53

Introducción a la Seguridad del SO

Capítulo por **Peter Reiher (UCLA)**

53.1 Introducción

La seguridad de los sistemas informáticos es un tema vital cuya importancia no hace más que aumentar. Se ha perdido mucho dinero y la vida de muchas personas se han visto perjudicadas cuando ha fallado la seguridad informática. Los ataques a los sistemas informáticos son tan comunes que son inevitables en casi cualquier escenario en el que se realiza informática. En general, todos los elementos de un sistema informático pueden ser objeto de ataque, y los fallos en cualquiera de ellos pueden dar a un atacante una oportunidad para hacer algo que se quiere evitar. Pero los sistemas operativos son especialmente importantes desde el punto de vista de la seguridad. ¿Por qué? Para empezar, casi todo se ejecuta sobre un sistema operativo, por regla general, si un software se ejecuta bajo un sistema operativo, una pieza de middleware, o cualquier otra cosa que sea insegura, luego ese software también será inseguro. Es como construir una casa sobre arena. Puedes construir una bonita y sólida estructura, pero una inundación puede arrastrar la base de tu casa, destruyéndola totalmente el cuidado que hayas tenido en su construcción. Del mismo modo, una aplicación puede que quizás no tenga ningún fallo de seguridad propio, pero si el atacante puede utilizar indebidamente el software que está debajo de para robar preciada información, colapsar el programa, o causar algún daño, dichos esfuerzos para asegurar el código puede ser en vano.

Este punto es especialmente importante para los sistemas operativos. Es posible que a un usuario no le importe la seguridad de un servidor web o un sistema de base de datos en particular si no ejecuta ese software, y puede que no le importe la seguridad de alguna plataforma de middleware que no utilice, pero todo el mundo ejecuta un sistema operativo, y hay relativamente pocas opciones entre las que elegir. Por lo tanto, las fallas de seguridad en un sistema operativo, especialmente uno ampliamente utilizado, tienen un inmenso impacto en muchos usuarios y muchas piezas de software.

Otra razón por la que la seguridad del sistema operativo es tan importante es que en última instancia, todo nuestro software depende del comportamiento adecuado del hardware subyacente: el procesador, la memoria y los dispositivos periféricos. ¿Qué tiene el control final de esos recursos de hardware? El sistema operativo.

Pensando en lo ya estudiado sobre la gestión de la memoria, la programación, los sistemas de archivos, la sincronización, etc., ¿qué pasaría con cada uno de estos componentes de tu sistema operativo si un adversario pudiera forzarlo a comportarse de alguna manera arbitraria? Si entiendes lo aprendido hasta ahora, deberías encontrar esta perspectiva profundamente preocupante[1]. Nuestras vidas informáticas dependen de que nuestros sistemas operativos se comporten como han sido definidos, y especialmente de que no se comporten de forma que beneficien a nuestros adversarios, en lugar de nosotros.

La tarea de asegurar un sistema operativo no es fácil, ya que los sistemas operativos modernos son grandes y complejos. No es muy complicado entender que cuanto mas grande y complejo sea el código, mas propenso es a tener fallos y vulnerabilidades. Los fallos en la seguridad del software generalmente surgen de este tipo de fallos. Los programas grandes y complejos son probablemente más difíciles de asegurar que los programas pequeños y simples. No hay muchos otros programas que sean tan grandes y complejos como un sistema operativo moderno.

Otro reto en la seguridad de los sistemas operativos es que, en su mayoría, están pensados para soportar múltiples procesos simultáneamente. Como se ha aprendido, hay muchos mecanismos en un sistema operativo destinados a separar los procesos entre sí, y para proteger las piezas compartidas de hardware para que no se utilicen de forma que interfieran con otros procesos. Si cada proceso pudiera hacer lo que quisiera con cualquier recurso del hardware y cualquier pieza de datos en la máquina sin dañar cualquier otro proceso, la seguridad del sistema sería mucho más fácil. Sin embargo, normalmente no confiamos en todo por igual. Al descargar y ejecutar un script de un sitio web no visitado antes, ¿realmente se quiere dar la capacidad de borrar todos los archivos del disco, matar todos los otros procesos, y empezar a usar tu interfaz de red para enviar correo electrónico basura a otras máquinas? Probablemente no, pero si usted es el propietario de su computadora, tiene todo el derecho de hacer esas cosas, si eso es lo que quiere hacer. Y a menos que el sistema operativo tenga cuidado, cualquier proceso que ejecute, incluyendo el el que ejecuta ese script descargado, puede hacer todo lo que tú puedes hacer.

