Seguridad de aplicaciones web: vulnerabilidades en los controles de acceso

MSc. Susana C. Romaniz

Grupo de Investigación en Seguridad de las Tecnologías de Información y Comunicaciones
Facultad Regional Santa Fe - Universidad Tecnológica Nacional
sromaniz@frsf.utn.edu.ar

Resumen. No hay dudas respecto de que la seguridad de las aplicaciones web es un tema de interés actual y cotidiano. Las complejas y sensibles funcionalidades de las actuales aplicaciones web ha movido el perímetro de seguridad de las organizaciones, y una parte significativa del mismo ahora reside en las propias aplicaciones web. Y los privilegios de acceso a funcionalidades y datos ya no son uniformes y abiertos, sino que requieren de complejos esquemas., resultando esencial la fortaleza de los mecanismos de control de acceso. Las debilidades en los controles de acceso pueden surgir de diferentes fuentes: un diseño pobre de la aplicación hace muy difícil y hasta imposible el chequeo por accesos no autorizados, un simple descuido puede dejar desprotegidas funcionalidades críticas, o suposiciones erróneas acerca del comportamiento de los usuarios dejan a una aplicación web sin protección y pasible de un quiebre de seguridad. En muchos casos, detectar una brecha en los controles de acceso puede resultar trivial, pero en otros casos, puede ser muy difícil, quedando ocultos defectos sutiles dentro de la lógica de la aplicación, especialmente en aplicaciones complejas y de alta seguridad. La lección más importante es que cuando se chequea la robustez de los controles de acceso se debe mirar en todas direcciones, debiendo ser paciente y testear cada paso particular de todas las funcionalidades de la aplicación.

Palabras Claves: Aplicaciones web, seguridad, vulnerabilidad, ataques, control de acceso.

1. INTRODUCCIÓN

No hay dudas respecto de que la seguridad de las aplicaciones web es un tema de interés actual y cotidiano. Para todos los que se encuentran implicados, los riesgos son altos: para las empresas que ven mejoradas sus ganancias con el comercio a través de la Internet, para los usuarios que confían en las aplicaciones web con información sensible, y para los criminales que pueden obtener importantes réditos. La reputación juega un rol crítico: pocas personas desean realizar transacciones sensibles con un sitio web no seguro, razón por la que muy pocas organizaciones revelan detalles acerca de sus propias vulnerabilidades o brechas. Por lo tanto, en la actualidad no es trivial obtener información confiable acerca del estado de la seguridad de las aplicaciones web.

2. EVOLUCIÓN DE LAS APLICACIONES WEB

En los primeros días de Internet, la *World Wide Web* sólo ofrecía sitios web consistentes en repositorios de información contenida en documentos estáticos; fueron creados los navegadores web como un medio de recuperación y visualización de dichos documentos. El flujo de información de interés era en un único sentido: desde el servidor hacia el navegador. La mayoría de los sitios no autenticaba usuarios debido a que cada usuario era tratado de la misma manera y le era presentada la misma información. Cualquier amenaza de seguridad emergente de hospedar un sitio web estaba principalmente relacionada con las vulnerabilidades del software del servidor web [1], que por ciertos eran varias. Si un atacante comprometía un servidor web, generalmente no lograba acceso a información sensible, debido a que la información soportada en el servidor ya estaba abierta al público. Lo común era que un atacante modificara los archivos almacenados en el servidor para desfigurar los contenidos del sitio, o para utilizar el almacenamiento del servidor o el ancho de banda de su conexión para distribuir *warez*.

En la actualidad, la *World Wide Web* resulta casi irreconocible respecto de su formato inicial. La mayoría de los sitios sobre la web son, de hecho, aplicaciones. Las mismas poseen funcionalidades significativas dentro de un dominio, y dependen del flujo de información en dos sentidos entre el servidor y el navegador. Soportan registración de inicio de sesión (*login*), transacciones bancarias y comerciales, búsqueda, y contenido dependiente del usuario. El contenido presentado a los usuarios se genera dinámicamente bajo demanda, y en general está adecua-

do a cada usuario particular. La mayoría de la información procesada es privada y altamente sensible. Por todo esto, la seguridad es una cuestión importante: nadie desea utilizar una aplicación web si considera que su información será divulgada a partes no autorizadas.

Las aplicaciones web traen consigo nuevas y significativas amenazas de seguridad, ya que cada aplicación es diferente y puede contener vulnerabilidades únicas; muchas de las aplicaciones son desarrolladas y mantenidas por equipos internos y, en muchos casos, por desarrolladores que poseen pocos conocimientos respecto de los problemas de seguridad que pueden emerger en el código que están produciendo.

2.1. Funcionalidades comunes de un servidor web

Las aplicaciones web han sido creadas para ejecutar prácticamente todas las funcionalidades pasibles de ser realizadas en línea (entre las que han emergido en los últimos años se incluyen compras, redes sociales, banca hogareña, búsquedas, subastas, juegos de azar, *blogs*, *webmail*, e información interactiva). Además han sido ampliamente adoptadas por las organizaciones para ejecutar funcionalidades internas esenciales (entre las que se incluyen servicios de recursos humanos y gestión de recursos). Y frecuentemente se las utiliza para proporcionar una interface de administración de los dispositivos de hardware (impresoras, ruteadores IP) y del software de soporte (servidores web, sistemas de detección de intrusión).

Numerosas aplicaciones anteriores al surgimiento de las aplicaciones web han sido migradas hacia esta tecnología (aplicaciones del tipo *Enterprise Resource Planning* -ERP-, que eran accedidas utilizando una aplicación propietaria de cliente delgado, se pueden acceder hoy utilizando un navegador web; servicios basados en software tales como correo electrónico, que inicialmente requería de un cliente separado, hoy se puede acceder vía interfaces web). Y esta tendencia continúa en la medida que las aplicaciones de oficina tradicionales, tales como procesadores de texto y planilla de cálculo, están siendo migradas hacia aplicaciones web.

Estamos próximos a que el único software cliente que necesite la mayoría de los usuarios sea un navegador web. Un importante rango de diferentes funcionalidades serán implementadas utilizando un conjunto común de protocolos y de tecnologías, y al hacerlo han de heredar un rango distintivo de vulnerabilidades de seguridad comunes.

2.2. Beneficios de las aplicaciones web

¿En qué se basa esta relevancia significativa de la que gozan las aplicaciones web? Son varios los factores técnicos y comerciales que colaboran en la revolución ocurrida en la manera de cómo se utilizar la Internet: (i) HTTP, el protocolo de comunicaciones central para acceder al *World Wide Web*, es liviano y sin-conexión; introduce resiliencia en la comunicación de eventos de error y evita la necesidad que el servidor mantenga abierta una conexión de red para cada usuario como es en el caso de muchas aplicaciones legacy cliente-servidor; además, HTTP puede ser proxeado y tunelizado con otros protocolos, haciendo posible una comunicación segura en cualquier configuración distribuida; (ii) todos los usuarios web ya cuentan con un navegador instalado en su puesto de trabajo; las aplicaciones web despliegan su interface de usuario dinámicamente dentro del contexto del navegador, evitando así la necesidad de distribuir y de gestionar un componente de software cliente particular; los cambios introducidos en la interface sólo necesitan ser implementados una vez, sobre el servidor, y toman efecto en forma inmediata; (iii) los navegadores actuales incluyen múltiples funcionalidades, posibilitando la construcción de interfaces de usuario ricas y satisfactorias; las interfaces web utilizan técnicas de navegación y controles de entrada estándares, evitando la necesidad de aprender de qué manera funcionan las aplicaciones particulares; la técnica de scripting del lado del cliente permite que las aplicaciones muevan parte de su procesamiento hacia el lado del cliente, y las capacidades de los navegadores se puedan extender de manera arbitraria utilizando componentes del cliente delgado donde resulte necesario; (iv) las tecnologías y los lenguajes básicos utilizados para desarrollar aplicaciones web son relativamente simples; se dispone de un amplio rango de plataformas y herramientas de desarrollo que facilitan la construcción de aplicaciones de alta eficacia por parte de desarrolladores de relativa experiencia, y se puede acceder a una importante cantidad de código fuente abierto y otros recursos para su incorporación en aplicaciones desarrolladas a medida.

