Módulo 2. Endpoints seguros

En este módulo aprenderemos a fortalecer la seguridad de los *endpoints* (equipos cliente y servidores, Windows y Linux) mediante configuraciones básicas de *hardening*. El contenido se divide en dos unidades: una dedicada a Windows y otra a Linux. Mantendremos un enfoque práctico, con explicaciones claras y ejercicios para aplicar los conceptos en entornos de pequeñas y medianas empresas (pymes) sin infraestructura compleja. Al final se incluyen actividades de laboratorio guiadas y un caso de estudio integrador para afianzar lo aprendido.

Unidad 1. Windows

En esta unidad abordaremos las medidas fundamentales para asegurar equipos con Windows. Veremos el uso de Objetos de Directiva de Grupo (GPO) para aplicar políticas básicas de seguridad, configuraremos las protecciones integradas de Windows Defender y el firewall de Windows, aprenderemos a habilitar auditorías y registros de eventos de seguridad, y repasaremos la importancia de las actualizaciones del sistema y del Control de cuentas de usuario (UAC).

Objetos de directiva de grupo (GPO) básicas

Las directivas de grupo permiten administrar de forma centralizada las configuraciones de usuarios y equipos con Windows, especialmente en entornos de dominio Active Directory.

Un objeto de directiva de grupo (GPO) es, esencialmente, un conjunto de configuraciones de directivas y permisos de seguridad que se pueden aplicar a usuarios y equipos. En un dominio Active Directory, los GPO se almacenan y replican en los controladores de dominio, y pueden vincularse a distintos niveles (sitios, dominios, unidades organizativas) para definir su ámbito de aplicación.

Porque permiten asegurar configuraciones clave de forma consistente en todos los equipos de la empresa. Por ejemplo, mediante GPO se pueden forzar políticas de contraseñas seguras, establecer reglas de bloqueo de cuentas tras intentos fallidos, desplegar software de seguridad o actualizaciones, prohibir el uso de dispositivos USB no autorizados, activar el *firewall* local en los equipos o incluso definir una página de inicio obligatoria en el navegador. Muchas de estas configuraciones (especialmente las relacionadas con contraseñas, bloqueo de pantalla, *firewall*, etc.) refuerzan la seguridad y no pueden ser modificadas por los usuarios, lo que garantiza el cumplimiento de las políticas corporativas.

En entornos de pymes sin un dominio Active Directory, no se cuenta con GPO centralizadas, pero sí es posible usar la directiva de grupo local de cada equipo (accediendo al editor de directivas de grupo local con `gpedit.msc`). A través de la directiva local, se pueden aplicar muchas de las políticas mencionadas a un solo equipo. Por ejemplo, en Windows Pro se puede usar «gpedit» para establecer requisitos de complejidad de contraseñas,

habilitar la auditoría de eventos de inicio de sesión o impedir el acceso al Panel de control a usuarios estándar. Estas configuraciones locales ofrecen un nivel básico de *hardening* inicial.

Ejemplos básicos de GPO o directiva de seguridad local para pymes

Entre las configuraciones que pueden aplicarse se encuentran las siguientes: definir una longitud mínima y complejidad de contraseña; establecer que, después de cinco intentos fallidos de inicio de sesión, la cuenta se bloquee temporalmente; impedir que Windows almacene contraseñas LM/NTHash antiguas; deshabilitar la función «Autoplay/Autorun» de dispositivos extraíbles (para evitar la ejecución automática de *malware* desde USB); y configurar políticas de contraseña del protector de pantalla para bloquear la sesión tras un periodo de inactividad. Todas estas opciones pueden establecerse desde la directiva de seguridad local del equipo o mediante GPO en un dominio.

Para aplicar estas políticas en un equipo local, se puede utilizar la «Directiva de seguridad local» (`secpol.msc`) o el «Editor de directivas de grupo local» (`gpedit.msc`). Por ejemplo, las opciones de auditoría, políticas de cuenta y restricciones locales se encuentran en `gpedit` en la ruta: «Configuración del equipo» > «Configuración de Windows» > «Configuración de seguridad».

En un entorno de dominio, esas mismas opciones se configuran mediante GPO a nivel de Active Directory. Las GPO incluyen secciones de «Configuración del equipo» (Computer Configuration) y «Configuración de usuario» (User Configuration), lo que permite controlar parámetros del sistema (por ejemplo, deshabilitar servicios, configurar actualizaciones, *firewall*, etc.) y aspectos del entorno del usuario (como la redirección de carpetas, asignación de unidades de red o políticas de Internet Explorer).

En resumen, las GPO (o las directivas locales, en su defecto) son una herramienta fundamental para estandarizar la postura de seguridad de todos los equipos Windows en la empresa. Con ellas nos aseguramos de que todos los usuarios y sistemas cumplan ciertas políticas mínimas de seguridad (contraseñas robustas, pantallas bloqueadas, antivirus activo, *firewall* habilitado, etc.) sin depender de la acción manual de cada usuario.

Nota: en Windows Home no está disponible el «Editor de directivas de grupo local», lo que limita la aplicación de políticas de manera local. En entornos pequeños sin Active Directory, esta carencia puede suplirse parcialmente mediante la configuración manual de las mismas opciones en el registro o mediante herramientas de línea de comandos (como `net accounts` para políticas de contraseñas o `auditpol` para auditorías), aunque esto requiere conocimientos más avanzados.

Defender / Firewall —

Windows 10 y Windows 11 cuentan con «Microsoft Defender Antivirus» y el «Firewall de Windows Defender» integrados de forma gratuita. Estas herramientas proporcionan una primera línea de defensa básica en *endpoints*

con Windows.

Windows Defender Antivirus

Es la solución *antimalware* integrada de Windows. Desde el primer momento tras instalar el sistema, «Defender» está activado y analiza el equipo en tiempo real. Con el paso de los años, ha mejorado notablemente su eficacia y hoy ofrece una protección competente sin coste adicional. «Defender» utiliza análisis de firmas y técnicas avanzadas de detección (como *machine learning* y análisis de comportamiento en la nube) para identificar virus, *spyware*, *ransomware* y otras amenazas.

Es importante mantener siempre habilitado y actualizado «Defender». Windows Update distribuye periódicamente actualizaciones de inteligencia de seguridad (firmas) para el antivirus. En la sección «Protección contra virus y amenazas» de la aplicación «Seguridad de Windows» (Windows Security), se puede ver el estado de «Defender», realizar análisis bajo demanda y configurar sus opciones. Lo recomendable es tener activadas todas las opciones de protección en tiempo real: «Protección en tiempo real», «Protección en la nube», «Envío de muestras automático» y «Control de aplicaciones potencialmente no deseadas». «Defender» también incluye funciones como la «Protección contra ransomware» (con copia de seguridad de carpetas en OneDrive), que conviene aprovechar si es posible.

En caso de contar con otra solución antivirus de terceros, Windows desactiva automáticamente «Defender» para evitar conflictos. Si esa solución caduca o se desinstala, «Defender» suele reactivarse de forma automática. En entornos pymes, si no se dispone de presupuesto para antivirus de pago, Microsoft Defender resulta suficiente y adecuado, siempre que esté actualizado y se combine con buenas prácticas por parte del usuario.

Firewall de Windows

El «Firewall de Windows Defender» viene activado por defecto y es esencial mantenerlo así. Este *firewall* ayuda a proteger el dispositivo filtrando el tráfico de red y bloqueando accesos no autorizados. Reduce el riesgo de amenazas al restringir o permitir el tráfico según reglas basadas en direcciones IP, puertos o aplicaciones.

En la configuración del *firewall*, Windows clasifica las redes en perfiles: «Privada» (redes confiables, como la de casa u oficina) o «Pública» (redes abiertas, por ejemplo, una red wifi de cafetería). Es recomendable marcar como pública cualquier red desconocida para aplicar las restricciones más estrictas.

De forma predeterminada, el *firewall* de Windows bloquea todas las conexiones entrantes no solicitadas (excepto las permitidas por reglas) y permite las conexiones salientes. Esto significa que, por ejemplo, otros equipos no pueden iniciar conexión con nuestro equipo a menos que se haya habilitado explícitamente una regla (como permitir el puerto 3389 para «Escritorio remoto» o abrir el puerto 445 si se comparten archivos). En cambio, las aplicaciones en el equipo sí pueden conectarse hacia afuera libremente (a menos que una regla lo impida).

Es posible crear reglas de entrada o salida según las necesidades. Por ejemplo, se puede abrir el puerto 80/TCP si el equipo actúa como servidor web, o bloquear completamente ciertas aplicaciones para que no tengan acceso a la red. Windows ofrece una interfaz gráfica para reglas avanzadas (en «Firewall de Windows con seguridad avanzada»), donde se pueden definir reglas por puerto, programa, protocolo, alcance de IP, etc.

Sin embargo, para la mayoría de los usuarios básicos, basta con mantener el *firewall* activo en todos los perfiles de red y permitir puntualmente alguna aplicación mediante el cuadro de diálogo del *firewall* cuando Windows lo solicite.

Vale la pena destacar algunas buenas prácticas con el *firewall*:

- Mantenerlo siempre habilitado en todos los perfiles de red («Dominio», «Privada» y «Pública»). Si en algún momento se desactiva (por prueba o por error), Windows mostrará alertas; es importante reactivarlo lo antes posible.
- No desactivar el *firewall*, aunque se cuente con otro *firewall* perimetral (por ejemplo, en el router). El *firewall* local agrega una capa de protección en profundidad.
- Utilizar la opción «Bloquear todas las conexiones entrantes» cuando se requiera máxima seguridad (por ejemplo, en entornos de alta sensibilidad o al conectarse a redes wifi públicas). Esta opción, disponible en la configuración de cada perfil de red, ignora incluso las reglas de aplicaciones permitidas y bloquea todo ingreso. Solo las conexiones salientes funcionarán; sin embargo, esto puede afectar a ciertas aplicaciones, por lo que debe usarse con precaución.
- Permitir aplicaciones a través del *firewall* en lugar de desactivarlo. Si una aplicación legítima es bloqueada (por ejemplo, un programa de escritorio remoto o un juego en red), se puede crear una regla específica para ella o abrir el puerto necesario. Nunca es buena idea deshabilitar completamente el *firewall* como solución, ya que deja el equipo expuesto.

En la aplicación «Seguridad de Windows» se puede ver el estado tanto del antivirus como del *firewall*. La sección «Firewall y protección de red» muestra los perfiles («Dominio», «Privado» y «Público») y permite activarlos o desactivarlos (lo cual requiere permisos de administrador). También ofrece un enlace a la configuración avanzada del *firewall*.