Considera la cuestión de la seguridad del sistema operativo desde una perspectiva diferente. Un papel de un sistema operativo es pro-

[1] Si no lo entiendes, tendras que releer mucho. Mucho

porcionar abstracciones útiles para que los programas de aplicación se basen en ellas. Estas aplicaciones deben confiar en las implementaciones del sistema operativo de las abstracciones para trabajar como están definidas. Brevemente, una parte de la definición de tales abstracciones es su comportamiento de seguridad. Por ejemplo, esperamos que el sistema de archivos del sistema operativo aplique las restricciones de acceso que se supone que debe aplicar. Las aplicaciones pueden basarse en esta expectativa para lograr los objetivos de seguridad que requieren, como contar con las garantías de acceso del sistema de archivos para asegurar que un archivo que han especificado como no escribible no sea alterado. Si las aplicaciones no pueden confiar en una implementación adecuada de las garantías de seguridad para las abstracciones del sistema operativo, entonces no pueden utilizar las garantías de acceso al sistema de archivos del sistema operativo, entonces no pueden usar estas abstracciones para lograr sus propios objetivos de seguridad. Como mínimo, eso implica mucho más trabajo en el desarrollo de aplicaciones, ya que se tendrán que tomar medidas adicionales para alcanzar los objetivos de seguridad deseados. Teniendo en cuenta nuestra discusión anterior, a menudo serán incapaces de lograr estos objetivos si las abstracciones en las que deben confiar (como la memoria virtual o una política de política de programación) no son de confianza.

Obviamente, la seguridad de los sistemas operativos es vital, pero difícil de conseguir. Así que ¿qué hacemos para asegurar nuestro sistema operativo? Abordar esta cuestión ha sido un reto para generaciones de informáticos, y todavía no hay una respuesta completa. Pero hay algunos principios importantes y herramientas que podemos utilizar para asegurar los sistemas operativos, que generalmente están incorporados en cualquiera de propósito general con el que se pueda trabajar, y alteran lo que se puede hacer con ese sistema y cómo se hace hacerlo. Así que puede que pienses que no estás interesado en la seguridad, pero hay que entender lo que el sistema operativo hace para asegurarse a sí mismo para también entender cómo conseguir que el sistema haga lo que se quiera.

EL PROBLEMA EN CUESTIÓN:
CÓMO ASEGURAR LOS RECURSOS DEL SO

Ante la existencia de múltiples procesos posiblemente concurrentes e interactivos ejecutan en la misma máquina, ¿cómo podemos garantizar que los recursos a los que cada proceso pueda acceder son exactamente aquellos a los que debería acceder, en exactamente de la forma que deseamos? ¿Qué primitivas necesita el sistema operativo? ¿Qué mecanismos de mecanismos debe proporcionar el hardware? ¿Cómo podemos utilizarlos para resolver los problemas de seguridad?

53.2 ¿Qué estamos protegiendo?

Es poco probable que logremos una buena protección a menos que tengamos una visión bastante completa de lo que estamos tratando de proteger cuando decimos que nuestro sistema operativo debe ser seguro. Afortunadamente, esa pregunta es fácil de responder para un sistema operativo al menos a alto nivel: todo. Esa respuesta no es muy reconfortante, pero es mejor tener un conocimiento realista de las amplias implicaciones de la seguridad del sistema operativo.

