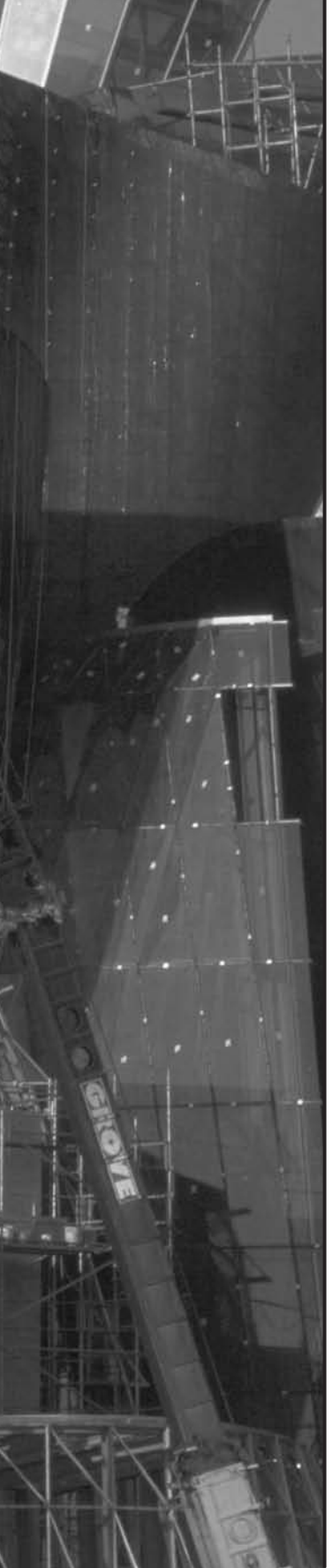

14

Ingeniería de seguridad

Objetivos

El objetivo de este capítulo es introducirlo a los temas que deben considerarse cuando se diseñan sistemas de aplicación segura. Al estudiar este capítulo:

- comprenderá la diferencia entre seguridad de aplicación y seguridad de infraestructura;
- identificará cómo la valoración del riesgo del ciclo de vida y la valoración del riesgo operativo permiten entender los conflictos de seguridad que afectan el diseño de un sistema;
- tendrá en cuenta las arquitecturas de software y los lineamientos de diseño para el desarrollo de sistemas seguros;
- aprenderá la noción de supervivencia del sistema y por qué es importante el análisis de supervivencia para sistemas de software complejos.

Contenido

14.1 Gestión del riesgo de seguridad
14.2 Diseño para la seguridad
14.3 Supervivencia del sistema

El uso extendido de Internet en la década de 1990 planteó un nuevo reto a los ingenieros de software: diseñar e implementar sistemas que fueran seguros. Conforme más y más sistemas se conectaban a Internet, se emprendió una variedad de ataques externos para amenazar estos sistemas. Los problemas para generar sistemas confiables aumentaron considerablemente. Los ingenieros de sistemas debían tomar en cuenta las amenazas de atacantes maliciosos y técnicamente hábiles, además de los problemas resultantes de errores accidentales en el proceso de desarrollo.

Ahora es esencial diseñar sistemas para soportar ataques externos y recuperarse de ellos. Sin precauciones de seguridad, es casi inevitable que los atacantes comprometerán un sistema en red. Los atacantes pueden hacer mal uso del hardware del sistema, robar datos confidenciales o dificultar los servicios que ofrece el sistema. Por lo tanto, la ingeniería de seguridad de sistemas es un aspecto cada vez más importante del proceso de ingeniería de sistemas.

La ingeniería de seguridad se interesa por el desarrollo y la evolución de los sistemas que pueden hacer frente a ataques maliciosos, cuya intención es perjudicar el sistema o los datos. La ingeniería de seguridad de software es la parte más general del campo de seguridad computacional. Ésta se ha convertido en una prioridad para las compañías y los individuos, pues más y más transgresores tratan de aprovecharse de los sistemas en red con propósitos ilícitos. Los ingenieros de software deben conocer tanto las amenazas a la seguridad que enfrentan los sistemas, como las formas en las que es posible neutralizar tales amenazas.

La intención de este capítulo es presentar la ingeniería de seguridad a los ingenieros de software, enfocándose en los conflictos de diseño que afectan la seguridad de la aplicación. Este apartado no trata la seguridad de la computadora en su conjunto ni cubre temas como encriptación, control de acceso, mecanismos de autorización, virus y caballos troyanos, etcétera, sino que describe a detalle textos generales sobre seguridad computacional (Anderson, 2008; Bishop, 2005; Pfleeger y Pfleeger, 2007).

El capítulo se suma al análisis de la seguridad que se presenta en otras partes del libro. Este material debe leerse junto con:

- Sección 10.1, que explica cómo la seguridad y la confiabilidad están estrechamente relacionadas;
- Sección 10.4, que introduce la terminología de seguridad;
- Sección 12.1, que expone la noción general de especificación dirigida por riesgo;
- Sección 12.4, que analiza los conflictos generales de la especificación de requerimientos de seguridad;
- Sección 15.3, que describe algunos enfoques a las pruebas de seguridad.

Al considerar los conflictos de seguridad, debe contemplarse tanto el software de aplicación (el sistema de control, el sistema de información, etcétera) como la infraestructura sobre la que se construye el sistema (figura 14.1). La infraestructura para aplicaciones complejas incluye:

- una plataforma de sistema operativo, tal como Linux o Windows;
- otras aplicaciones genéricas que operan en dicho sistema, tales como navegadores Web y clientes de correo electrónico;
- un sistema de gestión de base de datos;

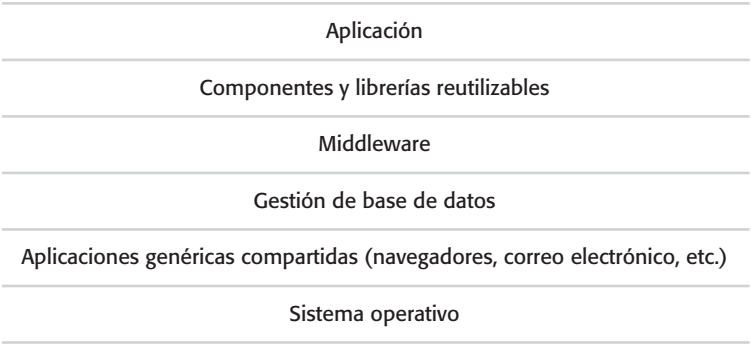

Figura 14.1 Capas de sistema donde puede comprometerse la seguridad

- middleware, que soporta computación distribuida y acceso a la base de datos;
- librerías de componentes reutilizables, que usa el software de aplicación.

La mayoría de los ataques externos se enfocan en infraestructuras de sistema debido a que los componentes de infraestructura (por ejemplo, navegadores Web) son bien conocidos y ampliamente disponibles. Los atacantes pueden probar dichos sistemas para localizar puntos débiles y buscar información compartida sobre las vulnerabilidades descubiertas. Puesto que muchas personas usan el mismo software, los ataques tienen amplia aplicabilidad. Las vulnerabilidades de infraestructura pueden conducir a que los atacantes obtengan sin autorización acceso a un sistema de aplicación y a los datos.

En la práctica, existe una importante diferencia entre seguridad de aplicación y seguridad de infraestructura:

1. La seguridad de aplicación es un problema de ingeniería de software; los ingenieros de software deben asegurarse de que el sistema se diseñó para soportar ataques.
2. La seguridad de infraestructura es un problema de gestión en el que los administradores del sistema configuran la infraestructura para resistir ataques. Los administradores de sistema deben configurar la infraestructura con la finalidad de usar de manera más efectiva cualquier característica de seguridad de infraestructura disponible. También deben corregir las vulnerabilidades de seguridad de infraestructura que salgan a la luz conforme se use el software.

La gestión de la seguridad del sistema no es una tarea sencilla, sino que debe incluir un rango de actividades como gestión de usuarios y permisos, implementación y mantenimiento del software del sistema, y monitorización, detección y recuperación de ataque.

1. La gestión de usuarios y permisos implica agregar y eliminar usuarios del sistema, asegurar que se coloquen mecanismos adecuados de autenticación de usuarios, y configurar los permisos en el sistema de forma que los usuarios sólo tengan acceso a los recursos que necesitan.
2. La implementación y el mantenimiento del software del sistema comprenden la instalación del software y middleware del sistema y configurar éste de manera adecuada, además de evitar las vulnerabilidades de seguridad. Asimismo, incluyen la

Ataques internos e ingeniería social

Los ataques internos son ataques a un sistema realizados por un individuo de confianza (un empleado de la organización) que abusa de la situación. Por ejemplo, una enfermera que trabaja en un hospital puede tener acceso a registros médicos confidenciales de pacientes a quienes no atiende. Los ataques internos son difíciles de contrarrestar porque las técnicas adicionales de seguridad que pudieran usarse afectarían a los usuarios confiables del sistema.

La ingeniería social es una forma de engañar a los usuarios acreditados para que muestren sus credenciales. En consecuencia, un atacante puede comportarse como empleado de la organización cuando ingresa al sistema.

http://www.SoftwareEngineering-9.com/Web/SecurityEng/insiders.html

actualización regular de este software con nuevas versiones o parches, que reparen los problemas de seguridad descubiertos.

3. La monitorización del ataque, detección y recuperación incluyen actividades que monitorizan el sistema en relación con acceso no autorizado, detectan e instauran estrategias para resistir ataques, así como actividades de respaldo de manera que, después de un ataque externo, pueda reanudarse la operación normal.

La gestión de la seguridad tiene una importancia vital, pero no suele considerarse como parte de la ingeniería de seguridad de aplicación. En vez de ello, este tipo de ingeniería se interesa por el diseño de un sistema que sea tan seguro como sea posible, dadas las restricciones presupuestales y de uso. Parte de este proceso consiste en "diseñar para administrar", es decir, diseñar sistemas para minimizar las probabilidades de errores en la administración de la seguridad que permitan ataques al sistema.

En sistemas de control crítico y sistemas embebidos, es práctica normal seleccionar una infraestructura adecuada para apoyar el sistema de aplicación. Por ejemplo, los desarrolladores de sistemas embebidos eligen por lo general un sistema operativo de tiempo real que ofrezca a la aplicación embebida las facilidades que necesita. Deben considerarse las vulnerabilidades conocidas y los requerimientos de seguridad, lo que significa que para la ingeniería de seguridad puede adoptarse un enfoque holístico. Los requerimientos de seguridad de aplicación se implementan mediante la infraestructura o la aplicación en sí.

