debe inundar múltiples réplicas para hacer que el servicio no esté disponible. Este ataque puede requerir tantos mensajes que, con un análisis cuidadoso de los registros de auditoría, se vuelva posible rastrear al adversario.

# Base de Confianza de Computación

Implementar el modelo de seguridad de la Sección 11.1.6 es un objetivo negativo y, por lo tanto, difícil. No existen métodos para verificar la corrección de una implementación que pretende lograr un objetivo negativo. Entonces, ¿cómo procedemos? La idea básica es minimizar el número de mecanismos que deben ser correctos para que el sistema sea seguro, el principio de economía de mecanismos, y seguir el enfoque de la red de seguridad (ser explícito y diseñar para la iteración).

Al diseñar un sistema seguro, organizamos el sistema en dos tipos de módulos: módulos no confiables y módulos confiables. La corrección de los módulos no confiables no afecta la seguridad de todo el sistema. Los módulos confiables son la parte que debe funcionar correctamente para hacer que el sistema sea seguro. Idealmente, queremos que los módulos confiables sean utilizables por otros módulos no confiables, para que el diseñador de un nuevo módulo no tenga que preocuparse por hacer bien los módulos confiables. La colección de módulos confiables se llama habitualmente la base de confianza de la computación (TCB, por sus siglas en inglés).

Establecer si un módulo es parte de la TCB puede ser difícil. Mirando un módulo individual, no hay ningún procedimiento simple para decidir si la seguridad del sistema depende del funcionamiento correcto de ese módulo. Por ejemplo, en UNIX, si un módulo se ejecuta en nombre del principal del superusuario, es probable que sea parte de la TCB porque si el adversario compromete el módulo, el adversario tiene privilegios completos.

Si el mismo módulo se ejecuta en nombre de un principal regular, a menudo no es parte de la base de confianza de la computación porque no puede realizar operaciones privilegiadas. Pero incluso entonces, el módulo podría ser parte de la TCB; puede ser parte de un servicio de nivel de usuario (por ejemplo, un servicio web) que toma decisiones sobre qué clientes tienen acceso. Un error en el código del módulo puede permitir que un adversario obtenga acceso no autorizado.

Al carecer de un procedimiento sistemático de decisión para decidir si un módulo está en la TCB, la decisión es difícil de tomar y fácil de equivocarse, sin embargo, una buena división es importante. Una mala división entre módulos confiables y no confiables puede resultar en una TCB grande y compleja, lo que dificulta razonar sobre la seguridad del sistema.

Si la TCB es grande, también significa que los usuarios ordinarios pueden realizar solo unos pocos cambios porque los usuarios ordinarios solo deben cambiar módulos fuera de la TCB que no afecten la seguridad. Si los usuarios ordinarios pueden cambiar el sistema solo de manera limitada, puede dificultarles realizar su trabajo de manera efectiva y resultar en malas experiencias de usuario. Una TCB grande también significa que gran parte del sistema puede ser modificada solo por principios de confianza, limitando la velocidad a la que puede evolucionar el sistema.

Los principios de diseño de la Sección 11.1.4 pueden guiar esta parte del proceso de diseño, pero típicamente la división debe ser elaborada por expertos en seguridad.

Una vez que se haya trabajado la división, el desafío se convierte en diseñar e implementar una TCB. Para tener éxito en este desafío, queremos trabajar de manera que maximice la probabilidad de que el diseño e implementación de la TCB sean correctos. Para hacerlo, queremos minimizar la probabilidad de errores y maximizar la tasa de descubrimiento de errores. Para lograr el primer objetivo, debemos minimizar el tamaño de la TCB. Para lograr el segundo objetivo, el proceso de diseño debe incluir retroalimentación para que podamos encontrar errores rápidamente.