Un sistema operativo básico típico tiene un control completo de todo (o casi todo) el hardware de la máquina y es capaz de hacer literalmente cualquier cosa que el hardware permita. Esto significa que puede controlar el procesador, leer y escribir todos los registros, examinar cualquier ubicación de la memoria principal y realizar cualquier operación que soporte uno de sus periféricos. Como resultado, entre las cosas que que el SO puede hacer son:

- examinar o alterar la memoria de cualquier proceso
- leer, escribir, borrar o corromper cualquier archivo en cualquier medio de almacenamiento persistente, incluidos los discos duros y las unidades flash
- cambiar la programación o incluso detener la ejecución de cualquier proceso
- enviar cualquier mensaje a cualquier lugar, incluyendo versiones alteradas de aquellos que un proceso desee enviar
- activar o desactivar cualquier dispositivo periférico
- dar a cualquier proceso acceso a los recursos de cualquier otro proceso
- quitar arbitrariamente cualquier recurso que controle un proceso
- responder a cualquier llamada del sistema con una mentira lo más perjudicial posible

En esencia, los procesos están a merced del sistema operativo. Es casi imposible imposible para un proceso "proteger" cualquier parte de sí mismo de un sistema operativo malicioso. sistema operativo malicioso. Normalmente asumimos que nuestro sistema operativo no es malicioso [2], pero un fallo que permita a un proceso malicioso hacer que el sistema operativo se comporte mal es casi igual de malo,

[2] Si sospecha que su sistema operativo es malicioso, es hora de conseguir uno nuevo.

APARTE: ENCLAVES DE SEGURIDAD

Un poco más atrás, dijimos que el sistema operativo controla "casi todo" el hardware de la máquina. Ese tipo de advertencia debería haberte hecho preguntar preguntando, "bueno, ¿qué partes del hardware no controla?" Originalmente, realmente era todo el hardware. Pero a partir de los años 90, los desarrolladores de hardware empezaron a ver la necesidad de mantener parte del hardware aislado, hasta cierto punto, del sistema operativo. El primer hardware de este tipo estaba destinado principalmente a proteger el proceso de arranque del sistema operativo. EL **TPM** o **Trusted Platform Module**, proporcionaba la seguridad de que se estaba arrancando la versión del sistema operativo que pretendía, protegiendo de ataques que intentaban arrancar versiones comprometidas del sistema. Más recientemente, elementos de hardware más generales han tratado de controlar lo que puede hacer en la máquina, típicamente con algunos datos particularmente importantes, a menudo datos relacionados con la criptografía. Estos elementos de hardware se denominan enclaves de seguridad, ya que están pensados para permitir sólo un uso seguro de estos datos, incluso por el código más poderoso y confiable del sistema, el propio sistema operativo. A menudo se utilizan para apoyar las operaciones en un entorno de computación en la nube, donde varios sistemas operativos pueden ejecutarse en máquinas virtuales que comparten el mismo hardware físico.

Esto resulta ser un truco más difícil de lo que se esperaba, los trucos de seguridad suelen serlo. Los enclaves de seguridad a menudo resultan no proporcionar tanto aislamiento como sus diseñadores esperaban, pero los ataques contra ellos tienden a ser tan sofisticados y difíciles, y normalmente requieren la capacidad de ejecutar código privilegiado en el sistema. Así que incluso si no logran sus objetivos completos, ponen una barrera de protección extra contra el código del sistema operativo comprometido.

ya que podría permitir que ese proceso obtenga cualquiera de los poderes del propio sistema operativo. Este punto debería hacerte pensar muy seriamente en la importancia de diseñar sistemas operativos seguros y más comúnmente, aplicar parches de seguridad a cualquiera. Los fallos de seguridad en tu sistema operativo pueden comprometer completamente de la máquina en la que se ejecuta el sistema, por lo que prevenirlos y arreglar los que se encuentren es de vital importancia.

53.3 Objetivos y Políticas de Seguridad

¿A qué nos referimos cuando decimos que queremos que un sis-

tema operativo, o cualquier sistema, sea seguro? Es una afirmación bastante vaga, lo que realmente queremos decir es que hay cosas que nos gustaría que sucedieran en el sistema y cosas que no queremos que ocurran, y nos gustaría tener un alto grado de seguridad de que conseguimos lo que queremos. Como en la mayoría de los aspectos de la vida, normalmente acabamos acabamos pagando por lo que obtenemos, así que merece la pena pensar en qué propiedades y efectos de seguridad que realmente necesitamos y luego pagar sólo por y pagar sólo por ellos, y no por otras cosas que no necesitemos. Esto se reduce a que queremos especificar los objetivos que tenemos para el comportamiento relevante para la seguridad de nuestro sistema y elegir los enfoques de defensa que probablemente logren esos objetivos a un coste razonable.