2.3. Seguridad de las aplicaciones web

Al igual que con cualquier tecnología nueva, las aplicaciones web vienen acompañadas con una nueva variedad de vulnerabilidades de seguridad. El conjunto de los defectos que se detectan con mayor frecuencia ha evolucionado a lo largo del tiempo: los nuevos ataques están concebidos de forma tal que resulta imposible considerarlos al momento del desarrollo de las aplicaciones, y, a su vez, han surgido nuevas tecnologías que introdujeron nuevas posibilidades de explotación. Por otra parte, algunos problemas han perdido importancia en la medida que se ha incrementado la concientización respecto de ellos, y algunas categorías de defectos se han eliminado como consecuencia de los cambios introducidos en el software de los navegadores web. Es indiscutible que en la actualidad la seguridad en las aplicaciones web es el principal campo de batalla entre atacantes y aquéllos que administran recursos y datos que se deben defender, y es probable que siga así en el futuro mediato.

2.4. "Este sitio es seguro"

En la actualidad, se está haciendo cada vez más importante entre los usuarios la conciencia respecto a que la seguridad en una cuestión relevante para las aplicaciones web; basta consultar la página de Preguntas Frecuentes de una aplicación típica, donde se alienta al usuario a considerar que se trata de una aplicación segura; por ejemplo: "*Este sitio es absolutamente seguro. Ha sido diseñado para utilizar tecnología 128-bit SSL a fin de evitar que usuarios no autorizados visualicen su información. Ud. puede utilizar este sitio con la tranquilidad de que sus datos están seguros con nosotros.*". Como vemos, las aplicaciones web declaran ser seguras debido a que utilizan SSL (*Secure Socket Layer*); asímismo, se les propone a los usuarios verificar el certificado del sitio, informarse sobre los protocolos de criptografía avanzada que se emplean, y a partir de todo ello, se los alienta a confiarles su información personal. Pero de hecho, las mayoría de las aplicaciones web son inseguras debido a condiciones que no tienen ninguna relación con SSL (de esto, no se tiene que inferir que es innecesario para la seguridad de dichas aplicaciones, ya que utilizado adecuadamente, provee una efectiva defensa contra varios ataques significativos).

Diferentes fuentes indican que, dado los resultados de testeos sobre cientos de aplicaciones web realizados en los dos últimos años, porcentajes elevados de ellas están afectadas por algunas categorías comunes de vulnerabilidades, las cuales se describen brevemente a continuación:

i. *Quiebre de la autenticación* (67%). Esta categoría de vulnerabilidad comprende diferentes defectos dentro del mecanismo de inicio de sesión de la aplicación, lo cual puede permitir que un atacante adivine contraseñas débiles, lance un ataque de fuerza bruta, o eluda por completo el proceso de inicio de sesión.
ii. *Quiebre de los controles de acceso* (78%). Esto comprende los casos en que la aplicación fracaza en proteger adecuadamente el acceso a sus datos y sus funcionalidades, potencialmente permitiendo que un atacante visualice datos sensibles de otro usuario hospedados en el servidor, o lleve a cabo acciones privilegiadas.
iii. *Inyección de SQL* (36%). Esta vulnerabilidad permite que un atacante envíe una entrada con un contenido tal que provoque una interferencia en la interacción de la aplicación con las bases de datos del back-end, y así ser capaz de recuperar datos arbitrarios vía la propia aplicación, interferir en la lógica de la misma, o ejecutar comandos sobre el servidor de base de datos.
iv. *Cross-site scripting* (91%). Esta vulnerabilidad permite que usuarios de la aplicación sean blanco de un atacante, el que logra acceder a los datos de dichos usuarios, ejecutando acciones no autorizadas en nombre de los mismos, o desencadenando ataques contra ellos.
v. *Fuga de información* (81%). Comprende los casos en que una aplicación divulga información sensible, la que es utilizada por un atacante para desencadenar un ataque contra la aplicación, mediante el manejo de un error derivado de un defecto.

3. EL PROBLEMA CENTRAL: LOS USUARIOS PUEDEN ENVIAR ENTRADAS ARBITRARIAS

A igual que las aplicaciones distribuidas, las aplicaciones web enfrentan un problema fundamental que debe ser atendido para volverlas seguras. Debido a que el cliente está fuera del control de la aplicación, los usuarios pueden enviar una entrada (*input*) completamente arbitraria hacia la aplicación del lado del servidor. Por lo tanto, la apli-

cación debe asumir que todas las entradas son maliciosas, y que debe tomar medidas para asegurar que los atacantes no pueden utilizar entradas conformadas para comprometer a la aplicación mediante la interferencia con su lógica, obteniendo acceso no autorizado a sus datos y su funcionalidad.

El problema central se manifiesta de diferentes maneras: (i) los usuarios pueden interferir con cualquier porción de los datos transmitidos entre el cliente y el servidor, incluídos parámetros de solicitud, cookies, y cabeceras de HTTP. Cualquier control de seguridad implementado del lado del cliente, tal como chequeos de validación, pueden ser fácilmente eludidos; (ii) los usuarios pueden enviar solicitudes en cualquier secuencia, y pueden enviar parámetros en una etapa diferente de la que espera la aplicación, más de una vez, o no enviarlos. Cualquier suposición que hagan los desarrolladores acerca de cómo han de interactuar los usuarios con la aplicación se pueden violar; (iii) los usuarios no están restringidos a utilizar sólo un navegador web para acceder a la aplicaciones; están disponibles una variada cantidad de herramientas que operan junto, o independientemente, de un navegador, y que ayudan a atacar las aplicaciones web; las mismas pueden realizar solicitudes que un navegador ordinario no podría realizar, y pueden generar una significativa cantidad de solicitudes a una altísima tasa a fin de detectar y explotar problemas.

La mayoría de los ataques contra las aplicaciones web incluyen el envío hacia el servidor de entradas conformadas hábilmente para causar algún evento no esperado o deseado por el diseñador de la aplicación. Entre algunos ejemplos de este tipo de entradas tenemos: cambio del precio de un producto transmitido dentro de un campo oculto form HTML, realizando una compra fraudulenta de un producto a un precio menor; modificar el token de sesión transmitido dentro de una cookie HTTP, y así secuestrar la sesión de un usuario autorizado; remover ciertos parámetros que se envían normalmente, para explotar una debilidad lógica en el procesamiento de la aplicación; alterar alguna entrada que será procesada por una base de datos del back-end, para inyectar una consulta maliciosa en la base de datos y acceder a datos sensibles.

4. FACTORES CLAVES DEL PROBLEMA

El problema principal de seguridad que deben enfrentar las aplicaciones web surge en cualquier situación en la que se deban aceptar y procesar datos no confiables que pueden ser maliciosos. Sin embargo, en el caso de las aplicaciones web, existen varios factores que se han combinado para exacerbar el problema, y que explican por qué en la actualidad tantas aplicaciones web disponibles sobre la Internet hacen tan poco por resolverlo.