El siguiente método muestra cómo construir una TCB de este tipo:

- Especificar los requisitos de seguridad para la TCB (por ejemplo, comunicación segura a través de redes no confiables). La razón principal de este paso es especificar explícitamente las suposiciones para poder decidir si son creíbles. Como parte de los requisitos, también se especifican los ataques contra los cuales está protegida la TCB para que los riesgos de seguridad sean evaluables. Al especificar qué hace y qué no hace la TCB, sabemos contra qué tipos de ataques estamos protegidos y a qué tipos somos vulnerables.
- Diseñar una TCB mínima. Utilizar buenas herramientas (como la lógica de autenticación, que discutiremos en la Sección 11.5) para expresar el diseño.
- Implementar la TCB. Es importante nuevamente utilizar buenas herramientas. Por ejemplo, los ataques de desbordamiento de búfer pueden evitarse utilizando un lenguaje que verifique los límites de los arreglos.
- Ejecutar la TCB e intentar vulnerar la seguridad.

La parte difícil en este método de diseño de varios pasos es verificar que los pasos sean consistentes: verificar que el diseño cumpla con la especificación, verificar que el diseño sea resistente a los ataques especificados, verificar que la implementación coincida con el diseño y verificar que el sistema que se ejecuta en la computadora sea el que realmente se implementó. Por ejemplo, como ha demostrado Thompson, es fácil para un adversario con experiencia en compiladores insertar Caballos de Troya en un sistema que es difícil de detectar [Consultas para lectura adicional 11.3.3 y 11.3.4].

El problema en la seguridad informática no suele ser inventar mecanismos y arquitecturas ingeniosas, sino asegurarse de que el sistema instalado cumpla realmente con el diseño y la implementación. Realizar una verificación de extremo a extremo es difícil. Por ejemplo, es común contratar a un equipo de tigres cuya misión es encontrar lagunas que podrían ser explotadas para romper la seguridad del sistema. El equipo de tigres puede encontrar algunas lagunas, pero, desafortunadamente, no puede garantizar que se hayan encontrado todas las lagunas.

El método de diseño también implica que cuando se detecta y se repara un error, el diseñador debe revisar las suposiciones para ver cuáles fueron incorrectas o faltantes, reparar las suposiciones y repetir este proceso hasta obtener suficiente confianza en la seguridad del sistema. Este enfoque elimina cualquier pensamiento confuso, hace que el sistema sea más confiable y construye lentamente la confianza de que el sistema es correcto.

El método también establece claramente qué riesgos se consideraron aceptables cuando se diseñó el sistema, porque el usuario potencial debe poder consultar la especificación para evaluar si el sistema cumple con los requisitos. Declarar qué riesgos son aceptables es importante porque gran parte del diseño de sistemas seguros está impulsado por limitaciones económicas. Los usuarios pueden considerar aceptable un riesgo de seguridad si el costo de un fallo de seguridad es pequeño en comparación con diseñar un sistema que elimine el riesgo.

# Authenticating Principals

La mayoría de las políticas de seguridad involucran a personas. Por ejemplo, una política simple podría decir que solo el propietario del archivo "x" debería poder leerlo. En esta declaración, el propietario corresponde a un humano. Para poder respaldar tal política, el servicio de archivos debe tener una forma de establecer un vínculo seguro entre un usuario del servicio y el origen de una solicitud. Establecer y verificar el vínculo son temas que caen en la provincia de la autenticación.

Volviendo a nuestro modelo de seguridad, la configuración para la autenticación se puede presentar de manera pictórica como en la Figura 11.3. Una persona (Alice) le pide a su computadora cliente que envíe un mensaje "Comprar 100 acciones de Generic Moneymaking, Inc." a su servicio de comercio electrónico favorito. Un adversario puede ser capaz de copiar el mensaje, eliminarlo, modificarlo o reemplazarlo. Como se explica en la Sección 11.1, cuando el servicio de comercio de Alice recibe este mensaje, el guardián debe establecer dos hechos importantes relacionados con la autenticidad:

1. ¿Quién es este principal que realiza la solicitud? El guardia debe establecer si el mensaje realmente proviene del principal que representa a la persona del mundo real "Alice". En un sentido más general, el guardia debe establecer el origen del mensaje.
2. ¿Es esta solicitud realmente la que hizo Alice? O, por ejemplo, ¿un adversario ha modificado el mensaje? El guardia debe establecer la integridad del mensaje.

