Módulo 3. Configuración segura e identidades

CÓRDOBA CREA FUTURO

1. Servidores Web

Los servidores web constituyen el componente central en la entrega de contenido y funcionalidades de una aplicación accesible a través de HTTP. Desde una perspectiva de seguridad, su configuración determina cómo se procesan las solicitudes, cómo se construyen las respuestas y bajo qué condiciones se ejecuta código en el navegador del cliente. En este marco, la seguridad no depende únicamente del desarrollo del *software*, sino también de los mecanismos de control que el servidor impone sobre la carga de recursos, la validación de datos y la interacción con el entorno del usuario.

En el contexto de riesgos de seguridad en aplicaciones, OWASP señala que cada debilidad debe analizarse considerando su explotabilidad, el impacto técnico y la probabilidad de ausencia de controles . En este sentido, la configuración del servidor influye directamente en esos factores, ya que puede reducir la superficie de exposición o, por el contrario, ampliarla cuando se permiten comportamientos no restringidos. La seguridad del servidor web se articula, por lo tanto, como un sistema de restricciones técnicas que limitan la ejecución de acciones no previstas.

Una de las dimensiones centrales de la seguridad en servidores web es el tratamiento de entradas no confiables. OWASP define los ataques de secuencias de comandos entre sitios como vulnerabilidades que ocurren cuando una aplicación incluye datos no validados dentro de la respuesta generada al usuario . Esta definición permite comprender que el servidor web funciona como intermediario entre la entrada externa y la salida renderizada en el navegador. Si el servidor no aplica controles adecuados, se convierte en el vehículo de ejecución de código malicioso.

La hoja de referencia de prevención de XSS de OWASP sostiene que la defensa exige que todas las variables sean sometidas a validación y posterior codificación o saneamiento antes de su inclusión en la interfaz . Desde la perspectiva de configuración, esto implica definir reglas de procesamiento que consideren cada contexto de salida —HTML, atributos, JavaScript, CSS o URL— como un entorno distinto con requerimientos específicos de codificación. La seguridad del servidor no se limita a bloquear ataques conocidos, sino que establece un modelo sistemático de tratamiento de datos.

Otro eje estructural en la configuración segura del servidor web es la Política de Seguridad de Contenido (Content Security Policy, CSP). La guía de MDN describe que el servidor puede enviar un encabezado *Content-Security-Policy* que restringe qué recursos puede cargar el documento y bajo qué condiciones . Esta política permite controlar la ejecución de scripts, deshabilitar código en línea y limitar la carga de recursos externos, introduciendo una capa adicional de restricción que opera directamente en el navegador.

La configuración también debe contemplar la gestión de solicitudes y estados de sesión. Aunque los ataques pueden clasificarse en diferentes categorías, como inyecciones de scripts o manipulación de solicitudes, todos comparten un punto común: la explotación de decisiones permisivas en el servidor. Cuando la arquitectura no distingue adecuadamente entre datos y código, o cuando no se verifican adecuadamente los permisos asociados a recursos internos, se amplía el margen de acción para vulnerabilidades como el acceso directo a objetos o la falsificación de solicitudes.

Finalmente, la seguridad en servidores web debe entenderse como un enfoque de defensa en profundidad. OWASP advierte que mecanismos como la CSP no sustituyen la validación y codificación de entrada, sino que actúan como una capa complementaria . De este modo, la configuración segura integra controles de validación, codificación contextual, restricciones de carga de recursos y limitaciones en la ejecución de código. Sobre esta base conceptual se desarrollarán los subtemas específicos: XSS reflejado y almacenado, CSRF tokens, IDOR y errores de configuración, cada uno analizado desde el punto de vista de controles y arquitectura del servidor.

XSS reflejado y almacenado

Los ataques de secuencias de comandos entre sitios (XSS) constituyen una vulnerabilidad de inyección que permite a un atacante ejecutar código en el navegador de la víctima mediante la inclusión de datos no confiables en la respuesta generada por la

aplicación. OWASP describe que esta situación ocurre cuando una aplicación utiliza información proveniente de una fuente no confiable y la incluye en el contenido dinámico enviado al usuario sin validarla ni codificarla adecuadamente . Desde la perspectiva del servidor web, el problema no reside únicamente en la entrada maliciosa, sino en la decisión de procesarla y reenviarla sin controles adecuados.

El XSS reflejado, también denominado no persistente, se produce cuando el script inyectado viaja en la solicitud HTTP —por ejemplo, en un parámetro de URL— y el servidor lo devuelve inmediatamente dentro de la respuesta . En este esquema, el servidor actúa como intermediario involuntario que refleja el contenido al navegador, el cual lo ejecuta al considerarlo proveniente de una fuente confiable. El control frente a esta modalidad exige mecanismos de validación y codificación antes de la construcción de la respuesta HTTP, de modo que el dato se trate estrictamente como texto y no como código ejecutable.

En contraste, el XSS almacenado o persistente implica que el contenido malicioso se guarda de forma permanente en el servidor, por ejemplo en una base de datos o en un sistema de comentarios, y se entrega posteriormente a otros usuarios cuando acceden al recurso afectado. En estas circunstancias, la vulnerabilidad no se manifiesta en un único ciclo de solicitud y respuesta, sino que se convierte en un problema sistémico que puede impactar a múltiples usuarios. El control debe aplicarse tanto en el momento de

almacenamiento como en el momento de presentación del contenido.