A continuación se resumen los factores más destacados, y sobre los que las organizaciones deben reflexionar: (1) *Inmadurez en el conocimiento de los problemas de seguridad*, es menor el nivel de concientización de los problemas de seguridad en las aplicaciones web que en áreas ya mejor establecidas, tales como redes y sistemas operativos, existe una amplia confusión y conceptos falsos respecto de muchos de los fundamentos implicados en la seguridad de aplicaciones web [2] (resulta frecuente encontrar desarrolladores de aplicaciones web experimentados para quienes la explicación de muchos de los tipos básicos de las debilidades resulta ser una completa revelación); (2) *Desarrollo interno*, la mayoría de las aplicaciones web se desarrollan dentro de la organización, por equipos propios o contratados, y aún cuando una aplicación emplee componentes de terceros, éstos generalmente se ajustan utilizando código nuevo; en esta situación, cada aplicación es diferente y puede contener sus propios defectos que resultan únicos, lo que contrasta con el despliegue de una infraestructura típica, en el que una organización puede adquirir un producto de primera línea e instalarlo en base a recomendaciones estándares de la industria [3]; (3) *Simplicidad engañosa*, con las actuales plataformas de aplicación web y de herramientas de desarrollo, resulta posible que un desarrollador novato cree una aplicación potente desde la nada en un corto tiempo; pero existe una enorme diferencia entre la producción de código funcional y de código seguro; muchas aplicaciones web son creadas por personas bien intencionadas pero que carecen del conocimiento y la experiencia para identificar dónde pueden surgir problemas de seguridad; (4) *Rápida evolución del perfil de amenazas*, una de las consecuencias de su relativa inmadurez es que la investigación sobre los ataques y las defensas en las aplicaciones web es un área próspera, en la que se conciben nuevos conceptos y amenazas a una tasa más rápida de lo que se observa en tecnologías más maduras; por ello, un equipo de desarrollo que comienza un proyecto con un completo conocimiento de las amenazas corrientes, bien puede perder este estatus al momento en que la aplicación se encuentre terminada y sea desplegada; (5) *Restricciones de recursos y de tiempo*, la mayoría de los proyectos de desarrollo

de software están sujetos a este tipo de restricciones; generalmente, no es posible contar en los equipos de diseño o de desarrollo con un especialista en seguridad con una dedicación completa, a lo que se agrega que, a menudo, el testeo de seguridad por parte de especialistas se posterga hasta las etapas avanzadas del ciclo de vida del proyecto; en el momento de balancear prioridades, la necesidad de producir una aplicación estable y funcional conforme una planificación posterga las consideraciones de seguridad que resultan menos tangibles; (6) *Tecnología que supera las capacidades*, muchas de las tecnologías esenciales que se emplean en las aplicaciones web hacen su aparición cuando el panorama de la *World Wide Web* era muy diferente, y desde entonces han estado siendo empujadas más allá de los propósitos con que fueron concebidas originalmente; en la medida que las expectativas depositadas en la funcionalidad de una aplicación web evolucionan rápidamente, las tecnologías utilizadas para implementar dicha funcionalidad van detrás; por ello, no debe sorprender que esta situación conduzca a la aparición de vulnerabilidades de seguridad no previstas.

5. EL NUEVO PERÍMETRO DE LA SEGURIDAD

Antes que surgieran las aplicaciones web, los esfuerzos realizados por las organizaciones en asegurarse a sí mismas contra ataques externos estuvieron fuertemente focalizados sobre el perímetro de la red; defender este perímetro significa fortalecer los servicios que se necesita exponer, y filtrar los accesos hacia los demás. Pero las aplicaciones web han cambiado todo lo anterior. Para que una aplicación sea accesible por sus usuarios, el filtrado perimetral debe permitir las conexiones entrantes hacia el servidor que utilizan HTTPS. Y para que la aplicación funcione, el servidor debe tener permitido conectarse con los sistemas que soportan el *back-end*; estos sistemas frecuentemente soportan las operaciones esenciales de la organización y residen detrás de varias capas de defensas a nivel de la infraestructura de comunicaciones. Si existe una vulnerabilidad dentro de una aplicación web, entonces un atacante sobre la Internet pública puede ser capaz de comprometer los sistemas esenciales de la organización localizados en el *back-end* simplemente mediante el envío de datos conformados desde su navegador web. Estos datos atravesarán todas las defensas de la infraestructura de comunicaciones de la organización, de la misma manera que lo hace el tráfico benigno hacia la aplicación web.

El efecto del despliegue generalizado de aplicaciones web es que se ha movido el perímetro de seguridad de una organización. Parte de ese perímetro aún se encuentra en los *firewalls* y los *bastions-hosts*. Pero una parte significativa del mismo ahora está ocupado por las aplicaciones web de la organización. Debido a las múltiples maneras en que las aplicaciones web reciben las entradas del usuario y las pasan hacia sistemas sensibles del *back-end*, son las potenciales puertas de entrada de un amplio rango de ataques, y las defensas contra los mismos se deben implementar dentro de las propias aplicaciones. Una simple línea de código defectuoso en una única aplicación web puede convertir en vulnerables los sistemas internos de una organización. Las estadísticas indicadas anteriormente de la incidencia de las vulnerabilidades dentro de este nuevo perímetro de seguridad, debería darle que pensar a cualquier organización.

Una segunda forma en la que las aplicaciones web han movido su perímetro de seguridad surge de las amenazas que los propios usuarios enfrentan cuando acceden a una aplicación vulnerable. Una atacante malicioso puede hacer uso de una aplicación web benigna pero vulnerable para atacar a cualquier usuario que la visite. Si el usuario está localizado en una red interna organizacional, el atacante puede controlar el navegador del usuario para lanzar un ataque contra la red interna desde la posición de confianza del usuario. Sin ninguna cooperación del usuario, el atacante puede ser capaz de realizar cualquier acción que pudiera ejecutar el usuario, si fuera malicioso.

Los administradores de red están familiarizados con la idea de impedir que sus usuarios visiten sitios web maliciosos, y los mismos usuarios gradualmente están adquiriendo más conciencia de estas amenazas. Pero la naturaleza de las vulnerabilidades de las aplicaciones web implica que una aplicación vulnerable puede representar para sus usuarios y para la organización a la que pertenecen una amenaza no menor que un sitio web que es claramente malicioso. En correspondencia, el nuevo perímetro de seguridad impone un deber de cuidado sobre todos los dueños de aplicaciones para proteger a sus usuarios respecto de ataques contra ellos viabilizados a través de la aplicación.

6. ATAQUE CONTRA LOS CONTROLES DE ACCESO

Como parte de los mecanismos básicos de seguridad de las aplicaciones, los controles de acceso se encuentran construidos por encima de los mecanismos de autenticación y de gestión de sesión; la principal razón por la que una aplicación necesita incluir estas funcionalidades, al menos en términos de la seguridad, es la necesidad de contar con algún mecanismo que le permita decidir si permite la ejecución de la acción indicada en una solicitud sobre los recursos indicados.

Cuando las funcionalidades asociadas al control de acceso son deficientes, un atacante puede, a menudo, comprometer la aplicación completa, tomando el control de la funcionalidad de administración y teniendo acceso a datos sensibles que pertenecen a otros usuarios. Como ya se comentara, la capacidad de los atacantes de quebrar los controles de acceso es una de las categorías que se encuentra con mayor frecuencia dentro de las vulnerabilidades de las aplicaciones web, afectando a un 78% de las aplicaciones recientemente testeadas; si bien puede resultar soprendente, es muy común encontrar aplicaciones que implementan mecanismos robustos de autenticación y de gestión de sesión, pero que fracasan por la desatención habida al momento de adoptar controles de acceso efectivos por encima de los mismos.

Las vulnerabilidades de control de acceso son, conceptualmente, muy simples: la aplicación está permitiendo que un usuario lleve a cabo algo que no debe; las diferencias entre las debilidades derivan de las diferentes maneras en las que se manifiesta este defecto fundamental, y de las diferentes técnicas que se requieren para detectar dicho defecto. A continuación se describen todas estas técnicas, mostrando de qué forma un atacante puede explotar diferentes tipos de comportamiento dentro de una aplicación para ejecutar acciones no autorizadas y tener acceso a datos protegidos.

