Medidas de seguridad y vulnerabilidades de los servidores web

SWAP

Hugo Maldonado Cózar
Aarón Rodríguez Bueno

Índice

Introducción

Para poder hablar de seguridad en servidores, tenemos que partir de la base de que los servidores no son más que computadores “convencionales” que proporcionan servicios a usuarios, aplicaciones, empresas…

Por tanto, antes de empezar a asegurar nuestro servidor, lo primero que tenemos que hacer es asegurar nuestro computador y su sistema operativo para evitar el máximo número de fallos de seguridad que nuestro computador pudiera tener.

Seguridad del Computador

Podemos definir la seguridad del computador como la protección asequible de un sistema de información automatizado al que le sean aplicables los objetivos de preservar la **integridad**, **disponibilidad** y **confidencialidad** (tríada **CIA**) de los recursos del Sistema:

- **Confidencialidad.** Evitar la divulgación no autorizada de información. Preservar las restricciones autorizadas sobre el acceso y revelación, incluyendo los medios para proteger la privacidad personal y la propieda de la información. Algunas formas en que podemos conseguirlo son:

    - **Cifrado o encriptación**. Transformación de la información utilizando un secreto (clave de cifrado) de forma que sólo pueda leerse con otro secreto (clave de descifrado).
    - **Control de acceso**. Reglas y políticas que limitan el acceso a la información confidencial a las personas o sistemas que necesitan conocerla.
    - **Autenticación.** Determinación de la identidad o rol que tiene un usuario en el sistema.
    - **Autorización.** Determinación de si una persona/sistema tiene permitido el acceso a los recursos, basado en una política de control de acceso.
    - **Seguridad Física.** Establecer barreras físicas para limitar el acceso a los recursos protegidos.

- **Integridad.** Guardar la información frente a modificaciones inadecuadas o destructivas, incluyendo asegurar la autenticidad y no repudiación de la información. Algunas formas en que podemos conseguirlo son:

    - **Copias de seguridad**. Archivar periódicamente los datos para una posterior restauración si se diese algún caso de alteración indebido.
    - **Sumas de verificación**. Funciones hash que mapean los contenidos de un archivo a un valor alfanumérico de forma que si se altera el contenido, el valor difiere.

    - **Códigos de corrección de datos**. Método para almacenar los datos de forma que pequeños cambios puedan ser detectados y corregidos.

- **Disponibilidad.** Permitir que la información sea accesible y modificable en el tiempo para quienes tienen autoridad para ello. Asegurar el acceso fiable y a tiempo al uso de la información o del sistema de información. Se puede conseguir mediante:

    - **Protección física**. Infraestructuras necesarias para mantener la disponibilidad de la información frente a desastres naturales.
    - **Redundancia computacional**. Replicar los dispositivos de cómputo, redes, almacenamiento...

La seguridad del computador está definida por la **política de seguridad**, que son un conjunto de declaraciones que particionan el sistema entre *estados seguros* e *inseguros*. Se puede asumir que un sistema es seguro si sólo transiciona de un estado seguro a otro seguro. En la práctica son un conjunto de documentos en lenguaje natural que establecen las reglas de un sistema, qué y quién debe proteger la información.

La política de seguridad no establece el cómo debe protegerse, de eso se encargan los **mecanismos de seguridad**. Son estos los que aplican cómo debe hacerse la transición de un estado seguro a otro seguro. Los mecanismos de seguridad se dividen en 3 categorías: **autenticación**, **autorización** y **cifrado**. Algunos ejemplos de mecanismos de seguridad son Control de Acceso Discrecional (DAC), Control de Acceso Obligatorio (MAC), Monitor de referencia…

Estos mecanismos de seguridad por sí solos no son suficientes, si no que, individualmente, pueden ser efectivos contra una amenaza específica. Por esto, se necesita detallar un **modelo de seguridad**, que es el que dicta cómo se estructura y aplica la política de seguridad.

Los modelos de seguridad especifican el uso de los mecanismos de seguridad, normalmente siguiendo una combinación de principios apoyados en un marco teórico para describir las propiedades de seguridad. El principio más común es la tríada CIA (comentada anteriormente). Algunos ejemplos de modelos de seguridad son: Matrix de Control de Acceso, Bell-La Padula, GCFA (Generalized Framework for Access Control), FLASK (Flux Advanced Security Kernel)...

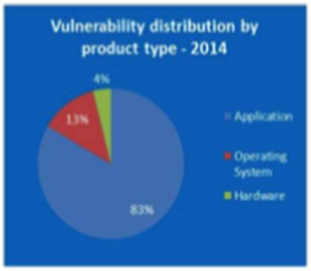
Seguridad del Sistema Operativo

Una vez tenemos estos conceptos teóricos básicos en mente, vamos a describir conceptos que se tiene que tener en cuenta de la seguridad de los sistemas operativos, así como relatar cómo endurecer el mismo para que nuestro servidor esté lo más seguro posible.

Tenemos que partir de la base en que todos los sistemas operativos, como software que son, tiene **vulnerabilidades**, es decir debilidades del sistema que pueden ser aprovechadas para causar daño en el mismo.