La hoja de referencia de prevención de XSS de OWASP establece que la protección requiere que todas las variables pasen por validación y posterior codificación o saneamiento antes de su inclusión en la interfaz . Este enfoque supone reconocer que cada contexto de salida —HTML, atributos HTML, JavaScript, CSS o URL— posee reglas de interpretación distintas por parte del navegador. Por lo tanto, la configuración segura debe contemplar mecanismos de codificación contextual específicos para cada tipo de salida.

Desde la arquitectura del servidor, el control de carga de recursos constituye otra línea de defensa. La Política de Seguridad de Contenido (CSP), enviada mediante el encabezado **Content-Security-Policy**, permite restringir qué recursos puede cargar un documento y bajo qué condiciones . Mediante directivas como **script-src**, el servidor puede limitar la ejecución de scripts únicamente a fuentes autorizadas o incluso bloquear completamente la ejecución de scripts en línea, reduciendo el impacto potencial de un XSS reflejado o almacenado.

La guía de CSP también describe el uso de nonces y *hashes* como mecanismo para autorizar explícitamente *scripts* legítimos . En una política basada en nonces, el servidor genera un valor aleatorio por cada respuesta y lo asocia a los scripts permitidos. El navegador ejecuta únicamente aquellos scripts cuyo valor coincida con el indicado en el encabezado. En este sentido, aunque un atacante

logre inyectar código en el contenido HTML, el navegador lo rechazará al no poseer el identificador autorizado.

No obstante, OWASP advierte que la CSP debe entenderse como una capa adicional dentro de una estrategia de defensa en profundidad . La codificación de salida continúa siendo necesaria, dado que la CSP no elimina la vulnerabilidad subyacente, sino que limita su explotación. Si la configuración del servidor permite la inclusión de datos sin validación, la aplicación seguirá siendo vulnerable desde el punto de vista lógico, aun cuando ciertas ejecuciones sean bloqueadas por políticas restrictivas.

Desde el análisis de riesgos propuesto en el marco del OWASP Top 10, la severidad de una vulnerabilidad se evalúa considerando factores como explotabilidad e impacto técnico . En el caso del XSS, la ausencia de controles incrementa la probabilidad de explotación, mientras que la posibilidad de acceder a *cookies*, *tokens* de sesión u otros datos sensibles eleva el impacto técnico. Por lo tanto, la configuración del servidor incide directamente en la reducción del riesgo, al limitar tanto la superficie de ataque como las consecuencias de una eventual explotación.

El tratamiento del XSS reflejado y almacenado desde la configuración y los controles del servidor implica un enfoque articulado que integra validación de entrada, codificación contextual de salida, restricciones de ejecución mediante CSP y limitación de APIs peligrosas. La seguridad no se alcanza mediante una única medida aislada, sino mediante la combinación coherente de mecanismos que impiden que datos no confiables se transformen en código ejecutable dentro del navegador del usuario.

Figura 1. XXS reflejado y almacenado

Fuente: elaboración propia.

CSRF tokens

El ataque ***Cross-Site Request Forgery*** (CSRF) consiste en inducir a un usuario autenticado a ejecutar una acción no deseada en una aplicación web en la que mantiene una sesión activa. Según OWASP, el ataque se basa en que el navegador envía automáticamente las

credenciales asociadas a la sesión —como *cookies* de autenticación — junto con cada solicitud HTTP dirigida al dominio legítimo. En este sentido, el problema no radica en la obtención de las credenciales, sino en la explotación del mecanismo automático de envío de las mismas.

Desde la perspectiva arquitectónica, el CSRF se produce cuando la aplicación no distingue entre una solicitud legítimamente generada por el usuario y una solicitud forzada desde un sitio externo. Si el servidor acepta cualquier solicitud autenticada sin verificar su origen o legitimidad contextual, se habilita la posibilidad de que un tercero envíe una petición maliciosa utilizando el navegador de la víctima como intermediario. La vulnerabilidad surge, por lo tanto, de la ausencia de controles adicionales que validen la intención del usuario.

El mecanismo técnico del ataque se basa en la confianza implícita que el servidor deposita en la sesión. Cuando el usuario ha iniciado sesión, su navegador adjunta automáticamente la *cookie* correspondiente en cada solicitud hacia el dominio objetivo. Un atacante puede construir un enlace, formulario o recurso incrustado que genere una solicitud HTTP válida hacia la aplicación vulnerable. Dado que la cookie se envía automáticamente, el servidor interpreta la solicitud como auténtica si no dispone de controles adicionales.

La principal estrategia de mitigación propuesta por OWASP consiste en el uso de tokens anti-CSRF. Estos tokens son valores impredecibles generados por el servidor y asociados a la sesión del

usuario. El token se incluye en formularios o solicitudes sensibles y debe ser verificado por el servidor al recibir la petición. Si el valor recibido no coincide con el token esperado, la solicitud es rechazada. Desde el punto de vista de configuración, esto implica que el servidor debe generar, almacenar y validar correctamente estos identificadores en cada operación que modifique estado.

Arquitectónicamente, el token anti-CSRF introduce un elemento de verificación adicional que no es automáticamente enviado por el navegador en solicitudes originadas desde otros dominios. A diferencia de las *cookies*, el atacante no puede conocer ni reproducir fácilmente el valor del token si este ha sido generado de forma impredecible y está vinculado a la sesión. De este modo, la arquitectura pasa de un modelo basado exclusivamente en autenticación a un modelo que también valida la intención explícita del usuario.