6.1. Vulnerabilidades comunes

Los controles de acceso se pueden dividir en dos grandes categorías: *vertical* y *horizontal* [4]. Los primeros permiten que diferentes tipos de usuarios tengan acceso a diferentes partes de la funcionalidad de una aplicación; en el caso más simple, generalmente comprende una división entre usuarios ordinarios y usuarios administradores; en casos más complejos, los controles de acceso verticales pueden involucrar roles de usuarios con granulometría fina (*fine-graned*) en base a los que se otorga acceso a funcionalidades específicas, en el que cada usuario tiene asignado un único rol o una combinación de roles diferentes. Los segundos permiten que los usuarios tengan acceso a un determinado subconjunto de una amplio rango de recursos del mismo tipo (por ejemplo, en una aplicación de workflow, un usuario particular puede tener permitido actualizar las tareas asignadas pero sólo leer las tareas asignadas a otros usuarios).

En muchos casos, los controles de acceso verticales y horizontales se entrelazan; por ejemplo, una aplicación de planificación de recursos puede permitir que cada auxiliar de cuentas facture a una unidad organizacional específica pero no a otras; por otro lado, puede permitir que el responsable facture a cualquier unidad; de manera similar, puede permitir que los auxiliares paguen facturas por montos bajos, en tanto que los montos mayores deben ser pagados por el responsable; puede permitir que el director financiero visualice los pagos y cobros de cada unidad organizacional dentro de la empresa, pero no permitir que realice pagos.

Como ya se indicara, los controles de acceso son quebrados en caso que cualquier usuario sea capaz de acceder a funcionalidades o a recursos para los que no posee autorización. Existen dos grandes tipos de ataques contra los controles de acceso, que se corresponden con las dos categorías de control principales: (i) *escalamiento de privilegio vertical*, ocurre cuando un usuario puede ejecutar funcionalidades que no están permitidas al rol que tiene asignado (por ejemplo, si un usuario ordinario puede ejecutar funcionalidades de administrador, o si, en el ejemplo anterior, un auxiliar puede pagar facturas de cualquier monto); (ii) *escalamiento de privilegio horizontal*, ocurre cuando un usuario puede visualizar o modificar recursos sobre los que no posee privilegios (por ejemplo, si un usuario puede utilizar una aplicación webmail para leer el correo electrónico de otras personas, o si, en el ejemplo anterior, un auxiliar puede facturar para una unidad organizacional distinta de la asignada).

Resulta común encontrar casos en los que una vulnerabilidad en la separación horizontal de privilegios de la aplicación puede conducir inmediatamente a un ataque de escalamiento vertical; por ejemplo, si un usuario descubre una manera de establecer una contraseña diferente de usuario, entonces dicho usuario puede atacar una cuenta de admi-

nistrador y tomar el control de la aplicación. En los casos descriptos hasta ahora, el quiebre de los controles de acceso permiten a usuarios que se han autenticado a sí mismos ante la aplicación dentro del contexto de un usuario particular, ejecutar acciones o acceder a datos para los que dicho contexto no los autoriza. Sin embargo, en los casos más serios de quiebre del control de acceso, resulta posible que usuarios completamente no autorizados obtenga acceso a funcionalidades o a datos que se espera sólo sean accedidos por usuarios autenticados privilegiados.

6.1.1. Funcionalidad completamente desprotegida

En muchos casos de quiebre de controles de acceso, funcionalidad y recursos sensibles pueden ser accedidos por cualquier que conozca el URL relevante. Por ejemplo, existen muchas aplicaciones en las cuales cualquiera que visite un URL específico es capaz de hacer uso total de sus funcionalidades de administrador; en esta situación, por lo general la aplicación obliga a un control de acceso sólo en el siguiente caso: los usuarios que han accedido a la aplicación como administradores verán un enlace hacia este URL sobre su interface de usuario, en tanto que los demás usuarios, no; esta diferencia "cosmética" es el único mecanismo existente para "proteger" una funcionalidad sensible contra su uso no autorizado. Algunas veces, el URL que otorga acceso a funcionalidades potentes puede ser menos fácil de conjeturar, e incluso puede resultar de alguna manera críptico; en este caso, las funcionalidades de administrador están protegidas por el supuesto que el atacante no conoce o no puede descubrir dicho URL; esta aplicación puede resultar más difícil de comprometer por parte de una persona externa de la organización, dado que resulta menos probable que lo adivine.

Se cae en el error de afirmar que ningún usuario de bajo privilegio conocerá el URL porque no se lo referencia en ningún lugar dentro de la aplicación; la ausencia de algún control de acceso genuino sigue siendo una vulnerabilidad seria, independientemente de cuán fácil pueda ser conjeturarlo, ya que un URL no posee el estatus de secreto, tanto dentro de la propia aplicación como en las manos de los usuarios: se lo visualiza en la pantalla, aparece en los historiales del navegador, genera registros de eventos de los servidores web y los servidores proxy, los usuarios los pueden marcar entre los preferidos o enviarlos a través de correo electrónico; además, no se los modifica periódicamente como se debe hacer con las contraseñas; cuando los usuarios cambian de roles, y se requiere disminuir su acceso a la funcionalidad de administración, no hay manera de "borrar" su conocimiento de un URL particular.

En algunas aplicaciones en las que la funcionalidad sensible está oculta por URLs que no son triviales de conjeturar, un atacante puede ser capaz de identificarlos mediante la inspección del código del lado del cliente. Muchas aplicaciones utilizan JavaScript para construir dinámicamente la interface de usuario dentro del cliente; generalmente esto trabaja estableciendo diferentes *flags* de acuerdo al estatus del usuario, y adicionando luego elementos individuales a la interface de usuario en base a dichos *flags*; en este caso, el atacante puede sencillamente revisar el JavaScript para identificar los URLs correspondientes a la funcionalidad administrativa e intentar acceder a los mismos; en otros casos, los comentarios HTML pueden contener referencias acerca de URLs que no están vinculados en el contexto que se despliega en la pantalla.

6.1.2. Funcionalidades basadas en identificador

Cuando se utiliza una funcionalidad de una aplicación para obtener acceso a un recurso específico, es muy común ver que se pasa un identificador del recurso solicitado dentro de un parámetro de la solicitud, ya sea dentro de un URL *query string* o del cuerpo de una solicitud POST. Por ejemplo, una aplicación puede utilizar este mecanismo para desplegar un documento particular que le pertenece a un usuario particular; en este caso, cuando dicho usuario inicia una sesión con la aplicación, se despliega un vínculo hacia el documento sobre la página desde donde se ofrece dicha funcionalidad; otros usuarios no pueden visualizar ese vínculo. Sin embargo, si se quiebran los controles de acceso, cualquier usuario que solicite el URL relevante puede ser capaz de visualizar el documento exactamente de la misma manera que si fuera el usuario autorizado.

Este tipo de vulnerabilidad frecuentemente se presenta cuando la aplicación principal posee una interface con un sistema externo o con un componente del *back-end*; puede resultar dificultoso compartir un modelo de seguridad basado en sesión entre diferentes sistemas (que pueden estar basados en tecnologías diferentes). Ante este problema, frecuentemente los desarrolladores toman un atajo y se alejan del modelo, utilizando parámetros enviados por el cliente para tomar decisiones de control de acceso.

En el ejemplo anterior, un atacante que intenta ganar acceso no autorizado necesita conocer no sólo el nombre de la página de la aplicación sino también el identificador del documento que desea visualizar; algunas veces, los identificadores de recurso se generan de una manera altamente impredecible -por ejemplo, de manera aleatoria; en otros casos, pueden ser muy fácilmente conjeturados -por ejemplo, si se generan como números secuenciales. Como ya se indicara, los URLs no poseen el estatus de secretos, y lo mismo ocurre con los identificadores de recursos. Frecuentemente, un atacante que desea descubrir los identificadores de los recursos de otro usuario podrá descubrir algún lugar de la aplicación que los revela, tales como registros de acceso. Aún cuando estos identificadores no sean fácilmente conjeturables, sigue siendo un mecanismo vulnerable para el control de acceso apropiado; en los casos en que el identificador es fácilmente predecible, el problema es todavía más serio y más sencillo de ser explotado.