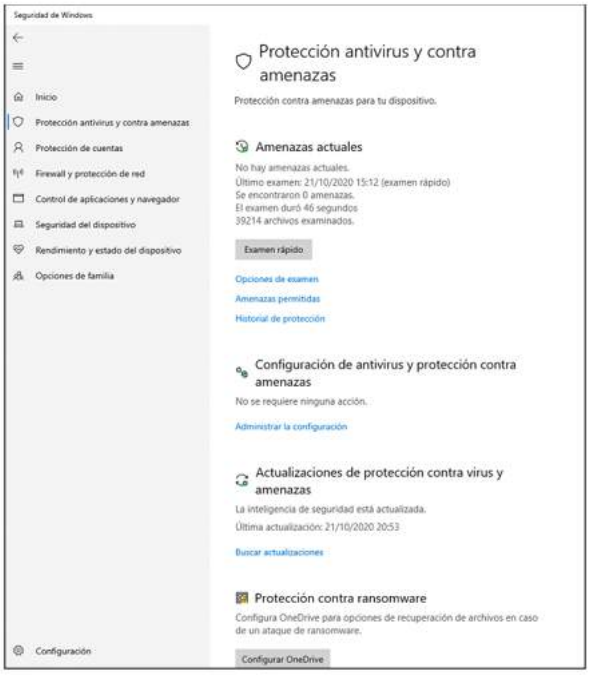
La sección «Protección contra virus y amenazas» muestra el estado del antivirus «Defender», permite ejecutar distintos tipos de examen (rápido, completo o personalizado) y gestionar configuraciones como exclusiones o el historial de amenazas detectadas.

La figura 1 muestra un ejemplo de la interfaz de «Seguridad de Windows», en la sección del antivirus «Defender».

En resumen, para *endpoints* con Windows en pymes, es fundamental mantener habilitados tanto «Windows Defender» como el «Firewall de Windows». Juntos, proporcionan protección antivirus y filtrado de red sin costo adicional. A esto se debe sumar la capacitación de los usuarios (por ejemplo, no ejecutar archivos adjuntos desconocidos o no desactivar el antivirus por cuenta propia).

Con un sistema actualizado (como se verá más adelante) y estas defensas activas, se cubren los controles de seguridad básicos en los equipos con Windows.

Figura 1. Pantalla de «Protección contra virus y amenazas» en la aplicación «Seguridad de Windows» (Windows 11)



Fuente: captura de pantalla de Windows (Microsoft, 2023)

Auditoría y logs

Configurar la auditoría de eventos de seguridad en Windows permite llevar un registro de sucesos importantes (inicios de sesión, accesos a objetos, cambios de configuración, etc.) que sirven para detectar actividades sospechosas o investigar incidentes. Windows, por defecto, ya registra ciertos eventos en el «Visor de eventos» (*Event Viewer*), pero muchos otros requieren habilitar explícitamente la auditoría.

Las principales categorías de auditoría incluyen, entre otras: «Eventos de inicio de sesión/cierre de sesión», «Acceso a objetos» (archivos, carpetas, claves de registro), «Eventos de privilegios especiales» (uso de privilegios

administrativos), «Cambios de políticas» (como GPO o la propia auditoría), «Eventos del servicio de directorio» (si el equipo es un controlador de dominio), etc.

Por ejemplo, para saber si alguien intentó iniciar sesión con una cuenta y falló la contraseña, se debe tener activada la auditoría de inicios de sesión fallidos. Para verificar si se accedió o eliminó un archivo sensible, es necesario habilitar la auditoría de acceso a objetos y configurar la SACL en ese archivo.

En equipos unidos a un dominio, las directivas de auditoría suelen configurarse mediante GPO, en la ruta: «Configuración del equipo» > «Configuración de Windows» > «Configuración de seguridad» > «Directivas locales» > «Directiva de auditoría». En un equipo individual, se puede usar la «Directiva de seguridad local» (`secpol.msc`) en la sección «Directivas locales» > «Directiva de auditoría» para activar las necesarias.

Por ejemplo, se puede activar la opción «Auditar eventos de inicio de sesión» tanto en éxitos como en errores (los éxitos permiten saber quién accede correctamente; los errores, detectar intentos fallidos). También es recomendable auditar «Eventos de inicio de sesión de cuenta» (diferentes de los anteriores, ya que registran eventos en el autenticador local o del dominio), «Acceso a objetos» (requiere especificar luego qué archivos o registros auditar), «Cambios de políticas», «Uso de privilegios especiales», entre otros, según el nivel de detalle que se busque.

> **Tip:** no es conveniente habilitar todas las categorías de auditoría de forma indiscriminada, ya que esto generaría un volumen excesivo de eventos —muchos de ellos irrelevantes— y dificultaría encontrar información útil. Conviene seleccionar aquellas categorías que sean relevantes para la seguridad, como los inicios de sesión, los intentos de elevación de privilegios o los accesos a recursos sensibles.

Todos los eventos auditados se almacenan en los «Registros de Windows», sección «Seguridad», visibles a través de la herramienta «Visor de eventos» (*Event Viewer*). El «Visor de eventos» es una de las herramientas de diagnóstico más importantes de Windows, ya que permite revisar errores, advertencias e información de diversos componentes del sistema. En particular, el registro «Seguridad» lista los eventos de auditoría de seguridad.

Cada evento tiene un ID, una fecha y hora, el usuario implicado, el resultado (éxito o fracaso) y detalles adicionales. Por ejemplo, el evento con ID 4625 indica un intento de inicio de sesión fallido (credenciales incorrectas); el ID 4624, un inicio de sesión exitoso; y los ID 4673/4674, el uso de privilegios asignados, entre otros. Revisar periódicamente estos registros o configurarlos para enviar alertas puede ayudar a detectar situaciones como múltiples intentos fallidos de inicio de sesión (posible ataque de fuerza bruta o usuario intentando adivinar contraseñas), accesos en horarios inusuales o cambios importantes de configuración.

En la siguiente figura se observa un ejemplo del «Visor de eventos» que muestra eventos de seguridad. Se ha filtrado por la categoría «Seguridad» dentro de «Registros de Windows», listando eventos de autenticación auditados. Al seleccionar un evento, en el panel inferior se muestran sus detalles (origen, ID, usuario, etc.).

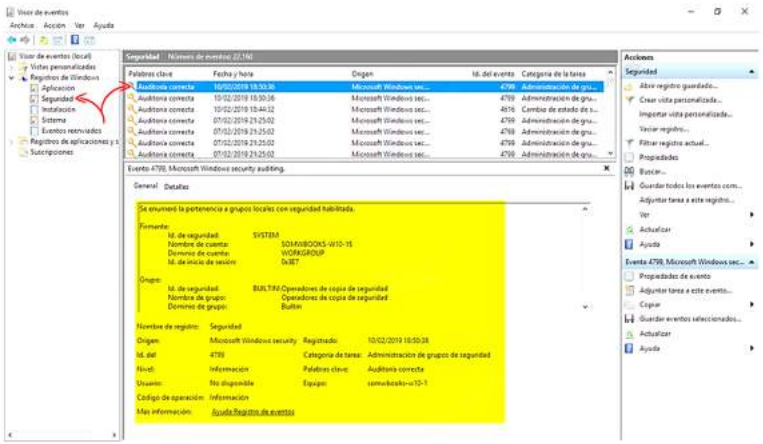
En el caso de una pyme, se recomienda al menos habilitar la auditoría de inicios de sesión (éxito y fracaso), y, eventualmente, la de accesos a objetos críticos (por ejemplo, una carpeta donde se almacenen documentos confidenciales). De este modo, ante un incidente, se podrá revisar el «Visor de eventos» para responder preguntas como las siguientes: ¿Quién inició sesión en este equipo a tal hora?, ¿hubo intentos de intrusión fallidos?, ¿qué usuario accedió o borró este archivo? Incluso sin un sistema SIEM complejo, el registro de eventos de Windows resulta valioso para la investigación forense básica y la detección de comportamientos anómalos.

Windows también registra otros eventos útiles en registros como «Aplicación» (eventos de aplicaciones instaladas), «Sistema» (eventos del sistema operativo y controladores) y «Seguridad». Es conveniente familiarizarse con el «Visor de eventos» y saber cómo ubicar la información. Por ejemplo, si un usuario reporta que «el sistema hizo algo extraño a las 15:00», se pueden revisar los eventos del registro «Sistema» en ese horario (tal vez falló un controlador o se reinició un servicio). En «Seguridad», como ya se mencionó, se pueden consultar intentos fallidos de inicio de sesión, elevaciones a administrador (eventos UAC), entre otros. Los administradores de sistemas deben monitorear regularmente estos registros o, al menos, revisarlos tras cualquier señal de alerta.

Finalmente, cabe mencionar que Windows permite configurar tareas de auditoría avanzada (disponibles desde Windows 7 y Windows Server 2008), que ofrecen mayor granularidad que las categorías básicas. Por ejemplo, en lugar de habilitar la auditoría genérica de «Acceso a objetos», se puede utilizar la política «Auditoría de acceso a archivos» con reglas avanzadas para auditar únicamente los accesos a un archivo específico. Estas configuraciones también se realizan mediante directiva, en la ruta: «Configuración de seguridad» > «Directivas de auditoría avanzadas». Para simplificar, si se está comenzando, el uso de las categorías estándar resulta suficiente.

En resumen, activar la auditoría de eventos de seguridad y revisar los registros debería formar parte de la rutina de administración de *endpoints* con Windows. Esto permite detectar y reaccionar ante actividades anómalas (por ejemplo, un *malware* que crea un usuario —quedaría registrado— o un empleado intentando acceder a recursos sin autorización). Aunque pueda parecer técnico, es una fuente de información invaluable para fortalecer la seguridad.

Figura 2. Eventos de seguridad en el «Visor de eventos» de Windows



Fuente: Ruiz, 2019, https://short.do/JVbRIn

Actualizaciones y Control de Cuentas de Usuario o UAC (User Account Control)

Mantener Windows actualizado es fundamental para la seguridad. Microsoft publica con regularidad parches que corrigen vulnerabilidades en el sistema operativo y otras aplicaciones integradas. No aplicar estas actualizaciones deja al equipo expuesto a amenazas conocidas que podrían ser explotadas fácilmente por atacantes.

Las actualizaciones de Windows proporcionan a usuarios y administradores las correcciones de seguridad necesarias y protegen a los dispositivos para que sus vulnerabilidades no parcheadas no sean aprovechadas. En otras palabras, constituyen una primera línea de defensa frente a *malware* y atacantes: tan pronto se descubre una vulnerabilidad (por ejemplo, en el protocolo RDP, en SMB o en el *kernel*), Microsoft lanza un parche. Si no se instala, el equipo sigue siendo una presa fácil para *exploit kits* y *ransomware* que busquen esa brecha.

Windows cuenta con el servicio «Windows Update», que en equipos modernos está activado por defecto para descargar e instalar actualizaciones automáticamente. Es recomendable dejar «Windows Update» en automático, de modo que el equipo reciba al menos las actualizaciones de seguridad mensuales (conocidas como «Patch Tuesday»), que Microsoft libera el segundo martes de cada mes. Estas actualizaciones acumulativas incluyen todos los parches de seguridad disponibles hasta la fecha.

En entornos corporativos más controlados, a veces se difieren las actualizaciones unos días para realizar pruebas de compatibilidad. Sin embargo, en una pyme pequeña, generalmente conviene instalarlas lo antes posible. Microsoft también publica ocasionalmente parches fuera

de banda (*out-of-band*, OOB) para vulnerabilidades críticas emergentes; «Windows Update» los entregará cuando corresponda.