Otra línea de control relevante en la configuración del servidor es la verificación de encabezados como Origin y Referer. OWASP indica que estos encabezados pueden utilizarse como mecanismo complementario para comprobar que la solicitud proviene del dominio esperado. Sin embargo, estos controles deben considerarse adicionales y no sustitutivos del uso de tokens, ya que pueden existir escenarios en los que dichos encabezados no estén presentes o puedan ser manipulados.

Desde la configuración de cookies, el atributo SameSite constituye una medida relevante descrita por Mozilla MDN. Este atributo permite indicar al navegador que restrinja el envío de cookies en solicitudes de contexto cruzado. Cuando se configura como SameSite=Strict o SameSite=Lax, el navegador limita el envío automático de cookies en determinadas solicitudes originadas desde otros sitios. En este sentido, la política de cookies se convierte en un componente de la arquitectura defensiva frente a CSRF.

No obstante, la utilización de SameSite no reemplaza el uso de tokens anti-CSRF, sino que actúa como una capa adicional de restricción. Existen escenarios, especialmente en aplicaciones complejas o con integraciones externas, donde la configuración debe equilibrar usabilidad y seguridad. Por lo tanto, la arquitectura debe contemplar explícitamente qué operaciones modifican estado y aplicar controles reforzados en dichos puntos críticos.

Desde el análisis de riesgos, la vulnerabilidad CSRF puede generar impactos significativos, ya que permite ejecutar acciones con los privilegios de la víctima autenticada. La ausencia de tokens, validaciones de origen y configuraciones adecuadas de *cookies* incrementa la probabilidad de explotación. En consecuencia, el diseño seguro del servidor web debe integrar generación de tokens impredecibles, validación estricta en el backend y políticas de

cookies restrictivas como parte de una estrategia coherente de protección frente a solicitudes falsificadas.

IDOR

El Insecure Direct Object Reference (IDOR) constituye una vulnerabilidad de control de acceso que se produce cuando una aplicación permite acceder a objetos internos —como registros de base de datos, archivos o recursos— mediante identificadores directos sin verificar adecuadamente los permisos del usuario. Desde la perspectiva conceptual, el problema no reside en el identificador en sí mismo, sino en la ausencia de una validación de autorización que determine si el solicitante tiene derecho a acceder al recurso referenciado.

Según las guías de autorización de OWASP, todo acceso a recursos debe estar sujeto a una verificación explícita de permisos en el lado del servidor. En este sentido, la autorización no puede depender de supuestos implícitos, como la dificultad de adivinar un identificador o la estructura interna de las URLs. La arquitectura debe asumir que cualquier identificador expuesto al cliente puede ser manipulado y que toda solicitud debe validarse contra las reglas de acceso definidas.

La vulnerabilidad IDOR se manifiesta típicamente cuando un usuario modifica un parámetro en la URL, en un cuerpo de solicitud o en un campo oculto de un formulario para acceder a un recurso que pertenece a otro usuario. Si el servidor no comprueba que el

objeto solicitado está efectivamente asociado al usuario autenticado, se produce una ruptura del control de acceso. En este sentido, el problema se ubica en la lógica de autorización, no en la autenticación.

Desde la arquitectura del servidor, el control debe implementarse mediante verificaciones centralizadas y sistemáticas. OWASP recomienda que cada solicitud que implique acceso a datos sensibles o recursos restringidos incluya una comprobación de autorización basada en el contexto del usuario. Esto implica consultar atributos como roles, identificadores de propietario o políticas de acceso antes de devolver cualquier objeto solicitado. La validación debe realizarse siempre en el backend, independientemente de las restricciones aplicadas en la interfaz.

PortSwigger señala que las vulnerabilidades de acceso directo a objetos forman parte de un conjunto más amplio de fallas de control de acceso. Estas fallas pueden surgir cuando la aplicación confía en identificadores secuenciales, estructuras predecibles o referencias expuestas sin protección adicional. En estas circunstancias, el atacante no necesita explotar una debilidad criptográfica ni técnica compleja, sino simplemente manipular valores accesibles desde el cliente.

Desde el punto de vista de configuración y diseño, una estrategia preventiva consiste en evitar exponer identificadores internos directos cuando no sea necesario. Sin embargo, incluso cuando se utilizan identificadores indirectos o no secuenciales, la verificación

de autorización sigue siendo obligatoria. La seguridad no puede depender únicamente de la ofuscación o complejidad de los identificadores, ya que la manipulación de parámetros es una práctica habitual en ataques de prueba y error.

El principio de autorización basada en políticas implica que cada operación debe evaluarse en función de reglas previamente definidas. Esto incluye tanto la verificación de que el usuario posee un rol adecuado como la confirmación de que el recurso solicitado le pertenece o está dentro de su ámbito permitido. La falta de este doble control —rol y pertenencia— es una causa frecuente de vulnerabilidades IDOR.

En términos de riesgo, las fallas de control de acceso se ubican entre las categorías de mayor impacto en aplicaciones web, dado que permiten la exposición de datos confidenciales o la modificación de información ajena. La explotabilidad suele ser elevada cuando los identificadores son fácilmente manipulables y no existen verificaciones adicionales. Por lo tanto, la implementación consistente de controles de autorización reduce tanto la probabilidad de explotación como el impacto técnico asociado.