Los registros de eventos de una aplicación a menudo son una mina de oro en cuanto a la información que poseen, ya que generalmente contienen numerosos ítems de datos que se pueden utilizar como identificadores para comprobar funcionalidad, a la que se accede de esta manera; usualmente, los identificadores que se encuentran dentro de registros de eventos de aplicación incluyen: nombres de usuarios, números identificadores de usuarios, números de documentos, grupos y roles de usuarios, y direcciones de correo electrónico. Además de ser utilizados como referencia hacia recursos basados en datos dentro de la aplicación, este tipo de identificadores también son utilizado a menudo para referenciar funcionalidades de la propia aplicación; es sabido que una aplicación pueden entregar diferentes funcionalidades vía una simple página, la cual accepta un nombre de función o un identificador como un parámetro (nuevamente, en esta situación, los controles de acceso pueden ejecutarse con no mayor robustez que en presencia/ausencia de un URL específicos en las interfaces de los diferentes tipos de usuario); si un atacante puede determinar el identificador correspondiente a una función sensible, puede tener la capcidad de acceder al mismo de la misma manera que si fuera un usuario más privilegiado.

6.1.3. Funcionalidades de múltiples etapas

Diferentes tipos de funcionalidades dentro de una aplicación se implementan a través de varias etapas, requiriendo del envío de múltiples solicitudes enviadas desde el cliente hacia el servidor; por ejemplo, una función que adiciona un nuevo usuario puede requerir la selección de esta opción desde un menú de mantenimiento de usuarios, o la selección del departamente y el rol de usuario a partir de listas desplegables, y luego ingresar el nombre de usuario nuevo, la contraseña inicial y otra información.

Es común encontrar aplicaciones en las que se han realizado esfuerzos para proteger este tipo de funcionalidad sensible contra el acceso no autorizado, pero donde los controles de acceso utilizado son pasibles de quiebre debido a suposiciones erróneas acerca de la manera en que ha de ser utilizada la funcionalidad. En el ejemplo anterior, cuando un usuario intenta cargar el menú de mantenimiento de usuarios, y selecciona la opción para adicionar un usuario nuevo, la aplicación puede verificar que el usuario posee los privilegios requeridos, y bloquear el acceso en caso que no los posea; sin embargo, si el atacante accede directamente a la etapa en que se especifica el departamente del usuario, este control de acceso no será efectivo. El desarrollador ha presupuesto inconscientemente que cualquier usuario que haya alcanzado las últimas etapas del proceso debe tener privilegios relevantes debido a que fue verificado en las etapas iniciales; el resultado es que cualquier usuario de la aplicación puede adicionar una cuenta de usuario administrador, y de esta manera, ganar control pleno de la aplicación, accediendo a muchas otras funcionalidades cuyo acceso es intrínsecamente robusto.

Este tipo de vulnerabilidad se puede encontrar en aplicaciones de seguridad crítica, como el caso de aplicaciones bancarias en línea; en una aplicación de este tipo, la realización de transferencias generalmente comprende varias etapas, en especial para evitar que los usuarios accidentalmente cometan errores cuando solicitan una transferencia; el proceso de múltiples etapas comprende la captura de diferentes ítems de datos que el usuario especifica en cada una de ellas, datos que son chequeados estrictamente cuando se ingresan por primera vez y luego generalmente se los pasa a la etapa subsiguientes, utilizando campos ocultos de un HTML *form*. Sin embargo, si la aplicación no revalida todos estos datos en la etapa final, entonces el atacante está en condiciones de eludir los chequeos del servidor. Por ejemplo, la aplicación podría verificar que la cuenta origen seleccionada para la transferencia pertenece al usuario actual y luego pedir detalles acerca de la cuenta destino y el monto de la transferencia; si un usuario intercepta la solicitud POST final del proceso y modifica el número de cuenta origen, pueden ejecu-

tar un escalamiento de privilegio horizontal y transferir fondos desde una cuenta que pertenece a un usuario diferente.

6.1.4. Archivos estáticos

En la mayoría de los casos, los usuarios ganan acceso a una funcionalidad y a recursos protegidos mediante la creación de solicitudes vía páginas dinámicas que se ejecutan sobre el servidor; es responsabilidad de cada una de estas páginas ejecutar los chequeos adecuados de control de acceso, y confirmar que el usuario posee los privilegios relevantes para ejecutar la acción que está intentando. Sin embargo, en algunos casos, las solicitudes por recursos protegidos se realizan directamente hacia recursos estáticos, los cuales se encuentran alojados dentro del *web root* del servidor. Por ejemplo, una aplicación que permite la publicación en línea puede permitirle a los usuarios navegar por su catálogo de libros y comprar *ebooks* descargables; una vez realizado el pago, el usuario es dirigido hacia un URL de descarga que apunta a un recurso completamente estático (el *ebook* adquirido), el cual no se ejecuta sobre el servidor, y su contenido es devuelto directamente por el servidor web; por lo tanto, el recurso en sí mismo no puede implementar ninguna lógica que le permita verificar que el usuario solicitante posee los privilegios requeridos. Cuando se accede de esta manera a recursos estáticos, resulta muy probable que no existan controles de acceso efectivos que los proteja y que cualquiera que conozca el esquema de nombres del URL puede explotarlo para acceder a cualquier recurso que desee.

Ciertos tipos de funcionalidades son particularmente propensas a este tipo de problemas, incluidos sitios web bancarios que ofrecen acceso a documentos estáticos relativos a empresas tales como informes anuales, vendedores de software que ofrecen la descarga de archivos binarios, y funcionalidad administrativas que dan acceso a archivos de registro estáticos u otros datos sensibles recolectados dentro de la aplicación.

6.1.5. Métodos de control de acceso no seguros

Algunas aplicaciones emplean un modelo de control de acceso particulamente no seguro, en el cual las decisiones de control de acceso se toman a partir de los parámetros de solicitud enviados por el cliente; en algunas versiones de este modelo, la aplicación determina un rol de usuario o un nivel de acceso en el momento que el usuario inicia la sesión y a partir de este momento transmite esa información vía el cliente dentro de un campo HTML *form* oculto, una *cookie* o un parámetro predefinido de HTTP *query string*. Cuando se procesa cada una de las subsiguientes solicitudes, la aplicación lee este parámetro de solicitud y decide qué acceso otorgarle al usuario de manera acorde. Por ejemplo, un administrador que utiliza una aplicación puede visualizar URLs que contienen parámetros ligados a su rol, mientras que los URLs visualizados por usuarios ordinarios contienen parámetros diferentes, o ningún parámetro; cualquier usuario que esté en conocimiento del parámetro asignado a los administradores puede sencillamente declararlo en sus propias solicitudes y de esta manera obtener acceso a funcionalidades de administrador.

En algunas circunstancias, este tipo de control de acceso puede ser difícil de detectar si no se utiliza la aplicación como un usuario que posee alto nivel de privilegio y así identificar qué solicitudes se realizan; existen técnicas que permiten descubrir parámetros de solicitud ocultos que resultan exitosas en el descubrimiento del mecanismo sólo cuando se trabaja como un usuario ordinario.

En otros modelos de control de acceso no seguros, la aplicación utiliza la cabecera HTTP '*Referer*' en base a la cual toma decisiones de control de acceso. Por ejemplo, una aplicación puede realizar un estricto control de acceso al menú principal de administración, en base a los privilegios de usuario; pero cuando un usuario realiza una solicitud por una función de administración particular, la aplicación puede simplemente chequear que esta solicitud fue referenciada desde la página del menú de administración y asumir que, si es así, entonces el usuario debe haber accedido a dicha página y por lo tanto posee los privilegios requeridos. Desde ya que este modelo es pasible de ser quebrado debido a que la cabecera HTTP '*Referer*' se encuentra completamente bajo control del usuario, quien le puede asignar cualquier valor.

