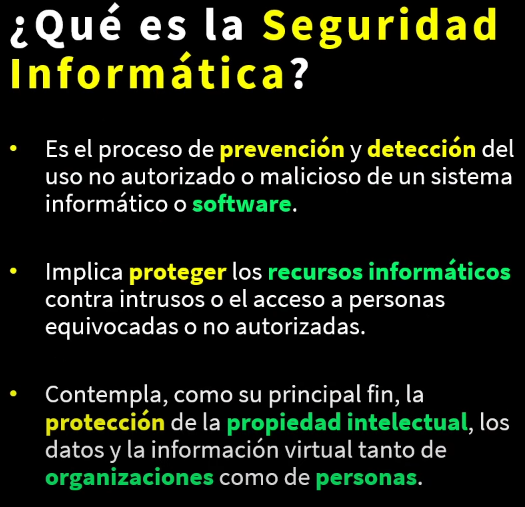
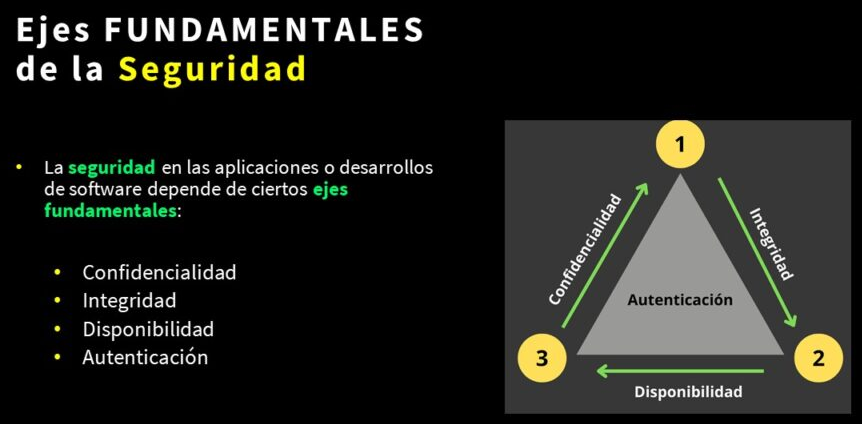
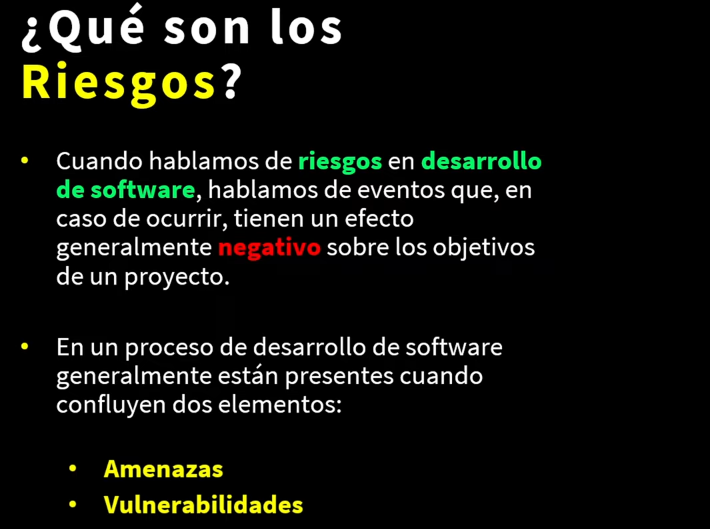
**Introducción a la Seguridad Informática**



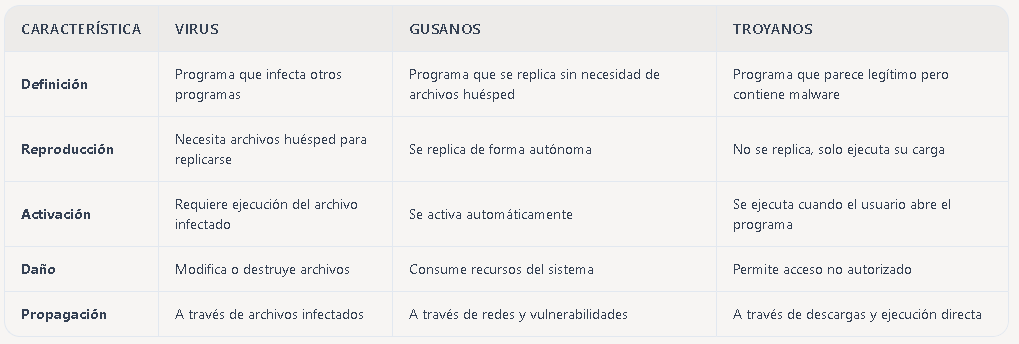


* **Confidencialidad:** Establece que sólo los usuarios autorizados pueden acceder a nuestros recursos, datos e información.
* **Integridad**: Sólo los usuarios autorizados pueden modificar los datos
* **Disponibilidad**: Es la condición donde los recursos pueden ser accedidos por los usuarios en un tiempo determinado.
* **Autenticación:** proceso de comprobación que comunica a las entidades.