El tratamiento del IDOR desde la perspectiva de control de acceso exige una arquitectura que valide sistemáticamente cada solicitud en el servidor, que no confíe en restricciones del lado del cliente y que aplique políticas de autorización

coherentes y centralizadas. La exposición de objetos debe considerarse un punto crítico de diseño, en el cual la verificación explícita de permisos constituye la única garantía efectiva frente a accesos indebidos.

Errores de configuración

Los errores de configuración constituyen una categoría de riesgo ampliamente reconocida en el ámbito de la seguridad de aplicaciones. OWASP Top 10 incluye la “Security Misconfiguration” como una debilidad recurrente que surge cuando sistemas, servidores, marcos de trabajo o componentes no se configuran de manera adecuada. Esta categoría no se limita a fallas de programación, sino que abarca decisiones técnicas relacionadas con parámetros, servicios habilitados, permisos, encabezados de seguridad y exposiciones innecesarias.

Desde una perspectiva conceptual, un error de configuración ocurre cuando el entorno operativo de la aplicación permite comportamientos que amplían la superficie de ataque. Esto puede incluir servicios innecesarios habilitados, directorios expuestos, mensajes de error detallados, configuraciones por defecto sin modificar o falta de endurecimiento del servidor. En este sentido, la seguridad no depende únicamente del código de la aplicación, sino del ecosistema técnico en el que se ejecuta.

OWASP señala que muchas vulnerabilidades explotables no se originan en defectos complejos, sino en configuraciones incompletas o inconsistentes entre entornos. La falta de segmentación adecuada, el uso de credenciales predeterminadas o la ausencia de políticas restrictivas en producción incrementan la probabilidad de explotación. Estas situaciones revelan que la configuración debe considerarse parte integral del diseño de seguridad y no una tarea secundaria posterior al desarrollo.

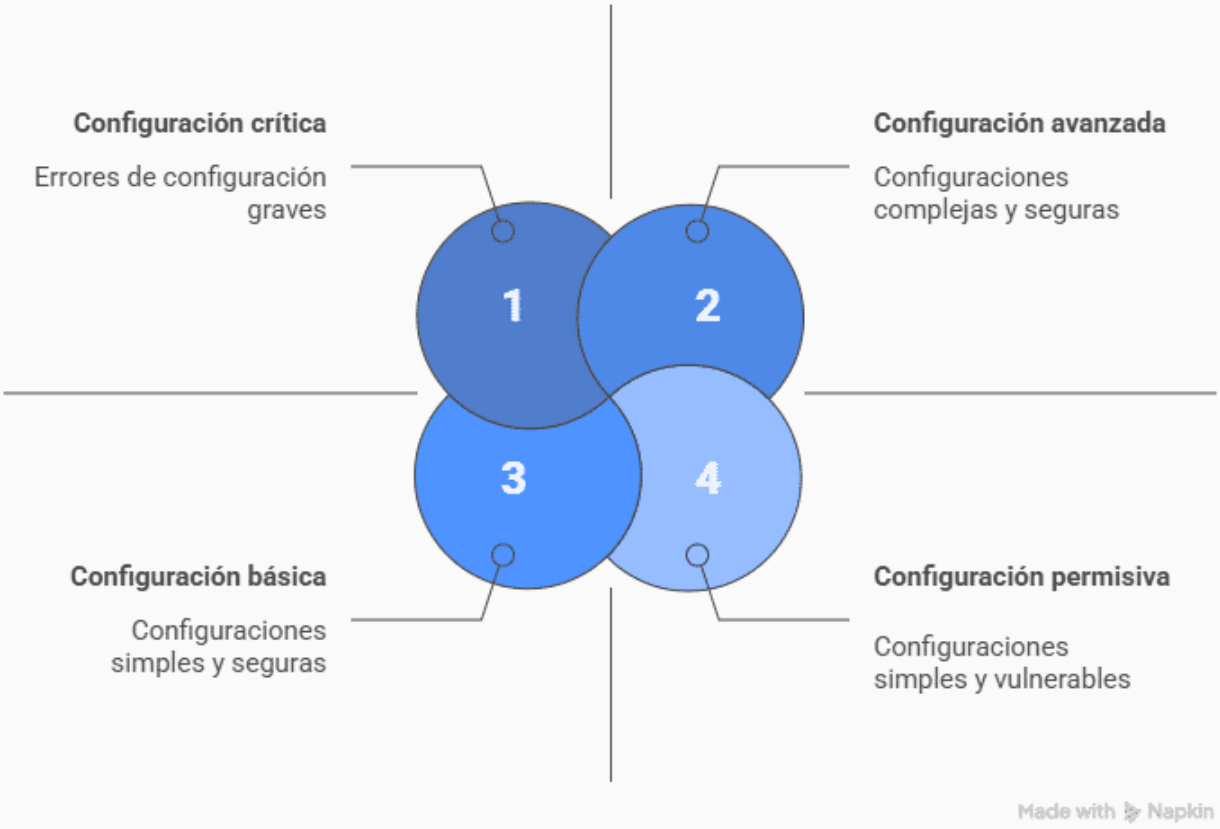
En el ámbito de los servidores web, los errores de configuración pueden manifestarse mediante la habilitación de métodos HTTP innecesarios, la exposición de archivos de respaldo, la visualización de listados de directorios o la divulgación de información técnica en encabezados de respuesta. Cada uno de estos elementos proporciona información que puede ser utilizada por un atacante para mapear el sistema y planificar vectores de explotación adicionales. La reducción de información expuesta se convierte así en una estrategia de control.