	(1)	(2)	(3)	(4)
Operating system	# of vulnerabilities	# of HIGH vulnerabilities	# of MEDIUM vulnerabilities	# of LOW vulnerabilities
Apple Mac OS X	147	64	67	16
Apple iOS	127	32	72	23
Linux Kernel	119	24	74	21
Microsoft Windows Server 2008	38	26	12	0
Microsoft Windows 7	36	25	11	0
Microsoft Windows Server 2012	38	24	14	0
Microsoft Windows 8	36	24	12	0
Microsoft Windows 8.1	36	24	12	0
Microsoft Windows Vista	34	23	11	0
Microsoft Windows RT	30	22	8	0

- C. Florian, GFI Blog, 2015 en "Most vulnerable operating systems and applications in 2014"
http://www.gfi.com/blog/most-vulnerable-operating-systems-and-applications-in-2014/
datos de la National Vulnerability Database (NVD).



Distribución	(1)	(2)	(3)	(4)
Ubuntu	39	7	27	5
Red Hat E.	27	6	17	4
OpenSuse	22	9	9	4
Fedora	15	3	9	4
Android[1]	6	4	1	1

[1] Sin incluir las del kernel

Todos los sistemas operativos tienen lo que se denomina **modelo de confianza**, que define el conjunto mínimo de software y datos sobre los cuales el sistema depende para hacer cumplir de forma correcta los **objetivos de seguridad** del mismo (operaciones que el sistema puede ejecutar mientras evita el acceso no autorizado). Para un SO, un modelo de confianza es sinónimo de la **Base de Computación Segura** (**TCB**) del sistema.

Forman parte del TCB:

- Mecanismos de arranque del sistema (que habilita la carga de los objetivos)
- SO completo. No hay fronteras de protección. La seguridad de un sistema siempre se establece por su punto más débil.
- Utilidades: login, ssh, smtp…
- Entorno que realiza servicios para todos los procesos del sistema y mecanismos de compartición...

Se garantiza la confianza del TCB mediante 3 mecanismos básicos:

- **Funcionamiento en modo dual**. Depende de qué servicio o proceso se ejecuta en modo kernel o en modo usuario.
- **Protección de memoria**. El SO construye un espacio de direcciones aislado para cada proceso. Un proceso no puede acceder a nada fuera de su espacio de direcciones, ni siquiera puede "nombrarlo".
- **Autorización** del uso de los recursos del sistema a quienes les está permitido, impidiendo el acceso a quienes no lo están.

Con estos conceptos descritos, vamos a pasar al apartado de **fortalecimiento del Sistema Operativo**, para asegurar el sistema reduciendo la superficie de amenaza.

Es un proceso que requiere planificación sobre los servicios a instalar en el sistema (cuantas más funciones realice el sistema, mayor será la posibilidad de encontrar una vulnerabilidad que pueda ser explotada).

Este proceso no tiene fin, el sistema no es estático y cada día aparecen nuevos ataques, por lo que hay que aumentar continuamente el fortalecimiento y su posterior auditoría.

Debemos considerar:

- Tipo de información almacenada, aplicaciones y servicios a suministrar y sus requisitos de seguridad.
- Categorizar a los usuarios: privilegios e información sobre la que pueden acceder.
- Mecanismos de autenticación (contraseñas, biométricos, 2 fases…).
- Cómo se gestiona el acceso a la información almacenada en el sistema.
- Acceso a la información almacenada en otros sistemas (servidores de bases de datos, de archivos…)
- Quién administra el sistema y cómo lo gestiona (local o remotamente).
- Firewall
- Antimalware
- etc.

El endurecimiento de la seguridad de un sistema abarca todas las etapas del sistema:

- # Instalación

  Es importante comprobar las fuente de la descarga, es decir, asegurarnos que el sistema operativo que tenemos es legítimo y no ha sido modificado durante la descarga (por ejemplo con algún ataque MITM).

  Otra consideración a tener en cuenta es (si vamos a instalar LINUX) compilar nosotros mismos el kernel del sistema, eligiendo detalladamente cada opción de éste, ya que puede ser bastante interesante no permitir ciertas funcionalidades que nosotros no vamos a utilizar en nuestro servidor para evitar un posible ataque por ahí (por ejemplo, deshabilitar los puertos USB, si los tiene, directamente en el kernel, para que nadie pudiera robar información si tiene acceso físico al servidor, o incluso destruirlo con los pen drives de “la muerte”). Además, algunas de las distribuciones precompiladas están pobremente configuradas con respecto a la seguridad, por lo que de esta forma podemos fortalecer mucho nuestro SO. Los mayores inconvenientes son que es una tarea bastante completa y hay que saber muy bien qué se está haciendo con cada opción de compilación y que con cada actualización del kernel del SO, tendríamos que volver a compilarlo nosotros mismos con las opciones que elijamos.

  La instalación siempre debe hacerse con el sistema desconectado de la red, para evitar posibles ataques antes de fortalecer el sistema.

  No se deben instalar paquetes que no vayamos a utilizar, ya que aumenta la superficie de ataque, por ejemplo, en un servidor, raramente vamos a necesitar herramientas de desarrollo, de impresión…

Y para el resto de aplicaciones determinar perfectamente quién y cómo pueden utilizarla y acceder, es decir, limitar el uso y utilización de herramientas, servicios, aplicaciones a usuarios concretos o grupos de usuarios para reducir los posibles ataques.

- ## Proceso de Arranque

Un atacante con acceso físico a nuestro servidor podría evitar mecanismos de seguridad del sistema y manipular el proceso de arranque para su beneficio.

Debemos tener la BIOS-UEFI protegida con contraseña, así como el cargador de arranque del sistema.

Es importante deshabilitar los arranques desde otros medios (CD-ROMS, Pen drives…).