Los investigadores en seguridad han pensado en esta cuestión en términos generales durante mucho tiempo. A un alto nivel conceptual, han definido tres grandes objetivos relacionados con la seguridad que son comunes a muchos sistemas, incluidos los sistemas operativos. Son los siguientes

- **Confidencialidad** - Si alguna pieza de información se supone que está oculta a los demás, no permitas que la descubran. Por ejemplo, no quieres que alguien se entere de cuál es tu número de tarjeta de crédito: quieres que ese número se mantenga confidencial.
- **Integridad** - Si alguna pieza de información o componente de un sistema se supone que está en un estado particular, no permitas que un adversario lo cambie. Por ejemplo, se ha hecho un pedido en línea para la entrega de una pizza de pepperoni, no se querrá que un bromista malintencionado cambie tu pedido a 1000 pizzas de anchoas. Un aspecto importante de la integridad es la autenticidad. A menudo es importante estar seguro no sólo de que la información no ha cambiado, sino de que ha sido creada por una parte determinada y no por un adversario.
- **Disponibilidad** - Si alguna información o servicio debe estar disponible para su uso o el de otros, asegúrese de que un atacante no pueda impedir su uso. Por ejemplo, si un negocio tiene una gran venta, no se querrá que sus competidores puedan bloquear las calles alrededor de su tienda, impidiendo que sus clientes lleguen a ella.

Una dimensión adicional importante de estos tres objetivos es que queremos que se comparta de forma controlada en nuestros sistemas, compartimos nuestros secretos con algunas personas y no con otras. Permitimos que algunas personas modifiquen las bases de datos de nuestra empresa, pero no cualquiera. Algunos sistemas deben estar disponibles para un conjunto concreto de usuarios preferidos (como

los que han pagado por jugar a su juego en línea) y no para otros (que no lo han hecho). Quién pide importa mucho, tanto en la informática como en la vida cotidiana.

Otro aspecto importante de la seguridad de los sistemas informáticos es que a menudo queremos estar seguros de que cuando alguien nos ha dicho algo, no poder negar después que lo ha hecho. Este aspecto suele denominarse **no repudio**.

Cuanto más difícil y costoso sea para alguien repudiar sus acciones, más fácil será responsabilizarlos por ellas y por lo tanto, menos probable será que la gente realice acciones maliciosas, al fin y al cabo, es muy posible que los descubran y será menos posible negar que lo hicieron.

Estos son grandes objetivos generales, en un sistema real, hay que profundizar en objetivos más detallados y específicos. En un sistema operativo típico, por ejemplo, podríamos tener un objetivo de confidencialidad que establezca que el espacio de memoria de un proceso no puede ser leído arbitrariamente por otro proceso. Podemos tener un objetivo de integridad que establezca que si un usuario escribe un registro en un archivo concreto, otro usuario que no debería poder escribir en ese archivo no pueda cambiar el registro. Podemos tener un objetivo de disponibilidad que establezca que un proceso que se ejecuta en el sistema no puede acaparar la CPU e impedir que otros procesos obtengan su parte de la CPU. Si piensas en lo que has aprendido sobre la abstracción de los procesos, la gestión de la memoria, la programación, los sistemas de archivos, el IPC y otros temas de esta clase, deberías ser capaz de pensar en algunos otros objetivos obvios de confidencialidad, integridad y disponibilidad que probablemente queramos en nuestros sistemas operativos.