**Amenazas**

* son posibles **eventos** o **acciones maliciosas** que buscan aprovechar las **vulnerabilidades** para comprometer la seguridad de un sistema. En su esencia, una amenaza en seguridad informática se define como cualquier evento o acción que tiene el potencial de causar daño o comprometer la seguridad de los sistemas de información.
* Pueden provenir de personas, malware, desastres naturales, errores humanos, entre otros. Las amenazas pueden aprovechar las vulnerabilidades para acceder, dañar o comprometer la integridad, confidencialidad o disponibilidad de los datos.
* Las **amenazas** en el ámbito de la **seguridad informática** representan una constante preocupación para individuos, organizaciones y la sociedad en su conjunto.
* Una de las **amenazas más comunes** en el panorama actual de la ciberseguridad son los **ataques de malware**.
* El **malware**, que incluye software como ser virus, gusanos, troyanos y/o ransomware, es software diseñado para infiltrarse en sistemas informáticos con el fin de dañarlos, robar información confidencial o extorsionar a los usuarios. Estos ataques pueden ocurrir a través de correos electrónicos maliciosos, descargas de software no seguras o vulnerabilidades en sistemas operativos y aplicaciones.



**Vulnerabilidades**

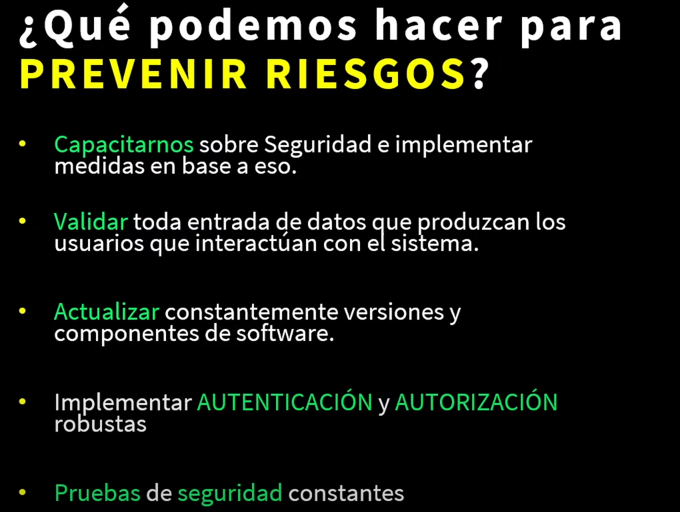
En el contexto de la seguridad informática, una vulnerabilidad es una debilidad o fallo en el diseño, implementación, configuración o mantenimiento de un sistema de software que podría ser explotada por un atacante para comprometer la seguridad del sistema.

Estas vulnerabilidades pueden manifestarse de diversas formas, como errores de programación, configuraciones inseguras, falta de parches de seguridad, entre otros.

Las vulnerabilidades en el software pueden surgir en cualquier etapa del ciclo de vida del desarrollo de software. Desde la fase de diseño, donde decisiones arquitectónicas pueden introducir potenciales puntos débiles, hasta la implementación, donde errores de codificación pueden pasar desapercibidos. Además, factores como la complejidad del software, la presión por cumplir plazos y la falta de conciencia sobre seguridad contribuyen a la proliferación de vulnerabilidades (en pocas palabras, errores humanos o vulgarmente llamados de «capa 8»).

Existen numerosos tipos de vulnerabilidades en el software, cada una con su propio mecanismo de explotación y potencial impacto. Algunos ejemplos comunes suelen ser:

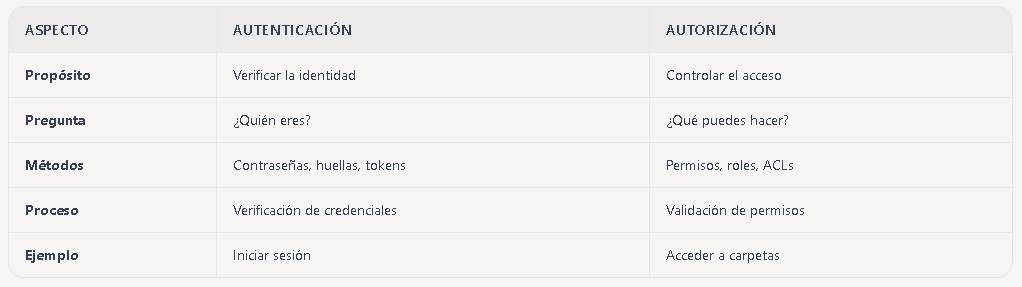
1. **Vulnerabilidades de inyección de código:** Estas vulnerabilidades permiten que un atacante inyecte y ejecute código malicioso en una aplicación, como en el caso de ataques de SQL injection o XSS (Cross-Site Scripting).
2. **Vulnerabilidades de autenticación y control de acceso:** Fallas en la gestión de credenciales de usuario pueden permitir a un atacante obtener acceso no autorizado a sistemas o datos sensibles.
3. **Vulnerabilidades de desbordamiento de búfer:** Este tipo de vulnerabilidad ocurre cuando un programa intenta almacenar más datos en un búfer de memoria del que puede contener, lo que puede llevar a la ejecución de código arbitrario.
4. **Vulnerabilidades de divulgación de información:** Errores en la gestión de datos confidenciales pueden resultar en la exposición no autorizada de información sensible.



**Autenticación y Autorización**

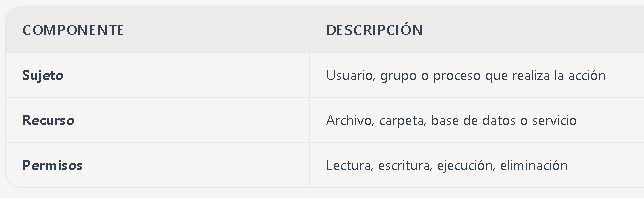
La **autenticación** se refiere al proceso de verificar la identidad de un usuario o entidad que intenta acceder a un sistema informático. En un contexto de software, esto generalmente implica solicitar al usuario que proporcione credenciales únicas, como un nombre de usuario y una contraseña, para demostrar su identidad. Sin embargo, la autenticación no se limita solo a las credenciales basadas en contraseñas; también puede incluir métodos más avanzados, como la autenticación multifactorial y la biometría.

***Una vez que un usuario ha sido autenticado con éxito, entra en juego el concepto de autorización***. La **autorización** se refiere al proceso de determinar qué acciones y recursos específicos tiene permitido un usuario una vez que ha sido autenticado. Esto se logra mediante la asignación de roles, permisos y políticas de acceso que definen los límites de las actividades que un usuario puede realizar en el sistema.



**Nota**:

Los ACLs (Access Control Lists o Listas de Control de Acceso) son reglas que definen quién puede realizar qué acciones sobre un recurso específico.



**Aplicaciones Stateful y Stateless**

Las **Aplicaciones Steful** son mejor conocidas como Aplicaciones con “**Seguridad basada en sesiones**”. Cuando un usuario se **autentica**, se crea una **sesión** en el servidor que contiene la información de dicho usuario (identidad, permisos y otros datos relacionados). El servidor utiliza esta información para **autorizar** las **solicitudes del usuario**y asegurarse de que solo tenga acceso a los recursos y funcionalidades permitidos (autorización).

Las aplicaciones stateful, como su nombre indica, mantienen un estado o contexto de sesión para cada usuario que interactúa con ellas. Esto significa que la aplicación almacena información sobre el estado de la sesión del usuario, como datos de sesión, historial de navegación, preferencias del usuario, entre otros. Esta información se guarda típicamente en la memoria del servidor o en una base de datos asociada con el usuario.

Para manejar los datos de estados, las aplicaciones Stateful pueden utilizar **cookies** u otros mecanismos para administrar y mantener la sesión del usuario en el servidor.

En cuanto a seguridad, las aplicaciones Stateful pueden ser **vulnerables** a ciertos tipos de **ataques** si no se implementan adecuadamente medidas de seguridad como la gestión adecuada de cookies, tokens anti-csrf, etc.

**Nota**: CSRF (Cross-Site Request Forgery) es un tipo de ataque donde un atacante fuerza a un usuario a realizar acciones no deseadas en una aplicación web que el usuario está autorizado a usar. Por ejemplo, Un atacante fuerza al usuario a transferir dinero sin su conocimiento

*Las características principales de las aplicaciones stateful son:*

* **Persistencia de Estado:** Mantienen información sobre el estado de la sesión del usuario a lo largo de múltiples interacciones.
* **Escalabilidad Limitada:** Debido a la necesidad de mantener el estado del usuario, las aplicaciones stateful pueden ser más difíciles de escalar horizontalmente, ya que cada instancia del servidor debe tener acceso al mismo estado.
* **Mayor Complejidad:** La gestión del estado puede agregar complejidad al diseño y desarrollo de la aplicación, especialmente en entornos distribuidos.