Otro aspecto interesante es proteger los servicios que acceden a la red en el arranque, por ejemplo, arrancar antes de conectar a la red **iptables**, **syslog** y herramientas similares para asegurarnos mayor seguridad (no conectar a la red y luego arrancar estas herramientas).

- ## Ejecución del Sistema operativo

Eliminar todo el software no necesario para nuestro servidor (o restringir su uso a un grupo de usuarios específico), así como los servicios que no vayamos a necesitar.

Alterar las cuentas que vienen por defecto, así como los mecanismos de acceso.

Establecer el **principio de menor privilegio**, es decir, establecer los permisos mínimos que necesitan los usuarios para realizar su cometido, así como sobre los archivos. También se puede hacer mediante la asignación de roles.

Control de recursos del sistema, para detectar posibles amenazas en ejecución.

Registrar todos los eventos del sistema y auditoría para detectar posibles ataques.

Muchas de estos cambios en nuestro sistema pueden tener efectos imprevistos, por lo que siempre es mejor realizarlos sobre máquinas de prueba antes que sobre nuestro servidor final para evitar problemas de disponibilidad y caídas de nuestros servicios.

- ## Actualizaciones

Uno de los elementos más críticos de la seguridad son las actualizaciones del SO y las aplicaciones/servicios, ya que continuamente se descubren nuevas vulnerabilidades y distintas formas de atacar y el software se actualiza para parchearlos.

Al igual que con la instalación, es necesario comprobar siempre la integridad de las actualizaciones.

Y si es posible, realizar las actualizaciones primero en máquinas de prueba y después incluso desconectados de la red para evitar problemas de ataques en el proceso.

- ## Controles adicionales

  Tener antimalware instalado en nuestro computador, así como firewalls eficientemente configurados.

  IDS / **IPS** para monitorizar el tráfico o comprobaciones de integridad de archivos, así como para ayudar en una mejor y más fuerte protección.

  Tener listas blancas de aplicaciones permitidas en el sistema, haciendo que todas las que no se encuentren en esta lista no podrán ejecutarse.

  Desarrollar nuestros servicios/aplicaciones teniendo muy presente la seguridad también es fundamental para evitar posibles fallos que se puedan atacar (como Buffer Overflow, Stack Smashing, Shellcode…).

Seguridad del Servidor Web

El hackeo de un servidor web consiste en explotar vulnerabilidades en el software del servidor web y paquetes de software asociados con muchos posibles fines distintos (extorsión, robo de información, secuestro del servidor, acceso a la información, hacktivismo…).

Vulnerabilidades

Para configurar un servidor web para prevenirlo de posibles ciberataques, es importante conocer las vulnerabilidades que se pueden aprovechar del mismo. Algunas vulnerabilidades son fáciles de detectar y atacar. Un atacante con recursos puede echar un servidor web vulnerable en minutos.
Algunas de las vulnerabilidades que trataremos son las asociadas al software más común en los servidores web: Microsoft IIS/ASP/ASP.NET, LAMP (Linux/Apache/MySQL/PHP), BEA WebLogic, IBM WebSphere, J2EE, … Las acompañaremos de algunos ejemplos concretos y reales:

- **Archivos de muestra**

  Las plataformas web presentan una larga lista de características, parámetros y funcionalidad. Para hacer más fácil su uso, los proveedores de software frecuentemente meten en su software archivos y scripts de cómo usar su producto. Por desgracia esto es un arma de doble filo, ya que muchas veces parte de esas

funcionalidades o están configuradas con una seguridad pobre o directamente accesibles al público.

Un caso de vulnerabilidad clásica de este apartado es Microsoft's IIS 4.0, el cual permitía a los atacantes descargarse el código fuente ASP.

Por suerte, los fabricantes cada vez utilizan menos estos archivos de muestra y tanto éstos como la documentación forman parte del proceso de revisión de seguridad previo.

- **Divulgación del código fuente**

  Permiten a un usuario malintencionado ver el código fuente de los archivos de aplicación en un servidor web vulnerable que pretende permanecer confidencial. Bajo ciertas condiciones, el atacante puede combinar esto con otras técnicas para ver archivos protegidos importantes como / etc / passwd, global.asa, etc.

  Algunas de las vulnerabilidades de divulgación de código fuente más clásicas incluyen la vulnerabilidad IIS + .htr y problemas similares con Apache Tomcat y BEA WebLogic relacionados con la adición de caracteres especiales a las solicitudes de Java Server Pages (JSP).

  Esta vulnerabilidad han sido removidas hace mucho tiempo, o se han publicado soluciones provisionales. Aún así, en nuevas versiones donde se pueda divulgar el código fuente podría volver a aparecer esta debilidad.