Para cualquier sistema particular, incluso los objetivos a este nivel no son lo suficientemente específicos. El objetivo de integridad al que hemos aludido anteriormente, en el que el archivo de un usuario no debe ser sobrescrito por otro usuario no autorizado a hacerlo, nos da una pista sobre la especificidad adicional que necesitamos en nuestros objetivos de seguridad para un sistema concreto. Quizá haya algún usuario que deba poder sobrescribir el archivo, como podría ser el caso de dos personas que colaboran en la redacción de un informe, pero eso no significa que un tercer usuario no relacionado deba poder escribir ese archivo, si no está colaborando en el informe almacenado allí. Tenemos que ser capaces de especificar ese detalle en nuestros objetivos de seguridad. Los sistemas operativos están escritos para ser utilizados por muchas personas diferentes con muchas necesidades diferentes, y la seguridad del sistema operativo debe reflejar esa generalidad. Lo que queremos en los mecanismos de seguridad de los sistemas operativos es flexibilidad en la descripción de nuestros objetivos de seguridad detallados.

En última instancia, por supuesto, el software del sistema operativo debe hacer todo lo posible para hacer cumplir esos objetivos

de seguridad flexibles, lo que implica que tendremos que codificar esos objetivos en formas que el software pueda entender. Por lo general, debemos convertir nuestra vaga comprensión de los objetivos de seguridad en políticas de seguridad muy específicas. Por ejemplo, en el caso del archivo descrito anteriormente, podríamos querer especificar una política como "los usuarios A y B pueden escribir en el archivo X, pero ningún otro usuario puede hacerlo". Con ese grado de especificidad, respaldado por mecanismos cuidadosamente diseñados e implementados, podemos esperar alcanzar nuestros objetivos de seguridad.

Nótese una implicación importante para la seguridad del sistema operativo: en muchos casos, un sistema operativo tendrá los mecanismos necesarios para implementar una política de seguridad deseada con un alto grado de garantía en su correcta aplicación, pero sólo si alguien le dice al sistema operativo precisamente cuál es esa política. Con algunas excepciones importantes (como mantener privado el espacio de direcciones de un proceso a menos que se indique específicamente lo contrario), el sistema operativo simplemente proporciona mecanismos generales que pueden implementar muchas políticas específicas. Sin un diseño inteligente de las políticas y una aplicación cuidadosa de los mecanismos, independiente de lo que el sistema operativo *debería* o *podría* hacer puede no ser lo que el sistema operativo hará.

53.4 Diseñando Sistemas Seguros

Pocos de lectores construirán su propio sistema operativo, ni siquiera harán cambios serios en algún sistema operativo existente, pero esperamos que muchos construyan grandes sistemas de software de algún tipo. La experiencia de muchos informáticos en el diseño de sistemas ha demostrado que hay ciertos principios de diseño que son útiles para construir sistemas con requisitos de seguridad. Estos principios fueron expuestos originalmente por Jerome Saltzer y Michael Schroeder en un influyente documento [SS75], aunque algunos de ellos provienen de observaciones anteriores de otros. Aunque ni los autores originales ni los comentaristas posteriores afirmarían que seguirlos garantizará que un sistema sea seguro, se ha demostrado que prestarles atención conduce a sistemas más seguros, mientras que si se ignoran será bajo su propio riesgo. Los discutiremos brevemente aquí. Si usted construye un gran sistema de software, valdría la pena buscar este documento (o comentarios más comentarios más detallados sobre el mismo) y estudiar los conceptos cuidadosamente.

1. **Economía de mecanismo** - Esto significa básicamente que hay que mantener el sistema lo más pequeño y simple posible, los sistemas simples tienen menos errores y es más fácil entender

APARTE: SEGURIDAD VS. TOLERANCIA A FALLOS
Al hablar de la abstracción del proceso, hablamos de cómo la virtualización protegía a un proceso de las acciones de otros procesos. Por ejemplo, no queríamos que la memoria de nuestro proceso fuera sobrescrita accidentalmente por otro proceso, por lo que nuestros mecanismos de virtualización debían impedir tal comportamiento. Entonces hablábamos sobre todo de fallos o errores en los procesos. ¿Es esto realmente diferente a preocuparse por el comportamiento malicioso, que es el contexto más común en el que discutimos la seguridad? ¿Hemos resuelto ya todos nuestros problemas virtualizando nuestros recursos?
Sí y no. (¿No es una frase útil?) Sí, si virtualizáramos perfectamente todo y no permitiéramos ninguna interacción entre nada, muy probablemente habríamos resuelto la mayoría de los problemas de malicia. Sin embargo, la mayoría de los mecanismos de virtualización no son totalmente a prueba de balas, funcionan bien cuando nadie intenta subvertirlos, pero pueden no ser perfectos contra todas las formas posibles de mal comportamiento. En segundo lugar, y quizás más importante, no queremos aislar totalmente los procesos entre sí. Los procesos comparten algunos recursos del sistema operativo por defecto (como los sistemas de archivos) y pueden optar por compartir otros. Estas relajaciones intencionadas de la virtualización no son problemáticas cuando se usan correctamente, pero las posibilidades de compartir legítimamente que abren son también canales potenciales para ataques maliciosos. Por último, el SO no siempre tiene el control total del hardware...

