1. Introducción a la seguridad de bases de datos. (Cheuk)

1.1. Introducción a las bases de datos

Una base de datos consiste en una colección de datos interrelacionados que representan información sobre una organización o área en particular. Estas bases de datos están organizadas según modelos de datos que definen la estructura, almacenamiento y manipulación de la información. El modelo principal es un modelo relacional que representa datos a través de tablas y crea relaciones entre ellos.

Los sistemas de gestión de bases de datos (DBMS) son programas informáticos que gestionan, modifican, consultan y crean bases de datos. Entre sus ventajas importantes se incluyen la independencia de los datos con respecto a la aplicación respectiva, la garantía de integridad mediante reglas de validación; seguridad de la información mediante control de acceso; y optimización del rendimiento a través de índices/vistas.

En el diseño de bases de datos intervienen una variedad de etapas, incluido el análisis de requisitos, el diseño conceptual, el diseño lógico y el diseño físico. Los modelos de entidad, atributos y relaciones se utilizan en el modelo de relación de entidad para representar datos en el diseño conceptual. El modelo relacional es la base del diseño lógico, ya que presenta datos a través de tablas, claves y restricciones.

El lenguaje utilizado para consultar la base de datos es un conjunto de instrucciones que permiten a los usuarios seleccionar, insertar, actualizar y eliminar datos. La mayoría de los DBMS pueden ser compatibles con SQL, qué es el lenguaje de consulta más utilizado.

En el ámbito de la programación de bases de datos, se emplean subrutinas, que son bloques de código encargados de llevar a cabo tareas específicas relacionadas con los datos. Estas subrutinas pueden manifestarse como procedimientos almacenados o funciones, diferenciándose en que los procedimientos almacenados pueden retornar varios valores o ninguno, mientras que las funciones solo pueden devolver un valor. La ejecución de estas subrutinas dentro del Sistema de Gestión de Bases de Datos (SGBD) contribuye a mejorar tanto la eficiencia como la seguridad de las bases de datos.

1.2. Introducción a la seguridad de base de datos

La seguridad de la base de datos abarca un conjunto de herramientas, medidas y controles diseñados para proteger la integridad, confidencialidad y disponibilidad de los datos almacenados en una base de datos. Estas “copias” están implementadas para evitar el acceso no autorizado, modificaciones inapropiadas o pérdidas accidentales. En este contexto, se cubrirán aspectos técnicos y organizativos, incluidas actividades como autenticación, auditoría, cumplimiento normativo, gestión de riesgos, educación, etc.

La importancia de la seguridad de las bases de datos reside en su función esencial para asegurar el funcionamiento adecuado de las organizaciones y la salvaguardia de la privacidad de las personas. Para preservar la integridad del sistema, resulta crucial hacer frente a amenazas como ataques internos, errores humanos, vulnerabilidades de software, ataques de inyección SQL, ataques de denegación de servicio y la presencia de malware.

#

Ante las amenazas mencionadas, se sugiere adoptar las mejores prácticas de ciberseguridad, que incluyen medidas como salvaguardar la integridad física, aplicar controles administrativos y de acceso a la red, asegurar dispositivos y cuentas de usuarios finales, proteger el software de bases de datos, garantizar la seguridad de servidores de aplicaciones y web, así como implementar precauciones en las copias de respaldo y llevar a cabo auditorías periódicas.

La seguridad de las bases de datos emerge como un tema de gran importancia y complejidad. En este contexto, se busca resguardar la información almacenada en dichas bases contra accesos no autorizados, alteraciones indebidas y pérdidas accidentales o maliciosas. La protección de la base de datos involucra diversos elementos críticos como:

- Datos de la base de datos: Esta información, que abarca desde nombres y contraseñas hasta números de tarjetas de crédito y debe mantenerse confidencial, íntegra y accesible únicamente para usuarios autorizados.
- Sistema de gestión de bases de datos (DBMS): Desempeña un papel fundamental al crear, administrar y manipular las bases de datos. Es esencial que el DBMS esté actualizado, configurado y protegido de manera adecuada para prevenir vulnerabilidades y ataques.
- Aplicaciones asociadas: Los programas que interactúan con la base de datos para realizar diversas operaciones deben validar y filtrar las entradas de los usuarios. Además, es crucial que utilicen consultas parametrizadas o procedimientos almacenados, y limiten los permisos y privilegios de los usuarios y las bases de datos.
- Servidor de base de datos físico y/o virtual y hardware subyacente: Estos dispositivos, ya sean físicos o virtuales, donde residen la base de datos y el DBMS, deben contar con protección física, sistemas de copia de seguridad y recuperación, así como medidas de seguridad de red, como firewalls, antivirus y cifrado.
- Infraestructura informática y/o de red para acceder a la base de datos: Los medios a través de los cuales usuarios y aplicaciones se comunican con la base de datos, como ordenadores, dispositivos móviles e internet, deben garantizar la seguridad y privacidad de las comunicaciones. Esto implica el uso de protocolos seguros, contraseñas robustas, certificados de seguridad, entre otras medidas.

#

La importancia de asegurar las bases de datos radica en la necesidad de prevenir o reducir al mínimo los riesgos y las consecuencias asociadas a los ciberataques. Estos eventos pueden resultar en daños irreparables tanto para los datos almacenados como para las organizaciones y los usuarios involucrados. Algunos de los ciberataques más comunes y peligrosos pueden ser:

- La inyección SQL: implica la introducción de código SQL malicioso en las entradas de los usuarios para alterar o acceder a la información de la base de datos. Este tipo de ataque puede ocasionar la pérdida o el robo de datos sensibles, la modificación o eliminación de información crucial, la ejecución de comandos arbitrarios en el servidor, o la revelación de datos internos o confidenciales de una organización.
- El robo de credenciales: se refiere a la obtención de contraseñas o nombres de usuario de usuarios o aplicaciones que acceden a la base de datos. El propósito de este ataque es suplantar identidades o llevar a cabo acciones no autorizadas, lo que puede resultar en acceso no autorizado a los datos, alteración o borrado de información, o la propagación de malware o virus.
- El ransomware: implica cifrar los datos de la base de datos o bloquear su acceso, con el objetivo de exigir un rescate a cambio de su liberación. Este tipo de ataque puede conducir a la inaccesibilidad de los datos, interrupciones en el funcionamiento del negocio o el pago de grandes sumas de dinero.

La seguridad de las bases de datos exige la implementación de un conjunto integral de medidas técnicas, organizativas y legales destinadas a resguardar los datos de posibles amenazas, tanto internas como externas, como hackers, empleados deshonestos, errores humanos, fallas de hardware o software, desastres naturales y violaciones normativas. Diversas prácticas, políticas y tecnologías pueden ser adoptadas para fortalecer la seguridad de las bases de datos, entre las que se encuentran:

- Diseñar una arquitectura de base de datos segura que separe los datos sensibles de los no sensibles, que minimice los puntos de acceso y que aplique el principio de mínimo privilegio.
- Implementar un sistema de gestión de bases de datos (DBMS) actualizado y configurado adecuadamente, que ofrezca funciones integradas de seguridad, tales como cifrado, control de acceso, registro de eventos y detección de anomalías.
- Desarrollar una aplicación segura que utilice métodos de conexión seguros, que evite la inyección de código malicioso, valide los datos de entrada y cifre la información en tránsito y en reposo.
- Llevar a cabo pruebas de seguridad periódicas que evalúen la vulnerabilidad de la base de datos e identifiquen y corrijan posibles debilidades o brechas.
- Impartir formación y concienciación a los usuarios y administradores de la base de datos acerca de las mejores prácticas de seguridad, como la utilización de contraseñas sólidas, el cambio regular de credenciales, el bloqueo de sesiones inactivas y la notificación de cualquier incidente sospechoso.

2. Riesgos y consecuencias de la inyección SQL (Cheuk)

2.1. Introducción a SQL

SQL (Structured Query Language) se trata de un lenguaje de programación especializado diseñado para la gestión y manipulación de bases de datos relacionales. Su función principal radica en permitir que los usuarios y las aplicaciones realicen consultas, inserten, actualicen y modifiquen datos almacenados en una base de datos.

Este lenguaje incorpora diversos comandos y cláusulas que posibilitan la interacción de los usuarios con los sistemas de gestión de bases de datos (DBMS), tales como MySQL, PostgreSQL, Oracle, Microsoft SQL Server, entre otros. Los comandos fundamentales de SQL abarcan desde la recuperación de datos mediante la cláusula SELECT, la inserción de datos mediante la cláusula INSERT, hasta la actualización y eliminación de datos mediante las cláusulas UPDATE y DELETE, respectivamente.