Esta sección proporciona las técnicas para responder a estas dos preguntas.

# Separar la Confianza de la Autenticación de Principales

La autenticación consiste en identificar de manera confiable al principal asociado con una solicitud. La autenticación puede ser proporcionada por medios técnicos como contraseñas y firma de mensajes. Los medios técnicos crean una cadena de evidencia que conecta de manera segura una solicitud entrante con un principal, quizás estableciendo que un mensaje proviene del mismo principal que un mensaje anterior. Incluso los medios técnicos pueden ser capaces de establecer la identidad del mundo real del principal.

Una vez que los mecanismos de autenticación han identificado al principal, hay un problema estrechamente relacionado pero distinto: ¿se puede confiar en el principal? Los medios de autenticación pueden ser capaces de establecer que la identidad del mundo real de un principal es la persona "Alice",

pero se requieren otras técnicas para decidir si y cuánto confiar en Alice. El servicio de comercio puede decidir considerar la solicitud de Alice porque el servicio de comercio puede, por medios técnicos, establecer que el número de tarjeta de crédito de Alice es válido. Para ser más precisos, el servicio de comercio confía en que la compañía de tarjetas de crédito cumplirá con el dinero y confía en la compañía de tarjetas de crédito para establecer la confianza de que Alice pagará su factura de tarjeta de crédito.

Los problemas de autenticidad y confianza están conectados a través del nombre del principal. Los medios técnicos establecen el nombre del principal. Los nombres de los principales vienen en muchas variedades: por ejemplo, el nombre podría ser simbólico, como "Alice", un número de tarjeta de crédito, un seudónimo o una clave criptográfica. Las técnicas psicológicas establecen la confianza en el nombre del principal. Por ejemplo, un reportero podría confiar en la información de un informante anónimo que tiene un seudónimo porque el contenido previo de los mensajes conectados con el seudónimo siempre ha sido correcto.

Para hacer más clara la separación de la confianza de la autenticación de los principales, consideremos el siguiente ejemplo. Te enteras de una librería en línea llamada "ShopWithUs.com". Inicialmente, es posible que no estés seguro de qué pensar acerca de esta tienda.

Observa su sitio web, hablas con amigos que han comprado libros allí, escuchas a una persona respetable decir públicamente que esta tienda es donde compra libros esa persona, y con toda esta información desarrollas cierta confianza de que tal vez esta librería es real y es segura para ordenar. Pides un libro en ShopWithUs.com y la tienda te lo entrega más rápido de lo que esperabas. Después de un tiempo, estás ordenando todos tus libros de ellos porque te ahorra el viaje a la librería local y has descubierto que aceptan libros defectuosos sin problemas.

Desarrollar confianza en ShopWithUs.com es la parte psicológica. El nombre ShopWithUs.com es el identificador principal en el que has aprendido que puedes confiar. Es el nombre que escuchaste de tus amigos, es el nombre que le dices a tu navegador web, y es el nombre que aparece en tu factura de tarjeta de crédito. Tu confianza se basa en ese nombre; cuando recibes una oferta por correo electrónico de "ShopHere.com", la tiras a la basura porque, aunque el nombre es similar, no coincide exactamente.

Cuando realmente compras un libro en ShopWithUs.com, entra en juego la autenticación del principal. Las técnicas mecánicas te permiten establecer un enlace de comunicación segura a un sitio web que afirma ser ShopWithUs.com, y verificar que este sitio web realmente tiene el nombre ShopWithUs.com. Las técnicas mecánicas no te dicen por sí mismas con quién estás tratando; solo te aseguran que sea quien sea, se llama ShopWithUs.com. Debes decidir tú mismo (el componente psicológico) quién es esa persona y cuánto confiar en ella.

En la dirección inversa, ShopWithUs.com quisiera asegurarse de que se le pague por los libros que envía. Lo hace pidiéndote un identificador principal, tu número de tarjeta de crédito, y subcontratando a la compañía de tarjetas de crédito el componente psicológico de desarrollar confianza en que pagarás tus facturas de tarjeta de crédito. El enlace de comunicación segura entre tu navegador y el sitio web de ShopWithUs.com asegura a ShopWithUs.com que el número de tarjeta de crédito que proporcionas está asociado de forma segura con la transacción, y un enlace de comunicación seguro con la compañía de

tarjetas de crédito asegura a ShopWithUs.com que el número de tarjeta de crédito es un identificador principal válido.