6.2. Ataques a los controles de acceso

Antes de comenzar a comprobar la aplicación con la finalidad de detectar alguna vulnerabilidad de control de acceso, se recomienda como primer paso establecer cuáles son los requerimientos reales de la aplicación en térmi-

nos de control de acceso a los fines de focalizar mucho mejor la atención. Las preguntas que se deben considerar al momento de examinar los controles de acceso de una aplicación incluyen: (i) ¿las funcionalidades de la aplicación le otorgan acceso a usuarios particulares a un subconjunto particular de datos que les pertenecen?; (ii) ¿existen diferentes niveles de usuarios, tales como administradores, supervisores, invitados, entre otros, a los que se le otorgan acceso a diferentes funcionalidades?; (iii) ¿los administradores utilizan una funcionalidad que está construida dentro de la misma aplicación, la que tiene como finalidad configurar y monitorear la propia aplicación?; (iv) ¿es posible identificar qué funcionalidades o recursos de datos dentro de la aplicación permiten escalar hasta los actuales privilegios corrientes?

La manera más sencilla y efectiva de testear la efectividad de los controles de acceso de una aplicación es acceder a la aplicación utilizando diferentes cuentas, y determinar si los recursos y las funcionalidad que pueden ser accedidas legítimamente por una cuenta pueden ser accedidas ilegítimamente por otra.

A continuación se presenta una secuencia de acciones que se puede ejecutar para explotar el conjunto de vulnerabilidades descriptas: (i) si la aplicación segrega el acceso de usuario en diferentes niveles de funcionalidad, en primer lugar se utilizará una cuenta alto privilegio para localizar todas las funcionalidades disponibles y luego intentar acceder a las mismas utilizando una cuenta de menor privilegio; (ii) si la aplicación segrega el acceso de usuario a los diferentes recursos (tales como documentos), utilizar dos cuentas diferentes de nivel de usuario para testear si los controles de acceso son efectivos o si es posible un escalamiento de privilegios horizontal (para ello se debe encontrar un documento que puede ser legítimamente accedido por un usuario pero no por el otro, e intentar accederlo utilizando la cuenta del segundo usuario, ya sea mediante la solicitud de un URL relevante o enviando los mismos parámetros POST dentro de la sesión del segundo usuario); (iii) si es posible automatizar algunos de estos testeos ejecutando una herramienta tipo *spider* dos o más veces sobre la aplicación, utilizando un contexto de usuario diferentes cada vez, y también en un contexto que no requiere autenticación (por ejemplo, ejecutar el *spider* como administrador, y luego obtener un token de sesión para un usuario de menor privilegio y volver a enviar los mismos vínculos pero reemplazando el token de sesión priviligiado por el token de menor privilegio); (iv) si al ejecutar la sesión con el *spider* como un usuario ordinario se descubren funcionalidades privilegiadas a las que sólo el administrador debería tener acceso, entonces esto puede representar una vulnerabilidad (no obstante, la efectivdad de este método depende del comportamiento de la aplicación: algunas aplicaciones le proporcionan a todos los usuarios los mismos vínculos de navegación y devuelven un mensaje "acceso denegado" (dentro de una respuesta HTTP 200) cuando se solicita una funcionalidad no autorizada.

En caso que se disponga solamente de una cuenta de nivel de usuario con la cual se accede a la aplicación (o no se accede), entonces se requiere de un trabajo adicional para testear la efectividad de los controles de acceso; de hecho, para realizar un testeo abarcativo, se necesita realizar un trabajo más avanzado, cualquiera sea el caso, debido a que puede exitir una funcionalidad pobremente protegida que no está vinculada explícitamente desde la interface de ningún usuario de la aplicación (tal como una funcionalidad antigua que no ha sido aún removida, o una funcionalidad nueva que ya ha sido desplegada pero que aún no ha quedado disponible a los usuarios).

Nuevamente, se presenta a continuación una secuencia de acciones que se puede ejecutar para explotar el conjunto de vulnerabilidades en un contexto más exigente: (i) hacer uso de técnicas de descubrimiento de contenido para identificar lo más que se pueda acerca de las funcionalidades de la aplicación, empleando un usuario de bajo privilegio, lo que a menudo resulta suficiente tanto para enumerar como para ganar acceso directo a funcionalidades sensibles; (ii) cuando se identifican páginas de aplicación que probablemente presentan diferentes funcionalidades o vínculos para usuarios ordinarios y administradores, tratar de adicionar parámetros triviales (tales como '*admin=true*' al URL *query string* y al cuerpo de solicitudes POST), para determinar si otorga acceso a alguna funcionalidad adicional respecto del contexto del usuario con que se está realizando el testo; (iii) testear si la aplicación utiliza la cabecera HTTP '*Referer*' en base a la cual toma decisiones de control de acceso, y para aquellas funcionalidad principales para las que el usuario en uso posee acceso, tratar de remover o de modificar el contenido de dicha cabecera, y así determinar si la solicitud continúa resultado exitosa; de no ser así, la aplicación puede estar confiando en la cabecera HTTP '*Referer*' de una manera no segura; (iv) analizar todos los HTML y *scripts* del lado del cliente para detectar referencias hacia funcionalidades ocultas o que se pueden manipular del lado del cliente, tales como interfaces de usuario basadas en *scripts*.

Una vez que se han enumerado todas las funcionalidades, resulta necesario testear si la segregación por usuario

para el acceso a los recursos se encuentra garantizada; en cada instancia donde la aplicación le otorga a un usuario acceso a un subconjunto de recursos del mismo tipo (tales como documentos o correos electrónicos), pueden existir oportunidades para que dicho usuario gane acceso no autorizado a otros recursos.

En este contexto, se proponen los siguientes pasos de testeo: (i) cuando la aplicación emplea identificadores de cualquier clase (identificadores de documentos, números de cuentas, referencias de orden, etc.) para especificar cuál recurso está solicitando un usuario, intentar descubrir los identificadores para los que no se posee acceso autorizado; (ii) si resulta posible generar una serie de tales identificadores en una rápida sucesión (por ejemplo, mediante la creación de múltiples documentos u órdenes nuevos), utilizar las técnicas indicadas anteriormente para tratar de descubrir cualquier secuencia de identificadores predecible producida por la aplicación; (iii) si no es posible generar identificadores nuevos, el testeo estará restringido a analizar los identificadores que ya han sido descubiertos, o intentar un plan de conjeturas; si el identificador posee el formato de un GUID *(Globally Unique Identifier*) es poco probable que tenga éxito cualquier tipo de intento basado en conjeturas; sin embargo, si se trata de un número relativamente pequeño, probar con otros números comprendidos en un rango cerrado, o con números aleatorios de la misma cantidad de dígitos; (iv) si se lograr establecer que los controles de acceso son quebrables, y que los identificadores de recurso son predecibles, se puede montar un ataque automatizado a fin de recolectar recursos sensibles e información; se recomienda el empleo de técnicas *bespoke* automatizadas.

En cada instancia en que una aplicación dé la impresión de estar forzando los controles de acceso de manera efectiva, pero dicha impresión no resulte totalmente convincente, se la debe comprobar más a fondo para determinar si se evidencia la existencia de suposiciones inadecuadas asumidas por los desarrolladores. En este caso, se recomienda: (i) donde una acción se lleve a cabo en múltiples etapas, enviar varias solicitudes diferentes desde el cliente hacia el servidor, testear cada solicitud individualmente para determinar si se le han aplicado los controles de acceso; (ii) tratar de determinar cualquier lugar donde la aplicación efectivamente asume que si el usuario ha alcanzado un punto particular, entonces debe haber arribado a través de medios legítimos; tratar de alcanzar ese punto por otras vías utilizando una cuenta de menor privilegio, a fin de detectar si es posible algún tipo de ataque de escalamiento de privilegios.