Además de estos comandos básicos, SQL ofrece la capacidad de crear y modificar esquemas de bases de datos mediante comandos de definición de datos, como CREATE, ALTER y DROP. Estos permiten a los usuarios establecer la estructura de la base de datos y sus objetos, como tablas, vistas, índices y procedimientos almacenados.

En términos generales, SQL provee a los usuarios de una herramienta poderosa y eficiente para la administración y manipulación efectiva de grandes volúmenes de datos. Esto lo posiciona como una herramienta esencial en el ámbito de la gestión de bases de datos y el desarrollo de software. Sin embargo, es crucial destacar que su uso indebido o incorrecto puede dar lugar a problemas de seguridad, como la inyección SQL, que puede comprometer la integridad y confidencialidad de los datos almacenados en la base de datos.

2.2. Inyección SQL

La inyección SQL es un tipo de ciberataque que se aprovecha de los errores existentes en aplicaciones web para meter código malicioso y atacar bases de datos de SQL. Al introducir el código van con el fin de quebrantar las medidas de seguridad y privacidad y así acceder a datos protegidos o de carácter sensible con el objetivo de eliminar información o incluso editar las bases de datos.

Existen tres clases de ataques principales de inyección SQL. Estos son:

- **Ataque por error:** Son los más comunes. Se puede obtener información de la base de datos.
- **Ataque por unión:** Una página web muestra más resultados de los que debería, entre los que se incluye la amenaza.
- **Ataque ciego:** Es el ataque más complicado de realizar. Se hacen preguntas cerradas a la base de datos

2.2.1. Riesgos que conlleva una inyección de SQL

Los atacantes de inyección SQL plantean una variedad de amenazas de seguridad para la organización afectada, afectando así los usuarios como las entidades que sufren este tipo de agresiones. Una vez los ciberdelincuentes se aprovechan de una vulnerabilidad, pueden:

- Robo de datos sensibles: los agresores tienen la capacidad de adquirir información confidencial, como contraseñas, detalles de tarjetas de crédito, datos personales o información de clientes. Este tipo de datos obtenidos pueden ser utilizados con intenciones maliciosas, como llevar a cabo fraudes, robar identidades o realizar chantajes.
- Modificación o eliminación de contenido de la base de datos: los atacantes pueden manipular o suprimir información almacenada en la base de datos. Esta acción puede tener consecuencias perjudiciales para el funcionamiento de la aplicación web, generando pérdidas económicas o dañando la reputación de la organización.
- Manipulación y/o ejecución de codigo: los atacantes pueden manipular o suprimir información almacenada en la base de datos. Esta acción puede tener consecuencias perjudiciales para el funcionamiento de la aplicación web, generando pérdidas económicas o dañando la reputación de la organización.
- Escalada de privilegios: Los atacantes pueden obtener privilegios de administrador o de usuario en la base de datos o en el servidor. Esto les permite llevar a cabo acciones que normalmente estarían fuera de su alcance, como la creación, modificación o eliminación de usuarios, tablas o registros.

2.2.2. Cómo evitar ataques e inyección SQL

Con el fin de prevenir la inyección SQL, es esencial implementar diversas medidas de seguridad, tanto por parte de los desarrolladores como de los usuarios. Algunas de estas medidas incluyen:

- Validar y filtrar las entradas de los usuarios para evitar la introducción de caracteres o comandos no deseados.
- Emplear consultas parametrizadas o procedimientos almacenados, los cuales impiden que el código SQL se modifique al ejecutarse.
- Restringir los permisos y privilegios de los usuarios y bases de datos para limitar el acceso y las acciones posibles.
- Mantener actualizados los sistemas, aplicaciones y bases de datos para corregir posibles vulnerabilidades o fallos de seguridad.
- Utilizar contraseñas seguras y gestores de contraseñas para salvaguardar las credenciales de acceso y prevenir posibles robos o adivinanzas.
- Proporcionar información personal únicamente en sitios web de confianza que cuenten con el protocolo HTTPS y un certificado de seguridad válido.