Es importante verificar que «Windows Update» no esté deshabilitado (en algunos casos, usuarios avanzados lo hacen por molestias con los reinicios). Esto se puede comprobar en «Configuración» > «Windows Update», observando si hay algún mensaje de error o si el sistema lleva mucho tiempo sin buscar actualizaciones. También conviene activar la opción «Recibir actualizaciones para otros productos Microsoft» en «Windows Update» para incluir aplicaciones como Microsoft Office.

Una buena práctica en pymes es programar las ventanas de mantenimiento (reinicios) fuera del horario laboral, de modo que las actualizaciones se apliquen sin interrumpir la productividad. Windows 10 y 11 permiten especificar «horas activas» para evitar reinicios automáticos en ciertos horarios. Aun así, es importante educar a los usuarios para que, cuando Windows solicite reiniciar por actualizaciones, lo hagan al finalizar su jornada.

Junto con las actualizaciones del sistema, también es recomendable mantener actualizadas las aplicaciones de terceros (como navegadores, lectores de PDF, Java, etc.), ya que también pueden contener vulnerabilidades. Herramientas gratuitas como Patch My PC o Ninite permiten automatizar parcialmente la actualización de programas comunes en estaciones de trabajo.

CONTROL DE CUENTAS DE USUARIO (UAC) | **CONFIGURACIÓN DE UAC**

El «Control de cuentas de usuario» (UAC) es una característica de seguridad presente desde Windows Vista, diseñada para prevenir cambios no autorizados en el sistema. Funciona solicitando confirmación —y credenciales de administrador si el usuario actual no lo es— cada vez que una tarea intenta realizar cambios con privilegios elevados en el equipo.

Seguramente todos hemos visto la ventana emergente que pregunta: «¿Desea permitir que esta aplicación haga cambios en su dispositivo?» con los botones «Sí» y «No». Esa es la acción del UAC en funcionamiento. En Windows 10 y 11, por defecto, el UAC está configurado en un nivel intermedio («Notificar solo cuando las aplicaciones intenten hacer cambios»). Esto significa que, si un programa intenta instalarse o modificar archivos del sistema, se mostrará la notificación; pero si es el propio usuario (con rol de administrador) quien cambia una configuración del sistema, no se notifica. Aun así, en ambos casos, los cambios solo se ejecutan con privilegios elevados si se acepta el cuadro de diálogo del UAC.

El UAC aporta una capa de protección importante. Sin él, cualquier *malware* que lograra ejecutarse podría obtener privilegios completos de inmediato. Con el UAC activado, si un *malware* intenta, por ejemplo, copiarse a una carpeta del sistema o instalar un servicio, Windows mostrará la ventana de confirmación, lo que brinda al usuario la oportunidad de sospechar («¿por qué se solicita permiso para esta acción que no reconozco?») y cancelar la operación, evitando así la infección. Incluso si el usuario, por descuido, hace clic en «Sí», el UAC sigue siendo útil porque obliga a que los procesos se inicien con permisos de usuario estándar por defecto, limitando su impacto hasta que se concedan explícitamente privilegios elevados.

Por ello, es altamente desaconsejable deshabilitar el UAC. Ejecutar todo con privilegios de administrador de forma permanente es una práctica riesgosa. La propia Microsoft recomienda mantener el UAC activado en un nivel que logre un equilibrio entre seguridad y usabilidad. El nivel predeterminado suele ser apropiado para la mayoría de los casos, ya que evita notificaciones innecesarias pero alerta en situaciones críticas. Solo en entornos controlados, con fines muy específicos, podría justificarse reducir su nivel (y nunca se recomienda desactivarlo por completo, salvo en servidores *core* administrados completamente mediante scripts).

Como práctica administrativa, conviene que los usuarios habituales no trabajen con una cuenta con permisos de administrador todo el tiempo. Lo ideal es aplicar el principio de «ejecutar como usuario estándar y usar la cuenta de administrador solo para cambios». En muchas pymes, los usuarios se configuran como administradores locales, lo cual, al menos, combinado con el UAC, obliga a confirmar los cuadros de diálogo. Si es posible, se debería evaluar la configuración de cuentas de usuario estándar en los equipos y mantener una cuenta de administrador separada; de este modo, cuando una acción requiera elevación, será necesario ingresar la contraseña de administrador. Esto agrega una capa adicional de seguridad, ya que el usuario deberá contactarse con el administrador o, al menos, reflexionar antes de instalar algo. Si por motivos prácticos esto no fuera viable, al menos se debe confiar en el UAC como medida de mitigación.

CONTROL DE CUENTAS DE USUARIO (UAC) | **CONFIGURACIÓN DE UAC**

A través de «Panel de control» > «Cuentas de usuario» > «Cambiar configuración de Control de cuentas de usuario», se accede a un control deslizante con cuatro niveles:

- **Nivel más alto – «Notificar siempre»**. Windows solicita confirmación ante cualquier cambio significativo, sin excepciones (incluso cambios de configuración iniciados por el propio

usuario). Es el nivel más seguro, pero puede resultar molesto, ya que incluso al realizar ajustes conscientes del sistema, se mostrará la ventana de confirmación.

- **Nivel predeterminado – «Notificar solo cuando las aplicaciones intenten hacer cambios en mi equipo»**: Windows no notifica si el propio usuario cambia una configuración del sistema (por ejemplo, desde el «Panel de control»), pero sí lo hace cuando un programa externo intenta realizar cambios o instalaciones. Además, durante la notificación, Windows atenúa el escritorio (lo hace inaccesible) para evitar interferencias de *malware*. Esta función se denomina «Escritorio seguro» (Secure Desktop). Este es el nivel de equilibrio predeterminado en Windows 10 y 11.
- **Nivel intermedio – «Notificar… (no atenuar mi escritorio)»**: Igual al anterior en cuanto al tipo de acciones que generan notificación, pero sin bloquear el escritorio mientras la alerta está visible. Esto reduce la molestia visual, aunque también disminuye levemente la seguridad, ya que un *malware* sofisticado podría, en teoría, interactuar con la ventana del UAC en segundo plano.
- **Nivel más bajo – «Nunca notificar»**: desactiva el UAC por completo. No se recomienda en absoluto. En este modo, cualquier programa obtiene privilegios de administrador sin requerir autorización, lo que deja el sistema extremadamente vulnerable a modificaciones no autorizadas. Equivale a trabajar siempre con permisos de administrador, como ocurría en Windows XP, con todos los riesgos que eso implica.

Para la gran mayoría de los usuarios, mantener el UAC en el nivel predeterminado (tercer nivel) es lo más aconsejable. Solo administradores de TI que realicen múltiples cambios diarios en el sistema —y tengan pleno conocimiento de lo que hacen— podrían considerar reducir un nivel la atenuación del escritorio, pero nunca desactivarlo. El UAC ha demostrado ser eficaz para limitar el impacto de muchos *malware* de generaciones anteriores que asumían que el usuario contaba con derechos administrativos totales. Además, todos los instaladores legítimos están preparados para funcionar con UAC (incluyen la marca para solicitar elevación); si una aplicación requiere desactivarlo para instalarse, conviene sospechar de ella.

En resumen, esta unidad dedicada a Windows ha abordado las políticas básicas mediante GPO o directivas locales, las protecciones integradas de antivirus y *firewall*, la auditoría de eventos de seguridad, y la importancia de las actualizaciones y del UAC. Con estos elementos implementados, un equipo con Windows estará razonablemente protegido frente a amenazas comunes.

Existen otras medidas adicionales (como el cifrado de disco con BitLocker, la restricción de software mediante AppLocker o la «Política de restricción de software», y el control de

dispositivos) que forman parte de un *hardening* más avanzado. Sin embargo, con lo tratado hasta aquí, se establecen bases sólidas de seguridad. En la siguiente unidad, aplicaremos una filosofía similar de *hardening* básico a sistemas Linux.

CONTINUAR

Unidad 2. Linux

En esta unidad nos enfocaremos en la seguridad básica de *endpoints* o servidores con Linux. Veremos cómo utilizar el *firewall* integrado (ya sea mediante «UFW» o directamente con iptables), aprenderemos a gestionar servicios y demonios con systemd, revisaremos la configuración adecuada de permisos de archivos y el archivo sudoers para el control de privilegios, y, finalmente, abordaremos la aplicación de parches y actualizaciones del *kernel* en sistemas Linux. Al igual que en Windows, estas medidas buscan fortalecer los sistemas Linux con herramientas gratuitas o incluidas por defecto, adecuadas para pymes.

UFW (UNCOMPLICATED FIREWALL) / IPTABLES | **EJEMPLO PRÁCTICO** | **SERVICIOS Y SYSTEMD**

Linux cuenta con un potente cortafuegos a nivel de núcleo llamado *netfilter/iptables*. iptables es la interfaz de línea de comandos tradicional para configurar las reglas de filtrado de paquetes del *kernel* de Linux. Sin embargo, usar iptables de forma directa puede resultar complejo para usuarios sin experiencia.

Aquí es donde entra en juego UFW (Uncomplicated Firewall), que, como su nombre indica, busca simplificar la gestión del *firewall*. UFW es, en esencia, una interfaz de alto nivel para iptables que ofrece una sintaxis más sencilla para definir reglas. En distribuciones basadas en Debian o Ubuntu, suele estar instalado por defecto (aunque inicialmente deshabilitado), con el fin de facilitar a administradores menos experimentados la configuración de un *firewall* local.

¿Qué hace un *firewall* en Linux?

Al igual que en Windows, permite controlar el tráfico entrante y saliente según una política definida. Es posible especificar qué puertos o servicios estarán permitidos (por ejemplo, permitir «SSH» en el puerto 22, «HTTP» en el 80, etc.) y cuáles serán denegados. Esto resulta fundamental en servidores o equipos que tengan servicios de red expuestos.

La seguridad básica establece que se debe denegar todo lo que no sea necesario y permitir solo lo mínimo indispensable (principio de mínimo privilegio, aplicado a puertos)

UFW suele tener una configuración inicial bastante segura: por defecto, bloquea todas las conexiones entrantes y permite todas las salientes, aplicando esta política tanto para IPv4 como para IPv6. Esto significa que, una vez habilitado, el equipo no aceptará ninguna conexión desde el exterior (a menos que sea una respuesta a una solicitud iniciada por él), pero podrá seguir navegando y comunicándose hacia afuera. Es un buen punto de partida, ya que cierra cualquier puerto que pudiera estar abierto inadvertidamente.

Para utilizar UFW, primero debemos habilitarlo. En Ubuntu, el comando es `sudo ufw enable`. Si «UFW» no estuviera instalado, se puede instalar con el comando `sudo apt install ufw` en sistemas Debian o Ubuntu.