En caso que los recursos protegidos por la aplicación sean accedidos, en última instancia, a través de URLs asociados a los propios archivos, se deberá testear si es posible que usuarios no autorizados soliciten directamente estos URLs. Para ello se recomiendan los siguientes testeos: (i) realizar el proceso normal que permite acceder a un recurso estático protegido, a fin de obtener un ejemplo del URL mediante el que dicho recurso se recupera finalmente; (ii) utilizando un contexto de diferente usuario (por ejemplo, un usuario de menor o ningún privilegio sobre el recurso) intentar acceder al mismo en forma directa empleando el URL identificado; (iii) si este ataque resulta exitoso, intentar comprender el esquema de nombre que está siendo usado para los archivos estáticos protegidos y, si es posible, construir un ataque automatizado para "rastrillar" la aplicación en busca de contenido que pueda resultar de utilidad o que contenga datos sensibles.

6.3. Aseguramiento de los controles de acceso

Los controles de acceso es una de las áreas de seguridad de aplicaciones web que resulta más fácil de comprender, aunque es necesario aplicar cuidadosamente una metodología global y correctamente documentada al momento de su implementación. En primer lugar, existen muchas trampas que se deben evitar, las que generalmente surgen de desconocer los requerimientos esenciales de un control de acceso efectivo, o de presupuestos incorrectos acerca de los tipos de solicitudes que realizarán los usuarios y contra las que la aplicación se necesita defender a sí misma: (i) no contar con la ignorancia de los usuarios respecto de los URLs o de los identificadores utilizados para especificar los recursos de la aplicación, sino asumir que conocen todos los URLs y los identificadores empleados por la aplicación, y asegurar que los controles de acceso de la misma son suficientes por sí mismos para prevenir accesos no autorizados; (ii) no confiar en ningún parámetro enviado por el usuario para conceder privilegios de acceso; (iii) no asumir que los usuarios accederán a las páginas de la aplicación solamente en la secuencia prevista; (iv) no confiar en que el usuario no manipula indebidamente los datos que son transmitidos a través del cliente, y si algunos de estos datos han sido validados y luego transmitidos, no basarse en los datos transmitidos sin realizar una revalidación.

A continuación se presenta una *metodología de mejores prácticas* para la implementación de controles de acceso efectivos dentro de las aplicaciones web: (1) evaluar y documentar explícitamente los requerimientos de control de acceso para cada funcionalidad de la aplicación; se debe incluir tanto quién puede legitimar el uso de la funcionalidad y a qué recursos pueden acceder los usuarios particulares a través de la funcionalidad; (2) conducir las decisiones de control de acceso desde la sesión de usuario; (3) utilizar un componente de aplicación centralizado para verificar los controles de acceso; (4) procesar todas las solicitudes de cada cliente particular a través de este componente, a fin de validar que el usuario que está realizando la solicitud tiene permitido el acceso a la funcionalidad y a los recursos solicitados; (5) utilizar técnicas sistemáticas para asegurar que no existen excepciones al punto anterior; una solución efectiva es obligar a que todas las páginas de la aplicación implementen una interface que es consultada por el mecanismo de control de acceso centralizado; de esta manera, al obligar a los desarrolladores a codificar explícitamente la lógica de control de acceso en todas las páginas, no habrá excusas ante omisiones; (6) en el caso de una funcionalidad particularmente sensible, tal como páginas de administración, y en caso que se trate de una intranet, se pueden establecer restricciones de acceso adicionales basadas en alguna información única del puesto de trabajo, a fin de asegurar que sólo los usuarios ubicados en dichos puestos tienen la capacidad de acceder a la funcionalidad, independientemente de la identidad conque han iniciado la sesión; (7) si se necesita proteger contenido estático, existen dos métodos para proporcionar acceso a los mismos: (i) los archivos estáticos pueden ser accedidos indirectamente mediante el pasaje de un nombre de archivo a una página dinámica del lado del servidor, la que implementa la lógica de control de acceso; (ii) el acceso directo a archivos dinámicos puede ser controlado utilizando autenticación HTTP u otras características del servidor de aplicación para encapsular la solicitud y chequear los permisos para el recurso antes de otorgar acceso; (8) los identificadores que especifican a cuál recurso desea acceder un usuario son vulnerables de manipulación siempre que son transmitidos a través del cliente; los servidores sólo deberán confiar en la integridad de los datos del lado del servidor; en cada oportunidad que los identificadores son transmitidos a través del cliente, necesitan ser revalidados a fin de asegurar que el usuario está autorizado a acceder al recurso solicitado; (9) en el caso de funcionalidades de aplicaciones en las que la seguridad es un requerimiento crítico, considerar la implementación de re-autenticación por transacción y autorización dual a fin de garantizar que la funcionalidad no está siendo utilizada por una parte no-autorizada; esto permitirá mitigar las consecuencias de otros posibles ataques, tales como secuestro de sesión; (10) registrar todos los eventos de acceso a datos sensibles o de acciones sensibles que se ejecutan; estos registros permiten la detección y la investigación de potenciales brechas en el control de acceso.

Frecuentemente, los desarrolladores de aplicaciones web implementan funciones de control de acceso sobre una base poco sistemática, adicionando código a las páginas individuales en aquellos casos que hayan registrado un requerimiento de algún tipo de control de acceso, cortando y pegando el mismo código de una página a otra para implementar requerimientos similares. Esta forma de trabajo convella un riesgo inherente de introducir defectos en el mecanismo de control de acceso resultante: muchos casos en los que se requieren controles son pasados por alto, los controles diseñados para un área pueden no operar de la manera esperada en otra área, y las modificaciones introducidas en algunos lugares de la aplicación pueden quebrar los controles existentes al ser violadas las suposiciones en las que se basan éstos.

A diferencia de esta solución, el método que se describió anteriormente que hace uso de un componente de aplicación centralizado para forzar la aplicación de los controles de acceso posee muchos beneficios: (1) incrementa la claridad de los controles de acceso dentro de la aplicación, y de esta manera permite que los diferentes desarrolladores comprendan más rápidamente los controles realizados por los demás; (2) vuelve más eficiente y confiable el mantenimiento, ya que la mayoría de los cambios necesitarán aplicarse una sola vez, a un único componente compartido, y no requerirá ser replicado en múltiples lugares; (3) mejora la adaptabilidad, de tal manera que a medida que van surgiendo nuevos requerimientos de control de acceso se los puede reflejar fácilmente a través de la interface de aplicación existente implementada por cada página de aplicación; (4) presenta una menor cantidad de errores y omisiones que si el código de control de acceso estuviera implementado por partes a lo largo de toda la aplicación

6.3.1. Un modelo de privilegios multi-capa

Los inconvenientes relacionados con el acceso no sólo incumben a las aplicaciones web en sí mismas sino tam-

bién a otras capas de infraestructura subyacente, en particular, el servidor de aplicación, la base de datos y el sistema operativo. La adopción de un solución de defensa en profundidad [5] para la seguridad implica la implementación de controles de acceso en cada una de estas capas a los fines de crear varias capas de protección. Esto proporciona una mayor garantía contra las amenazas de acceso no autorizado, debido a que si el atacante tiene éxito en comprometer las defensas de una capa, el ataque aún puede ser bloqueado por las defensas de otra capa.

La implementación de controles de acceso efectivos mediante una solución multi-capa se puede aplicar de diferentes maneras, como por ejemplo: (i) se puede utilizar el servidor de aplicación para controlar el acceso sobre el URL *path* completo en base a los roles de usuario definidos en la capa del servidor de aplicación; (ii) la aplicación puede emplear una cuenta de base de datos diferente cuando realiza acciones de diferentes usuarios (para los usuarios que sólo consultan datos, se deberá utilizar una cuenta con privilegios de sólo lectura); (iii) se puede implementar control *fine-grained* sobre el acceso a las diferentes tablas de la base de datos dentro de la misma base, utilizando una tabla de privilegios; (iv) las cuentas de sistema operativos que se utilizan para ejecutar cada componente dentro de la infraestructura deben estar restringidas a los mínimos privilegios que requiere el componente.

En el caso de una aplicación compleja con requerimientos críticos de seguridad, este tipo de defensas en profundidad se puede diseñar con la ayuda de una matriz, en la que se definen los diferentes roles de usuario dentro de la aplicación y los diferentes privilegios, en cada capa, que deberían se asignados a cada rol.