3. Tipos de ataques de inyección SQL + ejemplos de casos reales (Javi)

Las vulnerabilidades de inyección SQL surgen de la falta de validación de la entrada de datos proporcionada por los usuarios en una página web. Esta brecha de seguridad permite a un usuario concatenar comandos en lenguaje SQL en una consulta de la aplicación, lo que facilita la ejecución de código en la página. Existen tres tipos principales de inyección SQL:

1. **Inyección SQL In-Band**: Este tipo de ataque permite al usuario obtener información de la base de datos de la aplicación web utilizando el mismo canal de comunicación que se utiliza para explotar la vulnerabilidad. Es decir, los resultados de la consulta maliciosa se muestran en la pantalla del sitio web si se ejecuta el código desde algún campo de entrada. Este tipo de inyección es uno de los más comunes.

2. **Inyección SQL basado en error**: Este tipo de inyección permite extraer información de la base de datos al inducir errores a propósito a través de la interacción del cliente. En ocasiones, estos errores revelan información valiosa sobre la estructura y contenido de la base de datos.

3. **Inyección SQL a ciegas**: Esta variante de ataque permite obtener información de la base de datos sin que los resultados de la consulta se muestren en la pantalla del sitio web. En lugar de ello, se infieren los resultados a partir de pruebas de verdadero o falso que se pueden ejecutar en los campos de consulta vulnerables.

##3
Existen diferentes tipos de de inyección SQL, aquí están las más comunes:

- **Inyección SQL clásica:** También conocida como inyección SQL en banda, es la forma más habitual en ciberataques. Al introducir código SQL perjudicial en los campos de entradas del usuario, los atacantes logran acceso no autorizado para manipular, eliminar e incluso ejecutar comandos administrativos en la base de datos afectada. Es crucial estar consciente de esta amenaza común y tomar medidas preventivas.
- **Inyección SQL a ciegas:** obligan a los atacantes a operar sin acceso directo a la salida de la base de datos. En la mayoría de los casos, quienes hacen los ataques SQL a ciegas se basan en consultas verdaderas y falsas para recopilar información. Esta metodología permite a los atacantes deducir el esquema y contenido de la base de datos de manera progresiva.
- **Inyección SQL basada en errores:** aprovechan los mensajes de error de la base de datos para revelar información sensible. Los atacantes detrás de estos ataques envían intencionalmente consultas SQL malformadas, induciendo a la base de datos a generar mensajes de error que contienen información valiosa. Al analizar minuciosamente estos mensajes, los ciberdelincuentes pueden comprender el funcionamiento interno del sistema e identificar posibles puntos vulnerables.
- **Inyección SQL a ciegas basada en el tiempo:** representan un tipo más específico de ataque de inyección SQL a ciegas. Los atacantes que realizan estos ataques se

centran en el tiempo de respuesta de la base de datos para deducir información. Al enviar consultas SQL que provocan retrasos en la respuesta, pueden obtener detalles específicos sobre la base de datos basándose en el tiempo que lleva recibir una respuesta.

- **Inyección SQL fuera de banda:** emplean un canal de comunicación independiente para enviar y recibir datos, en lugar del canal directo entre la aplicación y la base de datos. Aunque menos frecuente, este método de ataque resulta poderoso, ya que permite a los atacantes evitar ciertas medidas de seguridad, como firewalls y sistemas de detección de intrusos, para cumplir con sus objetivos maliciosos.

##

Identificación de Vulnerabilidades

La elección del tipo de inyección SQL adecuado para una aplicación web en particular depende de las vulnerabilidades específicas que se encuentren. Para facilitar este proceso, se puede recurrir a herramientas automatizadas, siendo **SQLmap** una de las más destacadas.

SQLmap, un programa de código abierto incluido en Kali Linux, es una herramienta que realiza una amplia variedad de pruebas para detectar fallos de inyección SQL en aplicaciones web. Además, proporciona una gama de "payloads" que pueden ejecutarse para explotar las consultas vulnerables. Esto simplifica considerablemente el trabajo del **pentester**, quien solo necesita enfocarse en analizar la aplicación y familiarizarse con el uso de esta herramienta.

El uso de **SQLmap** permite llevar a cabo ataques de inyección SQL sin requerir un profundo conocimiento de SQL, siendo especialmente útil para pruebas de penetración de nivel fácil o intermedio. Sin embargo, es valioso adquirir conocimientos más avanzados en este lenguaje para comprender y prevenir ataques de mayor complejidad.