- **Ataques de Canonicalización**

  Los recursos de la computadora y de la red se pueden a menudo dirigir usando más de una representación. Por ejemplo, el archivo C: \ text.txt también puede ser accedido por la sintaxis .. \ text.txt o \\ computer \ C $ \ text.txt. El proceso de resolver un recurso a un estándar (Canónico) nombre se llama Canonicalización. Aplicaciones que toman decisiones de seguridad basado en el nombre del recurso puede ser fácilmente engañado en la realización de acciones imprevistas utilizando los llamados ataques de canonización.
  La vulnerabilidad ASP::$DATA en IIS de Microsoft fue uno de los primeros temas de canonización publicados en una gran plataforma web. Esta vulnerabilidad permite al atacante descargar el código fuente de Active Server Pages (ASP) en lugar de hacerlos renderizar dinámicamente por el motor IIS ASP. La hazaña es fácil, simplemente utilizando el siguiente formato de URL cuando se descubra una página ASP:

  http://servidor/scripts/archivo.asp::$DATA

  También se encontró que Apache contenía una vulnerabilidad de canonización cuando se instalaba en servidores que ejecutan Windows. Si el directorio que contenía los scripts del servidor se encontraba dentro del directorio raíz del

documento, podría obtener el código fuente de los scripts CGI haciendo una solicitud directa para el archivo de script con, por ejemplo, la siguiente configuración:

DocumentRoot "C:/Documents and Settings/http/site/docroot"

ScriptAlias /cgi-bin/ "C:/Documents and Settings/http/site/docroot/cgi-bin /"

El uso normal haría una solicitud POST a http://[target]/cgi-bin/foo (observe la minúscula "cgi-bin"). Sin embargo, un atacante podría recuperar el origen en el script foo simplemente solicitando http://[target]/CGI-BIN/foo (anote las letras mayúsculas). Esta se produce debido a que los algoritmos de enrutamiento de solicitud de Apache son sensibles a mayúsculas y minúsculas, mientras que el sistema de archivos de Windows no distingue entre mayúsculas y minúsculas.

Una vulnerabilidad de canonización común fue la de Unicode / Double Decode, también en IIS. Fue explotada en su momento por el famoso gusano Nimda.

- **Extensiones del servidor**

Por sí solo, un servidor web proporciona un mínimo de funcionalidad. Gran parte viene en forma de extensiones, que son bibliotecas de código que se agregan al núcleo del motor HTTP para proporcionar características tales como ejecución de script dinámico, seguridad, almacenamiento en caché y mucho más.

Por desgracia, las extensiones a menudo traen problemas a lo largo del tiempo. La historia está llena de vulnerabilidades en las extensiones de servidores web: la extensión de indexación de Microsoft, que fue víctima de los desbordamientos de búfer; Internet Printing Protocol (IPP), otra extensión de Microsoft que fue víctima de ataques de desbordamiento de búfer alrededor de IIS5; Creación y revisión automatizada de versiones web (WebDAV); Secure Sockets Layer (SSL, por ejemplo, las vulnerabilidades de desbordamiento de búfer de mod_ssl de Apache y el conjunto de bibliotecas de Netscape Network Security Services); y así.

Estos módulos complementarios que se convirtieron en gloria -y se desvanecieron en la infamia en muchos casos- deberían servir como un recordatorio de las compensaciones entre funcionalidad adicional y seguridad. Las extensiones de WebDAV han sido particularmente afectadas por vulnerabilidades. Diseñado para permitir que varias personas accedan, carguen y modifiquen archivos a un servidor web, ha habido muchos problemas serios identificados en las implementaciones de WebDAV de Microsoft y Apache.

El problema de Microsoft WebDAV Translate: f es un ejemplo particularmente bueno de lo que sucede cuando un atacante envía una entrada inesperada que hace que el servidor web bifurque la ejecución a una biblioteca vulnerable complementaria. La vulnerabilidad Translate: f se explota mediante el envío de una solicitud HTTP GET mal formada para un script ejecutable del lado del servidor o un tipo de archivo relacionado, como los archivos Active Server Pages (.asp) o global.asa.

Con frecuencia, estos archivos están diseñados para ejecutarse en el servidor y nunca se deben renderizar en el cliente para proteger la confidencialidad de la lógica de programación, las variables privadas, etc. La solicitud mal formada hace que IIS envíe el contenido de dicho archivo al cliente remoto en lugar de ejecutarlo utilizando el motor de secuencias de comandos apropiado. Los aspectos clave de la solicitud HTTP GET mal formada incluyen un encabezado especializado con Traducir: f al final de la misma y una barra invertida final (\) anexada al final de la URL especificada en la solicitud.

Es una desafortunada realidad que muchos sitios todavía codifiquen contraseñas de aplicaciones en archivos .asp y .asa, y aquí es donde el riesgo de mayor penetración es más alto. Si unl atacante consiguió descargar el archivo .asa de las contraseñas, ha obtenido contraseñas para varios servidores back-end, incluido un sistema LDAP. Los scripts exploit de Perl que simplifican la vulnerabilidad anterior basada en netcat están disponibles en Internet.

Traducir: f surge de un problema con WebDAV, que se implementa en IIS como un filtro ISAPI llamado httpext.dll que interpreta las solicitudes web antes que el núcleo de IIS. El encabezado Translate: f señala el filtro WebDAV para manejar la solicitud, y la barra invertida final confunde al filtro, por lo que envía la solicitud directamente al SO subyacente. Windows 2000 devuelve felizmente el archivo al sistema del atacante en lugar de ejecutarlo en el servidor. Este es también un buen ejemplo de un tema de canonización. Especificar una de las diversas formas equivalentes de un nombre de archivo canónico en una solicitud puede hacer que la solicitud sea manejada por diferentes aspectos de IIS o del sistema operativo.

La vulnerabilidad anteriormente comentada ::$DATA en IIS es un buen ejemplo de un problema de canonización al solicitar el mismo archivo con un nombre diferente, un atacante puede hacer que el archivo se devuelva al navegador de una manera inadecuada. Parece que Translate: f funciona de manera similar. Al confundir WebDAV y especificar "false" para traducir, un atacante puede hacer que el flujo del archivo sea devuelto al navegador.