Antes de habilitarlo, es importante permitir el puerto `22` si se administra el sistema de forma remota mediante «SSH», ya que, de lo contrario, se podría bloquear el propio acceso. Por ejemplo, el siguiente comando añade una regla para permitir «SSH»: `sudo ufw allow 22/tcp`. De hecho, UFW cuenta con perfiles predefinidos para servicios comunes. Así, el comando `sudo ufw allow ssh` realiza la misma acción que el anterior, ya que reconoce «ssh» como el puerto `22/tcp`.

Una vez habilitado, se puede comprobar el estado de «UFW» con el siguiente comando: `sudo ufw status`. Si está activo, mostrará `Status: active` y listará las reglas cargadas. Al principio, puede que no haya reglas explícitas, aparte de las políticas por defecto. A partir de allí, es posible añadir reglas fácilmente con comandos como:

- `sudo ufw allow <puerto>/<protocolo>`
- `sudo ufw deny <puerto>`

Estos permiten autorizar o denegar puertos específicos. También se pueden especificar rangos de direcciones IP de origen o interfaces de red.
Veamos algunos ejemplos:

- `sudo ufw allow 80/tcp`. Habilita el acceso entrante al puerto `80` (HTTP) desde cualquier origen. Tiene sentido en un servidor web.
- `sudo ufw deny 23`. Deniega el tráfico en el puerto `23` (Telnet), por razones de seguridad. Telnet no cifra los datos y no debería utilizarse.
- `sudo ufw allow from 192.168.1.100 to any port 3306`. Permite que solo la dirección IP `192.168.1.100` acceda al puerto `3306` (MySQL), bloqueando otros orígenes.

Para obtener una vista más detallada del estado del *firewall*, se puede usar `sudo ufw status verbose`. Este comando muestra información adicional sobre las reglas activas y las políticas por defecto, como: `Default: deny (incoming), allow (outgoing), deny (routed)`.

Cabe mencionar que UFW también cuenta con una interfaz gráfica opcional llamada «GUFW», disponible para entornos de escritorio. Es útil para quienes prefieren no usar la terminal. Sin embargo, en servidores —donde no suele haber entorno gráfico— se utiliza habitualmente desde la línea de comandos.

Si bien «UFW» cubre las necesidades típicas, es útil comprender que, en su funcionamiento subyacente, utiliza `iptables`. En algunos casos avanzados, puede que se necesite una regla que no esté contemplada directamente por la sintaxis de «UFW». Para estos casos, se puede recurrir a `iptables` de forma manual o utilizar las denominadas *reglas raw* en UFW. Sin embargo, para un proceso de *hardening* básico, «UFW» resulta suficiente.

En aquellas distribuciones que no incluyen UFW (por ejemplo, CentOS o Fedora, que utilizan «firewalld» y `nftables`), el principio sigue siendo el mismo: asegurarse de que el *firewall* esté activo y correctamente configurado. En RHEL o CentOS, por ejemplo, «firewalld» cumple una función similar (y por debajo emplea `nftables`). No entraremos en detalle sobre cada sistema, pero la recomendación general para Linux es: utilizar el *firewall* propio de la distribución y cerrar todos los puertos no utilizados.

A menudo, los servicios instalados agregan sus propias reglas automáticamente si se emplea el *firewall* nativo. Por ejemplo, en «firewalld» existen servicios predefinidos como «http» o «ssh», similares a los perfiles de UFW.

Para quienes administran routers con Linux o entornos complejos, aprender `iptables` o `nftables` directamente es una inversión valiosa. Sin embargo, en escenarios de pymes con servidores sencillos, «UFW» ofrece un equilibrio ideal entre simplicidad y seguridad.

Conviene recordar que `iptables` y `nftables` permiten mucho más que bloquear puertos: pueden realizar tareas de NAT (como enmascaramiento o redirección de puertos), limitar tasas de conexión (*rate limiting*) para mitigar ataques de fuerza bruta, entre otras funciones. UFW admite parcialmente algunas de estas capacidades. Por ejemplo, el comando `sudo ufw limit ssh/tcp` permite aplicar una limitación de conexiones al servicio «SSH» para mitigar intentos de fuerza bruta, generando internamente reglas de *rate limiting* en `iptables`.

En resumen, se debe activar y configurar el *firewall* en Linux. En sistemas Debian o Ubuntu, UFW permite facilitar esta tarea: con unos pocos comandos es posible bloquear todas las

conexiones no esenciales. Se deben añadir excepciones únicamente para los servicios que realmente se estén ofreciendo (por ejemplo, «SSH», web, etc.). Luego, conviene verificar qué reglas están aplicadas mediante `sudo ufw status` o, en su defecto, `sudo iptables -L`. También es recomendable realizar pruebas desde otro equipo: intentar conectarse a un puerto no permitido (la conexión debería fallar) y a uno permitido (debería funcionar). De este modo, se confirma que el *firewall* cumple su función.

Supongamos un servidor Linux recién instalado que alojará una página web y será administrado mediante «SSH». En ese caso, luego de la instalación, podrían ejecutarse los siguientes comandos:

- `sudo ufw allow 22/tcp`
- `sudo ufw allow 80/tcp`
- `sudo ufw allow 443/tcp`
- `sudo ufw enable`

Esto dejará abiertos los puertos 22, 80 y 443 a todos los orígenes (aunque podrían restringirse por IP si se desea). Todos los demás puertos (como 3306 de MySQL, 3389, etc.) permanecerán bloqueados. El sistema podrá seguir accediendo a Internet, realizar actualizaciones, resolver nombres de dominio, etc., ya que las salidas están permitidas por defecto.

Nota: es importante revisar los registros del *firewall*. UFW guarda un registro en `/var/log/ufw.log` (si se habilita con `sudo ufw logging on`). Allí se puede observar qué está siendo bloqueado, lo que permite diagnosticar posibles reglas demasiado restrictivas o detectar intentos no deseados (como múltiples accesos al puerto 22 desde direcciones desconocidas, indicio de ataques de fuerza bruta, los cuales quedarán registrados como bloqueados en el *log*).

En Linux, muchos procesos de fondo (*daemons*) se ejecutan como servicios del sistema. Por ejemplo, el servidor web «Apache», el servicio «SSH» o el motor de base de datos son servicios que normalmente se inician automáticamente al arrancar el sistema y permanecen en ejecución. La gestión de estos servicios, en la mayoría de las distribuciones modernas, está a cargo de `systemd`, el sistema de inicio (*init*) y gestor de servicios predeterminado en Linux. Ha sido adoptado por distribuciones como Ubuntu, Debian, RHEL, CentOS, Fedora, SUSE, entre otras.

Un servicio (o *daemon*) es simplemente un programa que se ejecuta en segundo plano, típicamente para brindar alguna funcionalidad continua, como esperar conexiones o ejecutar tareas programadas. Por razones de seguridad, es fundamental mantener activos solo los servicios necesarios y deshabilitar o eliminar los innecesarios, ya que cada servicio representa una posible puerta de entrada o, al menos, una superficie de ataque. Por ejemplo, un servicio de base de datos que escuche conexiones en red sin estar en uso podría ser explotado.

`systemd` es un administrador del sistema y de servicios que reemplazó a los antiguos sistemas *init* de SysV en la mayoría de las distribuciones Linux. Coordina el arranque del sistema y el lanzamiento de todos los servicios, permitiendo definir dependencias, realizar un inicio paralelo más eficiente y supervisar los servicios para reiniciarlos en caso de fallos, entre muchas otras funcionalidades.

`systemd` es compatible con scripts de inicio SysV por motivos de retrocompatibilidad y aporta características como el inicio paralelo, la activación bajo demanda de demonios y el control de servicios basado en dependencias. En esencia, unifica la gestión de *daemons*: todos los servicios se definen mediante unidades (*units*), generalmente archivos con extensión `.service`, y se administran a través del comando central `systemctl`.

A continuación, se describen algunas acciones comunes con `systemctl`, que deben ejecutarse con privilegios de administrador o mediante `sudo`:

- **Listar servicios**: `systemctl list-units --type service --all`. Muestra todos los servicios y su estado actual (activos, inactivos, fallidos, etc.).
- **Ver el estado de un servicio**: `systemctl status nombre-servicio`. Por ejemplo, `systemctl status ssh` indica si el servicio «SSH» está en ejecución, desde cuándo, su PID y las últimas entradas de registro asociadas.
- **Iniciar o detener un servicio**:
  - `systemctl start nombre-servicio`

- systemctl stop nombre-servicio

Estas acciones afectan solo a la sesión actual y no persisten tras un reinicio del sistema.

- **Habilitar o deshabilitar un servicio**:
  - systemctl enable nombre-servicio
  - systemctl disable nombre-servicio

La opción enable configura el servicio para que se inicie automáticamente al arrancar el sistema, mientras que disable elimina ese inicio automático. Cabe aclarar que deshabilitar un servicio no lo detiene si ya se encuentra en ejecución, sino que evita que se inicie en el próximo arranque.

Por ejemplo, en una máquina Linux estándar pueden estar activos servicios como «cron» (planificación de tareas), «cups» (impresión), «apache2» o «nginx» (servicios web), «mysql» (base de datos), entre otros. Si se sabe que en un *endpoint* no se utilizará determinado componente, resulta conveniente deshabilitar el servicio correspondiente.

Un caso habitual es «cups»: en muchas instalaciones por defecto, el demonio de impresión se encuentra en ejecución incluso cuando el equipo no utilizará impresoras. Deshabilitarlo reduce el consumo de recursos y elimina una posible superficie de ataque. Lo mismo ocurre con «bluetooth» en un servidor sin adaptador Bluetooth, o con «rpcbind» si no se utilizan servicios RPC o NFS.

systemd facilita notablemente esta gestión. Además de iniciar o detener servicios, permite bloquearlos por completo mediante el comando systemctl mask nombre-servicio. En este caso, el servicio queda completamente inutilizado, ya que no podrá iniciarse ni siquiera de forma manual, salvo que se revierta la acción. Por ejemplo, se podría aplicar systemctl mask telnet.socket para evitar que el servicio de Telnet sea activado accidentalmente o de forma indebida.

Revisión de servicios habilitados

Una buena práctica de *hardening* consiste en revisar qué servicios se inician automáticamente al arrancar el sistema. Para ello, se puede utilizar el siguiente comando systemctl list-unit-files --state=enabled --type=service. Este comando lista los servicios que se encuentran en estado *enabled*. Conviene inspeccionar esa lista de forma crítica y plantearse la pregunta: «¿realmente

es necesario este servicio en este equipo?». Si la respuesta es negativa, se debería considerar su deshabilitación. Cada servicio innecesario que se apaga representa una posible superficie de ataque menos.

En entornos de escritorio, a menudo hay numerosos servicios habilitados por comodidad o funcionalidad adicional (por ejemplo, «avahi» para descubrimiento de red, «cups» para impresión o «NetworkManager-wait-online»). Dependiendo del caso de uso concreto, algunos de estos servicios pueden desactivarse sin afectar el funcionamiento esperado del sistema.

Journal y *logs*

Cabe mencionar que systemd incluye el servicio journald, encargado de recopilar los *logs* de los servicios del sistema. El comando journalctl -u nombre-servicio permite visualizar los *logs* de un servicio específico, lo cual resulta útil para detectar errores o comportamientos anómalos en un *daemon*. Por ejemplo, si un servicio se detiene repetidamente debido a un fallo, en el *journal* quedarán registradas las causas.