Desde el enfoque de los CIS Benchmarks, la configuración segura implica aplicar principios de endurecimiento sistemático. Aunque estos lineamientos suelen presentarse como listas de verificación, su valor conceptual radica en promover configuraciones mínimas, deshabilitar servicios no requeridos y establecer parámetros restrictivos por defecto. La idea subyacente es que todo componente activo debe justificarse en función de una necesidad operativa concreta.

La arquitectura de despliegue también influye en la aparición de errores de configuración. Entornos de desarrollo, prueba y producción suelen diferir en parámetros de depuración, niveles de registro y exposición de servicios. Cuando estas diferencias no se gestionan adecuadamente, es posible que configuraciones permisivas diseñadas para pruebas se mantengan en producción. En estas circunstancias, la falta de controles sistemáticos amplía el riesgo operacional.

La configuración de encabezados de seguridad constituye otro eje relevante. La ausencia de políticas como Content Security Policy, encabezados de protección frente a clickjacking o restricciones de tipo MIME puede facilitar la explotación de otras vulnerabilidades. En este sentido, la configuración no solo previene errores aislados, sino que fortalece la postura general frente a múltiples categorías de ataque.

Figura 2. Configurar con seguridad



Fuente: elaboracion propia.

Desde el análisis de riesgos propuesto por OWASP, la probabilidad de ausencia de controles y el impacto técnico asociado determinan la severidad de una categoría. En el caso de los errores de configuración, la incidencia suele ser elevada debido a la diversidad de entornos y tecnologías involucradas. La configuración incorrecta no requiere necesariamente un conocimiento profundo del sistema por parte del atacante, sino la identificación de un punto expuesto o una omisión en los controles.

Los errores de configuración representan una debilidad estructural que emerge cuando el servidor web y sus componentes no se ajustan a criterios restrictivos y coherentes. La mitigación exige una arquitectura orientada al principio de mínima exposición, la revisión sistemática de parámetros activos y la aplicación consistente de políticas de endurecimiento. La configuración segura no se limita a activar mecanismos defensivos, sino que implica eliminar toda funcionalidad innecesaria que pueda transformarse en un vector de ataque.

CONTINUAR

2. Gestión de identidades y mínimos privilegios o IAM (Identity and Access Management) básico

La gestión de identidades y accesos, conocida como IAM (Identity and Access Management), constituye el conjunto de principios, políticas y mecanismos destinados a garantizar que los usuarios accedan únicamente a los recursos que les corresponden dentro de un sistema. Desde una perspectiva teórica, IAM articula tres dimensiones interrelacionadas: identificación, autenticación y autorización. Mientras la identificación determina quién es el usuario, la autenticación verifica su identidad y la autorización define qué acciones puede realizar una vez autenticado.

En el contexto de la seguridad en aplicaciones web, la gestión de identidades no puede limitarse a la validación de credenciales. El diseño seguro requiere establecer reglas explícitas que definan cómo se asignan permisos, cómo se controlan los accesos y cómo se supervisa el uso de privilegios. En este sentido, IAM se integra con la arquitectura del servidor y con los mecanismos de control de acceso abordados previamente, ampliando la protección más allá del perímetro técnico.

Uno de los principios estructurales en la gestión de accesos es la separación entre autenticación y autorización. Un usuario puede estar correctamente autenticado y, sin embargo, no estar autorizado para acceder a determinados recursos o ejecutar ciertas acciones. Esta distinción evita que la autenticación sea interpretada como un permiso generalizado. La arquitectura debe garantizar que cada solicitud sea evaluada no solo en función de la identidad del usuario, sino también de sus privilegios asignados.

El modelo de roles y permisos constituye un enfoque ampliamente utilizado para estructurar la autorización. En lugar de asignar privilegios de manera individual a cada usuario, se definen roles con conjuntos de permisos preestablecidos. De este modo, la gestión se vuelve más sistemática y coherente. No obstante, la correcta definición de roles requiere un análisis previo de funciones y responsabilidades dentro del sistema, evitando superposiciones innecesarias o acumulación excesiva de privilegios.

El principio de mínimo privilegio establece que cada usuario debe contar únicamente con los permisos estrictamente necesarios para cumplir su función. Esta directriz reduce la superficie de impacto en caso de compromiso de credenciales o uso indebido de una cuenta. Desde el punto de vista arquitectónico, implica diseñar estructuras de permisos restrictivas por defecto y otorgar accesos adicionales solo cuando exista una justificación explícita.

En entornos dinámicos, la gestión de identidades también debe contemplar la temporalidad del acceso. Los privilegios no necesariamente deben ser permanentes, sino que pueden concederse por períodos limitados bajo condiciones controladas. Este enfoque se relaciona con modelos de acceso bajo demanda, donde la activación de permisos adicionales se encuentra sujeta a validaciones adicionales y registros de auditoría.

Finalmente, la gestión de identidades incorpora mecanismos de supervisión y revisión periódica de accesos. La auditoría y recertificación permiten verificar que los permisos asignados continúan siendo apropiados en función de los cambios organizativos o técnicos. En ausencia de estos procesos, los sistemas tienden a acumular privilegios innecesarios, lo que incrementa el riesgo de exposición. En consecuencia, IAM no se limita a la asignación inicial de accesos, sino que comprende un ciclo continuo de definición, control y revisión.

Roles y permisos

El modelo de roles y permisos constituye un mecanismo estructurado para la gestión de autorizaciones dentro de sistemas de información. Desde el enfoque de la guía de autorización de OWASP, toda decisión de acceso debe basarse en reglas explícitas y verificables que determinen si un usuario puede ejecutar una acción

sobre un recurso determinado. En este sentido, los permisos representan las acciones autorizadas y los roles funcionan como agrupadores lógicos de dichos permisos.