Las **Aplicaciones Stateless** son mejor conocidas como Aplicaciones con “**Seguridad basada en tokens de autenticación**”.

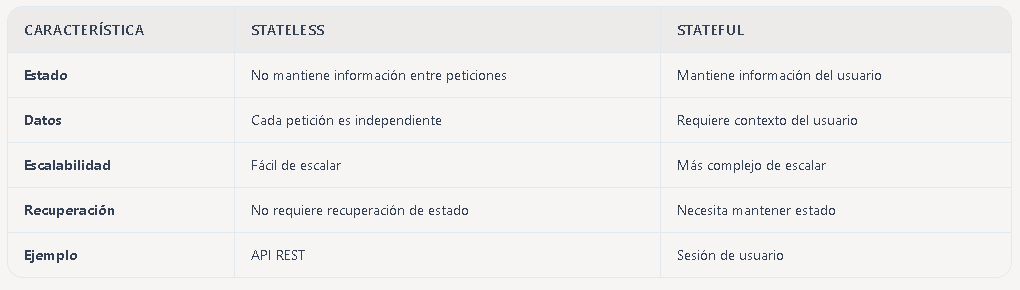
Las aplicaciones stateless no mantienen información sobre el estado de las sesiones de los usuarios entre solicitudes. Cada solicitud que realiza un usuario se procesa de manera independiente y sin conocimiento del contexto anterior. Esto significa que la aplicación no guarda ningún estado del usuario entre solicitudes, lo que simplifica significativamente la arquitectura y el diseño de la aplicación.

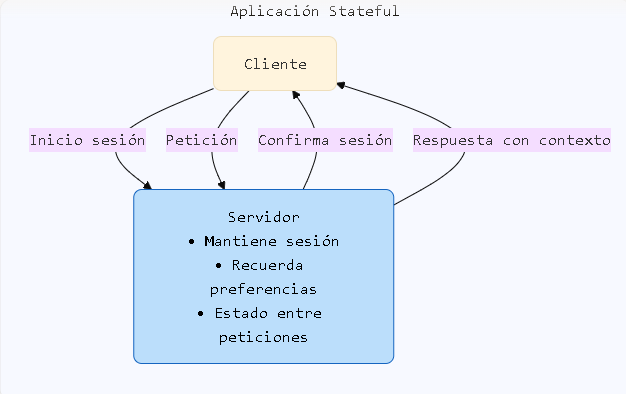
En las aplicaciones Stateless, después de que se logra una **autenticación** exitosa, el servidor emite un **token** de autenticación al cliente que contiene información sobre la **identidad del usuario**y cualquier **autorización** relevante.

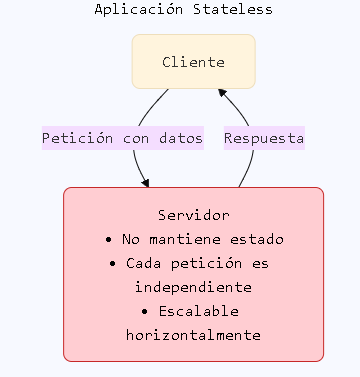
En cada solicitud siguiente, el cliente presenta este **token** al servidor para autenticar y autorizar la solicitud. Como el servidor no mantiene estado del usuario, es más fácil **escalar** y **distribuir** la aplicación. Ejemplo: El cliente puede **solicitar recursos** de cualquier servidor, lo que facilita la implementación de **sistemas distribuidos**y **balanceadores de carga**.

*Las características principales de las aplicaciones stateless son:*

* **Simplicidad:** Al no mantener estado de sesión, las aplicaciones stateless son más simples de diseñar, desarrollar y mantener.
* **Escalabilidad Mejorada:** Debido a su naturaleza sin estado, las aplicaciones stateless son más fáciles de escalar horizontalmente, ya que cada instancia del servidor es independiente y no requiere compartir estado.
* **Menor Uso de Recursos:** Al no tener que almacenar ni gestionar información sobre el estado de sesión, las aplicaciones stateless pueden requerir menos recursos de memoria y procesamiento.







**Ejemplo Práctico**

En un servicio de streaming:

Stateless: Cada petición de video es independiente, el servidor solo necesita la URL del video

Stateful: El sistema recuerda qué episodio estabas viendo, tus preferencias y la posición del video

Ambos enfoques son válidos y se utilizan según las necesidades específicas de la aplicación. **Las aplicaciones stateless son ideales para servicios que requieren alta escalabilidad, mientras que las stateful son mejores cuando se necesita mantener un contexto del usuario a lo largo del tiempo.**