## Autenticación de Principales

Cuando el servicio de comercio recibe el mensaje, el guardia sabe que el mensaje afirma provenir de la persona llamada "Alice", pero no sabe si la afirmación es cierta o no. El guardia debe verificar la afirmación de que el identificador Alice corresponde al principal que envió el mensaje.

La mayoría de los sistemas de autenticación siguen este modelo: el remitente le dice al guardia su identidad principal, y el guardia verifica esa afirmación. Este protocolo de verificación consta de dos etapas:

1. Un paso de encuentro, en el cual una persona del mundo real visita físicamente a una autoridad que configura al guardia. La autoridad verifica la identidad de la persona del mundo real, crea un identificador principal para la persona y acuerda un método mediante el cual el guardia pueda identificar más tarde el identificador principal de la persona. Es necesario ser particularmente cauteloso al verificar la identidad del mundo real de un principal porque un adversario podría ser capaz de falsificarla.
2. Una verificación de identidad, que ocurre en diversos momentos posteriores. El remitente presenta un identificador principal reclamado y el guardia utiliza el método acordado para verificar el identificador principal reclamado. Si el guardia puede verificar el identificador principal reclamado, entonces la fuente está autenticada. Si no, el guardia deniega el acceso y emite una alerta.

El método de verificación acordado por el usuario y el guardia durante el paso de encuentro cae en tres categorías amplias:

- El método utiliza una propiedad física única del usuario. Por ejemplo, se asume que caras, voces, huellas dactilares, etc., identifican de manera única a un humano. Para algunas de estas propiedades, es posible diseñar una interfaz de verificación aceptable para los usuarios: por ejemplo, un usuario pronuncia una frase en un micrófono y el sistema compara la huella vocal con una huella vocal previa almacenada.

  Para otras propiedades, es difícil diseñar una interfaz de usuario aceptable; por ejemplo, un sistema informático que pide "por favor, dé una muestra de sangre" probablemente no tendrá mucho éxito. La singularidad de la propiedad física y si es fácil de reproducir (por ejemplo, reproducir una voz grabada) determinan la fortaleza de este enfoque de identificación. La identificación física a veces es una combinación de varias técnicas (por ejemplo, voz y reconocimiento facial o de iris) y se combina con otros métodos de verificación.
- El método utiliza algo único que el usuario posee. El usuario podría tener una tarjeta de identificación con un identificador escrito en una banda magnética que puede ser leído por un ordenador. O, la tarjeta podría contener un pequeño ordenador que almacena un secreto; estas tarjetas se llaman tarjetas inteligentes. La seguridad de este método depende de (1) los usuarios que no entreguen su tarjeta a otra persona o la pierdan, y (2) que un adversario no pueda

reproducir una tarjeta que contenga el secreto (por ejemplo, copiar el contenido de la banda magnética). Estas restricciones son difíciles de aplicar, ya que un adversario podría sobornar al usuario o amenazar físicamente al usuario para que le dé la tarjeta al adversario. También es difícil hacer dispositivos a prueba de manipulaciones que no revelen su secreto.

- El método utiliza algo que solo el usuario sabe. El usuario recuerda una cadena secreta, por ejemplo, una contraseña, un número de identificación personal (PIN) o, como se presentará en la Sección 11.3, una clave criptográfica. La fortaleza de este método depende de (1) que el usuario no revele (voluntaria o involuntariamente) la contraseña y (2) de la dificultad para que un adversario adivine el secreto del usuario. El nombre de soltera de tu madre y los PIN de 4 dígitos son secretos débiles.