su comportamiento. Si no entiendes el comportamiento de tu sistema, no podrás saber si logra sus objetivos de seguridad.

2. **Predeterminados a prueba de fallos** - Siempre recurrir a la seguridad por defecto, no la inseguridad. Si se pueden establecer políticas para determinar el comportamiento de un sistema, haz que el valor por defecto de esas políticas sea más seguro, no menos.

3. **Mediación completa** - Este término de seguridad significa que se debe comprobar si una acción a realizar cumple con las políticas de seguridad cada vez que se realiza la acción [3].

[3] Este principio en particular es a menudo ignorado en muchos sistemas, en favor de una menor sobrecarga o usabilidad. Una característica primordial de todo diseño de ingeniería es que a menudo hay que equilibrar los objetivos, como hemos visto antes en el curso, como en los capítulos de programación. Hablaremos de esto en el contexto de la seguridad más adelante.

4. **Diseño abierto** - Suponga que su adversario conoce todos los detalles de su diseño. Si el sistema puede alcanzar sus objetivos de seguridad de todos modos, estás en buena forma. Este principio no significa necesariamente que le cuentes a todo el mundo todos los detalles, sino que bases tu seguridad en la suposición de que el atacante lo ha aprendido todo. En la práctica, frecuentemente lo ha hecho.
5. **Separación de privilegios** - Exija partes o credenciales separadas para realizar acciones críticas. Por ejemplo, la autenticación de dos factores, en la que se utiliza tanto una contraseña como la posesión de una pieza de hardware para determinar la identidad, es más segura que utilizar cualquiera de esos métodos por separado.
6. **Mínimo privilegio** - Dé a un usuario o a un proceso los privilegios mínimos necesarios para realizar las acciones que desea permitir. Cuantos más privilegios le des a una parte, mayor será el peligro de que abuse de esos privilegios, incluso si estás seguro de que la parte no es maliciosa, si se comete un error, un adversario puede aprovechar su error para utilizar sus privilegios superfluos de manera perjudicial.
7. **Mecanismo menos común** - Para diferentes usuarios o procesos, utiliza estructuras de datos o mecanismos separados para manejarlos Por ejemplo, cada proceso tiene su propia tabla de páginas en un sistema de memoria virtual, asegurando que un proceso no pueda acceder a las páginas de otro.
8. **Aceptabilidad** - Una propiedad crítica que no es del agrado de muchos programadores. Si sus usuarios no lo utilizan, su sistema no tiene valor. Demasiados sistemas seguros prometedores han sido abandonados porque exigían demasiado a sus usuarios.

Estos no son los únicos consejos útiles sobre el diseño de sistemas seguros que existen. También hay mucho material bueno sobre cómo dar el siguiente paso, convertir un buen diseño en código que logre la seguridad pretendida, y otro material sobre cómo evaluar si un sistema construido cumple realmente con esos objetivos. Estos temas están más allá del alcance de este curso, pero son extremadamente importantes cuando llega el momento de construir sistemas grandes y complejos. Para la discusión de los enfoques de la programación segura, se puede empezar con Seacord [SE13], si estás trabajando en C. Si se trabaja en otro lenguaje, sería bueno buscar un texto similar específico para ese lenguaje, ya que muchos problemas de programación segura están relacionados con detalles del lenguaje. Para un tratamiento completo sobre cómo evaluar si un sistema es seguro, comenzar con el trabajo de Dowd et al. [D+07].