El control de acceso basado en roles, conocido como RBAC (Role-Based Access Control), establece que los usuarios no reciben permisos directamente, sino que se les asignan roles que contienen conjuntos definidos de privilegios. Microsoft, en su documentación sobre seguridad y modelo Zero Trust, describe este enfoque como una manera de simplificar la administración y reducir la complejidad operativa al estructurar las autorizaciones según funciones organizativas. De este modo, la relación entre identidad y acción se intermedia mediante una capa abstracta de roles.

Desde una perspectiva normativa, el NIST SP 800-53, en su familia de controles de Access Control (AC), establece que las organizaciones deben definir políticas que regulen el acceso lógico a sistemas y recursos. Estas políticas deben especificar quién puede acceder, a qué recursos y bajo qué condiciones. La formalización de roles se integra dentro de este marco, dado que permite materializar las políticas de acceso en estructuras técnicas verificables y auditables.

Conceptualmente, los permisos se definen como autorizaciones específicas para realizar operaciones tales como leer, modificar, eliminar o ejecutar recursos. Estos permisos deben asociarse a objetos concretos del sistema y no formularse de manera genérica. OWASP enfatiza que la autorización debe aplicarse en cada solicitud del lado del servidor, evitando confiar en controles implementados

únicamente en la interfaz del usuario. En consecuencia, la asignación de permisos debe traducirse en verificaciones activas en el backend.

El diseño de roles requiere un análisis previo de las funciones existentes en el entorno organizativo o técnico. La definición excesivamente amplia de un rol puede conducir a acumulación innecesaria de privilegios, mientras que una segmentación excesiva puede dificultar la administración. Por lo tanto, la estructuración de roles debe equilibrar granularidad y operatividad, manteniendo coherencia con las responsabilidades reales asociadas a cada perfil.

El modelo RBAC también introduce el principio de separación de funciones, el cual implica que determinadas acciones críticas no deben concentrarse en un único rol cuando ello pueda generar riesgos de abuso o conflicto de interés. Esta directriz, alineada con marcos normativos como NIST, busca reducir la posibilidad de uso indebido de privilegios al distribuir responsabilidades sensibles entre distintos perfiles de acceso.

Desde la arquitectura de autorización, la implementación de roles y permisos debe estar centralizada y ser consistente en toda la aplicación. Las decisiones de acceso no deben replicarse de manera aislada en distintos módulos, ya que esto puede generar inconsistencias. OWASP recomienda que las verificaciones de autorización sean uniformes y sistemáticas, aplicándose en cada punto donde se accede a recursos protegidos.

En entornos modernos, los roles pueden complementarse con atributos adicionales que contextualizan la decisión de acceso, tales como ubicación, horario o nivel de riesgo de la sesión. No obstante, incluso en estos modelos ampliados, el rol continúa siendo la estructura organizadora principal de los permisos. La combinación de roles con controles contextuales permite refinar la decisión sin abandonar la lógica estructural de RBAC.

El modelo de roles y permisos constituye una estrategia organizada para traducir políticas de acceso en reglas técnicas aplicables. Su correcta implementación exige definir permisos específicos, agruparlos en roles coherentes, aplicar verificaciones en el servidor y mantener alineación con marcos normativos de control de acceso. De este modo, la gestión de identidades adquiere una estructura sistemática que reduce la probabilidad de accesos indebidos y facilita la supervisión de privilegios asignados.

Mínimo privilegio

El principio de mínimo privilegio establece que cada usuario, proceso o sistema debe disponer únicamente de los accesos estrictamente necesarios para cumplir su función y no más. Según la definición del NIST, este principio implica limitar la autorización al conjunto mínimo de permisos requeridos para realizar tareas autorizadas. En términos conceptuales, el mínimo privilegio no se refiere a restringir arbitrariamente el acceso, sino a definirlo con precisión y justificación.

Desde la perspectiva de gestión de identidades, el mínimo privilegio opera como criterio orientador en la asignación de roles y permisos. Si los roles se diseñan sin considerar este principio, es probable que acumulen privilegios innecesarios que incrementen la superficie de riesgo. En este sentido, el mínimo privilegio actúa como mecanismo de contención frente a posibles compromisos de cuentas, reduciendo el alcance de acciones que un atacante podría ejecutar utilizando credenciales válidas.

En el marco normativo del NIST SP 800-53, los controles de acceso establecen que las organizaciones deben implementar restricciones técnicas que limiten el acceso lógico a información y recursos del sistema. Estas restricciones deben basarse en necesidades operativas verificables y revisarse periódicamente. El mínimo privilegio se integra así dentro de un esquema formal de control que exige definición explícita, documentación y supervisión.

Desde el enfoque arquitectónico, la aplicación del mínimo privilegio requiere segmentar permisos según funciones concretas y evitar privilegios amplios por defecto. Esto implica que los usuarios no deben recibir permisos administrativos o de alto impacto salvo que exista una justificación técnica clara. La asignación debe realizarse de manera diferenciada, garantizando que cada cuenta tenga un ámbito de acción acotado.

El modelo Zero Trust promovido por CISA refuerza este principio al establecer que ningún usuario o sistema debe considerarse confiable por defecto, incluso si se encuentra dentro del perímetro de la organización. Bajo este enfoque, el acceso se concede tras una verificación continua y contextual. El mínimo privilegio se convierte en una condición permanente, no en una configuración inicial estática.