Sin embargo, los sistemas de aplicación en una organización por lo general se implementan utilizando la infraestructura existente (sistema operativo, base de datos, etcétera). Por lo tanto, deben considerarse los riesgos de utilizar dicha infraestructura y las características de seguridad como parte del proceso de diseño del sistema.

14.1 Gestión del riesgo de seguridad

Para la ingeniería de seguridad efectiva son esenciales tanto la valoración como la gestión del riesgo de seguridad. Esta última se encarga de evaluar las posibles pérdidas que se derivan de ataques a los activos en el sistema, y de equilibrar dichas pérdidas frente

a los costos de los procedimientos de seguridad encaminados a reducir las pérdidas. Las compañías de tarjetas de crédito actúan continuamente de esa forma. Es relativamente sencillo introducir nueva tecnología para reducir la probabilidad de fraude con las tarjetas de crédito. Sin embargo, con frecuencia es más económico que las empresas compensen a los usuarios por sus pérdidas como consecuencia de los fraudes, que comprar e implementar tecnología que permita reducir estos últimos. Conforme los costos disminuyen y aumentan los ataques, este balance puede cambiar. Por ejemplo, las compañías de tarjetas de crédito ahora codifican información en un chip sobre la tarjeta y no sobre la banda magnética. Esto hace que sea mucho más difícil copiar la tarjeta.

La gestión del riesgo es un conflicto empresarial más que un conflicto técnico, así que los ingenieros de software no son quienes deben decidir qué controles incluir en un sistema. Es responsabilidad de los altos ejecutivos de la empresa decidir si aceptan el costo de la seguridad o se exponen a las consecuencias de una falta de procedimientos de seguridad. En cambio, el papel de los ingenieros de software es ofrecer guía y orientación técnica informada acerca de conflictos de seguridad. Por lo tanto, ellos son participantes esenciales dentro del proceso de gestión del riesgo.

Como se explicó en el capítulo 12, una entrada crítica al proceso de valoración y gestión del riesgo es la política de seguridad de la organización. Ésta se aplica a todos los sistemas y establece lo que debe permitirse y lo que no. La política de seguridad establece las condiciones que deben mantenerse siempre mediante un sistema de seguridad, y además ayuda a identificar los riesgos y las amenazas que pudieran surgir. En consecuencia, la política de seguridad define lo que se permite y lo que no se permite. En el proceso de ingeniería de seguridad se diseñan los mecanismos para aplicar esta política.

La valoración del riesgo se inicia antes de decidir la adquisición del sistema y debe continuar a lo largo del proceso de desarrollo de éste, y después de que se pone en uso (Alberts y Dorofee, 2002). En el capítulo 12 se introdujo la idea de que esta valoración del riesgo es un proceso por etapas:

1. *Valoración preliminar del riesgo* En esta etapa aún no se toman decisiones sobre los requerimientos detallados del sistema, del diseño o de la tecnología de implementación. La meta de este proceso de valoración es decidir si puede lograrse un nivel adecuado de seguridad a un costo razonable. Si éste es el caso, entonces es posible establecer requerimientos de seguridad específicos para el sistema. No se tiene información de las vulnerabilidades potenciales en el sistema o los controles que se incluyen en componentes reutilizados del sistema o middleware.

2. *Valoración del riesgo del ciclo de vida* Esta valoración del riesgo tiene lugar durante el ciclo de vida de desarrollo del sistema, y es un informe de las decisiones técnicas de diseño e implementación del sistema. Los resultados de la valoración pueden conducir a cambios en los requerimientos de seguridad y a agregar otros. Se identifican las vulnerabilidades conocidas y potenciales. Este conocimiento se emplea para la toma de decisiones informada sobre la funcionalidad del sistema y de cómo ésta se implementará, probará y desplegará.

3. *Valoración del riesgo operativo* Después de que un sistema se implementa y pone en operación, se debe realizar una valoración del riesgo para tomar en cuenta cómo se usa el sistema y las propuestas para los nuevos y cambiantes requerimientos. Las suposiciones referentes a los requerimientos operativos que se hicieron cuando

el sistema se especificaba pueden ser incorrectas. Los cambios en la organización significan que el sistema podría utilizarse en diferentes formas a las planeadas originalmente. Por consiguiente, la valoración del riesgo operativo conduce a nuevos requerimientos de seguridad que deben implementarse a medida que evoluciona el sistema.

La valoración preliminar del riesgo se enfoca en la derivación de requerimientos de seguridad. En el capítulo 12 se mostró cómo puede derivarse un conjunto inicial de requerimientos de seguridad a partir de una valoración preliminar del riesgo. Esta sección se concentra en la valoración del riesgo del ciclo de vida y operacional, para ilustrar cómo la especificación y el diseño de un sistema reciben influencia de la tecnología y de las formas como se utiliza el sistema.

Para realizar una valoración del riesgo, es necesario identificar las posibles amenazas a un sistema. Una forma de hacerlo es desarrollar un conjunto de "casos de mal uso" (Alexander, 2003; Sindre y Opdahl, 2005). Ya se examinó cómo utilizar los casos de uso, es decir, las interacciones típicas con un sistema, para derivar los requerimientos de este último. Los casos de mal uso son escenarios que representan interacciones maliciosas con un sistema. Al desarrollar casos de mal uso es posible analizar e identificar posibles amenazas y, por lo tanto, también determinar los requerimientos de seguridad del sistema. Pueden usarse junto con los casos de uso para establecer los requerimientos del sistema.

Pfleeger y Pfleeger (2007) clasifican las amenazas en cuatro categorías, las cuales constituyen un punto de partida para identificar posibles casos de mal uso. Esas categorías son las siguientes:

1. Amenazas de intercepción, que permiten a un atacante conseguir acceso a un activo. De este modo, un posible caso de mal uso para el MHC-PMS puede ser una situación en que un atacante logra tener acceso a los registros de un paciente que es una celebridad.
2. Amenazas de interrupción, las cuales permiten a un atacante tomar parte de la indisposición del sistema. Por ejemplo, un posible caso de mal uso es un ataque de negación de servicio al servidor de la base de datos del sistema.
3. Amenazas de modificación, que permiten a un atacante sabotear un activo del sistema. En el MHC-PMS, esto podría representarse con un caso de mal uso en que un atacante cambia la información en el registro de un paciente.
4. Amenazas de fabricación, las cuales permiten a un atacante insertar información falsa en un sistema. Posiblemente ésta no es una amenaza verosímil en el MHC-PMS, pero sin duda sería una amenaza en un sistema bancario, donde podrían agregarse transacciones falsas al sistema para transferir dinero a la cuenta bancaria del transgresor.

Los casos de mal uso no sólo son útiles en la valoración preliminar del riesgo, sino también pueden usarse para analizar la seguridad en términos del riesgo del ciclo de vida y el riesgo operativo. Proporcionan una base útil para efectuar ataques hipotéticos al sistema y valorar las implicaciones de seguridad de las decisiones de diseño realizadas.

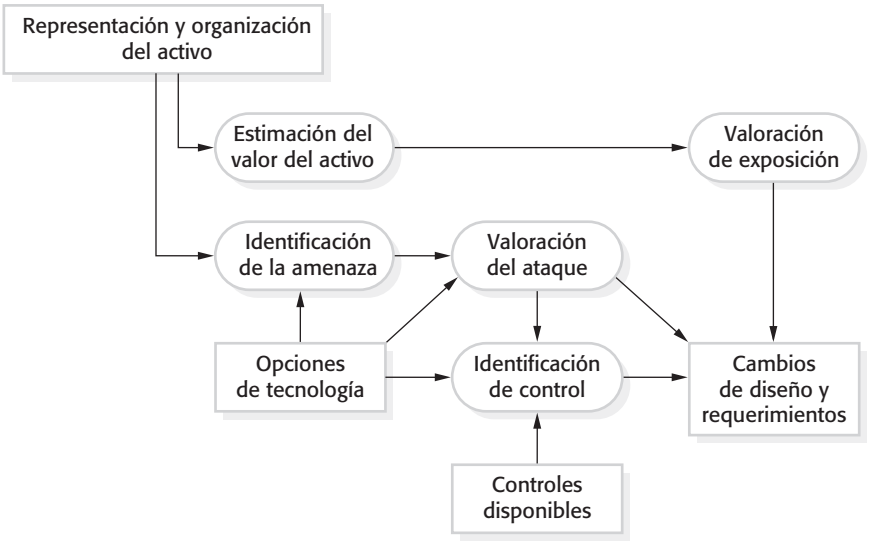

Figura 14.2 Análisis del riesgo del ciclo de vida

14.1.1 Valoración del riesgo del ciclo de vida

Con base en las políticas de seguridad de la organización, la valoración preliminar del riesgo debe identificar los requerimientos de seguridad más importantes para un sistema. Ello refleja cómo se debe implementar la política de seguridad en dicha aplicación, permite identificar los activos a proteger y ayuda a decidir qué enfoque podría usarse para ofrecer tal protección. Mantener la seguridad implica poner atención a detalle. Sin embargo, es imposible que los requerimientos iniciales de seguridad tomen en cuenta todos los detalles que afectan la seguridad.

La valoración del riesgo del ciclo de vida identifica los detalles de diseño e implementación que afectan la seguridad. Ésta es la diferencia importante entre valoración del riesgo del ciclo de vida y valoración preliminar del riesgo. La valoración del riesgo del ciclo de vida afecta la interpretación de los requerimientos de seguridad existentes, genera nuevos requerimientos e influye en el diseño global del sistema.

Cuando se valoran riesgos en esta etapa, es indispensable tener información mucho más detallada sobre qué se necesita proteger, y también conocer algo referente a las vulnerabilidades del sistema. Algunas de estas vulnerabilidades serán inherentes a las decisiones que se tomen en materia de diseño. Por ejemplo, una vulnerabilidad en todos los sistemas basados en contraseñas es que algún usuario autorizado revele su contraseña a otro usuario no autorizado. O bien, si una organización tiene una política para desarrollar software en C, sabrá que la aplicación puede tener vulnerabilidades, pues el lenguaje no incluye comprobación de límite de arreglo.