Métodos Manuales Simples

Si se está realizando una prueba de penetración de forma manual y se desea verificar la posibilidad de realizar inyecciones SQL, existen métodos sencillos para llevarlo a cabo.

El primer test consiste en enviar una comilla simple como entrada:

`

Si la entrada es vulnerable, la aplicación generará un error.

El siguiente test implica enviar el siguiente valor:

```
1' and '1'='1
```

Si la entrada se ejecuta sin problemas, indica que se está inyectando código SQL, ya que la condición '1'='1' siempre es verdadera.

Para verificar, se puede enviar el comando:

```
1' and '1'='0'
```

Si la entrada NO se ejecuta, se confirma que es posible inyectar código SQL en la entrada, ya que la condición '1'='0' es falsa y, por lo tanto, el comando no se ejecutará.

EJEMPLOS

1. **Incidente en Heartland Payment Systems (2008)**

En el invierno de 2008, el mundo financiero fue sacudido por un acto delictivo de proporciones digitales. Albert González, un conocido pirata informático, lideró un audaz ataque contra Heartland Payment Systems, una empresa encargada de procesar transacciones con tarjetas de crédito y débito. A lo largo de varias semanas, el grupo de González utilizó una técnica conocida como inyección SQL para infiltrarse en las defensas de Heartland.

Cada día que pasaba, Heartland se convertía inadvertidamente en un tesoro para González. El ataque resultó en una asombrosa violación que afectó a 130 millones de números de tarjetas de crédito y débito. Desde individuos hasta pequeñas empresas, nadie quedó inmune a la magnitud de esta brecha, que se extendió por todo Estados Unidos.

El daño financiero fue abrumador. Heartland enfrentó pérdidas de más de 145 millones de dólares en indemnizaciones por pagos fraudulentos. Como respuesta, Heartland reforzó sus protocolos de ciberseguridad, implementando la encriptación de extremo a extremo para proteger los datos de las tarjetas, lo que estableció un nuevo estándar en la industria.

Albert González fue condenado a 20 años de prisión en 2010, marcando el fin de su reinado de terror digital. La brecha en Heartland sirvió como un recordatorio contundente de la escala, alcance y potencial devastador de los ataques de inyección SQL.

2. **Incidente en Sony Pictures (2011)**

En 2011, un ciberataque envió ondas de choque a través de Hollywood, con Sony Pictures como víctima. El grupo de hackers conocido como LulzSec explotó una vulnerabilidad de inyección SQL para infiltrarse en la base de datos de Sony Pictures, dando inicio a una pesadilla digital de varias semanas.

El impacto del ataque se extendió tan ampliamente como la influencia global de Sony Pictures. Información confidencial se filtró en la red, incluyendo documentos, correos electrónicos e incluso películas no estrenadas. Fue un ataque directo a la industria del entretenimiento, afectando a individuos, empresas y gobiernos asociados con Sony Pictures.

Las consecuencias fueron significativas. Sony Pictures reportó una pérdida de 15 millones de dólares en su informe fiscal anual, directamente atribuida al ataque. El daño a la reputación de la empresa fue aún más notable.

Después de que el polvo se asentó, Sony Pictures fortaleció sus medidas de ciberseguridad y estableció estrictas prácticas de seguridad para prevenir futuros desastres. A diferencia de LulzSec, varios de sus miembros enfrentaron consecuencias legales en los años siguientes, subrayando las serias repercusiones legales de la ciberdelincuencia.

3. **Brecha en Yahoo! (2012)**

En el verano de 2012, Yahoo!, el gigante digital, sufrió un golpe devastador. El grupo de piratas informáticos conocido como D33Ds Company aprovechó una vulnerabilidad de inyección SQL en la base de datos de Yahoo!, obteniendo acceso a una gran cantidad de datos personales.

Este ataque internacional afectó a aproximadamente 450.000 usuarios de Yahoo! en todo el mundo. Las víctimas abarcaron desde usuarios comunes hasta pequeñas y medianas empresas que confiaban en Yahoo! para su comunicación digital diaria. Los datos comprometidos incluyeron nombres de usuario, direcciones de correo electrónico y contraseñas encriptadas de manera insuficiente.