En entornos dinámicos, la aplicación del mínimo privilegio también implica considerar la temporalidad del acceso. No todos los permisos deben mantenerse de manera indefinida; algunos pueden activarse únicamente cuando se requiere una tarea específica. Este planteamiento reduce la exposición continua a riesgos derivados de privilegios elevados permanentes.

Desde el punto de vista de la seguridad operativa, el incumplimiento del mínimo privilegio puede generar efectos acumulativos. La asignación excesiva de permisos facilita la escalada de privilegios y amplifica el impacto de errores humanos o compromisos de credenciales. Cuando múltiples usuarios disponen de accesos amplios, se incrementa la probabilidad de modificaciones no autorizadas o exposiciones accidentales de información.

La implementación efectiva del mínimo privilegio exige controles técnicos y administrativos complementarios. Técnicamente, deben establecerse mecanismos que restrinjan acciones sensibles y

registren el uso de privilegios elevados. Administrativamente, deben existir políticas claras que definan criterios de asignación y revisión periódica. La coherencia entre ambos niveles garantiza que el principio no quede reducido a una declaración formal sin aplicación práctica.

El principio de mínimo privilegio constituye un criterio estructural dentro de la gestión de identidades y accesos. Su aplicación requiere limitar permisos a lo estrictamente necesario, revisar periódicamente las asignaciones y adoptar un enfoque de confianza limitada en línea con los modelos Zero Trust. Al restringir el alcance de cada cuenta, se reduce la probabilidad de uso indebido y el impacto potencial de incidentes de seguridad.

Figura 3. Principio de Mínimo Privilegio

Principio de Mínimo Privilegio

Fuente: elaboración propia.

JIT Access

El ***Just-In-Time Access*** (JIT Access) constituye un modelo de gestión de privilegios que busca reducir la exposición permanente a accesos elevados mediante la concesión temporal y controlada de permisos. En lugar de asignar privilegios administrativos de forma continua, el acceso se habilita únicamente cuando existe una necesidad específica y por un período limitado. Desde el punto de vista conceptual, este enfoque traslada la gestión de privilegios desde una lógica estática hacia una lógica dinámica y contextual.

Microsoft, en su estrategia de acceso privilegiado, describe el acceso Just-In-Time como parte de un marco más amplio de protección frente a amenazas dirigidas a cuentas con altos privilegios. La idea central es minimizar el tiempo durante el cual una cuenta dispone de permisos sensibles. En estas circunstancias, incluso si las credenciales fueran comprometidas, el margen de acción del atacante se vería significativamente reducido.

Desde la arquitectura de control de acceso, el JIT se apoya en mecanismos que permiten solicitar, aprobar y registrar la activación de privilegios elevados. Esto implica que el sistema debe contar con procesos definidos para la elevación temporal de permisos, así como con registros que documenten cuándo, por qué y por cuánto tiempo

se concedió el acceso. La trazabilidad se convierte así en un componente estructural del modelo.

El enfoque descrito por BeyondTrust subraya que el acceso privilegiado permanente constituye uno de los principales vectores de riesgo en entornos corporativos. Las cuentas administrativas con permisos continuos amplían la superficie de exposición frente a ataques internos o externos. El modelo JIT busca eliminar esta exposición constante, manteniendo los privilegios inactivos hasta que sean estrictamente necesarios.

Desde el punto de vista del principio de mínimo privilegio, el JIT puede considerarse una extensión operativa del mismo. Mientras el mínimo privilegio limita el alcance de los permisos asignados, el JIT limita además su duración. La combinación de ambos criterios reduce tanto el ámbito como el tiempo de exposición, generando un entorno de control más restrictivo y verificable.

La implementación de JIT suele requerir autenticación reforzada antes de conceder privilegios elevados, así como procesos de aprobación basados en políticas. Esto implica que la activación no debe ser automática ni indiscriminada, sino sujeta a validaciones adicionales. De este modo, el sistema incorpora controles adicionales en los puntos donde se concentran mayores riesgos.

En el contexto de modelos Zero Trust, el acceso *Just-In-Time* se integra como un mecanismo coherente con la premisa de no otorgar confianza implícita. Incluso los usuarios con responsabilidades administrativas deben justificar y activar temporalmente sus privilegios. Este enfoque rompe con la práctica tradicional de cuentas administrativas permanentes, introduciendo un esquema más controlado y medible.

La trazabilidad asociada al JIT también facilita procesos de auditoría. Al existir registros detallados de cada activación de privilegios, resulta posible evaluar patrones de uso, detectar anomalías y revisar la adecuación de las solicitudes. En este sentido, el JIT no solo reduce exposición, sino que mejora la visibilidad sobre el uso de accesos sensibles.

El *Just-In-Time Access* constituye un modelo de gestión de privilegios basado en temporalidad, control y registro. Al conceder accesos elevados únicamente cuando son requeridos y durante intervalos limitados, se reduce la superficie de riesgo asociada a cuentas privilegiadas permanentes. Integrado con el principio de mínimo privilegio y con mecanismos de supervisión, el JIT fortalece la arquitectura de gestión de identidades al introducir una dimensión temporal en el control de accesos.