La valoración del riesgo de seguridad debe ser parte de las actividades del ciclo de vida de la ingeniería, desde la ingeniería de requerimientos hasta la implementación del sistema. El proceso que se sigue es similar al proceso de valoración preliminar del riesgo con la adición de actividades concernientes a la identificación y valoración de la vulnerabilidad del diseño. El resultado de la valoración del riesgo es un conjunto de decisiones de ingeniería que afectan el diseño o la implementación del sistema, o limitan la forma en que se usa.

En la figura 14.2 se muestra un modelo del proceso de análisis del riesgo del ciclo de vida, basado en el proceso de análisis preliminar del riesgo que se describe en la figura 12.9. La diferencia más importante entre dichos procesos es que ahora se tiene información de la representación y distribución de información, así como de la organización de la base de datos para los activos de alto nivel que deben protegerse. También se está al tanto de importantes decisiones de diseño, tales como el software a reutilizar, los controles y la protección de infraestructura, etcétera. Con base en esta información, el análisis permite identificar cambios necesarios en los requerimientos de seguridad y en el diseño del sistema para brindar protección adicional a los activos importantes del sistema.

Dos ejemplos ilustran cómo los requerimientos de protección reciben influencia de las decisiones concernientes a la representación y distribución de información:

1. Es posible tomar una decisión de diseño para separar la información personal de un paciente y la información de los tratamientos recibidos mediante una clave que vincule ambos registros. La información del tratamiento es mucho menos sensible que la información personal del paciente, de manera que quizá no necesite una protección tan profunda. Si la clave está protegida, entonces un atacante sólo podrá tener acceso a información de rutina, sin poder vincular esto con los datos individuales de un paciente.
2. Suponga que, al comienzo de la sesión, se toma una decisión de diseño para copiar registros de pacientes a un sistema cliente local. Esto permite que el trabajo continúe si el servidor no está disponible. Ello posibilita que un empleado de atención a la salud tenga acceso a los registros de los pacientes desde una laptop, incluso si no está disponible una conexión a red. Sin embargo, ahora tiene dos conjuntos de registros para proteger y las copias del cliente están sujetas a riesgos adicionales, como el robo de la laptop. Por lo tanto, habrá que pensar en qué controles ayudarían a reducir el riesgo. Por ejemplo, es posible que deban encriptarse los registros del cliente en la laptop.

Para ilustrar cómo las decisiones sobre las tecnologías de desarrollo influyen en la seguridad, suponga que el proveedor de atención a la salud decide construir un MHC-PMS mediante un sistema de información comercial para mantener registros de pacientes. Este sistema tiene que configurarse para cada tipo de clínica en la que se utilice. Esta decisión se toma porque parece ofrecer la funcionalidad más amplia en términos del costo de desarrollo más bajo y el tiempo de despliegue más rápido.

Cuando se elabora una aplicación con la reutilización de un sistema existente, deben aceptarse las decisiones de diseño tomadas por los desarrolladores de dicho sistema. Suponga que algunas de tales decisiones de diseño son las siguientes:

1. Los usuarios del sistema se autentican a través de una combinación de nombre/contraseña de acceso. No existe otro método de autenticación.
2. La arquitectura del sistema es cliente-servidor, y los clientes acceden a los datos mediante un navegador Web estándar en una PC cliente.
3. La información se presenta a los usuarios como un formato Web editable. Ellos pueden cambiar la información en el lugar y subir al servidor la información revisada.

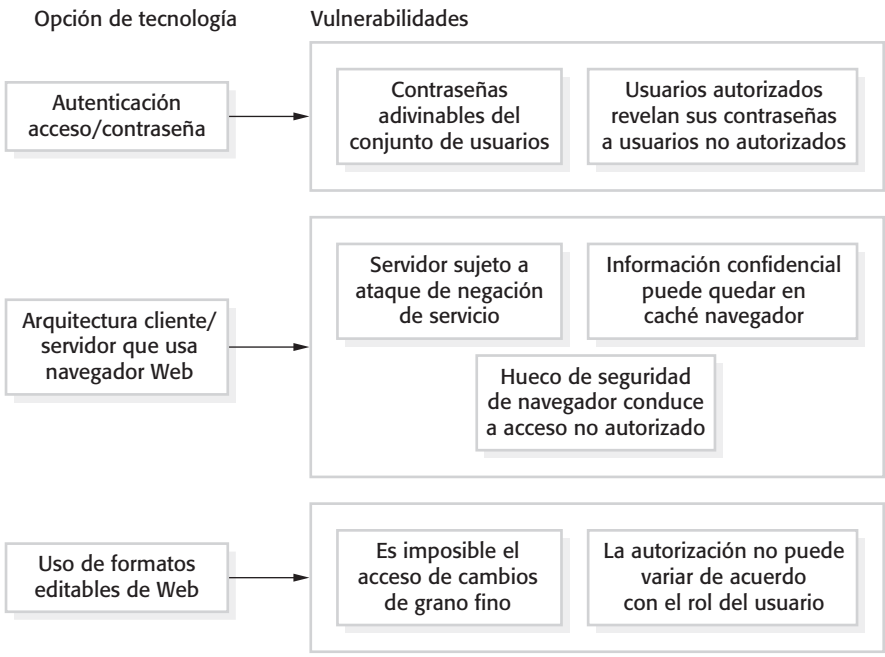

Figura 14.3 Vulnerabilidades asociadas con las opciones de tecnología

Para un sistema genérico, estas decisiones de diseño son perfectamente aceptables, pero un análisis del riesgo del ciclo de vida revela que tienen vulnerabilidades asociadas. En la figura 14.3 se presentan ejemplos de posibles vulnerabilidades.

Una vez identificadas las vulnerabilidades, debe tomarse entonces una decisión sobre qué pasos habrá que dar para reducir los riesgos asociados. Con frecuencia, esto implicará la toma de decisiones respecto a los requerimientos de seguridad de sistema adicional o el proceso operacional de usar el sistema. Aun cuando aquí no hay espacio para analizar todos los requerimientos que pueden proponerse al examinar las vulnerabilidades inherentes, algunos ejemplos de requerimientos son los siguientes:

1. Un programa de comprobación de contraseña debe estar disponible y funcionar continuamente. Las contraseñas de los usuarios que aparezcan en el diccionario del sistema deben identificarse; los usuarios con contraseñas vulnerables deben reportarse a los administradores del sistema.
2. El acceso al sistema sólo debe permitirse a computadoras cliente que aprobaron y registraron los administradores del sistema.
3. Todas las computadoras cliente deben tener un solo navegador Web instalado, y éste debe recibir la aprobación de los administradores del sistema.

Conforme se use un sistema comercial, no será posible incluir un verificador de contraseña en el sistema de aplicación en sí, de manera que se debe usar un sistema separado. Los verificadores de contraseñas analizan la fortaleza de las contraseñas de los usuarios cuando se configuran, y notifican a los usuarios si eligieron contraseñas vulnerables. Por consiguiente, las contraseñas vulnerables pueden identificarse de manera relativamente

rápida después de que se configuraron, y entonces será posible emprender una acción para garantizar que los usuarios cambien sus contraseñas.

El segundo requerimiento y el tercero significan que todos los usuarios siempre accederán al sistema a través del mismo navegador. Se puede decidir cuál es el navegador más seguro cuando el sistema se despliega e instala en todas las computadoras cliente. Las actualizaciones de seguridad se simplifican porque no hay necesidad de actualizar diferentes navegadores cuando se descubren y corrigen vulnerabilidades de seguridad.

14.1.2 Valoración del riesgo operativo

La valoración del riesgo de seguridad podría continuar a lo largo de la vida del sistema con la finalidad de identificar riesgos emergentes y cambios al sistema que puedan requerirse para lidiar con tales riesgos. Este proceso se llama valoración del riesgo operativo. Es probable que surjan nuevos riesgos debido a requerimientos cambiantes del sistema, cambios en la infraestructura del sistema, o cambios en el entorno donde se utiliza el sistema.

El proceso de valoración del riesgo operativo es similar al proceso de valoración del riesgo del ciclo de vida, pero implica agregar más información acerca del entorno en el que se emplea el sistema. El entorno es importante porque sus características pueden conducir al sistema a nuevos riesgos. Por ejemplo, suponga que un sistema se usará en un entorno donde los usuarios son interrumpidos con frecuencia. Un riesgo es que la interrupción signifique que el usuario deje de prestar atención a su computadora. En tal caso, es posible que una persona no autorizada consiga acceso a la información en el sistema. Entonces esto podría generar un requerimiento para que un protector de pantalla solicite una contraseña para operar después de un breve periodo de inactividad.

14.2 Diseño para la seguridad

En general, es cierto que resulta muy difícil adicionar seguridad a un sistema después de que se implementó. Por lo tanto, se deben tomar en cuenta los riesgos de seguridad durante el proceso de diseño del sistema. Esta sección se enfocará principalmente en los conflictos del diseño de un sistema, porque, por lo regular, en los libros de seguridad computacional no se concede a este tema la atención que merece. Los conflictos y errores de implementación también tienen una gran repercusión sobre la seguridad, pero éstos dependen normalmente de la tecnología específica utilizada. Se recomienda leer el libro de Viega y McGraw (2002), como una buena introducción a la programación para la seguridad.

Aquí el enfoque será sobre algunos conflictos generales, independientes de aplicación, relevantes para el diseño de sistemas seguros:

1. Diseño arquitectónico: ¿cómo afectan las decisiones de diseño arquitectónico la seguridad de un sistema?
2. Buena práctica: ¿qué se acepta como buena práctica al diseñar sistemas seguros?
3. Diseño para implementación: ¿qué soporte debe diseñarse en los sistemas para evitar la introducción de vulnerabilidades cuando un sistema se despliega para su uso?

Ataques de negación de servicio

Los ataques de negación de servicio tratan de derrumbar un sistema en red al bombardearlo con un enorme número de peticiones de servicio. Esto supone una carga adicional sobre el sistema, para la cual no está diseñado, y excluye las peticiones legítimas de servicio. En consecuencia, el sistema puede volverse inaccesible porque se cae ante la enorme carga, o bien, los administradores del sistema deben sacarlo de línea para detener el flujo de peticiones.

http://www.SoftwareEngineering-9.com/Web/Security/DoS.html

Desde luego, éstos no son los únicos conflictos de diseño importantes para la seguridad. Toda aplicación es diferente y el diseño de seguridad también debe tomar en cuenta el propósito, el carácter crítico y el entorno operacional de la aplicación. Por ejemplo, si se diseña un sistema militar, es necesario adoptar un modelo de clasificación de seguridad (secreto, ultrasecreto, etcétera). Si se diseña un sistema que mantiene información personal, debe considerarse la legislación sobre protección de datos que impone restricciones respecto a cómo se administra la información.