RECOMENDACIÓN: CUIDADO CON EL ESLABÓN MAS DÉBIL
Conviene recordar que las personas que atacan tus sistemas comparten muchas características contigo. En particular, probablemente sean bastante inteligentes y probablemente sean algo perezosos, en el sentido positivo de que no hacen el trabajo que no necesitan hacer. Eso implica que los atacantes tienden a buscar la forma más fácil posible de superar la seguridad de tu sistema, no van a buscar un *buffer overflow* de día cero si has elegido "password" como contraseña para acceder al sistema.
La implicación práctica para ti es que deberías dedicar la mayor parte del tiempo que dedicas a asegurar tu sistema a identificar y reforzar tu eslabón más débil. Tu eslabón más débil es la parte menos protegida de tu sistema, la más fácil de atacar, la que no puedes ocultar o aumentar con algún sistema de seguridad externo. A menudo, el eslabón más débil de un sistema en funcionamiento son sus usuarios humanos, no su software. Le será difícil difícil cambiar el comportamiento de las personas, pero puedes diseñar el software teniendo en cuenta que los atacantes pueden intentar engañar a los usuarios legítimos para que hagan un mal uso de él. ¿Recuerda el principio del mínimo privilegio? Si un atacante puede engañar a un usuario que tiene todos los privilegios para que haga un mal uso del sistema, será mucho peor que engañar a un usuario que sólo puede dañar sus propios bienes.
En general, pensar en la seguridad es un poco diferente a pensar en muchas otras cuestiones de diseño de sistemas, es más adverso. Si quieres aprender más sobre buenas formas de pensar en la seguridad de los sistemas que construyes, consulta el libro de Schneier "Secretos y mentiras" [SC00]

53.5 Los fundamentos de la seguridad del sistema operativo

En un sistema operativo típico, entonces, tenemos un conjunto de objetivos de seguridad, centrados en varios aspectos de confidencialidad, integridad y disponibilidad, algunos de estos objetivos tienden a ser incorporados en el modelo del sistema operativo, mientras que otros son controlados por los propietarios o usuarios del sistema. Los objetivos incorporados son aquellos que son extremadamente comunes, o que deben ser asegurados para que los objetivos más específicos sean alcanzables. La mayoría de estos objetivos integrados están relacionados con el control del acceso de los procesos a partes del hardware, esto se debe a que el hardware es compartido por todos los procesos de un sistema, y a menos que se controle cuidadosamente el uso compartido, un proceso puede interferir con los objetivos de seguridad de otro proceso. Otros objetivos integrados

están relacionados con los servicios que ofrece el sistema operativo, como los sistemas de archivos, la gestión de la memoria y las comunicaciones entre procesos. Si estos servicios no se controlan cuidadosamente, los procesos pueden subvertir los objetivos de seguridad del sistema.

Está claro que gran parte de la seguridad del sistema va a estar relacionada con la gestión de los procesos. Si el sistema operativo puede mantener una separación limpia de los procesos que sólo puede romperse con la ayuda del sistema operativo, entonces ni el hardware compartido ni los servicios del sistema operativo pueden utilizarse para subvertir nuestros objetivos de seguridad. Este requisito implica que el sistema operativo debe ser cuidadoso a la hora de permitir el uso del hardware y de sus servicios. En muchos casos, el sistema operativo tiene buenas oportunidades para aplicar esa precaución, por ejemplo, el sistema operativo controla la memoria virtual, que a su vez controla completamente a qué direcciones de memoria física puede acceder cada proceso. El soporte de hardware impide que un proceso pueda siquiera nombrar una dirección de memoria física que no esté mapeada en su espacio de memoria virtual. (La gente de software entre nosotros debería recordar agradecer regularmente a la gente de hardware todo el gran material que nos han dado para trabajar).