Auditoría y recertificación

La auditoría y recertificación de accesos constituyen mecanismos de control destinados a verificar que los permisos asignados a usuarios y sistemas continúan siendo apropiados en función de sus responsabilidades actuales. Desde el marco del NIST SP 800-53, la familia de controles Audit and Accountability (AU) establece la necesidad de generar, proteger y revisar registros de actividad que permitan reconstruir eventos relevantes relacionados con el acceso a recursos. En este sentido, la auditoría no se limita al registro técnico, sino que se integra en un sistema de supervisión estructurado.

El concepto de responsabilidad implica que toda acción ejecutada dentro del sistema debe poder asociarse a una identidad verificable. Para ello, los sistemas deben registrar eventos significativos como inicios de sesión, intentos fallidos de autenticación, modificaciones de permisos y accesos a información sensible. Estos registros permiten detectar anomalías, investigar incidentes y evaluar el cumplimiento de las políticas de acceso establecidas.

Desde la perspectiva normativa de ISO/IEC 27001, el control de accesos requiere no solo la definición inicial de permisos, sino también la revisión periódica de los mismos. La recertificación implica confirmar que cada usuario mantiene únicamente los privilegios que corresponden a su función vigente. Cuando los cambios organizativos no se reflejan en las asignaciones de acceso, se produce acumulación de privilegios innecesarios, lo que incrementa la exposición a riesgos.

La auditoría técnica constituye la base sobre la cual se sustenta la recertificación. Sin registros adecuados, resulta imposible evaluar el uso real de los permisos otorgados. En estas circunstancias, la gestión de identidades pierde visibilidad sobre cómo se utilizan los privilegios en la práctica. La combinación de registros detallados y revisiones periódicas permite identificar cuentas inactivas, accesos excesivos o asignaciones que ya no se justifican.

Microsoft, en el marco de Identity Governance y Access Reviews, describe procesos estructurados de revisión de accesos que involucran a responsables designados para validar periódicamente los permisos asignados a usuarios o grupos. Este enfoque introduce una dimensión formal de revisión en la que se documenta la confirmación o revocación de privilegios. De este modo, la recertificación deja de depender de verificaciones informales y se integra en un ciclo sistemático.

Desde el punto de vista arquitectónico, la auditoría requiere que los registros estén protegidos contra alteraciones y accesos no autorizados. Los controles del NIST establecen que los logs deben almacenarse de manera segura y mantenerse durante períodos definidos. Si los registros pueden modificarse o eliminarse sin control, se pierde la capacidad de reconstruir eventos y atribuir responsabilidades.

La recertificación periódica también contribuye al cumplimiento del principio de mínimo privilegio. Aun cuando los permisos hayan sido correctamente asignados en un inicio, el paso del tiempo puede

generar desalineación entre responsabilidades y privilegios. La revisión estructurada permite ajustar las asignaciones y reducir acumulaciones derivadas de cambios de rol, promociones o transferencias internas.

En términos de gestión de riesgos, la ausencia de auditoría y recertificación incrementa la probabilidad de accesos indebidos prolongados en el tiempo. Una cuenta con privilegios excesivos puede permanecer activa durante meses o años sin detección si no existen procesos formales de revisión. La supervisión periódica introduce un mecanismo de corrección que limita la persistencia de exposiciones innecesarias.

La auditoría y recertificación de accesos constituyen componentes estructurales de la gestión de identidades. A través del registro sistemático de eventos, la protección de logs y la revisión periódica de permisos, se garantiza que las asignaciones de acceso permanezcan alineadas con las políticas vigentes. Integradas dentro de un marco normativo y técnico coherente, estas prácticas fortalecen la responsabilidad, reducen acumulaciones indebidas y mejoran la visibilidad sobre el uso real de privilegios.

CONTINUAR

Referencias

BeyondTrust. (s. f.). Just-in-time access. https://www.beyondtrust.com/resources/glossary/just-in-time-access

Cloud Security Alliance & Cybersecurity and Infrastructure Security Agency (CISA). (2023). Zero trust maturity model. https://www.cisa.gov/zero-trust-maturity-model

International Organization for Standardization. (2022). ISO/IEC 27001:2022 Information security, cybersecurity and privacy protection — Information security management systems — Requirements. https://www.iso.org/isoiec-27001-information-security.html

Microsoft. (s. f.). Privileged access strategy. https://learn.microsoft.com/en-us/security/privileged-access-workstations/privileged-access-strategy

Microsoft. (s. f.). Role-based access control (RBAC). https://learn.microsoft.com/en-us/security/zero-trust/identity

Microsoft. (s. f.). Identity governance and access reviews. https://learn.microsoft.com/en-us/azure/active-directory/governance/

National Institute of Standards and Technology. (2020). Security and privacy controls for information systems and organizations (SP 800-53 Rev. 5). https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final

National Institute of Standards and Technology. (s. f.). Least privilege. https://csrc.nist.gov/glossary/term/least_privilege

Mozilla. (s. f.). Content Security Policy (CSP). https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CSP

Mozilla. (s. f.). Set-Cookie: SameSite attribute. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie/SameSite

OWASP Foundation. (2021). OWASP Top 10:2021. https://owasp.org/www-project-top-ten/

OWASP Foundation. (s. f.). Authorization cheat sheet. https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html

OWASP Foundation. (s. f.). Cross-site request forgery (CSRF). https://owasp.org/www-community/attacks/csrf

OWASP Foundation. (s. f.). Cross-site scripting (XSS). https://owasp.org/www-community/attacks/xss/

OWASP Foundation. (s. f.). Cross-site request forgery prevention cheat sheet. https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

CONTINUAR