Existe una relación estrecha entre confiabilidad y seguridad. El uso de redundancia y diversidad, que es fundamental para lograr confiabilidad, puede significar que un sistema resista y se recupere de ataques dirigidos a características o diseños específicos de implementación. Mecanismos de apoyo a alto nivel de disponibilidad pueden ayudar al sistema a recuperarse de los llamados ataques de negación de servicio, en los que la meta del atacante es hacer caer al sistema y detener su funcionamiento correcto.

Diseñar un sistema para que sea seguro implica inevitablemente un compromiso. Desde luego, es posible desarrollar múltiples medidas de seguridad en un sistema, las cuales reducirán las probabilidades de que un ataque logre su cometido. Sin embargo, las medidas de seguridad requieren a menudo una gran labor adicional de computación y, además, afectan el rendimiento global de un sistema. Por ejemplo, se pueden reducir las posibilidades de que se muestre información confidencial al encriptar dicha información. No obstante, esto significa que los usuarios de la información tienen que esperar a que ésta se desencripte, lo cual hará más lento su trabajo.

También existen tensiones entre seguridad y usabilidad. En ocasiones, las medidas de seguridad requieren que el usuario recuerde y proporcione información adicional (por ejemplo, contraseñas múltiples). Sin embargo, algunas veces el usuario olvida dicha información, de manera que la seguridad adicional significa que no puede usar el sistema. Por consiguiente, los diseñadores deben encontrar un equilibrio entre seguridad, rendimiento y usabilidad. Esto dependerá del tipo de sistema y de dónde se utilizará. Por ejemplo, en un sistema militar, los usuarios están familiarizados con los sistemas de alta seguridad, y también están deseosos de aceptar y seguir procesos que requieren comprobaciones periódicas. Sin embargo, en un sistema de comercio de acciones, las interrupciones de la operación para comprobaciones de seguridad serían completamente inaceptables.

14.2.1 Diseño arquitectónico

Como se estudió en el capítulo 11, la elección de arquitectura de software puede tener profundos efectos sobre las propiedades emergentes de un sistema. Si se usa una arquitectura inadecuada, tal vez sea muy difícil mantener en el sistema la confidencialidad e

integridad de la información, o garantizar un nivel requerido de disponibilidad del sistema.

Al diseñar una arquitectura de sistema que conserve la seguridad, se deben considerar dos temas fundamentales:

1. Protección: ¿cómo debe organizarse el sistema de manera que puedan protegerse los activos críticos contra ataques externos?
2. Distribución: ¿cómo deben distribuirse los activos del sistema de manera que sean mínimos los efectos de un ataque?

Estos temas son potencialmente conflictivos. Si se colocan todos los activos en un lugar, entonces es posible construir capas de protección en torno a ellos. Puesto que se debe construir un solo sistema de protección, es posible costear un sistema fuerte con muchas capas de protección. Si a pesar de ello, dicha protección fracasa, entonces todos los activos estarán comprometidos. Añadir más capas de protección también afecta la usabilidad de un sistema, lo que puede significar que sea más difícil satisfacer los requerimientos de usabilidad y rendimiento del sistema.

Por otra parte, si se distribuyen los activos, será más costoso protegerlos, ya que deben implementarse sistemas de protección para cada copia. Normalmente, no es posible costear tantas capas de protección. Hay más posibilidades de que la protección se rompa. Pero, si esto sucede, no habrá una pérdida total. Es posible duplicar y distribuir activos de información de manera que, si una copia se corrompe o es inaccesible, puede usarse entonces la otra copia. No obstante, si la información es confidencial, el conservar copias adicionales aumenta el riesgo de que un intruso obtenga el acceso a esta información.

Para el sistema de registros de pacientes, es adecuado usar una arquitectura de base de datos centralizada. Para brindar protección, se emplea una arquitectura en capas con los activos críticos protegidos en el nivel más bajo del sistema, con varias capas de protección en torno a ellos. La figura 14.4 ilustra esto para el sistema de registros de pacientes en el que los activos críticos a proteger son los registros individuales de cada paciente.

Si un atacante quiere tener acceso a los registros de pacientes y modificarlos, debe penetrar tres capas del sistema:

1. *Protección a nivel de plataforma* El nivel superior controla el acceso a la plataforma donde opera el sistema de registro de pacientes. Por lo general, esto implica el ingreso de un usuario a una computadora particular. La plataforma también incluirá comúnmente soporte para mantener la integridad de los archivos en el sistema, respaldos, etcétera.
2. *Protección a nivel de aplicación* El siguiente nivel de protección se construye en la aplicación en sí. Implica el acceso de un usuario a la aplicación, su autenticación y la obtención de permiso para realizar acciones como ver o modificar datos. Puede estar disponible soporte de gestión de integridad específico de la aplicación.
3. *Protección a nivel de registro* Este nivel se demanda cuando se requiere acceso a registros específicos, e incluye comprobar que un usuario está autorizado para realizar las operaciones solicitadas sobre tal registro. La protección a este nivel también

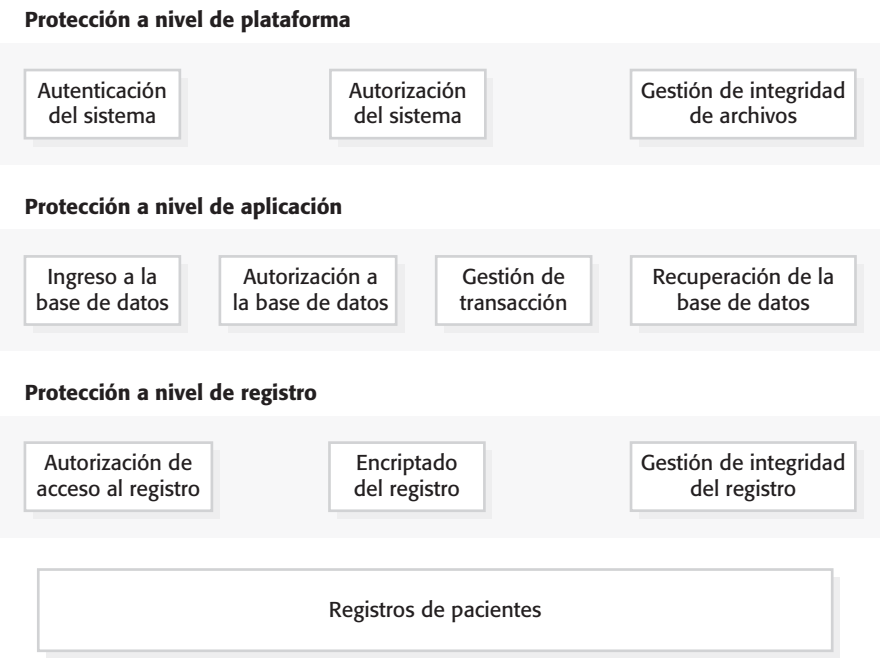

Figura 14.4 Arquitectura de protección en capas

podría implicar encriptado para garantizar que sea imposible consultar los registros utilizando un navegador de archivo. La comprobación de integridad mediante sumas de verificación criptográficas, por ejemplo, permite detectar cambios que se hayan realizado fuera de los mecanismos normales de actualización de registro.

El número de capas de protección que se necesitan en cualquier aplicación particular depende del carácter crítico de los datos. No todas las aplicaciones requieren protección a nivel registro y, por lo tanto, se usa comúnmente control de acceso de grano más grueso. Para lograr seguridad, no debe permitirse el uso de las mismas credenciales de usuario en cada nivel. Lo ideal sería que si se tiene un sistema basado en contraseñas, entonces la contraseña de la aplicación debe ser diferente tanto de la contraseña del sistema como de la contraseña a nivel de registro. Sin embargo, es difícil que los usuarios recuerden múltiples contraseñas y encuentran molestas las reiteradas peticiones de autenticación. Por ende, con frecuencia se tendrá que comprometer la seguridad en favor de la usabilidad del sistema.

Si la protección de datos es un requerimiento crítico, debe usarse en tal caso una arquitectura cliente-servidor, con los mecanismos de protección construidos en el servidor. No obstante, si se compromete la protección, es probable que las pérdidas asociadas con un ataque sean altas, al igual que los costos de recuperación (por ejemplo, tienen que emitirse nuevamente todas las credenciales de usuario). El sistema es vulnerable a los ataques de negación de servicio, que sobrecargan el servidor y hacen imposible que alguien acceda a la base de datos del sistema.

Si se considera que los ataques de negación de servicio son un riesgo mayor, se puede optar por una arquitectura de objeto distribuida para la aplicación. En esta situación, que se ilustra en la figura 14.5, los activos del sistema se distribuyen a través de algunas plataformas diferentes, con mecanismos de protección separados para cada una de ellas. Un