Por ejemplo, cuando Alice creó una cuenta de trading, es posible que el guardia le haya pedido un identificador principal y una contraseña (una cadena de caracteres secreta), que el guardia almacena. Este paso es el paso de encuentro. Más tarde, cuando Alice envía un mensaje para hacer trading, incluye en el mensaje su identificador principal reclamado ("Alice") y su contraseña, que el guardia verifica comparándola con su copia almacenada. Si la contraseña en el mensaje coincide, el guardia sabe que este mensaje proviene del principal Alice, suponiendo que Alice no haya revelado su contraseña a nadie más voluntaria o involuntariamente. Este paso es el paso de verificación.

En la autenticación en la vida real, típicamente usamos un proceso similar. Por ejemplo, primero obtenemos un pasaporte presentándonos en la oficina de pasaportes, donde respondemos preguntas, proporcionamos evidencia de nuestra identidad y una fotografía. Este paso es el paso de encuentro. Más tarde, presentamos el pasaporte en un puesto fronterizo. El guardia fronterizo examina la información en el pasaporte (altura, color de pelo, etc.) y observa cuidadosamente la fotografía. Este paso es el paso de verificación.

La seguridad de autenticar principales depende, entre otras cosas, de cuán cuidadosamente se ejecuta el paso de encuentro. Como vimos anteriormente, un proceso común es que antes de que a un usuario se le permita usar un sistema informático, el usuario debe ver a un administrador en persona y demostrar al administrador su identidad. El administrador podría pedirle al usuario potencial, por ejemplo, un pasaporte o una licencia de conducir. En ese caso, el administrador confía en la agencia que emitió el pasaporte o la licencia de conducir para hacer un buen trabajo en establecer la identidad de la persona.

En otras aplicaciones, el paso de encuentro es un procedimiento ligero y el guardia no puede confiar mucho en la identidad reclamada del principal. En el ejemplo del servicio de trading, Alice elige su identificador principal y contraseña. El servicio simplemente almacena el identificador principal y la contraseña en su tabla, pero no tiene una forma directa de verificar la identidad de Alice; es poco probable que Alice pueda ver al administrador del sistema del servicio de trading en persona porque podría estar en una computadora al otro lado del mundo.

Dado que el servicio de trading no puede verificar la identidad de Alice, el servicio no confía mucho en ninguna conexión reclamada entre el identificador principal y una persona del mundo real. La cuenta existe para la conveniencia de Alice para revisar, por ejemplo, sus operaciones; cuando ella realmente compra algo, el servicio no verifica la identidad de Alice, sino que verifica algo más (por ejemplo, el número de su tarjeta de crédito).

El servicio confía en la compañía de tarjetas de crédito para verificar el principal asociado con el número de tarjeta de crédito. Algunas compañías de tarjetas de crédito tienen esquemas de verificación débiles, que pueden ser explotados por adversarios para el robo de identidad.

# Funciones de Hash Criptográficas, Seguridad Computacional, Ventana de Validez

El método más comúnmente empleado para verificar identidades en sistemas informáticos se basa en contraseñas porque tiene una interfaz de usuario conveniente; los usuarios pueden simplemente escribir su nombre y contraseña en un teclado. Sin embargo, hay varias debilidades en este enfoque. Una debilidad es que la copia almacenada de la contraseña se convierte en un objetivo atractivo para los adversarios. Una forma de eliminar esta debilidad es almacenar un hash criptográfico de la contraseña en el archivo de contraseñas del sistema, en lugar de la contraseña misma.

Una función de hash criptográfico mapea un conjunto de bytes de tamaño arbitrario M a un valor de longitud fija V, y tiene las siguientes propiedades:

1. Para una entrada dada M, es fácil calcular $V \leftarrow H(M)$, donde H es la función de hash.
2. Es difícil calcular M conociendo solo V.
3. Es difícil encontrar otra entrada M' tal que $H(M') = H(M)$.
4. El valor calculado V es lo más corto posible, pero lo suficientemente largo como para que H tenga una baja probabilidad de colisión: la probabilidad de que dos entradas diferentes se conviertan en el mismo valor V debe ser tan baja que se pueda despreciar en la práctica. Un tamaño típico para V es de 160 a 256 bits.