Desde el punto de vista de la seguridad, la revisión de los *logs* de los servicios es fundamental. Esto incluye, entre otros, los intentos de autenticación en «SSH» o las peticiones recibidas por un servidor web. Aunque este tema excede en parte el alcance de este módulo, es importante conocerlo: gracias a systemd y journalctl, es posible acceder de forma centralizada a los *logs* del sistema, sin necesidad de buscar archivos de texto dispersos.

En resumen, systemd nos brinda un control centralizado y completo sobre qué procesos se ejecutan en un sistema Linux. Desde una perspectiva de seguridad, debe utilizarse para minimizar la exposición: mantener activos únicamente los servicios necesarios y asegurarse de que estén correctamente configurados (por ejemplo, «SSH» con opciones seguras o servicios internos escuchando solo en localhost). Un sistema Linux endurecido a nivel básico es aquel en el que, al ejecutar systemctl status, se observa que solo está en funcionamiento lo estrictamente esencial.

Por ejemplo, en un servidor web dedicado, los servicios esenciales suelen ser «nginx» o «apache», «ssh», «cron» y journald, y poco más. Servicios como «ftp» o «telnet» no deberían ni siquiera instalarse; para transferencias de archivos es preferible utilizar «sftp» sobre «SSH». En un equipo de usuario, los servicios esenciales incluyen el entorno gráfico, «NetworkManager» para la conectividad de red y, según el uso, «bluetooth» o «cups». Si no se utilizan, conviene

deshabilitarlos. Menos servicios implican menos puertos abiertos y, por lo tanto, menos posibles vulnerabilidades.

Finalmente, cabe mencionar que `systemd` abarca muchas más funcionalidades (temporizadores, tareas programadas, montaje de discos, entre otras). Sin embargo, desde el punto de vista de la seguridad, el foco principal es conocer y controlar los servicios (*daemons*) en ejecución. En administración de sistemas suele decirse que «un servicio que no está corriendo no puede ser atacado». Aplicar este principio resulta clave para fortalecer la seguridad de los *endpoints* Linux.

Permisos y sudoers —

El modelo de permisos de archivos en Linux (y en sistemas UNIX en general) es un pilar fundamental de la seguridad. Cada archivo y directorio tiene un propietario y permisos asociados que determinan quién puede leerlos, escribirlos o ejecutarlos.

En Linux, cada objeto tiene tres niveles de asignación de permisos: el **usuario propietario** (*owner*), el **grupo propietario** (*group*) y **otros** (el resto de los usuarios del sistema). Para cada uno de estos niveles se definen tres permisos básicos: lectura (r), escritura (w) y ejecución (x).

Por ejemplo, consideremos un archivo con permisos `rwxr-x---`. En términos de niveles, esto significa lo siguiente:

- **Usuario propietario**: `rwx` (puede leer, escribir y ejecutar el archivo).
- **Grupo propietario**: `r-x` (los usuarios pertenecientes al grupo asignado pueden leer y ejecutar el archivo, pero no modificarlo).
- **Otros**: `---` (ningún otro usuario del sistema tiene permisos sobre el archivo).

Este esquema se observa habitualmente mediante el comando `ls -l` que muestra líneas del siguiente tipo:

`-rw-r----- 1 alice contabilidad 524288 sep 1 12:00 balance.csv`
En este caso, `-rw-r-----` indica los permisos del archivo: `rw-` para «alice» (usuario propietario), `r--` para el grupo «contabilidad» y `---` para el resto de los usuarios. Esto implica que «alice» puede leer y modificar el archivo; los miembros del grupo «contabilidad» pueden leerlo, pero no alterarlo; y ningún otro usuario del sistema puede acceder a él.

Gestionar correctamente los permisos implica aplicar el **principio de mínimo privilegio**: cada usuario o proceso debe tener acceso únicamente a los recursos que necesita. Por ejemplo, archivos de configuración sensibles,

como /etc/shadow (que almacena los *hashes* de contraseñas), cuentan con permisos muy restrictivos (-rw-------), lo que los hace accesibles solo para el usuario «root».

En cambio, los comandos de uso general ubicados en /usr/bin suelen ser ejecutables por todos los usuarios (-rwxr-xr-x), de modo que cualquiera pueda utilizarlos, pero sin posibilidad de modificarlos.
Algunos aspectos importantes a recordar sobre los permisos:

- **Permiso de ejecución en directorios**. El permiso de ejecución (x) es necesario para poder acceder a un directorio (por ejemplo, mediante cd). Para listar su contenido, normalmente también se requiere el permiso de lectura (r).
- **Permisos especiales**: Existen bits especiales como *setuid*, *setgid* y *sticky bit*.
    - El bit *setuid* en un binario indica que, al ejecutarse, el proceso adquiere los permisos del propietario del archivo. Muchos comandos del sistema utilizan *setuid* con propietario «root» para realizar tareas privilegiadas; un ejemplo típico es /usr/bin/passwd.
    - El *sticky bit* aplicado a un directorio (como /tmp) garantiza que, aunque todos los usuarios tengan permiso de escritura, solo el propietario de un archivo pueda eliminarlo.

Es importante conocer la existencia de estos permisos especiales y utilizarlos con cuidado, ya que una configuración incorrecta puede introducir vulnerabilidades. Por ejemplo, un binario con *setuid root* mal protegido podría ser explotado para realizar una escalada de privilegios.

Como administradores, debemos asegurarnos de que los archivos y directorios importantes tengan los permisos adecuados. Algunas configuraciones incorrectas que conviene evitar incluyen, por ejemplo, dejar archivos confidenciales con permisos rw-rw-rw- (cualquier usuario podría leerlos o modificarlos) o mantener scripts ejecutables con propietario «root» que sean escribibles por otros usuarios. En este último caso, un usuario podría modificar el script e insertar comandos maliciosos que luego serían ejecutados con privilegios elevados.

En general, la revisión periódica de permisos en directorios como /etc, así como en archivos de aplicaciones y configuraciones, forma parte de las tareas de *hardening* del sistema. Afortunadamente, la mayoría de las distribuciones Linux incluyen permisos seguros por defecto en las ubicaciones sensibles. El mayor riesgo suele aparecer cuando se crean archivos o se instalan componentes de forma manual sin ajustar correctamente la propiedad y los permisos.

Por ejemplo, si se monta un directorio compartido y se deja con propietarios o permisos inapropiados, otros usuarios del sistema podrían acceder a su contenido o modificarlo sin autorización. Conviene plantearse siempre la pregunta: «¿quién necesita acceder a este recurso?» y asignar la propiedad y los permisos en consecuencia. El

uso de grupos de UNIX resulta especialmente útil para gestionar accesos compartidos; por ejemplo, crear un grupo «ventas» con acceso a determinados archivos, en lugar de conceder permisos amplios a «otros».

En sistemas Linux de escritorio, suele existir un único usuario principal con privilegios administrativos, además de «root», mientras que en servidores pueden coexistir varios usuarios legítimos. En ambos escenarios, es recomendable evitar, en la medida de lo posible, el uso directo de la cuenta «root» en sesiones interactivas, con el fin de reducir errores graves y minimizar vectores de ataque. En este contexto cobra relevancia el uso del sistema `sudo`, que permite una gestión más segura de los privilegios.

sudo y archivo sudoers —

La utilidad `sudo` permite a un usuario ejecutar comandos con privilegios de superusuario («root») de forma controlada, generalmente solicitando su propia contraseña. Qué usuarios pueden utilizar `sudo` y para ejecutar qué comandos se define en el archivo `/etc/sudoers`. Este archivo establece las políticas de uso de `sudo`. Mediante su configuración, es posible otorgar a determinados usuarios o grupos privilegios administrativos completos o limitarlos a la ejecución de comandos específicos.

En distribuciones como Ubuntu, el primer usuario creado durante la instalación suele pertenecer al grupo «sudo». Gracias a una entrada en el archivo `sudoers`, como `%sudo ALL=(ALL:ALL) ALL` se permite que los miembros de ese grupo utilicen `sudo` para ejecutar cualquier comando con privilegios elevados. Esto equivale, en la práctica, a contar con permisos administrativos completos, con la diferencia de que el usuario debe anteponer `sudo` al comando y, por defecto, ingresar su contraseña para confirmar la acción.

Este mecanismo aporta varias ventajas desde el punto de vista de la seguridad: por un lado, ofrece trazabilidad, ya que cada uso de `sudo` queda registrado en los *logs* de autenticación (por ejemplo, en `/var/log/auth.log`); por otro, introduce una pausa consciente antes de ejecutar acciones críticas, lo que ayuda a reducir errores accidentales.

El archivo `sudoers` debe editarse con extrema precaución. Como recomendación general, siempre se debe utilizar el comando `visudo`. Esta herramienta verifica la sintaxis antes de guardar los cambios y evita que varias personas editen el archivo de forma simultánea. Un error de sintaxis en `sudoers` puede dejar el sistema sin acceso a `sudo` y, si no se dispone de acceso directo como «root», provocar una situación crítica de administración.

La sintaxis típica de `sudoers` permite definir qué usuario o grupo puede ejecutar determinados comandos, en qué hosts (en entornos simples, normalmente «ALL»), como qué usuario (por ejemplo, «root») y qué comandos específicos están autorizados.

Un ejemplo sencillo de una entrada en `sudoers` es el siguiente:`alice ALL=(root) /usr/bin/apt, /usr/bin/systemctl`.

Esta regla permite que la usuaria «alice» ejecute únicamente los comandos /usr/bin/apt y /usr/bin/systemctl con privilegios de «root». Al utilizarlos mediante sudo, se le solicitará su contraseña. Cualquier otro comando será denegado al intentar ejecutarse con sudo.

Este tipo de configuración restrictiva resulta muy útil cuando se desea delegar tareas administrativas concretas a usuarios que no son «root», sin otorgarles acceso administrativo completo. Por ejemplo, se podrían conceder permisos para ejecutar systemctl restart apache2 al equipo de desarrolladores web, sin darles privilegios totales sobre el sistema.

En el archivo sudoers también es posible definir *aliases* para agrupar comandos, hosts o usuarios, lo que facilita la lectura y el mantenimiento de políticas más complejas. En pymes pequeñas, suele ser suficiente contar con uno o dos usuarios administradores con permisos completos (ALL=(ALL) ALL) y mantener al resto de los usuarios sin acceso a sudo, o con permisos puntuales si el caso lo requiere.

Lo importante es no utilizar la cuenta «root» de forma directa, sino operar siempre a través de sudo, y auditar periódicamente su uso. Los *logs* ubicados en /var/log/auth.log registran cada invocación de sudo, incluyendo la marca temporal y el usuario que ejecutó el comando.