- **Desbordamiento del buffer**

El temido ataque de desbordamiento de búfer simboliza el golpe de gracia de la piratería. Dadas las condiciones apropiadas, los desbordamientos de búfer a menudo resultan en la capacidad de ejecutar comandos arbitrarios en la máquina víctima, normalmente con niveles de privilegio muy altos. Los desbordamientos de búfer han sido un chink en la armadura de la seguridad digital durante muchos años.

Los desbordamientos más fáciles de explotar se denominan sobrecostos de búfer basados en pila, que indican la ubicación de código arbitrario en la pila de ejecución de la CPU. Más recientemente, los denominados desbordamientos de búfer basados en heap también se han vuelto populares, donde el código se inyecta en el montón y se ejecuta.

El software de servidor Web no es diferente de ningún otro, y también, es potencialmente vulnerable a los errores comunes de programación que son la causa raíz de los desbordamientos de búfer. Desafortunadamente, debido a su posición en las líneas de frente de la mayoría de las redes, los desbordamientos de búfer en el software de servidor web pueden ser verdaderamente devastadores, permitiendo a los atacantes saltar de un compromiso de borde simple en el corazón de una organización con facilidad. Por lo tanto, estos fallos de seguridad son los que se deben evitar a cualquier costo.

Algunos de los desbordamientos más graves en software fueron:

La vulnerabilidad de desbordamiento de pila de IIS ASP que afectaba a Microsoft IIS 5.0, 5.1 y 6.0. Permitía a un atacante que puede colocar archivos en el servidor web para ejecutar código de máquina arbitrario en el contexto del software del servidor web. Se publicó un exploit para esta vulnerabilidad en 2006.

La vulnerabilidad de desbordamiento de hembra de transferencia de codificación troceada HTR de IIS afectaba a Microsoft IIS 4.0, 5.0 y 5.1. Esto conducía potencialmente a la denegación de servicio remota o a la ejecución remota de código en el nivel de privilegios IWAM_MACHINENAME. Se publicó un exploit para esta vulnerabilidad.

IIS también sufrió desbordamientos de búfer en la extensión de servicio de Index Server (idq.dll), que podía explotarse mediante el envío de solicitudes .ida o .idq a un servidor vulnerable. Esta vulnerabilidad resultó en el infame gusano Code Red.

Otros "antiguos pero buenos" desbordamientos de búfer de IIS incluyen la vulnerabilidad del Protocolo de Impresión de Internet (IPP), y una de las primeras vulnerabilidades graves de desbordamiento de búfer identificadas en un servidor web comercial, IISHack. Al igual que muchos servicios de Windows, IIS también se vio afectado por las vulnerabilidades de la biblioteca de protocolos ASN.1.
Para no quedarse atrás, las plataformas web de código abierto también sufrieron algunas vulnerabilidades graves de desbordamiento de búfer. La vulnerabilidad de mod_rewrite de Apache afectó a todas las versiones hasta e incluyendo Apache 2.2.0 y resulta en la ejecución remota de código en el contexto del servidor web. La vulnerabilidad de Apache mod_ssl (también conocida como gusano Slapper) afectó a todas las versiones hasta e incluyendo Apache 2.0.40 y resultó en la ejecución remota de código en el nivel de superusuario. Apache también sufría una vulnerabilidad en la forma en que manejaba las solicitudes HTTP codificadas con codificación fragmentada que resultó en un gusano llamado "Scalper", que se cree que es el primer gusano Apache.

Normalmente, la forma más fácil de contrarrestar vulnerabilidades de desbordamiento de búfer es aplicar un parche de software, preferiblemente de una fuente confiable.

Vulnerabilidades de aplicaciones web

Nos referimos a ataques a las propias aplicaciones, en contraposición al software del servidor web en el que se ejecutan estas aplicaciones. La piratería de aplicaciones web implica muchas de las mismas técnicas que la piratería de servidores web, incluyendo ataques de validación de entrada, ataques de revelación de código fuente, etc.

La principal diferencia es que el atacante se centra ahora en el código de aplicación personalizada y no en el software de servidor estándar. Como tal, el enfoque requiere más paciencia y sofisticación. Vamos a esbozar algunas de las herramientas y técnicas de hacking de aplicaciones web en esta sección:

- **Encontrar aplicaciones web vulnerables mediante buscadores web**

  Los motores de búsqueda indexan un gran número de páginas web y otros recursos. Los hackers pueden utilizar estos motores para realizar ataques anónimos, encontrar víctimas fáciles y obtener los conocimientos necesarios para montar un poderoso ataque contra una red. Los motores de búsqueda son peligrosos en gran parte porque los usuarios son descuidados. Además, los motores de búsqueda pueden ayudar a los hackers a evitar la identificación. Los motores de búsqueda hacen descubrir máquinas candidatas casi sin esfuerzo.

  En la última década, los motores de búsqueda han acumulado una gran cantidad de atención negativa para exponer información sensible. Como resultado, muchas de las consultas más "interesantes" ya no devuelven resultados útiles.