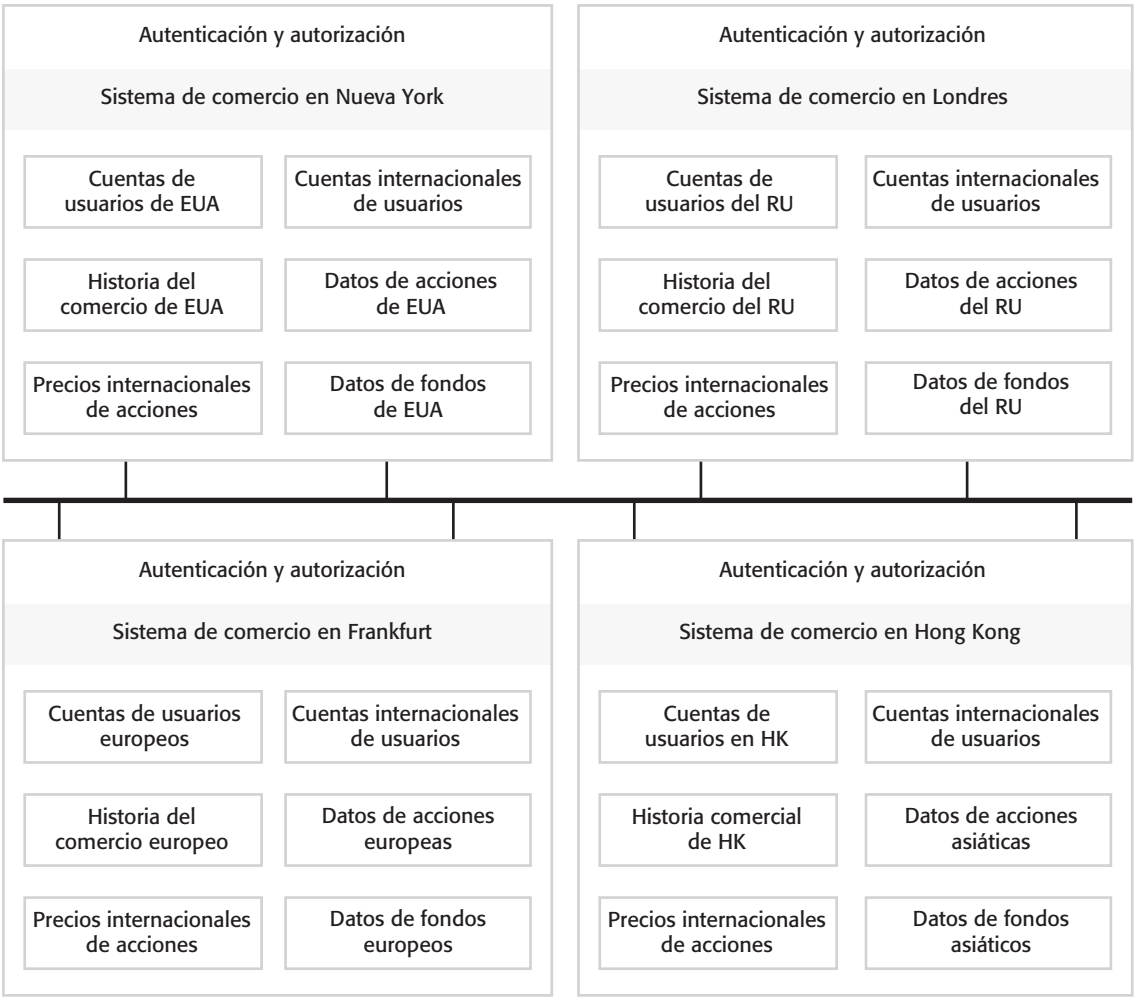

Figura 14.5 Activos distribuidos en un sistema de comercio de acciones

ataque en un nodo puede significar que algunos activos no estén disponibles, pero todavía es posible ofrecer algunos servicios del sistema. Es posible duplicar datos a través de los nodos en el sistema, de forma que se simplifica la recuperación de ataques.

La figura 14.5 muestra la arquitectura de un sistema bancario para comerciar acciones y fondos en los mercados de Nueva York, Londres, Frankfurt y Hong Kong. El sistema es distribuido, así que los datos de cada mercado se mantienen por separado. Los activos requeridos para soportar la actividad crítica de comercio de acciones (cuentas de usuarios y precios) se duplican y están disponibles en todos los nodos. Si un nodo del sistema sufre un ataque y deja de estar disponible, la actividad crítica del comercio de acciones puede transferirse a otro país y, por lo tanto, seguir disponible para los usuarios.

Ya se habló del problema de encontrar un equilibrio entre seguridad y rendimiento del sistema. Un problema de diseño seguro del sistema es que en muchos casos el estilo arquitectónico, que es más adecuado para satisfacer los requerimientos de seguridad, tal vez no sea el mejor para satisfacer los requerimientos de rendimiento. Por ejemplo, suponga que una aplicación tiene un requerimiento absoluto para mantener la confidencialidad de una

gran base de datos, y otro requerimiento para acceso muy rápido a dichos datos. Un alto nivel de protección sugiere la necesidad de capas de protección, lo que significa que debe haber comunicaciones entre las capas del sistema. Esto tiene una inevitable sobrecarga de rendimiento, lo que hace más lento el acceso a los datos. Si se usa una arquitectura alternativa, entonces quizá sea más difícil y costoso implementar protección y garantizar confidencialidad. Ante tal situación, es necesario discutir los conflictos inherentes con el cliente del sistema y acordar cómo se resolverán.

14.2.2 Lineamientos de diseño

No existen reglas rígidas y rápidas sobre cómo lograr seguridad del sistema. Diferentes tipos de sistemas requieren distintas medidas técnicas para lograr un nivel de seguridad que sea aceptable para el propietario del sistema. Las actitudes y los requerimientos de diversos grupos de usuarios afectan profundamente lo que es aceptable y lo que no lo es. Por ejemplo, en un banco, es probable que los usuarios acepten un alto nivel de seguridad, y más procedimientos de seguridad anti-intrusos en comparación con los usuarios en una universidad.

Sin embargo, existen lineamientos generales con amplia aplicabilidad al diseñar soluciones de seguridad de un sistema, los cuales encapsulan buenas prácticas de diseño para la ingeniería de sistemas seguros. Los lineamientos generales de diseño para seguridad, como los que se analizan más adelante, tienen dos usos principales:

1. Ayudan a crear conciencia acerca de los temas de seguridad en un equipo de ingeniería de software. Los ingenieros de software se enfocan con frecuencia en la meta a corto plazo de hacer el software operativo y entregarlo a los clientes. Para ellos es fácil pasar por alto los temas de seguridad. El conocimiento de dichos lineamientos puede significar que los temas de seguridad se consideren cuando se toman las decisiones de diseño de software.
2. Pueden usarse como una lista de verificación que será útil en el proceso de validación del sistema. A partir de los lineamientos de nivel superior estudiados aquí, es posible derivar cuestiones más específicas que examinen cómo se sometió a ingeniería la seguridad de un sistema.

Los 10 lineamientos de diseño, que se resumen en la figura 14.6, resultaron de varias fuentes (Schneier, 2000; Viega y McGraw, 2002; Wheeler, 2003). Aquí el enfoque es sobre los lineamientos que son aplicables en particular a los procesos de especificación y diseño del software. Principios más generales, como "Asegura el vínculo más débil del sistema", "Hazlo simple" y "Evita seguridad mediante oscuridad" también son importantes, pero menos relevantes directamente para la toma de decisiones de ingeniería.

Lineamiento 1: Decisiones de seguridad con base en una política explícita de seguridad

Una política de seguridad es un enunciado de alto nivel que establece las condiciones fundamentales de seguridad para una organización. Define el "qué" de la seguridad en lugar del "cómo", de manera que la política no debe definir los mecanismos a usar para

Lineamientos de seguridad
1. Decisiones de seguridad con base en una política explícita de seguridad.
2. Evite un solo punto de falla.
3. Seguridad en caso de falla.
4. Equilibrio entre seguridad y usabilidad.
5. Registre las acciones del usuario.
6. Use redundancia y diversidad para reducir el riesgo.
7. Valide todas las entradas.
8. Compartimente sus activos.
9. Diseñe para implementar.
10. Diseñe para facilitar la recuperabilidad.

Figura 14.6 Lineamientos de diseño para la ingeniería de sistemas seguros

ofrecer seguridad y reforzarla. En un inicio, todos los aspectos políticos de seguridad deben reflejarse en los requerimientos del sistema. En la práctica, en especial si se usa un proceso rápido de desarrollo de aplicación, es improbable que esto suceda. Por lo tanto, los diseñadores deben consultar la política de seguridad, ya que ésta constituye un marco para tomar y evaluar las decisiones de diseño.

Por ejemplo, suponga que se diseña un sistema de control de acceso para el MHC-PMS. La política de seguridad del hospital puede establecer que sólo el personal médico acreditado está autorizado para modificar los registros electrónicos de los pacientes. De ahí que el sistema deba incluir mecanismos que comprueben la acreditación de cualquiera que intente modificar el sistema, y que rechacen modificaciones de personas no acreditadas.

El problema que se enfrenta es que muchas organizaciones no tienen una política explícita de seguridad de sistemas. Con el tiempo, pueden hacerse cambios a los sistemas en respuesta a problemas identificados, aunque sin un documento de política global que guíe la evolución de un sistema. Ante tales situaciones, se requiere trabajar y documentar la política a partir de ejemplos, y confirmarla con los administradores de la compañía.

Lineamiento 2: Evite un solo punto de falla

En cualquier sistema crítico, una buena práctica de diseño es tratar de evitar un solo punto de falla. Esto significa que una sola falla en parte del sistema no debe dar por resultado una falla global del sistema. En términos de seguridad, significa que no debe apoyarse en un solo mecanismo para garantizar la seguridad; en vez de ello, se deben emplear muchas técnicas diferentes. En ocasiones esto se conoce como “defensa en profundidad”.

Por ejemplo, si se solicita una contraseña para autenticar a los usuarios de un sistema, también puede incluirse un mecanismo de autenticación pregunta/respuesta donde los usuarios deban registrar previamente preguntas y respuestas en el sistema. Después de la autenticación de contraseña, deben responder correctamente las preguntas antes de que

se les permita el acceso. Para proteger la integridad de los datos en un sistema, hay que mantener una bitácora ejecutable de los cambios realizados a los datos (véase el lineamiento 5). En caso de una falla, podrá reproducirse la bitácora para recrear el conjunto de datos. También es conveniente hacer una copia de todos los datos que se modifiquen antes de realizar algún cambio.

Lineamiento 3: Seguridad en caso de falla

Las fallas son inevitables en todos los sistemas y, en la misma forma en que los sistemas críticos de protección siempre deben ser a prueba de fallas, los sistemas críticos para la seguridad siempre deben ser "seguros en caso de falla". Cuando falle el sistema, no se deben usar procedimientos de estado de emergencia que sean menos seguros que el sistema en sí. La falla del sistema tampoco debe significar que un atacante acceda a datos a los que normalmente no tendría acceso.

Por ejemplo, en el sistema de información de pacientes se sugiere un requerimiento de que los datos de los pacientes deben descargarse a un sistema cliente al comienzo de una sesión clínica. Esto acelera el acceso y significa que el acceso es posible si el servidor no está disponible. Por lo general, el servidor borra estos datos al final de la sesión clínica. Sin embargo, si el servidor falla, entonces existe la posibilidad de que la información se conserve en el cliente. Un método de seguridad en caso de falla en dichas circunstancias consiste en encriptar todos los datos almacenados de los pacientes en el cliente. Esto significa que un usuario no autorizado no podrá leer los datos.