Volviendo al modelo de permisos de archivos, la combinación de una gestión adecuada de permisos junto con el uso correcto de sudo proporciona una arquitectura de privilegios robusta. Los usuarios normales no pueden modificar archivos del sistema; solo pueden hacerlo mediante sudo, y únicamente si se les ha concedido permiso. Si un atacante logra comprometer una cuenta sin privilegios administrativos, tendrá mayores dificultades para escalar a «root» si dicha cuenta no figura en sudoers o solo dispone de permisos muy limitados.

Por este motivo, conviene que únicamente los usuarios de confianza pertenezcan al grupo «sudo». En Ubuntu, la membresía a este grupo define quién puede utilizar sudo. Es recomendable revisar periódicamente esta configuración con el comando grep '^sudo' /etc/group y eliminar cualquier usuario que no deba estar incluido. De igual manera, otros grupos especiales (como «docker» o «lxd») pueden otorgar cierto nivel de privilegio, por lo que también conviene revisarlos como parte del proceso de *hardening*.

En cuanto a los permisos especiales mencionados, algunos binarios necesitan tener *setuid root*. Por ejemplo, «ping» solía utilizarlo para poder abrir *raw sockets*, aunque en sistemas actuales esto se maneja mediante *capabilities*. Sin embargo, no se deben crear ni dejar binarios de terceros con *setuid root*, ya que, si contienen fallos, un usuario sin privilegios podría explotarlos para elevar permisos. En sistemas modernos, normalmente solo los binarios estrictamente necesarios cuentan con *setuid*. Como medida adicional, es posible quitar el bit *setuid* de binarios que no lo requieran, por ejemplo mediante chmod u-s, aunque esto ya forma parte de un *hardening* más avanzado.

En resumen, una gestión adecuada de permisos y del uso de sudo es fundamental para la seguridad en sistemas Linux.

En primer lugar, es necesario comprender y aplicar correctamente los permisos en archivos y directorios. Se deben usar propietarios y grupos para segmentar accesos y evitar, salvo casos muy puntuales y temporales, otorgar permisos excesivos como 777 (lectura, escritura y ejecución para todos). Los archivos críticos del sistema deben ser editables únicamente por «root», algo que, en general, ya viene correctamente configurado por defecto.

En servidores multiusuario, conviene revisar que los directorios personales de los usuarios no permitan accesos indebidos por parte de otros. Para ello, es importante contar con una *umask* adecuada: valores como 022 suelen ser correctos para directorios personales, mientras que archivos privados pueden requerir permisos más restrictivos, como 600.

Para las tareas administrativas, se debe utilizar sudo en lugar de trabajar directamente como «root». También aquí se aplica el principio de mínimo privilegio: si un usuario solo necesita reiniciar un servicio como «Apache», no debería concedérsele acceso completo a sudo, sino únicamente el permiso específico para esa acción. El archivo sudoers debe protegerse especialmente, editarse siempre mediante visudo y documentar los cambios usando comentarios, indicando quién recibe cada permiso y con qué finalidad.

En entornos Linux de pymes, donde a menudo existe una única cuenta de usuario en un equipo de escritorio, algunas de estas consideraciones pueden parecer menos relevantes. Sin embargo, incluso en esos casos, la cuenta principal debe usarse con criterio: no realizar todas las tareas como «root». Cuando sea necesario instalar *software* o modificar configuraciones del sistema, se debe utilizar sudo e ingresar la contraseña correspondiente. De este modo, el sistema introduce una pausa de seguridad ante acciones críticas y deja registro de lo realizado.

Parches y kernel —

Finalmente, abordamos la gestión de parches en Linux y las actualizaciones del *kernel*. Al igual que en Windows, los sistemas Linux reciben actualizaciones de seguridad frecuentes para corregir vulnerabilidades tanto en el *kernel* (núcleo del sistema operativo) como en los paquetes de espacio de usuario (servicios, bibliotecas y aplicaciones). Mantener los sistemas Linux correctamente parcheados es un requisito básico de seguridad.

La administración de parches en Linux puede presentar ciertos desafíos (variedad de distribuciones, dependencias, etc.), pero las buenas prácticas recomiendan establecer una rutina de actualización constante y predecible.

En distribuciones como Debian o Ubuntu, esta tarea se simplifica mediante las herramientas de gestión de paquetes APT. Por ejemplo, es suficiente ejecutar periódicamente:

- `sudo apt update` (actualiza la lista de paquetes disponibles)
- `sudo apt upgrade` (instala las nuevas versiones de paquetes disponibles)

Idealmente, este proceso debería realizarse al menos una vez al mes, o de forma inmediata cuando se anuncie una vulnerabilidad crítica que afecte al sistema. De hecho, se recomienda aplicar los parches de seguridad críticos dentro de las primeras 48 horas desde su publicación, con el objetivo de minimizar la ventana de exposición al riesgo.

Muchas distribuciones ofrecen mecanismos de actualización automática. En Ubuntu, por ejemplo, se puede habilitar el paquete `unattended-upgrades`, que instala automáticamente las actualizaciones de seguridad sin intervención del usuario. En servidores, suele configurarse para aplicar únicamente parches de seguridad y evitar actualizaciones mayores que puedan afectar el funcionamiento de los servicios. En entornos de escritorio, herramientas gráficas como «GNOME Software» o «Discover» en KDE permiten notificar y aplicar actualizaciones de forma sencilla, facilitando que los sistemas se mantengan al día sin necesidad de usar la línea de comandos.

Actualización del *kernel*

Aquí hay un aspecto a tener en cuenta: cuando se actualiza el *kernel* de Linux, normalmente es necesario reiniciar el sistema para que la nueva versión entre en funcionamiento, ya que el *kernel* cargado en memoria es el que se encuentra en ejecución. Esto puede llevar a que algunos administradores retrasen la instalación de nuevos *kernels* en servidores que requieren alta disponibilidad.

Sin embargo, en la actualidad existen soluciones de *live patching* (parcheo en vivo), como «Livepatch» de Canonical —gratuito para hasta tres máquinas Ubuntu LTS— y otras alternativas disponibles en distintas distribuciones, como Ksplice o kpatch. Estas tecnologías permiten aplicar parches de seguridad al *kernel* sin necesidad de reiniciar el sistema. Aunque resultan muy útiles, en muchos casos exceden el alcance de una pyme promedio, salvo que se trate de servidores especialmente críticos. Para la mayoría de los entornos, suele ser suficiente programar un reinicio fuera del horario laboral después de actualizar el *kernel*.

Asimismo, es fundamental asegurarse de que la distribución utilizada cuente con soporte vigente. El uso de distribuciones o versiones fuera de soporte es riesgoso, ya que dejan de recibir parches de seguridad. Por ejemplo, Debian 9 o Ubuntu 16.04 ya no cuentan con soporte estándar y, salvo que se contrate un servicio de mantenimiento extendido, quedan expuestas a vulnerabilidades conocidas. Por este motivo, conviene planificar con anticipación las actualizaciones mayores del sistema antes de que finalice el período de soporte.

¿Qué parches aplicar primero?

Se deben priorizar los parches marcados como de seguridad, algo que muchas herramientas de gestión de paquetes, como APT, indican claramente. En sistemas de la familia Red Hat, las actualizaciones se clasifican además por nivel de severidad (importante, crítico, etc.). Un enfoque razonable es instalar los parches de seguridad y las correcciones de estabilidad. Las actualizaciones mayores de versión pueden evaluarse caso por caso, pero ignorar parches de seguridad no es una opción. La mayoría de las distribuciones separan los repositorios de seguridad, lo que facilita su aplicación.

Gestión de dependencias

Una diferencia respecto de Windows es que, en Linux, la actualización de un paquete puede implicar la actualización de otros debido a las dependencias. Los gestores de paquetes suelen resolver este proceso de forma automática, pero es importante verificar que, tras la actualización, todo continúe funcionando correctamente. Por este motivo, en servidores conviene, cuando sea posible, contar con entornos de prueba para aplicar actualizaciones críticas antes de hacerlo en producción. En entornos de escritorio, en general, las actualizaciones se aplican directamente sin mayores inconvenientes.

***Kernel*: cuidado con módulos de terceros**

Si se utilizan *drivers* privativos (por ejemplo, NVIDIA) o módulos gestionados mediante DKMS, al actualizar el *kernel* es importante asegurarse de que dichos módulos se recompilen para la nueva versión. De lo contrario, podría perderse funcionalidad (por ejemplo, tras una actualización del *kernel*, la GPU NVIDIA puede requerir la reinstalación del módulo correspondiente). En general, herramientas como APT o DNF se encargan de este proceso automáticamente cuando se utiliza DKMS. Conviene simplemente tenerlo en cuenta.

Reinicio pendiente

En Ubuntu se crea el archivo `/var/run/reboot-required` cuando se instalan actualizaciones que requieren un reinicio del sistema (normalmente del *kernel* o de bibliotecas críticas como `glibc`). Tras aplicar una tanda de actualizaciones, se puede comprobar su existencia con el comando `ls -l /var/run/reboot*`.
Si el archivo está presente, conviene planificar un reinicio a la brevedad. Mantener el sistema funcionando durante mucho tiempo sin reiniciar, cuando hay parches de *kernel* pendientes, implica que ciertas vulnerabilidades corregidas aún no están efectivamente mitigadas hasta que el nuevo *kernel* se carga.

Automatización y buenas prácticas

En entornos con múltiples sistemas, es posible utilizar herramientas como Ansible o scripts de *shell* para aplicar parches de forma masiva. Sin embargo, en una pyme con pocos servidores, suele ser suficiente conectarse por «SSH» y ejecutar periódicamente `apt upgrade`.

Resulta importante documentar un calendario de actualizaciones (por ejemplo, «el primer viernes de cada mes se actualizan los servidores» o habilitar actualizaciones automáticas) y registrar los cambios realizados. Llevar un control de cuándo y qué se actualizó permite, en caso de que surja un problema, saber dónde investigar.

En entornos *enterprise*, la gestión de parches suele estar altamente formalizada (ventanas de mantenimiento, pruebas en entornos de *staging*, etc.). En una pyme, quizá no exista ese nivel de formalismo, pero al menos debe haber conciencia del riesgo: no es aceptable dejar un servidor Linux funcionando durante años sin actualizar, ya que acumulará vulnerabilidades conocidas.

De hecho, muchas infecciones de *ransomware* en NAS o servidores Linux se producen porque estos ejecutaban *kernels* antiguos o servicios desactualizados (por ejemplo, versiones antiguas de «Samba» con fallos conocidos). En muchos de estos casos, la simple aplicación de parches habría evitado el incidente.

Conclusión

Las actualizaciones del sistema se deben aplicar de forma rutinaria, tanto las de seguridad como otras actualizaciones importantes. Linux es, en general, muy estable en términos de mantenibilidad, por lo que normalmente una actualización no rompe componentes críticos, especialmente cuando se utilizan versiones LTS.

En el caso poco frecuente de que una actualización genere un problema de compatibilidad, siempre resulta preferible gestionar ese inconveniente antes que operar un sistema comprometido por no haber sido parchado. Una gestión proactiva de parches reduce de manera significativa la superficie de ataque de los *endpoints* Linux.