  Por ejemplo, utilizando Google se puede obtener trivialmente una lista de páginas accesibles al público en un sitio web, simplemente usando los operadores de búsqueda avanzada:

  • site:example.com
  • inurl:example.com

  Para encontrar directorios no protegidos /admin, /password /mail y su contenido, busca las siguientes palabras clave en Google:

  • “Index of /admin”
  • “Index of /password”
  • “Index of /mail”
  • “Index of /” +banques +fi letype:xls (para Francia)
  • “Index of /” +passwd
  • “Index of /” password.txt

  Para encontrar las aplicaciones de sugerencias de contraseñas mal configuradas, escriba lo siguiente en http://www.google.com (muchos de estos enumeran usuarios, dan consejos para contraseñas, o contraseñas de la cuenta de correo a una dirección de correo electrónico que se le especifique):

• password hint
• password hint –email
• show password hint –email
• fi letype:htaccess user

- **Rastreo web**

  Un atacante serio se toma su tiempo para familiarizarse con la aplicación. Esto incluye la descarga de todo el contenido del sitio web de destino y la búsqueda de "la recogida de los frutos", como información de ruta local, nombres de servidor de fondo y direcciones IP, cadenas de consulta SQL con contraseñas, comentarios informativos y otros datos confidenciales en los elementos siguientes:

  • Páginas estáticas y dinámicas
  • Incluye y otros archivos de soporte
  • Código fuente
  • Cabeceras de respuesta del servidor
  • Cookies

  - **Herramientas de rastreo web**

    ¿Cuál es la mejor manera de obtener esta información? Debido a que recuperar un sitio web completo es por naturaleza tedioso y repetitivo, es un trabajo muy adecuado para la automatización. Afortunadamente, existen muchas buenas herramientas para realizar el rastreo web, como wget y HTTrack.

    Wget: es un paquete de software libre para recuperar archivos mediante HTTP, HTTPS y FTP, los protocolos de Internet más utilizados. Es una herramienta de línea de comandos no interactiva, por lo que se puede llamar fácilmente desde scripts, trabajos cron y terminales sin soporte X.

    La Copiadora de Sitios Web HTTrack: es una aplicación multiplataforma gratuita que permite a un atacante descargar un número ilimitado de sus sitios web favoritos y sitios FTP para su posterior visualización, edición y navegación fuera de línea. Las opciones de línea de comandos proporcionan la capacidad de secuencias de comandos y una interfaz gráfica fácil de usar, y WinHTTrack está disponible para Windows.

    Debido a que la navegación del sitio se realiza en código ejecutado en el navegador del cliente, AJAX y otras técnicas dinámicas de programación web pueden confundir incluso al mejor rastreador. Sin embargo, hay herramientas para analizar y rastrear aplicaciones AJAX. Crawljax, una de estas herramientas, realiza un análisis dinámico para reconstruir los cambios de estado de la interfaz de usuario y crear un gráfico de flujo de estado.

- **Evaluación de aplicaciones web**

Una vez que el contenido de la aplicación de destino se ha rastreado y analizado a fondo, el atacante normalmente hará un sondeo más profundo de las principales características de la aplicación. El objetivo final de esta actividad es entender a fondo la arquitectura y el diseño de la aplicación, señalar los posibles puntos débiles y romper la aplicación de forma lógica.

Para lograr este objetivo, cada componente principal de la aplicación será examinado desde un punto de vista no autenticado, así como desde la perspectiva autenticada si se conocen las credenciales adecuadas (por ejemplo, el sitio puede permitir el registro gratuito de nuevos usuarios, o tal vez el atacante ya ha recogido las credenciales de rastrear el sitio). Los ataques de aplicaciones Web suelen centrarse en las siguientes funciones:

• Autenticación
• Gestión de sesiones
• Interacción con la base de datos
• Validación de entrada genérica
• Lógica de la aplicación

Debido a que muchas de las fallas más serias en las aplicaciones web no pueden analizarse sin las herramientas adecuadas, comenzamos con una enumeración de herramientas comúnmente utilizadas para realizar el hackeo de aplicaciones web, entre ellas:

• Plug-ins del navegador
• Conjunto de herramientas gratuitas
• Escáneres de aplicaciones web comerciales

- **Explorador de Plug-ins**

Los complementos del navegador le permiten ver y modificar los datos que envía al servidor remoto en tiempo real mientras navega por el sitio web. Estas herramientas son útiles durante la fase de descubrimiento, cuando se intenta averiguar la estructura y la funcionalidad de la aplicación web, y son invaluables cuando se intenta confirmar vulnerabilidades en la fase de verificación.

El concepto detrás de las herramientas de seguridad del complemento del navegador es ingenioso y sencillo: instale una pieza de software en el navegador web que supervisa las solicitudes a medida que se envían al servidor remoto. Cuando se observe una nueva solicitud, pausarla temporalmente, mostrar la solicitud al usuario y dejar que la modifiquen antes de que se desconecte.

Como atacante, estas herramientas son invaluables para identificar campos de formulario ocultos, modificar argumentos de consulta y encabezados de solicitud e inspeccionar la respuesta del servidor remoto. Actualmente casi todos los navegadores proporcionan un mecanismo sencillo para crear complementos multiplataforma y ricos en funciones.

Por ejemplo, el plug-in TamperData le da al atacante control completo sobre los datos que su navegador envía al servidor. Las solicitudes se pueden modificar antes de que se envíen, y un registro de todo el tráfico se mantiene, lo que permite al usuario modificar y reproducir solicitudes anteriores. Junto con una herramienta como NoScript para desactivar JavaScript de forma selectiva, un hacker tiene todo lo necesario para la piratería de sitios web ad hoc.

Al evaluar las aplicaciones web que utilizan mucho JavaScript, puede ser útil tener un depurador que le permita examinar y recorrer el JavaScript de una página a medida que se ejecuta. El depurador de JavaScript de Venkman proporciona esta funcionalidad para Firefox. Microsoft proporciona el Editor de secuencias de comandos de Microsoft como parte de la suite de Office, que habilita la depuración de JavaScript en IE.