Dentro de un modelo de seguridad de este tipo, se puede observar el uso de varios conceptos relativos al control de acceso [6]: (1) *control sistemático*, la matriz de privilegios individuales se almacena como una tabla dentro de la base de datos, y se aplica en forma sistemática para forzar las decisiones de control de acceso; la clasificación de los roles de usuario proporciona un medio rápido para la realización de ciertas verificaciones de control de acceso, y esto también se aplica de manera sistemática; este tipo de control puede alcanzar un nivel de granularidad muy fino y se puede conformar una lógica arbitrariamente compleja embebida dentro del proceso de toma de decisión de control de acceso que forma parte de la aplicación; (2) *control de acceso discrecional* (*DAC*), existen dos modelos: (i) DAC cerrado, en el cual el acceso está denegado a menos que se otorgue explícitamente, y (ii) DAC abierto, en el que el acceso está permitido a menos que lo quite explícitamente; (3) *control de acceso basado en rol* (*RBAC*), en el que existen roles con nombres, los cuales contienen diferentes conjuntos de privilegios particulares, y cada usuario está asignado a uno de estos roles; ésta es una vía rápida para la asignación y aplicación forzada de los diferentes privilegios y es necesario para ayudar en la gestión del control de acceso en aplicaciones complejas; el empleo de roles para la verficación anticipada de acceso de las solicitudes de usuario permite que muchas solicitudes no autorizadas sean rápidamente rechazadas con una mínima cantidad de procesamiento (un ejemplo de esta solución es la protección de URL *paths* a los que pueden acceder un tipo de usuario específico); cuando se diseñan mecanismos de control basado en roles, resulta necesario balancear el número de roles de tal manera que sigan siendo una herramiento útil en la gestión de privilegios dentro de un aplicación; (4) *control declarativo*, emplea diferentes cuentas para los diferentes grupos de usuarios, y cada cuenta posee el mínimo nivel de privilegio necesario para llevar a cabo las acciones que el grupo correspondiente tiene permitido ejecutar; este tipo de controles se declaran por fuera de la aplicación; aún en el caso que un usuario descubra una brecha en los controles de acceso implementados dentro de la capa de aplicación tal que le permita ejecutar una acción sensible (como adicionar una usuario nuevo) se verá impedido de hacerlo debido a que la cuenta de base de datos que está utilizando no posee los privilegios requeridos dentro de la misma; un medio diferente de aplicar control de acceso declarativo existe en la capa del servidor de aplicación, vía el empleo de descriptores de archivos, los cuales se aplican durante el proceso de despliegue de la aplicación; sin embargo, éstos pueden ser instrumentos relativamente toscos, que no siempre escalan adecuadamente en la gestión de privilegios de granulometría fina existentes en una aplicación compleja.

Cuando se intenta atacar una aplicación que emplea un modelo de privilegios multi-capa de este tipo, es probable que puedan ser prevenidos muchos de los errores más obvios que se producen en el empleo de controles de acceso; lo que se debe determinar es que si se eluden los controles implementados dentro de la aplicación resulta imposible ir mucho más allá debido a la protección existente en otras capas; con esto en mente, aún están disponibles líneas de ataques potenciales. Lo más importante es comprender las limitaciones de cada tipo de control, y en qué términos no se puede satisfacer la protección deseada; esto ayudará a identificar las vulnerabilidades que muy probablemente afecten al esquema: (i) realizar chequeos sistemáticos dentro de la capa de aplicación, la que puede ser

susceptible de ataques basados en inyección; (ii) los roles definidos en la capa de aplicación a menudo están toscamente definidos y pueden ser incompletos; (iii) cuando los componentes de aplicación se ejecutan utilizando cuentas del sistema operativo con bajos privilegios, las mismas aún pueden poseer la capacidad de muchos tipos de datos potenciales sensibles dentro del sistema de archivos del servidor; cualquier vulnerabilidad que otorgue arbitrariamente acceso a los archivos aún se puede ser explotadas; (iv) las vulnerabilidades existentes en el software del servidor de aplicación por lo general serán capaces de hacer fracasar todos los controles de acceso implementados dentro de la capa de aplicación; no obstante, aún puede persistir un acceso limitado a la base de datos y al sistema operativo; (v) una única vulnerabilidad explotable en el control de acceso ubicada en el lugar preciso aún puede proporcionar un punto de arranque para un escalamiento de privilegios importante (por ejemplo, si se descubre una manera de modificar el rol asociado al usuario con que se está llevando adelante el ataque, entonces es posible que volviendo a iniciar la sesión con dicha cuenta se le otorgue un acceso mejorado tanto a la capa de aplicación como a la de base de datos).

7. EL FUTURO DE LA SEGURIDAD DE LAS APLICACIONES WEB

Transcurridos varios años después de su amplia adopción, las aplicaciones web desplegadas en la Internet aún hoy están están plagadas de vulnerabilidades [7, 8]. Comprender las amenazas de seguridad que se ciernen sobre estas aplicaciones, y la manera efectiva de enfrentarlas, permanece inmadura en la industria. Actualmente, existen pocos indicios que los factores antes descriptos vayan a ser superados en un futuro cercano.

Esto significa que los detalles del alcance de la seguridad de las aplicaciones web no es estático; en tanto las vulnerabilidades más antiguas y bien comprendidas, tal como inyección de SQL, continúan apareciendo, su preponderancia está disminuyendo gradualmente. Más aún, las instancias que aún perduran se están volviendo más difíciles de encontrar y de explotar. Se está investigando muy activamente en técnicas avanzadas que atacan manifestaciones más sutiles de vulnerabilidades que hace algunos años podían ser muy fácilmente detectadas y explotadas utilizando sólo un navegador web.

Otra tendencia destacada es el gradual desplazamiento de la atención de los ataques tradicionales contra la parte del servidor de la aplicación hacia los usuarios. Los últimos tipos de ataques aún aprovechan defectos dentro de la aplicación misma, pero que generalmente involucran algún tipo de interacción con otro usuario y comprometen las acciones del mismo con la aplicación vulnerable. Esta es una tendencia que se está reproduciendo en otras área de seguridad de software. A medida que madura la concientización de las amenazas de la seguridad, las debilidades del lado del servidor son las primeras en ser comprendidas y atendidas, dejando al lado del cliente como un campo de batalla clave en el que continúa el proceso de aprendizaje. De todos los ataques existentes, aquéllos contra otros usuarios están evolucionando más rápidamente, y son objeto de la mayoría de las investigaciones en la actualidad.

REFERENCIAS

1. Ristic, I. “Apache Security”. ISBN 10-0-596-00724-8. O’Reilly Media Inc. Estados Unidos de América. 2005.
2. Baier, D. “Devoloping More-Secure Microsoft ASP.NET 2.0 Applications”. ISBN 10-0-7356-2331-7. Microsoft Press. Estados Unidos de América. 2006.
3. Donahue, G. “Network Warrior”. ISBN 10-0-596-10151-1. O’Reilly. 2007.
4. Peikari, C. y Chuvakin, A. “Security Warrior”. ISBN 13-978-0-596-00545-0. O’Reilly Media Inc. Estados Unidos de América. 2004.
5. Santos, O. “End-to-End Network Security: Defense-in-Depth”. ISBN10-1-58705-332-2. Cisco Press. Networking Technology Series. 2007.
6. Ferraiolo, D., Khun, D. y Chandramouli, R. “Role-Based Access Control” 2ed. ISBN 10-1-59693-113-2. Artech House. Estados Unidos de América. 2007.
7. Letinen, R., Russell, D. y Gangemi, G.T. “Computer Security Basics”. 2ed. ISBN 10-0-596-00669-1. O’Reilly Media Inc. Estados Unidos de América. 2006.
8. Smith, B. y Komar, B. “Microsoft Windows Security Resource Kit”. ISBN 10-0-7356-1868-2. Microsoft Press. Estados Unidos de América. 2003.