Actividades prácticas y laboratorio

A continuación, se proponen una serie de ejercicios prácticos para afianzar la comprensión de los conceptos vistos. Estas propuestas no son de realización obligatoria. Los laboratorios están diseñados para entornos simulados (por ejemplo, máquinas virtuales o equipos de prueba), de modo que se puedan realizar cambios de configuración sin riesgo para los sistemas de producción.

PRÁCTICA 1. HARDENING BÁSICO DE WINDOWS 10

PRÁCTICA 2. HARDENING BÁSICO DE LINUX (UBUNTU SERVER)

Objetivo: aplicar en un equipo con Windows las medidas fundamentales de *hardening* aprendidas: establecer políticas locales de seguridad, revisar «Defender» y el *firewall*, habilitar auditorías y comprobar los *logs*, y configurar adecuadamente «Windows Update» y el UAC.

Instrucciones

1. **Directiva de contraseñas y bloqueo de cuenta**

En un equipo de prueba con Windows 10 Pro, abre el «Editor de directivas de seguridad local» (secpol.msc). Navega a «Directivas de cuenta» > «Directiva de contraseñas» y **configura** valores como una longitud mínima de contraseña de 8 caracteres, la complejidad de contraseña habilitada y un período máximo de vigencia de 90 días.

Luego, haz clic en «Directivas de cuenta» > «Directiva de bloqueo de cuenta» y establece, por ejemplo, un umbral de bloqueo de 5 intentos fallidos y una duración del bloqueo de 15 minutos. Aplica los cambios y cierra la herramienta.

En el caso de Windows Home, estas opciones no están disponibles en secpol.msc, pero **puedes** configurarlas mediante comandos como net accounts desde la línea de comandos.

2. **Comprobación de Windows Defender**

Abre «Seguridad de Windows» > «Protección antivirus y contra amenazas». Realiza un «Examen rápido» de forma manual. Luego, ingresa en «Configuración de antivirus y protección contra amenazas» y verifica que estén activadas las siguientes opciones: «Protección en tiempo real», «Protección basada en la nube», «Envío de muestras automático», entre otras.

A continuación, intenta descargar el archivo de prueba EICAR (cadena de prueba de antivirus) desde *eicar.org* para comprobar que «Defender» lo detecta y bloquea. El antivirus debería mostrar una notificación indicando que se ha detectado una amenaza.

Si cuentas con conexión a internet, ejecuta también la opción «Buscar actualizaciones de protección» para asegurarte de que el antivirus dispone de las definiciones más recientes.

3. **Comprobación del Firewall de Windows**

Accede a «Firewall y protección de red». Debe mostrarse el mensaje «Firewall de Microsoft Defender está activado» en los distintos perfiles de red. Si alguno aparece desactivado, **actívalo**.

Luego, **haz clic** en «Configuración avanzada». Esto abre la consola del *firewall* avanzado. Allí, **crea** una regla de entrada personalizada. Por ejemplo, bloquea completamente el puerto 445 (SMB) desde cualquier origen. Esta acción es ilustrativa; en Windows 10 el puerto suele estar bloqueado en redes públicas, pero se realizará manualmente para el ejercicio.

Para ello, sigue estos pasos: «Reglas de entrada» > «Nueva regla» > «Puerto» > TCP 445 > «Bloquear la conexión» > aplicar a todos los perfiles > asigna el nombre «Bloquear SMB». Guarda

la regla. Comprueba que la regla existe y se encuentra habilitada.

Nota: si dispones de dos equipos, puedes probar antes y después intentando acceder a `\\IP\c$` para verificar que, en una red privada, el acceso funcionaba previamente y que, tras crear la regla, queda bloqueado. Si no cuentas con un segundo equipo, basta con verificar la existencia de la regla.

4. **Habilitar auditoría de inicios de sesión**

Abre el «Editor de directivas de grupo local» (`gpedit.msc`) o la «Directiva de seguridad local» (`secpol.msc`), según disponibilidad. Ve a «Configuración de Windows» > «Configuración de seguridad» > «Directivas locales» > «Directiva de auditoría».

Habilita la opción «Auditar eventos de inicio de sesión» tanto para «Éxito» como para «Error». Aplica los cambios. (En un entorno de dominio, esta configuración se realizaría mediante GPO).

A continuación, genera eventos de prueba: intenta iniciar sesión con un usuario local utilizando una contraseña incorrecta para provocar un fallo. Por ejemplo, desde la pantalla de bloqueo, introduce la contraseña de forma incorrecta un par de veces y luego correctamente.

5. **Revisar el Visor de eventos**

Abre el «Visor de eventos» > «Registros de Windows» > «Seguridad». Usa la opción «Actualizar» para refrescar la vista y busca eventos con ID `4625` (inicio de sesión fallido) y `4624` (inicio de sesión exitoso). Deberían aparecer con la hora correspondiente a las pruebas realizadas.

Haz doble clic en alguno de los eventos para ver sus detalles y verifica información como el nombre de la cuenta, el equipo de origen, el tipo de inicio de sesión, entre otros datos. Esto confirma que la auditoría está funcionando correctamente. Deja habilitada esta auditoría para monitorear futuros intentos sospechosos.

6. **Configuración de UAC**

Ve a «Panel de control» > «Cuentas de usuario» > «Cambiar configuración de Control de cuentas de usuario». Verifica que el nivel esté configurado en el valor predeterminado (segundo escalón comenzando desde arriba, que indica «Notificarme solo cuando las aplicaciones intenten hacer cambios en mi equipo»). Si no lo está, ajústalo y acepta los cambios.

Si dispones de tiempo, prueba bajar el control al nivel mínimo y confirma la acción. Luego, intenta ejecutar cualquier instalador (`.exe`) y observa que no aparece el aviso del UAC.

Finalmente, vuelve a configurar el nivel recomendado. Esta prueba es únicamente para observar la diferencia de comportamiento.

7. **Simular Windows Update**

No es necesario instalar actualizaciones reales si no quieres. Ve a «Configuración» > «Windows Update» y haz clic en «Buscar actualizaciones». Si el sistema encuentra y descarga actualizaciones, deja que se instalen. Observa si se requiere un reinicio, lo cual es habitual tras paquetes de mayor tamaño.

Luego, practica la configuración de las «Horas activas» para evitar reinicios automáticos en horarios productivos. Para ello, ve a «Windows Update» > «Opciones avanzadas» > «Establecer horas activas». Reflexiona sobre cómo aplicarías esta configuración en un entorno corporativo (por ejemplo, mediante directivas de grupo para definir un horario común de actualización o el uso de un servidor WSUS).

8. *Cleanup*

Si lo deseas, restaura las políticas modificadas durante la práctica (o, al menos, recuerda revertir contraseñas de prueba y configuraciones temporales). Al tratarse de un equipo de laboratorio, también puedes dejar los cambios aplicados.

Resultado esperado

Tras completar esta práctica, habrás endurecido un sistema Windows 10 en varios frentes: política de contraseñas y bloqueo configurada, «Defender» actualizado (posiblemente con detección del archivo EICAR), *firewall* con una regla adicional restrictiva, auditoría de inicios de sesión habilitada y verificada en los *logs*, UAC configurado en un nivel seguro y sistema actualizado. En conjunto, estas medidas incrementan de forma notable la seguridad de un equipo Windows frente a su estado inicial de fábrica.

Objetivo: implementar en un sistema Linux (para el ejercicio se asume Ubuntu Server 22.04 LTS) las configuraciones de seguridad esenciales: activar y configurar «UFW», gestionar algunos servicios con `systemd`, ajustar permisos de archivos y realizar la actualización de paquetes, incluyendo el *kernel*.

Instrucciones

1. **Configurar UFW para un servidor básico**

Imagina que este sistema Ubuntu actúa como un servidor web. Primero, **instala** un servicio de ejemplo. Por ejemplo, `sudo apt install nginx -y`. Luego, **configura y habilita** el *firewall* «UFW»:

1. **Permite SSH**: `sudo ufw allow 22/tcp`
2. **Permite HTTP**: `sudo ufw allow 80/tcp`. Si planeas acceder vía HTTPS, permite también el puerto correspondiente: `sudo ufw allow 443/tcp`.
3. **Activa UFW**: `sudo ufw enable`. Confirma la acción con `y`.
4. **Comprueba el estado del *firewall***: `sudo ufw status verbose`

Deberías ver `Status: active`, reglas de tipo `ALLOW` para los puertos `22` y `80`, y las políticas por defecto (denegar conexiones entrantes y permitir las salientes).

Ahora, **prueba el *firewall***. Desde otra máquina —o desde el mismo sistema usando pruebas locales— **verifica** que el puerto `80` esté accesible. Al visitar `http://<IP>`, debería mostrarse la página predeterminada de Nginx.

También **comprueba** que otros puertos no permitidos, por ejemplo el 25, estén bloqueados. Una prueba como `telnet <IP> 25`, que no debería establecer conexión.

Opcional: configura una regla más restrictiva para permitir acceso solo desde una IP o red específica. Por ejemplo, si deseas limitar el acceso SSH a una red de confianza, `sudo ufw allow from 192.168.1.0/24 to any port 22`.

En un escenario real, podrías además eliminar la regla general del puerto `22` para restringir completamente el acceso SSH a la LAN.

2. **Gestionar servicios con systemd**

- **Lista los servicios activos**: `systemctl --type=service --state=running`. Observa qué servicios están en ejecución (por ejemplo, «nginx» debería estar activo, al igual que «sshd», entre otros).
- **Deshabilitar un servicio no necesario**: imagina que este servidor no va a utilizar servicios de correo. Es posible que exista un servicio como «postfix» (Ubuntu a veces lo instala para correo local). Para endurecer el sistema, supongamos que no se desea utilizarlo. Ejecuta `sudo systemctl disable postfix --now`. Este comando deshabilita el inicio automático del servicio y lo detiene inmediatamente.

- **Comprueba** el resultado con `systemctl status postfix`. Debería mostrarse como inactivo y con el estado de habilitación en «disabled».
- **Reinicia el servidor**: `sudo reboot`. Tras volver a iniciar sesión (por ejemplo, vía «SSH»), **ejecuta** nuevamente `systemctl status postfix` y **confirma** que el servicio no está en ejecución ni habilitado.

Supongamos ahora que tampoco se utilizan servicios de descubrimiento en red. En un servidor Ubuntu, se puede bloquear completamente «avahi-daemon», que proporciona descubrimiento de servicios en la LAN y no suele ser necesario en servidores. Para ello, ejecuta `sudo systemctl mask avahi-daemon`. Esto impedirá que el servicio se inicie incluso si algún *socket* intenta activarlo.

Comprueba el resultado con `systemctl status avahi-daemon`. El servicio debería aparecer como «masked». (Aunque Ubuntu Server no instala «avahi» por defecto, este paso ilustra el procedimiento para cualquier servicio no deseado).