- **Suites de herramientas**

Normalmente construidas alrededor de proxies web que se interponen entre el cliente web y el servidor web, las suites de herramientas son más potentes que los complementos de navegador. Invisibles para el navegador web del cliente, los proxies también pueden utilizarse en situaciones en las que el cliente no es un navegador, sino algún otro tipo de aplicación (como un servicio web). La integración de las herramientas de prueba con un proxy proporciona una herramienta eficaz para pruebas ad hoc de aplicaciones web. Por ejemplo, Fiddler es un servidor proxy que actúa como un hombre en el medio durante una sesión HTTP. Desarrollado por Microsoft, se integra con cualquier aplicación construida en la biblioteca WinINET, incluyendo Internet Explorer, Outlook, Office y muchos más. Cuando está habilitado, Fiddler interceptará y registrará todas las solicitudes y respuestas. Pueden establecerse puntos de interrupción, lo que le permite modificar solicitudes antes de que salgan al servidor web y manipular la respuesta del servidor antes de devolverlo a la aplicación cliente. Fiddler también proporciona un conjunto de herramientas para realizar transformaciones de texto y probar los efectos de un ancho de banda bajo y conexiones degradadas.

Otra herramienta es WebScarab, la cual es un framework de pruebas de seguridad de aplicaciones web basado en Java, desarrollado como parte del Proyecto de Seguridad de Aplicaciones Web Abiertas (OWASP). Construido alrededor de un motor proxy extensible, WebScarab incluye una serie de herramientas para analizar aplicaciones web, incluyendo spidering, análisis de ID de sesión y examen de contenido. WebScarab también incluye herramientas "fuzzing". Fuzzing es un término genérico para lanzar datos aleatorios en una interfaz (ya sea una API de programación o un formulario web) y examinar los resultados para detectar signos de posibles errores de seguridad. Debido a que está escrito en Java, WebScarab se ejecuta en un gran número de plataformas y se puede ampliar fácilmente utilizando una interfaz integrada de Bean. Las herramientas de WebScarab para analizar y visualizar identificadores de sesión proporcionan una manera fácil de identificar implementaciones de administración de sesiones débiles.

Más que un simple proxy, Burp Suite es un conjunto completo de herramientas para atacar aplicaciones web. Burp Proxy proporciona la funcionalidad habitual para interceptar y modificar el tráfico web, incluyendo la intercepción condicional y el reemplazo automático de cadenas basado en patrones. Las solicitudes pueden modificarse y reproducirse utilizando la herramienta Burp Repeater y Burp Sequencer puede utilizarse para evaluar la solidez de la gestión de sesiones de la aplicación. Burp Spider recopila información sobre el sitio web de destino, analiza HTML y JavaScript para proporcionar a los atacantes una imagen completa de la aplicación. Una vez que haya utilizado las herramientas de Burp Proxy y Spider para comprender el objetivo, puede utilizar Burp Intruder para comenzar a atacarlo.

Burp Intruder es una poderosa herramienta para crear ataques automatizados contra aplicaciones web. El atacante define una plantilla de solicitud de ataque, selecciona un conjunto de cargas útiles para incorporarlas a las plantillas de ataque y, a continuación, suelta una descarga de solicitudes. Burp Intruder procesa las respuestas y presenta los resultados de los ataques.

Medidas de seguridad preventivas contra ciberataques

- Cerciorarse de que todos los archivos de muestra que pudiera haber en nuestro servidor que no sean necesarios sean removidos del mismo.

- Se debe asumir que la lógica de las páginas de su aplicación web estará expuesta a ojos indiscretos, y nunca debe almacenar datos confidenciales, como contraseñas de base de datos o claves de cifrado, en el origen de la aplicación.

- Se deberían almacenar los scripts del servidor fuera del árbol de documentos, ya que si por ejemplo el software web es sensible a mayúsculas pero el sistema operativo del servidor no, puede producirse un fallo de seguridad (ataque de canonicalización).

- Mantenerse al día en la plataforma web (aparte de las actualizaciones, las posibles vulnerabilidades que se pudieran encontrar recientemente) y compartimentar adecuadamente la estructura del directorio de aplicaciones.

- Se recomienda restringir la entrada usando soluciones de capa de plataforma como URLScan de Microsoft (en IIS), que pueda eliminar las URL que contienen caracteres Unicode o doble hexadecimal antes de que lleguen al servidor.

- No codificar contraseñas (ni datos sensibles) de aplicaciones en archivos .asp y .asa, .php... porque aquí es donde el riesgo de mayor penetración es más alto. Es mejor tener un fichero de configuración donde estipulemos este tipo de datos sensibles y carguemos ese fichero en memoria durante la ejecución.

- Parchear o deshabilitar las extensiones vulnerables conocidas (preferentemente hacer las dos cosas).

Maneras de detectar vulnerabilidades

Afortunadamente, existen varias herramientas que automatizan el proceso de análisis de servidores web para la multitud de vulnerabilidades que continúan transmitiéndose a la comunidad de hackers.