Las secuelas fueron un testimonio del potencial destructivo de los ataques de inyección SQL. La reputación de Yahoo! se vio afectada y la compañía tuvo que enfrentar batallas legales y acuerdos durante años después de la brecha. A pesar de los considerables daños, Yahoo! utilizó esta experiencia como catalizador para el cambio. Reforzaron sus medidas de seguridad, introduciendo protecciones más robustas y promoviendo mejores prácticas de contraseñas entre sus usuarios.

A pesar de que la Compañía D33D no fue identificada, el ataque marcó su punto más alto y también su declive. La comunidad internacional y las fuerzas del orden iniciaron una búsqueda que finalmente llevó a la disolución del grupo.

4. **Vulnerabilidad en Drupal (2014)**

En octubre de 2014, se detectó una alarma silenciosa en las oficinas de Drupal, un sistema de gestión de contenidos muy popular. Se encontró una vulnerabilidad de inyección SQL en su sistema, que podría potencialmente afectar a millones de sitios web en todo el mundo.

La diversidad de usuarios de Drupal, desde individuos y pequeñas empresas hasta gobiernos y grandes corporaciones, estaban todos dentro del alcance de esta ciberamenaza internacional. La naturaleza de los datos en peligro era tan variada como los usuarios de Drupal, desde detalles personales hasta datos corporativos sensibles.

Aunque el daño financiero fue difícil de cuantificar debido a la naturaleza generalizada del ataque, las ondas de choque se sintieron en todo el mundo digital. A las siete horas de descubrir la vulnerabilidad, Drupal publicó una actualización de software para corregir el agujero de seguridad. Sin embargo, también anunciaron que cualquier sitio no

parcheado dentro de esa ventana debía considerarse comprometido, lo que provocó una loca revuelta entre sus usuarios.

A pesar de la ausencia de un autor conocido o de un número preciso de afectados, el incidente de Drupal subrayó el potencial alcance de los ataques de inyección SQL. El anonimato de los atacantes fue un escalofriante recordatorio de la naturaleza escurridiza de la ciberdelincuencia. Pero ante la adversidad, la comunidad digital se unió, actualizando sus sistemas y compartiendo medidas defensivas para capear juntos el temporal.

5. **Ataque a TalkTalk (2015)**

En otoño de 2015, TalkTalk, un gigante británico de las telecomunicaciones, se encontró en el punto de mira de un despiadado ciberataque. Un grupo de jóvenes piratas informáticos, liderados por un adolescente de 17 años, explotó una vulnerabilidad de inyección SQL para burlar las defensas digitales de TalkTalk.

El ataque se produjo a escala nacional, en todo el Reino Unido. Puso al descubierto los datos personales y financieros de 157.000 clientes de TalkTalk. Números de tarjetas de crédito, datos bancarios, nombres, direcciones, fechas de nacimiento, números de teléfono y direcciones de correo electrónico estaban en juego en este robo masivo de datos.

Las consecuencias financieras fueron monumentales. TalkTalk declaró unas pérdidas antes de impuestos de 60 millones de libras al año siguiente, consecuencia directa del ataque. También se enfrentaron a una multa récord de 400.000 libras de la Oficina del Comisionado de Información por fallos de seguridad.

A raíz de ello, TalkTalk reforzó sus medidas de seguridad, tranquilizó a sus clientes y trabajó incansablemente para restaurar su empañada reputación. Los jóvenes hackers fueron detenidos y se enfrentaron a cargos penales, un severo recordatorio de las graves consecuencias legales de la ciberdelincuencia.

6. **Base de Datos de Salud de Estonia (2020)**

En 2020, la pacífica nación del norte de Europa, Estonia, se enfrentó a un ciberataque de una magnitud sin precedentes. A pesar de su reputación de gobierno digital avanzado, Estonia no fue inmune a los ataques de inyección SQL.

El objetivo fue la Base de Datos Central de Salud de Estonia, un tesoro de registros médicos sensibles. Mediante un ataque de inyección SQL, los ciberdelincuentes comprometieron potencialmente los historiales médicos de casi todos los ciudadanos de Estonia, una brecha que abarcó toda la nación.

El daño financiero sigue sin revelarse, pero el impacto fue profundo, ya que expuso datos sanitarios privados y sacudió la confianza pública. Este ataque supuso una grave intrusión en la privacidad, que afectó tanto a particulares como a entidades gubernamentales.