Lineamiento 4: Equilibrio entre seguridad y usabilidad

Las demandas de seguridad y usabilidad a menudo son contradictorias. Para hacer seguro un sistema, debe introducir comprobaciones de que los usuarios están autorizados para utilizar el sistema y que actúan en concordancia con políticas seguras. Inevitablemente, todo ello supone hacer peticiones a los usuarios: deben recordar nombres y contraseñas de acceso, usar solamente el sistema de ciertas computadoras, etcétera. Esto significa que los usuarios tardan más tiempo para iniciar el sistema y usarlo de manera efectiva. Conforme se agregan características de seguridad a un sistema, es inevitable que éste se vuelva menos usable. Se recomienda la lectura del libro de Cranor y Garfinkel (2005), que examina un amplio rango de conflictos en el área general de seguridad y usabilidad.

Se llega a un punto en que es contraproducente seguir agregando nuevas características de seguridad a costa de la usabilidad. Por ejemplo, si se requiere que los usuarios ingresen múltiples contraseñas o que cambien sus contraseñas a intervalos frecuentes por cadenas de caracteres imposibles de recordar, simplemente anotarán en algún lugar dichas contraseñas. Un atacante (en especial uno interno) podría encontrar las contraseñas que se anotaron y así conseguir acceso al sistema.

Lineamiento 5: Registre las acciones del usuario

Si es prácticamente posible hacerlo, debe mantener siempre una bitácora de las acciones del usuario. Esta bitácora, al menos, debe registrar quién hizo qué, los activos que utilizó, y la hora y fecha de acción. Como se explicó en el lineamiento 2, si mantiene esto como lista de comandos ejecutables, tendrá la opción de reproducir la bitácora para recuperarse de las fallas. Desde luego, también necesita herramientas que le permitan analizar la bitácora y detectar acciones potencialmente anómalas. Dichas herramientas pueden esca-

near la bitácora y descubrir las acciones anómalas, y además ayudar a detectar ataques y rastrear cómo el atacante consiguió acceso al sistema.

Además de ayudar a recuperarse de la falla, una bitácora de acciones del usuario es útil porque actúa como un disuasivo a los ataques internos. Si las personas saben que sus acciones se registran, entonces es menos probable que intenten actuar de formas no autorizadas. Esto es más efectivo para ataques casuales, como una enfermera que busca registros de pacientes, o para detectar ataques donde se robaron credenciales de usuario legítimas mediante ingeniería social. Desde luego, esto no es infalible, ya que los internos técnicamente habilidosos también pueden tener acceso a la bitácora y modificarla.

Lineamiento 6: Use redundancia y diversidad para reducir el riesgo

Redundancia significa conservar más de una versión del software o de los datos en un sistema. Diversidad, cuando se aplica al software, significa que las diferentes versiones no deben apoyarse en la misma plataforma o implementarse usando las mismas tecnologías. Por lo tanto, una vulnerabilidad de plataforma o tecnología no afectará a todas las versiones y, en consecuencia, no conducirá a una falla común. En el capítulo 13 se explicó cómo redundancia y diversidad son mecanismos fundamentales que se usan en la ingeniería de confiabilidad.

Ya se estudiaron ejemplos de redundancia: mantener información de pacientes tanto en el servidor como en el cliente, primero en el sistema de atención a la salud mental, y luego en el sistema distribuido de comercio de acciones que se muestra en la figura 14.5. En el sistema de registro de pacientes, podría usar varios sistemas operativos en el cliente y el servidor (por ejemplo, Linux en el servidor, Windows en el cliente). Esto garantiza que un ataque basado en una vulnerabilidad de sistema operativo no afectará tanto al servidor como al cliente. Desde luego, habrá que negociar tales beneficios contra el creciente costo administrativo de mantener diferentes sistemas operativos en una organización.

Lineamiento 7: Valide todas las entradas

Un ataque común a un sistema implica proporcionar al sistema entradas inesperadas que hacen que se comporte en forma no anticipada. Esto simplemente puede causar la caída de un sistema, lo que deriva en pérdida de servicio, o bien, las entradas podrían constituir un código malicioso que se ejecute en el sistema. Las vulnerabilidades de desbordamiento de buffer, que primero se demostraron en el gusano de Internet (Spafford, 1989) y que utilizan comúnmente los atacantes (Berghel, 2001), pueden activarse mediante cadenas largas de entrada. Otro ataque bastante común es el llamado “envenenamiento SQL”, en el que un usuario malicioso ingresa un fragmento SQL que interpreta un servidor.

Como se explicó en el capítulo 13, es posible evitar muchos de estos problemas al diseñar la validación de entrada en su sistema. En esencia, nunca debe aceptar cualquier entrada sin aplicar algunas verificaciones. Como parte de los requerimientos, tiene que definir las comprobaciones que se apliquen. Debe usar el conocimiento de la entrada para definir dichas comprobaciones. Por ejemplo, si se ingresa un sobrenombre, puede comprobar que no existen espacios embebidos y que el único signo de puntuación utilizado es el guión. También puede verificar el número de caracteres de entrada y rechazar las entradas muy largas. Por ejemplo, nadie tiene un nombre familiar con más de 40 caracteres ni direcciones con más de 100 caracteres de largo. Si usa menús para presentar entradas permitidas, evitará algunos de los problemas de validación de entrada.

Lineamiento 8: Compartimente sus activos

Compartimentar significa que no debe permitir el acceso a la información en un sistema con el criterio de todo o nada. En vez de ello, debe organizar en compartimentos la información en un sistema. Los usuarios sólo deben tener acceso a la información que necesitan, y no a toda la información en un sistema. Esto significa que es posible que los efectos de un ataque se contengan. Tal vez se pierda o se dañe alguna información, pero es improbable que resulte afectada toda la información en el sistema.

Por ejemplo, en el sistema de información de pacientes, debe diseñar el sistema de manera que, en cualquier clínica, el personal hospitalario tenga normalmente sólo acceso a los registros de los pacientes que tienen una cita en esa clínica. Por lo general, no deben tener acceso a todos los registros de pacientes en el sistema. Esto no sólo limita la pérdida potencial de ataques internos, sino también significa que, si un intruso roba sus credenciales, la cantidad de daño que pueda causar será limitada.

Por otro lado, tal vez también deba tener mecanismos en el sistema para garantizar el acceso inesperado, por ejemplo, de un paciente seriamente enfermo que requiere tratamiento urgente sin tener cita. En tales circunstancias, puede usar algún mecanismo seguro alternativo para superar la compartimentación en el sistema. En esas situaciones, donde la seguridad se relaja para mantener la disponibilidad del sistema, es esencial contar con un mecanismo de acceso para registrar el uso del sistema. Siendo así, se puede comprobar las bitácoras con la finalidad de rastrear cualquier uso no autorizado.

Lineamiento 9: Diseñe para implementar

Muchos problemas de seguridad surgen porque el sistema no está configurado correctamente al implementarse en su entorno operacional. Por lo tanto, siempre debe diseñar su sistema de forma que se incluyan instalaciones para simplificar la implementación en el entorno del cliente, y comprobar errores y omisiones potenciales de configuración en el sistema implementado. Éste es un tema importante, que se analiza a detalle en la sección 14.2.3.

Lineamiento 10: Diseñe para facilitar la recuperabilidad

Sin importar cuánto esfuerzo se use en mantener la seguridad de los sistemas, siempre debe diseñar su sistema con la idea de que podría ocurrir una falla de seguridad. De ahí que deba pensar en cómo recuperarse de posibles fallas y restaurar el sistema a un estado operativo seguro. Por ejemplo, podría incluir un sistema de autenticación de respaldo en caso de que se comprometa su autenticación de contraseña.

Suponga que una persona no autorizada, externa a la clínica, consigue acceso al sistema de registros de los pacientes, y usted no sabe cómo obtuvo una combinación de nombre y contraseña válida. Necesita reiniciar el sistema de autenticación y no sólo cambiar las credenciales usadas por el intruso. Esto es esencial, porque el intruso también puede conseguir acceso a otras contraseñas de usuario. Por lo tanto, es preciso asegurarse de que todos los usuarios autorizados cambien sus contraseñas. También debe cerciorarse de que la persona sin autorización no tiene acceso al mecanismo de cambio de contraseñas.

En consecuencia, debe diseñar su sistema para negar el acceso a todos (hasta que se cambien sus contraseñas y se autentique a los usuarios reales para cambiar la contraseña), y suponer que sus contraseñas seleccionadas pueden no ser seguras. Una forma de hacer esto es mediante un mecanismo de pregunta/respuesta, donde los usuarios deben responder preguntas para las cuales tienen respuestas previamente registradas. Esto sólo

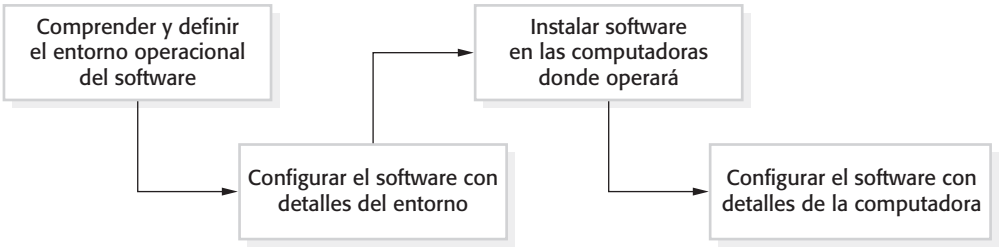

Figura 14.7 Implementación del software

se solicita al cambiar las contraseñas, lo que permite la recuperación de un ataque con relativamente poca perturbación del usuario.

14.2.3 Diseño para implementar

La implementación de un sistema implica configurar el software para funcionar en un entorno operacional, instalar el sistema en las computadoras en ese entorno y, posteriormente, configurar el sistema instalado para dichas computadoras (figura 14.7). La configuración puede ser un proceso simple que implique establecer algunos parámetros internos en el software para reflejar las preferencias del usuario. Sin embargo, en ocasiones, la configuración es compleja y requiere de la definición específica de los modelos y las reglas empresariales que afectan la ejecución del software.

Con frecuencia, en esta etapa del proceso se introducen accidentalmente vulnerabilidades en el software. Por ejemplo, durante la instalación, el software tiene a menudo que configurarse con una lista de usuarios permitidos. Cuando esta lista se entrega, consta simplemente de un acceso de administrador genérico como "admin" y una contraseña por defecto, como "password". Lo anterior facilita que un administrador configure el sistema. La primera acción debe ser introducir un nuevo nombre de acceso y contraseña, y borrar el nombre de acceso genérico. No obstante, es fácil olvidar hacer esto. Un atacante que conozca el acceso por defecto podrá obtener acceso privilegiado al sistema.