El desafío en el diseño de una función de hash criptográfica es encontrar una función que tenga todas estas propiedades. En particular, proporcionar la propiedad 3 es desafiante. La Sección 11.8 describe una implementación del Algoritmo de Hash Seguro (SHA), que es una familia estándar de algoritmos hash del gobierno de EE. UU. y de la OCDE.

Las funciones de hash criptográficas, al igual que la mayoría de las funciones criptográficas, son seguras computacionalmente. Están diseñadas de tal manera que es computacionalmente inviable romperlas, en lugar de ser imposible de romper. La idea es que si se requiere un número inimaginable de años de computación para romper una función en particular, entonces podemos considerar que la función es segura.

La seguridad computacional se mide y cuantifica utilizando un factor de trabajo. Para las funciones de hash criptográficas, el factor de trabajo es la cantidad mínima de trabajo requerido para calcular un mensaje M' tal que para un M dado, $H(M') = H(M)$. El trabajo se mide en operaciones primitivas (por ejemplo, ciclos de procesador). Si el factor de trabajo es de muchos años, entonces, para todos los propósitos prácticos, la función es tan segura como una inquebrantable porque en ambos casos probablemente existe un enfoque de ataque más fácil basado en la explotación de la falibilidad humana.

En la práctica, la seguridad computacional se mide mediante un factor de trabajo histórico. El factor de trabajo histórico es el factor de trabajo basado en los algoritmos mejor conocidos actualmente y en la tecnología de vanguardia para romper una función criptográfica. Este método de evaluación corre el

riesgo de que un adversario pueda idear un mejor algoritmo para romper una función criptográfica que los actualmente conocidos, y además los cambios tecnológicos pueden reducir el factor de trabajo. Dadas las complejidades de diseñar y analizar una función criptográfica, es recomendable utilizar solo aquellas que han existido el tiempo suficiente, como SHA-256, y que han sido sometidas a una revisión pública y cuidadosa.

Los teóricos han desarrollado modelos bajo los cuales pueden hacer afirmaciones absolutas sobre la dificultad de algunas funciones criptográficas. La creación de buenos modelos que coincidan con la práctica y el análisis teórico de los componentes de seguridad es un área de investigación activa con una gran cantidad de progreso en las últimas tres décadas, pero también con muchos problemas abiertos.

Dado que d(technology)/dt es tan alto en los sistemas informáticos y la criptografía es un campo de rápido desarrollo, es una buena práctica considerar la ventana de validez para una función criptográfica específica. La ventana de validez de una función criptográfica es el mínimo entre el tiempo de compromiso de todos sus componentes. La ventana de validez para las funciones de hash criptográficas es el mínimo entre el tiempo para comprometer el algoritmo de hash y el tiempo para encontrar un mensaje M' tal que para un M dado, H(M') = H(M). La ventana de validez de un sistema de autenticación basado en contraseñas es el mínimo entre la ventana de validez del algoritmo de hash, el tiempo para probar todas las contraseñas posibles y el tiempo para comprometer una contraseña.

Un desafío en el diseño de sistemas es que la ventana de validez de una función criptográfica puede ser más corta que la vida útil del sistema. Por ejemplo, SHA, ahora denominada "SHA-0" y que produce un valor de 160 bits para V, fue publicada por primera vez en 1993 y fue reemplazada solo dos años después por SHA-1 para corregir una posible debilidad.

De hecho, en 2004, un investigador criptográfico encontró una forma de derivar sistemáticamente ejemplos de mensajes M y M' que dan el mismo valor de hash SHA-0. La investigación publicada en 2005 sugiere debilidades en SHA-1, pero hasta 2007 nadie ha encontrado una forma sistemática de comprometer ese algoritmo de hash ampliamente utilizado (es decir, para un M dado, nadie ha encontrado un M' que dé el mismo valor de H(M)). Como precaución, sin embargo, el Instituto Nacional de Estándares y Tecnología está recomendando que para 2010 los usuarios cambien a versiones de SHA (por ejemplo, SHA-256) que produzcan valores más largos para V. Un diseñador de sistemas debe estar preparado para que durante la vida útil de un sistema informático, la función de hash criptográfica pueda tener que ser reemplazada, quizás más de una vez.