Las *syscall* ofrecen al sistema operativo otra oportunidad de proporcionar protección. En la mayoría de los sistemas operativos, los procesos acceden a los servicios del sistema haciendo una llamada explícita al sistema, como se ha comentado en capítulos anteriores. Como has aprendido, las llamadas al sistema cambian el modo de ejecución del modo de usuario del procesador a su modo de supervisor, invocando una parte apropiada del código del sistema operativo mientras lo hacen. Ese código puede determinar qué proceso hizo la *syscall* y qué servicio solicitó el proceso. Anteriormente, sólo hemos hablado de cómo esto podría permitir al sistema operativo llamar a la pieza apropiada de código del sistema para realizar el servicio, y mantener un registro de a quién devolver el control cuando el servicio se ha completado. Pero el mismo mecanismo da al sistema operativo la oportunidad de comprobar si el servicio solicitado debe ser permitido bajo la política de seguridad del sistema. Dado que el acceso a los dispositivos periféricos se realiza a través de los controladores de dispositivos, a los que también se suele acceder mediante una *syscall*, el mismo mecanismo puede garantizar la correcta aplicación de las políticas de seguridad para el acceso al hardware.

Cuando un proceso realiza una *syscall*, el sistema operativo utilizará el identificador del proceso en el bloque de control del proceso o una estructura similar para determinar la identidad del proceso. El sistema operativo puede entonces utilizar **mecanismos de control de acceso** para decidir si el proceso identificado está **autorizado** a realizar la acción solicitada. En caso afirmativo, el SO realiza la acción

por sí mismo o se encarga de que el proceso la realice sin más intervención del sistema, si el proceso no está autorizado, el SO puede simplemente generar un código de error para la *syscall* y devolver el control al proceso, si el algoritmo de programación lo permite.

53.6 Resumen

La seguridad del sistema operativo es vital tanto para él como para sus aplicaciones. Los fallos de seguridad en este software permiten que las consecuencias negativas sean esencialmente ilimitadas, aunque lograr la seguridad del sistema es un reto, hay principios de diseño conocidos que pueden ayudar. Estos principios son útiles no sólo en el diseño de sistemas operativos, sino en el diseño de cualquier gran sistema de software.

Lograr la seguridad en los sistemas operativos depende de los objetivos de seguridad que se tengan, estos objetivos suelen incluir objetivos relacionados con la confidencialidad, la integridad y la disponibilidad. En cualquier sistema dado, los detalles más concretos de estos objetivos de seguridad varían, lo que implica que diferentes sistemas tendrán diferentes políticas de seguridad destinadas a ayudarles a cumplir sus objetivos de seguridad específicos. Al igual que en otras áreas del diseño de sistemas operativos, manejamos estas necesidades variables separando las políticas específicas utilizadas por cualquier sistema particular de los mecanismos generales utilizados para implementar las políticas para todos los sistemas.

La siguiente pregunta que hay que plantear es: ¿qué mecanismos debe proporcionar nuestro sistema operativo para ayudarnos a soportar las políticas de seguridad generales? La virtualización de los procesos y la memoria es un mecanismo útil, ya que nos permite controlar en gran medida el comportamiento de los procesos. Describiremos otros mecanismos útiles de seguridad del sistema operativo en los próximos capítulos.

Referencias

[D+07] "The Art of Software Security Assessment" by Mark Dowd, John McDonald, and Justin Schuh. Addison-Wesley, 2007. *Un tratamiento largo y completo de cómo determinar si su sistema de software cumple sus objetivos de seguridad. También contiene consejos útiles para evitar problemas de seguridad en la programación.*

[SC00] "Secrets and Lies" by Bruce Schneier. Wiley Computer Publishing, 2000. *Una buena perspectiva de alto nivel de los retos de la seguridad informática, desarrollada en la extensión de un libro. Destinado a un público de lectores medianamente sofisticados técnicamente, y bien considerados en la comunidad de seguridad. Una lectura obligada si se pretende trabajar en ese campo.*

[SE13] "Secure Coding in C and C++" by Robert Seacord. Addison-Wesley, 2013. *Un libro bien considerado sobre cómo evitar los principales errores de seguridad programando en C.*

[SS75] "The Protection of Information in Computer Systems" by Jerome Saltzer and Michael Schroeder. Proceedings of the IEEE, Vol. 63, No. 9, September 1975. *Un paper muy influyente, especialmente su programación de los principios para el diseño de sistemas seguros.*