- **Escáneres de vulnerabilidad de servidores Web**

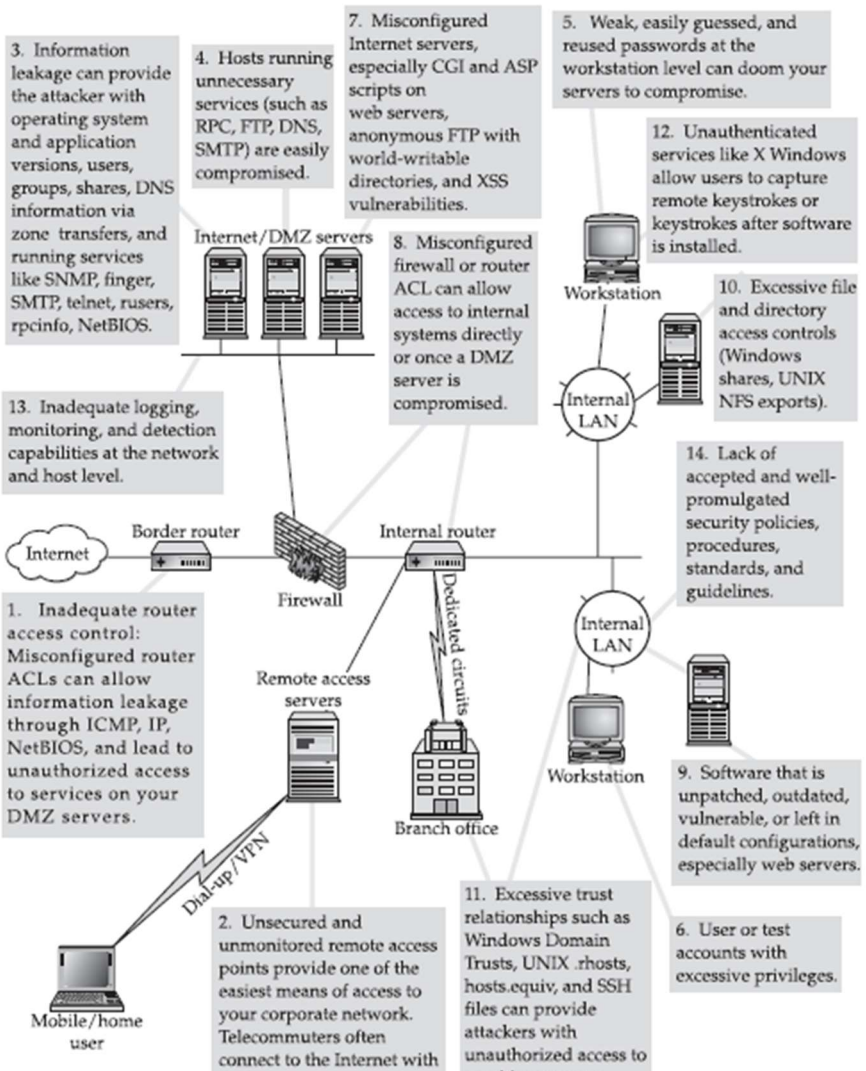
  Estos tipos de herramientas buscarán decenas de vulnerabilidades bien conocidas. Los atacantes pueden utilizar su tiempo de manera más eficiente en la explotación de las vulnerabilidades encontradas por la herramienta, por lo que dichos fallos se deben subsanar conforme los detecte la herramienta. Además, muchos no sólo sirven para buscar vulnerabilidades, también pueden buscar posible malware.

  Actualmente hay muchos escáneres de este tipo, tanto online como instalables, y de pago o gratis. Algunos online son:

  - Scan My Server
  - Quttera
  - Detectify

Las vulnerabilidades de seguridad más comunes

En la red de un servidor web, los elementos más comúnmente vulnerables y sus razones son las que aparecen en la imagen siguiente (no las vamos a describir ya que la imagen se explica por sí misma):


3. Information leakage can provide the attacker with operating system and application versions, users, groups, shares, DNS information via zone transfers, and running services like SNMP, finger, SMTP, telnet, rusers, rpcinfo, NetBIOS.
4. Hosts running unnecessary services (such as RPC, FTP, DNS, SMTP) are easily compromised.
7. Misconfigured Internet servers, especially CGI and ASP scripts on web servers, anonymous FTP with world-writable directories, and XSS vulnerabilities.
5. Weak, easily guessed, and reused passwords at the workstation level can doom your servers to compromise.
12. Unauthenticated services like X Windows allow users to capture remote keystrokes or keystrokes after software is installed.
Internet/DMZ servers
8. Misconfigured firewall or router ACL can allow access to internal systems directly or once a DMZ server is compromised.
Workstation
10. Excessive file and directory access controls (Windows shares, UNIX NFS exports).
Internal LAN
13. Inadequate logging, monitoring, and detection capabilities at the network and host level.
14. Lack of accepted and well-promulgated security policies, procedures, standards, and guidelines.
Internet
Border router
Internal router
Firewall
Dedicated circuits
1. Inadequate router access control: Misconfigured router ACLs can allow information leakage through ICMP, IP, NetBIOS, and lead to unauthorized access to services on your DMZ servers.
Internal LAN
Remote access servers
Workstation
9. Software that is unpatched, outdated, vulnerable, or left in default configurations, especially web servers.
Branch office
Dial-up/VPN
11. Excessive trust relationships such as Windows Domain Trusts, UNIX .rhosts, hosts.equiv, and SSH files can provide attackers with unauthorized access to sensitive systems.
6. User or test accounts with excessive privileges.
2. Unsecured and unmonitored remote access points provide one of the easiest means of access to your corporate network. Telecommuters often connect to the Internet with little protection, exposing sensitive files to attack.
Mobile/home user