Configuración e implementación se ven a menudo como temas de administración del sistema y, por consiguiente, se consideran fuera del ámbito de los procesos de ingeniería de software. Desde luego, la buena práctica administrativa ayuda a evitar muchos problemas de seguridad que surgen de errores de configuración e implementación. Sin embargo, los diseñadores de software tienen la responsabilidad de "diseñar para la implementación". Siempre se debe brindar soporte interno para la implementación, lo cual reducirá la probabilidad de que los administradores del sistema (o usuarios) cometan errores al configurar el software.

Se recomiendan cuatro formas de incorporar soporte de implementación en un sistema:

1. *Incluir soporte para ver y analizar las configuraciones* Siempre es conveniente adicionar instalaciones en un sistema que permitan a los administradores o a los usuarios examinar la configuración actual del sistema. De manera sorprendente, esta facilidad está ausente en la mayoría de los sistemas de software, y los usuarios se frustran ante las dificultades de encontrar parámetros de configuración. Por ejemplo, en la versión del procesador de texto que se usó para escribir este capítulo, es imposible ver o imprimir en una sola pantalla los parámetros de todas las preferencias del sistema. No obstante, si un administrador es capaz de obtener una imagen completa

de una configuración, es más probable que especifique los errores y las omisiones. Lo ideal es que una pantalla de configuración también pueda destacar los aspectos de la configuración que sean potencialmente inseguros, por ejemplo, si no se estableció una contraseña.

2. *Minimizar los privilegios por defecto* El software debe diseñarse de forma que la configuración por defecto de un sistema ofrezca mínimos privilegios esenciales. De esta forma, el daño que cualquier atacante pudiera hacer quedará limitado. Por ejemplo, la autenticación por defecto del administrador del sistema sólo debe permitir el acceso a un programa que permita a un administrador establecer nuevas credenciales. Debe prohibir el acceso a alguna otra instalación del sistema. Una vez establecidas las nuevas credenciales, deben borrarse automáticamente por defecto el nombre y la contraseña.
3. *Localizar parámetros de configuración* Cuando se diseñe soporte para la configuración del sistema, habrá que asegurarse de que todo en una configuración que afecte la misma parte de un sistema se configure en el mismo lugar. Nuevamente con el ejemplo del procesador de texto, en la versión que se utilizó, pudo configurarse alguna información de seguridad, como una contraseña para controlar el acceso al documento, en el menú Preferencias/Seguridad. Otra información se configura en el menú Herramientas/Proteger documento. Si no se localiza la información de configuración, es fácil olvidar configurarla o, en algunos casos, ni siquiera se estará al tanto de que en el sistema se incluyen algunas instalaciones de seguridad.
4. *Proporcionar formas sencillas de corregir vulnerabilidades de seguridad* Hay que incluir mecanismos directos para actualizar el sistema y reparar las vulnerabilidades de seguridad que se hayan descubierto. Éstos podrían incluir comprobación automática para actualizaciones de seguridad, o la descarga de dichas actualizaciones tan pronto como estén disponibles. Es importante que los usuarios no puedan pasar por alto dichos mecanismos pues, inevitablemente, considerarán otro trabajo como más importante. Existen muchos ejemplos registrados de problemas de seguridad mayores que surgen (por ejemplo, falla completa de una red hospitalaria) porque los usuarios no actualizaron su software cuando se les pidió que lo hicieran.

14.3 Supervivencia del sistema

Hasta el momento, se estudió la ingeniería de seguridad desde la perspectiva de una aplicación que está bajo desarrollo. El proveedor y el desarrollador del sistema tienen control sobre todos los aspectos del sistema que pudieran atacarse. En realidad, como se sugiere en la figura 14.1, los modernos sistemas distribuidos se apoyan necesariamente en una infraestructura que incluye sistemas comerciales y componentes reutilizables desarrollados por diferentes organizaciones. La seguridad de dichos sistemas no depende sólo de las decisiones de diseño locales, sino también de la seguridad de aplicaciones externas, servicios Web y la infraestructura de red.

Esto significa que, sin importar cuánta atención se ponga a la seguridad, no es posible garantizar que un sistema podrá resistir ataques externos. En consecuencia, para sistemas complejos en red, debe suponerse que es posible la penetración y que la integridad del sistema no está garantizada. Por lo tanto, se debe pensar en cómo hacer que el sistema sea resistente, de manera que sobreviva para entregar servicios esenciales a los usuarios.

La supervivencia o resiliencia (Westmark, 2004) es una propiedad emergente de un sistema como totalidad, y no una propiedad de componentes individuales, que en sí mismos podrían no ser supervivientes. La supervivencia de un sistema refleja su capacidad para continuar la entrega de servicios empresariales esenciales o críticos para la misión, y para legitimar a los usuarios mientras está bajo ataque o después de que se dañó parte del sistema. El daño podría ser causado por un ataque o una falla del sistema.

El trabajo en la supervivencia del sistema apunta al hecho de que las vidas económica y social dependen de una infraestructura crítica controlada por computadoras. Ésta incluye la infraestructura para entregar servicios públicos (electricidad, agua, gas, etcétera) y la infraestructura para entregar y gestionar información (teléfonos, Internet, servicio postal, etcétera). Sin embargo, la supervivencia no es simplemente un asunto crítico de infraestructura . Cualquier organización que se apoye en sistemas de cómputo críticos en red debe interesarse por la forma como resultaría afectada si sus sistemas no sobreviven a un ataque malicioso o a una falla catastrófica del sistema. Por lo tanto, para sistemas empresariales críticos, el análisis y el diseño de supervivencia deben ser parte del proceso de ingeniería de seguridad.

Conservar la disponibilidad de los servicios críticos es la esencia de la supervivencia. Esto significa que usted debe conocer:

- los servicios del sistema más críticos para una empresa;
- la calidad mínima del servicio a preservar;
- cómo pueden comprometerse dichos servicios;
- cómo pueden protegerse tales servicios;
- cómo recuperarse rápidamente si los servicios dejan de estar disponibles.

Por ejemplo, en un sistema que administra la salida de ambulancias en respuesta a llamadas de emergencia, los servicios críticos son aquellos relacionados con la recepción de llamadas y el envío de ambulancias a la emergencia médica. Otros servicios, como el registro de las llamadas y la administración de la ubicación de las ambulancias, son menos críticos, ya sea porque no requieren procesamiento en tiempo real o porque existen mecanismos alternativos. Por ejemplo, para encontrar la ubicación de una ambulancia es posible llamar al personal de la unidad y preguntarle la ubicación.

Ellison y sus colaboradores (1999a; 199b; 2002) diseñaron un método de análisis llamado Survivable Systems Analysis (análisis de sistemas supervivientes). Se usa para valorar las vulnerabilidades en los sistemas y apoyar el diseño de arquitecturas y características de los sistemas que promueven la supervivencia de éstos. Estos investigadores argumentan que el logro de la supervivencia depende de tres estrategias complementarias:

1. *Resistencia* Evitar los problemas mediante la construcción de capacidades en el sistema para repeler ataques. Por ejemplo, un sistema puede usar certificados digitales para autenticar a los usuarios, lo que dificulta el hecho de que usuarios no autorizados tengan acceso.
2. *Reconocimiento* Detectar los problemas mediante la construcción de capacidades en el sistema para descubrir ataques y fallas, además de valorar el daño resultante. Por ejemplo, podrían asociarse sumas de verificación con datos críticos, de manera que pueda detectarse la corrupción de los datos.
3. *Recuperación* Tolerar los problemas por medio de la construcción de capacidades en el sistema para entregar servicios esenciales mientras está bajo ataque, y recuperar la

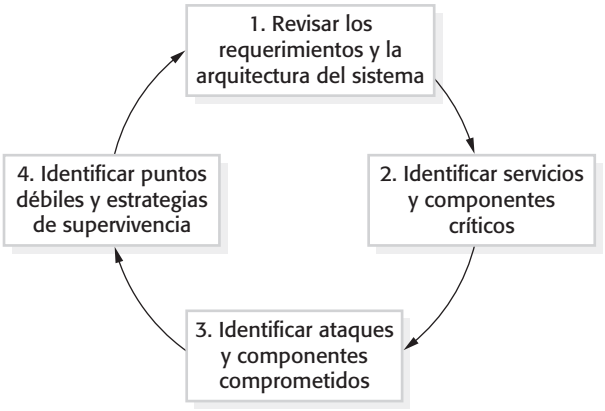

Figura 14.8 Etapas en el análisis de supervivencia

funcionalidad plena después de un ataque. Por ejemplo, los mecanismos de tolerancia a fallas en el desarrollo que usan diversas implementaciones de la misma funcionalidad pueden incluirse para lidiar con una pérdida de servicio de una parte del sistema.

El análisis de sistemas supervivientes es un proceso de cuatro etapas (figura 14.8), que analiza los requerimientos y la arquitectura actual o las propuestas del sistema; identifica servicios críticos, escenarios de ataque y "puntos débiles" del sistema, y propone cambios para mejorar la supervivencia de un sistema. Las actividades clave en cada una de dichas etapas son las siguientes:

1. *Comprensión del sistema* Para un sistema existente o propuesto, se revisan las metas del sistema (llamadas en ocasiones "objetivos de la misión"), los requerimientos y la arquitectura del sistema.
2. *Identificación de servicios críticos* Se identifican los servicios que siempre deben mantenerse y los componentes que se requieren para conservar dichos servicios.
3. *Simulación de ataque* Se identifican los escenarios o casos de uso para posibles ataques, junto con los componentes del sistema que resultarían afectados por dichos ataques.
4. *Análisis de supervivencia* Se identifican los componentes que son esenciales y que podrían resultar comprometidos por un ataque, así como las estrategias de supervivencia basadas en resistencia, reconocimiento y recuperación.

Ellison y sus colaboradores presentan un excelente estudio de caso del método basado en un sistema para apoyar el tratamiento de salud mental (1999b). Este sistema es similar al MHC-PMS que se usó como ejemplo en este libro. En vez de repetir su análisis, se utiliza el sistema de comercio de acciones, que se muestra en la figura 14.5, para ilustrar algunas de las características del análisis de supervivencia.