2. **Revisión general.** Por último, revisa qué servicios están configurados para iniciarse automáticamente con `systemctl list-unit-files --state=enabled --type=service`. Identifica algún servicio prescindible en el contexto de este laboratorio y deshabilítalo. Documenta cuál servicio se deshabilitó y por qué. Recuerda que detener servicios críticos (como red, «cron» o «dbus») puede afectar el funcionamiento del sistema; céntrate en servicios de funcionalidad adicional, como impresión, descubrimiento de red o correo local.
3. **Comprobación de permisos en archivos clave:**
   - Lista el contenido de `/home` con `ls -l /home`. Asegúrate de que cada usuario tenga acceso únicamente a su propio directorio personal. Por ejemplo, si existen los usuarios «alice» y «bob», los directorios `/home/alice` y `/home/bob` deberían tener como propietario `alice:alice` y `bob:bob`, respectivamente, y permisos típicos como `drwx------` o `drwxr-xr-x`, según la *umask*. Si los permisos estuvieran demasiado abiertos (777, etc.), restríngelos con `chmod`. En Ubuntu, normalmente la *umask* hace que los permisos sean 750 (propietario con todos los permisos, grupo con lectura y ejecución, otros sin acceso). Esto suele ser adecuado, salvo que haya usuarios colaborando dentro de un mismo grupo.
   - Crea un archivo de prueba y experimenta con permisos: `echo "prueba" > /home/$(whoami)/test.txt`. Observa sus permisos con `ls -l`. Cambia los permisos con `chmod`: hazlo accesible a otros con `chmod 777 test.txt` y *known* luego intenta leerlo desde otro usuario (si tienes otro, o crea uno con `sudo adduser testuser`). Después, devuelve permisos seguros con `chmod 600 test.txt`. Observa cómo cambia la salida de `ls`

`-l`. Este ejercicio sirve para practicar la notación octal: `600 = rw-------`, `644 = rw-r--r--`, `700 = rwx------`, etc.

- Revisa algunos archivos del sistema. Por ejemplo, `ls -l /etc/shadow` (debe ser accesible solo por «root»); `ls -l /etc/passwd` (debe ser legible por todos, pero solo escribible por «root»: `-rw-r--r--`). Piensa por qué uno es legible y el otro no (`passwd` no contiene contraseñas, `shadow` sí).
- Opcional avanzado: verifica el ***sticky bit*** en `/tmp`. En este caso, `ls -ld /tmp` debería mostrar `drwxrwxrwt` (la `t` final indica que el ***sticky bit*** está activo). Si apareciera como `drwxrwxrw-` sin la `t`, podrías habilitarlo con `chmod +t /tmp`. Esto evita que un usuario borre archivos temporales de otro. Ubuntu lo configura así por defecto, por lo que este paso es solo de verificación.

4. **Configuración de sudoers segura**
   - Abre el archivo `sudoers` con `visudo`: `sudo visudo`. Observa su contenido. Deberían aparecer líneas como `%sudo ALL=(ALL:ALL) ALL`. Esto indica que el grupo «sudo» tiene privilegios completos. En algunas distribuciones también puede aparecer una línea como `%admin ALL=(ALL) ALL`, correspondiente a un grupo administrativo antiguo. Comprueba qué permisos tiene tu usuario ejecutando `sudo -l`. Deberían listarse los comandos permitidos, normalmente `(ALL)`.
   - Crea un caso de prueba. Añade, usando `visudo` (por ejemplo, al final del archivo), la siguiente línea `%marketing ALL=(ALL) /usr/bin/apt update, /usr/bin/apt upgrade`. Esto permitirá que los usuarios del grupo «marketing» ejecuten únicamente esos comandos con `sudo`. Guarda los cambios y sal del editor.
   - Ahora crea el grupo y un usuario de prueba:

- `sudo addgroup marketing`
- `sudo adduser mark1`
- `sudo usermod -aG marketing mark1`

Cambia al usuario de prueba `su - mark1`.

4. Ejecuta `sudo apt update`. El comando debería permitirse y solicitar la contraseña del usuario «mark1». Luego intenta `sudo apt install nginx`. Este comando debería ser denegado, indicando que no está autorizado en `sudoers`. Este ejercicio muestra cómo limitar el uso de `sudo` a comandos específicos.

5. Una vez finalizada la prueba, elimina la entrada añadida en `sudoers` o borra el usuario y el grupo de prueba para limpiar el sistema. Ten especial cuidado de no dejar errores de sintaxis al editar con `visudo`. Si no deseas ejecutar realmente actualizaciones durante el curso, puedes simular el resultado comprobando los permisos con `sudo -l -U mark1`. Esto mostrará qué comandos tendría permitidos ese usuario.

6. **Aplicación de actualizaciones:**

- Ejecuta:
- `sudo apt update`
- `sudo apt upgrade -y`

5. Observa qué paquetes se actualizan. Si la actualización incluye el *kernel* (por ejemplo, un paquete llamado `linux-image-...`), al finalizar verás un mensaje que sugiere reiniciar el sistema. También puedes verificar si hay un reinicio pendiente comprobando el archivo correspondiente que Ubuntu crea tras ciertas actualizaciones.

6. Antes de reiniciar, comprueba la versión actual del *kernel* con `uname -r`. Anota ese valor. Luego, reinicia el sistema: `sudo reboot`. Tras volver a arrancar, ejecuta nuevamente `uname -r`.

7. Comprueba si la versión cambió. Si es así, el sistema está ejecutando el *kernel* nuevo y parcheado. Si no cambió, es posible que no se haya instalado un *kernel* nuevo o que el reinicio no se haya realizado.

8. Configura actualizaciones automáticas de seguridad. **Instala** el paquete correspondiente si no está presente: `sudo apt install unattended-upgrades`. Luego ejecuta `sudo dpkg-reconfigure unattended-upgrades` y selecciona la opción afirmativa. Esto permitirá que Ubuntu instale automáticamente las actualizaciones de seguridad en segundo plano cada día. Puedes verificar la configuración en `/etc/apt/apt.conf.d/50unattended-upgrades`.

Ten en cuenta que este mecanismo aplica los parches, pero no reinicia el sistema automáticamente cuando se actualiza el *kernel*. El reinicio sigue siendo necesario para que los parches del *kernel* entren en vigor. Existen opciones para habilitar el reinicio automático con un horario definido, pero para este laboratorio es suficiente con conocer que la funcionalidad existe.

Si utilizas otra distribución, el procedimiento variará. Por ejemplo, en CentOS se emplearía `yum update`.

6. Opcionalmente, prueba alguna herramienta automática de auditoría de hardening para Linux, como «Lynis». Por ejemplo, puedes instalarla con `sudo apt install lynis -y` (o bien

descargar el *tarball* desde CISOfy). Luego, ejecuta `sudo lynis audit system`. Lynis realizará un escaneo de la configuración de seguridad del sistema y generará un informe con sugerencias. Observa cuántas de las recomendaciones corresponden a configuraciones que ya aplicaste durante esta práctica (por ejemplo, *firewall* activo o uso controlado de `sudo`) y cuáles apuntan a aspectos que aún no se han tratado (por ejemplo, configurar una *umask* más restrictiva o habilitar auditoría de procesos). Lynis es una herramienta útil para identificar posibles puntos de mejora en el proceso de *hardening* del sistema.

Resultado esperado

Tras completar esta práctica, tu servidor Ubuntu estará mejor asegurado: el *firewall* «UFW» activo y permitiendo solo los puertos necesarios, los servicios innecesarios detenidos y deshabilitados, los usuarios sin privilegios correctamente limitados (y `sudo` configurado de forma mínima para quienes lo requieran), los permisos de archivos sensibles verificados y el sistema actualizado a la última versión de paquetes y del *kernel*.

Además, habrás adquirido mayor soltura en el uso de `systemctl` y UFW, en la manipulación de permisos con `chmod` y `chown`, y en la edición segura del archivo `sudoers` mediante `visudo`, todas ellas habilidades fundamentales para administrar sistemas Linux de forma segura.

CONTINUAR

Referencias

Ruiz, P. (2019). Usar el visor de eventos en Windows 10. *SomeBooks*. https://somebooks.es/usar-el-visor-de-even windows-10/

Referencias bibliográficas de consulta

Achirou. (s. f.). *¿Debería usar el Firewall de Windows?* https://achirou.com/deberia-usar-el-firewall-de-windows/

Hostinger, (s.f.). *Cómo configurar un Firewall en Ubuntu con UFW*. https://www.hostinger.com/es/tutoriales configurar-firewall-ubuntu

Hostinger. (s. f.). *Administrar y listar servicios en Linux*. https://www.hostinger.com/es/tutoriales/administrar-y servicios-en-linux

Hostinger. (s. f.). *Cómo usar sudo y el archivo sudoers*. https://www.hostinger.com/es/tutoriales/usar-comando-s archivo-sudo

INCIBE, (2020). *Microsoft Defender: el antivirus de Windows a tu servicio*. https://www.incibe.es/ciudadania/blog/mic defender-el-antivirus-de-windows-tu-servicio

LabEx. (s. f.). *Guía del archivo sudoers*. https://labex.io/es/tutorials/nmap-how-to-configure-sudoers-file-correctly-419582

Mancera, C. (2021). *Uso de iptables*. https://webirix.com/uso-de-iptables/

Microsoft, (s.f.). *Firewall y protección de red en la aplicación Seguridad de Windows*. https://support.microsoft.c es/windows/firewall-y-protecci%C3%B3n-de-red-en-la-aplicaci%C3%B3n-seguridad-de-windows-ec0844f7-aebd-058 67fe-601ecf5d774f

Microsoft, (2025). *Introducción a la directiva de grupo para Windows Server*. https://learn.microsoft.com/es-es/wi server/identity/ad-ds/manage/group-policy/group-policy-overview

Netwrix Blog, (s.f.). *Mejores prácticas de Group Policy: Una guía completa para administra* https://netwrix.com/es/resources/guides/group-policy-best-practices/

Red Hat. (s. f.). *Gestión de servicios con systemd*. En *Red Hat Enterprise Linux 8: Configuración básica del sistema*. https://docs.redhat.com/es/documentation/red_hat_enterprise_linux/8/html/configuring_basic_system_settings/man services-with-systemd_configuring-basic-system-settings

Reddit. (s. f.). *¿Es Windows Defender suficiente para proteg* https://www.reddit.com/r/windows/comments/17eq15v/is_windows_defender_enough_to_protect_me/

Red Hat. (s. f.). *Gestión de permisos de archivos en RHEL 8*. En *Red Hat Enterprise Linux 8: Configuración bás sistema*. https://docs.redhat.com/es/documentation/red_hat_enterprise_linux/8/html/configuring_basic_system_settings/file-permissions-rhel8_configuring-basic-system-settings

SimeonOnSecurity, (2023). *Buenas prácticas básicas de Windows Hardening para asegurar Windows 10 y Wind* https://es.simeononsecurity.com/articles/windows-hardening-best-practices-secure-windows-10-windows-11/

Splashtop. (s. f.). *Gestión de parches en Linux: mejores prácticas*. https://www.splashtop.com/es/blog/linux-management

Xataka. (s. f.). *UAC de Windows: qué es, cómo funciona y cómo se configura el Control de cuentas de usuario*. https://www.xataka.com/basics/uac-windows-que-como-funciona-como-se-configura-control-cuentas-usuario

CONTINUAR