Como se observa en la figura 14.5, este sistema ya hizo alguna provisión de supervivencia. Las cuentas de usuarios y los precios de las acciones se duplican a través de los servidores, de manera que pueden realizarse pedidos incluso si el servidor local no está disponible. Suponga que la capacidad de que usuarios autorizados realicen pedidos de acciones es el servicio clave que debe mantenerse. Para garantizar que los usuarios confíen en el sistema, es esencial mantener la integridad. Los pedidos deben ser precisos y reflejar las ventas o compras reales hechas por un usuario del sistema.

Ataque	Resistencia	Reconocimiento	Recuperación
Usuario no autorizado realiza pedidos maliciosos	Para realizar pedidos, se solicita una contraseña de negociación diferente de la de acceso.	Enviar por correo electrónico copia del pedido al usuario autorizado, con número telefónico de contacto (de forma que puedan detectarse pedidos maliciosos). Mantener un historial de pedidos de usuario y verificar patrones de negociación inusuales.	Proporcionar mecanismos para "deshacer" automáticamente las negociaciones y restaurar las cuentas de usuario. Reembolsar a los usuarios las pérdidas que se deben a negociación maliciosa. Adquirir seguros contra pérdidas derivadas de ello.
Corrupción de base de datos de transacciones	Requerir la autorización de usuarios privilegiados mediante un mecanismo de autenticación más confiable, como certificados digitales.	Mantener copias de sólo lectura de las transacciones para una oficina en un servidor internacional. Comparar periódicamente las transacciones para detectar casos de corrupción. Mantener sumas de verificación criptográficas con todos los registros de transacción para detectar la corrupción.	Recuperar la base de datos a partir de copias de respaldo. Ofrecer un mecanismo para reproducir las negociaciones desde una hora especificada para recrear la base de datos de transacciones.

Figura 14.9 Análisis de supervivencia en un sistema de comercio de acciones

Para mantener este servicio de pedidos, existen tres componentes del sistema que se utilizan:

1. *Autenticación de usuario* Esto permite que usuarios autorizados ingresen al sistema.
2. *Cotización de precios* Esto permite la cotización del precio de compra y venta de una acción.
3. *Colocación de pedidos* Esto permite la realización de pedidos de compra y venta a un precio dado.

Obviamente, dichos componentes usan los activos de datos esenciales, como base de datos de cuentas de usuarios, una base de datos de precios y una base de datos de transacción de pedidos. Éstos deben sobrevivir a los ataques si se desea mantener el servicio.

Existen muchos tipos diferentes de ataques sobre este sistema que podrían cometerse. Considere aquí dos posibilidades:

1. Un usuario malicioso siente rencor contra un usuario acreditado del sistema. Consigue acceso al sistema usando sus credenciales. Coloca pedidos maliciosos y compra y vende acciones, con la intención de causar problemas al usuario autorizado.
2. Un usuario no autorizado corrompe la base de datos de transacciones al conseguir permiso para emitir directamente comandos SQL. Por lo tanto, es imposible la reconciliación de ventas y compras.

La figura 14.9 muestra ejemplos de estrategias de resistencia, reconocimiento y recuperación que ayudan a contrarrestar estos ataques.

Desde luego, aumentar la supervivencia o la resiliencia de un sistema cuesta dinero. Las compañías tal vez se muestren renuentes a invertir en supervivencia si nunca han sufrido un ataque serio o pérdidas asociadas. Sin embargo, así como es mejor comprar buenas cerraduras y una alarma para evitar que se introduzcan ladrones a su casa, también es mejor invertir en supervivencia antes de sufrir un ataque. El análisis de supervivencia todavía no es parte de la mayoría de los procesos de ingeniería de software pero, conforme más sistemas se convierten en críticos para la empresa, es probable que tales análisis se usen más ampliamente.

PUNTOS CLAVE

- La ingeniería de seguridad se enfoca en cómo desarrollar y mantener sistemas de software capaces de resistir ataques maliciosos, los cuales tienen la intención de dañar un sistema basado en computadoras o sus datos.
- Las amenazas a la seguridad pueden ser amenazas a la confidencialidad, integridad o disponibilidad de un sistema o sus datos.
- La gestión del riesgo de seguridad implica diseñar una arquitectura segura de sistema, seguir buenas prácticas para diseñar sistemas seguros e incluir funcionalidad para minimizar la posibilidad de introducir vulnerabilidades al implementar el sistema.
- El diseño de seguridad implica diseñar una arquitectura de sistema que conserve la seguridad, seguir buenas prácticas para el diseño de sistemas seguros e incluir funcionalidad para minimizar la posibilidad de introducir vulnerabilidades cuando se despliega el sistema.
- Los temas clave cuando se diseña una arquitectura segura de sistemas incluyen organizar la estructura del sistema para proteger los activos clave y distribuir los activos del sistema para minimizar las pérdidas que podría ocasionar un ataque.
- Los lineamientos de diseño de seguridad sensibilizan a los diseñadores del sistema ante los asuntos de seguridad que quizá pasaron por alto. Ofrecen una base para elaborar listas de verificación que permitan revisar la seguridad.
- Para apoyar la implementación segura debe existir una forma de implementar y analizar las configuraciones del sistema, localizar parámetros de configuración de forma que no se olviden configuraciones importantes, minimizar los privilegios por defecto asignados a los usuarios del sistema, y ofrecer formas para reparar vulnerabilidades de seguridad.
- La supervivencia del sistema refleja su capacidad de continuar la entrega de servicios esenciales para la empresa o críticos para la misión, con la finalidad de legitimar a los usuarios mientras el sistema se encuentra bajo ataque, o después de que parte del sistema se ha dañado.

LECTURAS SUGERIDAS

"Survivable Network System Analysis: A Case Study." Un excelente ensayo que introduce la noción de supervivencia del sistema; refiere un estudio de caso de un sistema de registros de tratamiento

de salud mental para ilustrar la aplicación de un método de supervivencia. (R. J. Ellison, R. C. Linger, T. Longstaff y N. R. Mead, *IEEE Software*, **16** (4), julio/agosto de 1999.)

Building Secure Software: How to Avoid Security Problems the Right Way. Un buen libro práctico que analiza la seguridad desde una perspectiva de programación. (J. Viega y G. McGraw, Addison-Wesley, 2002.)

Security Engineering: A Guide to Building Dependable Distributed Systems, 2nd edition. Se trata de una discusión amplia y profunda de los problemas para construir sistemas seguros. Se enfoca en los sistemas y no en la ingeniería del software; ofrece una amplia cobertura de hardware y redes, mediante excelentes ejemplos extraídos de fallas de sistemas reales. (R. Anderson, John Wiley & Sons, 2008.)

EJERCICIOS

14.1. Explique las importantes diferencias entre ingeniería de seguridad de aplicación e ingeniería de seguridad de infraestructura.

14.2. Para el MHC-PMS, sugiera un ejemplo de activo, exposición, vulnerabilidad, ataque, amenaza y control.

14.3. Explique por qué hay necesidad de que la valoración del riesgo sea un proceso continuo, desde las primeras etapas de la ingeniería de requerimientos hasta el uso operativo de un sistema.

14.4. Con base en sus respuestas a la pregunta 2 acerca del MHC-PMS, valore los riesgos asociados con ese sistema y proponga dos requerimientos de sistema que permitan reducir tales riesgos.

14.5. Explique, con una analogía extraída de un contexto de ingeniería, no de software, por qué debe usarse un enfoque en capas para la protección de activos.

14.6. Explique por qué es importante usar tecnologías diversas para dar soporte a sistemas distribuidos en situaciones en que la disponibilidad del sistema es un asunto crítico.

14.7. ¿Qué es ingeniería social? ¿Por qué es difícil protegerse contra ella en las organizaciones grandes?

14.8. Para cualquier sistema de software comercial que use (por ejemplo, Microsoft Word), analice las instalaciones de seguridad incluidas y examine cualquier problema que encuentre.

14.9. Explique cómo pueden usarse las estrategias complementarias de resistencia, reconocimiento y recuperación para mejorar la supervivencia de un sistema.

14.10. Para el sistema de comercio de acciones que se describe en la sección 14.2.1, cuya arquitectura se ilustra en la figura 14.5, sugiera dos posibles ataques ulteriores sobre el sistema y proponga algunas estrategias que pudieran contrarrestar dichos ataques.

REFERENCIAS

Alberts, C. y Dorofee, A. (2002). *Managing Information Security Risks: The OCTAVE Approach*. Boston: Addison-Wesley.

Alexander, I. (2003). "Misuse Cases: Use Cases with Hostile Intent". *IEEE Software*, **20** (1), 58–66.

Anderson, R. (2008). *Security Engineering, 2nd edition*. Chichester: John Wiley & Sons.

Berghel, H. (2001). "The Code Red Worm". *Comm. ACM*, **44** (12), 15–19.

Bishop, M. (2005). *Introduction to Computer Security*. Boston: Addison-Wesley.

Cranor, L. y Garfinkel, S. (2005). *Security and Usability: Designing secure systems that people can use*. Sebastopol, Calif.: O'Reilly Media Inc.

Ellison, R., Linger, R., Lipson, H., Mead, N. y Moore, A. (2002). "Foundations of Survivable Systems Engineering". *Crosstalk: The Journal of Defense Software Engineering*, **12**, 10–15.

Ellison, R. J., Fisher, D. A., Linger, R. C., Lipson, H. F., Longstaff, T. A. y Mead, N. R. (1999a). "Survivability: Protecting Your Critical Systems". *IEEE Internet Computing*, **3** (6), 55–63.

Ellison, R. J., Linger, R. C., Longstaff, T. y Mead, N. R. (1999b). "Survivable Network System Analysis: A Case Study". *IEEE Software*, **16** (4), 70–7.

Pfleeger, C. P. y Pfleeger, S. L. (2007). *Security in Computing, 4th edition*. Boston: Addison-Wesley.

Schneier, B. (2000). Secrets and Lies: *Digital Security in a Networked World*. Nueva York: John Wiley & Sons.

Sindre, G. y Opdahl, A. L. (2005). "Eliciting Security Requirements through Misuse Cases". *Requirements Engineering*, **10** (1), 34–44.

Spafford, E. (1989). "The Internet Worm: Crisis and Aftermath". *Comm ACM*, **32** (6), 678–87.

Viega, J. y McGraw, G. (2002). *Building Secure Software*. Boston: Addison-Wesley.

Westmark, V. R. (2004). "A Definition for Information System Survivability". 37th Hawaii Int. Conf. on System Sciences, Hawaii: 903–1003.

Wheeler, D. A. (2003). *Secure Programming for Linux and UNix HOWTO*. Web published: http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/index.html.