En respuesta, Estonia emprendió inmediatamente amplias contramedidas, mejorando sus protocolos de ciberseguridad e implantando sistemas avanzados de detección de amenazas para evitar ataques similares en el futuro.

Aunque la identidad de los autores sigue siendo desconocida, el incidente envió un duro recordatorio a todo el mundo sobre la importancia de unas medidas de ciberseguridad sólidas para salvaguardar la información sensible. La respuesta, la capacidad de recuperación y el compromiso de Estonia con la seguridad digital sirvieron de faro para otras naciones que navegan por los procelosos mares de las ciberamenazas.

4. Buenas prácticas en el diseño de bases de datos para prevenir la inyección SQL (Sami)

Con el objetivo de evitar la inyección SQL, se crearon unos estándares para poder aplicar unas “técnicas” adecuadas para evitar estos problemas. Algunos de ellos son:

1.Emplear parámetros en lugar de realizar consultas concatenadas. Estas últimas son conocidas como una de las técnicas más frecuentes para la inserción de código malicioso. La utilización de parámetros previene que el código malicioso sea inyectado en la consulta. Debido a que las consultas concatenadas incluyen directamente variables que pueden tener un contenido malicioso a la hora de hacer la consulta.

2.Asegurarse de validar todas las entradas de usuario. Este parámetro es crucial para garantizar que no se incluye código malicioso. La validación puede lograrse mediante diversas técnicas, incluyendo la comprobación del formato, la longitud y el contenido de las entradas.

3.Utiliza filtros de salida para proteger los datos sensibles de posibles exposiciones. Estos filtros son eficaces para eliminar caracteres especiales que puedan ser aprovechados para inyectar código malicioso.

Consultas Concatenadas:

Las consultas concatenadas como hemos dicho anteriormente, al permitir la inyección de código malicioso a través de la simple concatenación con las entradas de usuario, representan un riesgo evidente, como se demuestra en este ejemplo:

*“SELECT * FROM users WHERE username = 'admin' AND password = '1234';”*

Un atacante o un “craker” puede llegar a aprovecharlo para inyectar código malicioso y eliminar la tabla de usuarios de la base de datos.
Para evitar la inyección SQL, resulta de crucial importancia sustituir las consultas concatenadas con el uso de parámetros. De esta manera, los valores se especifican en la consulta, dificultando la inyección de código malicioso. Por ejemplo:

*“SELECT * FROM users WHERE username = :username AND password = :password;”*

Aquí, el desarrollador especifica los valores de username y password como parámetros.

- La validación de todas las entradas de usuario es un paso esencial para asegurarse de las buenas prácticas. También es de debida importancia asegurarse de que las entradas cumplan con formatos específicos, limitar la longitud y eliminar caracteres especiales es clave para prevenir la inyección de código malicioso.

- Los filtros de salida son efectivos para eliminar caracteres especiales, como punto y coma (;), numeral (#), comilla simple ('), comilla doble ("), menor que (<) y mayor que (>), reduciendo el riesgo de inyección SQL.

Además de las buenas prácticas que se mencionaron en la introducción, existen otras medidas que se pueden tomar en el diseño de bases de datos para prevenir la inyección SQL y que están dentro de las buenas prácticas para poder evitar exploits o funcionamientos inadecuados.

Estas medidas son clasificables como las siguientes:

1.Utilizar tablas separadas para datos sensibles. Los datos sensibles, como las contraseñas, las tarjetas de crédito y los números de seguridad social, deben almacenarse en tablas separadas. Esto dificulta que un atacante acceda a estos datos si logra inyectar código malicioso en la base de datos.

Los datos sensibles, como las contraseñas, las tarjetas de crédito y los números de seguridad social, deben almacenarse en tablas separadas. Esto dificulta que un atacante acceda a estos datos si logra inyectar código malicioso en la base de datos.
Por ejemplo, una aplicación web que almacena datos de usuarios podría tener una tabla separada para almacenar contraseñas. Esta tabla solo debería permitir el acceso a usuarios autorizados, como administradores.

2.Utilizar funciones hash para almacenar contraseñas. Las contraseñas no deben almacenarse en texto plano en la base de datos. En su lugar, deben almacenarse en forma de hash. Esto hace que sea más difícil para un atacante descifrar las contraseñas si logra robarlas. Esto se debe a que las tablas hashes están encriptadas por su función hash.

Las contraseñas no deben almacenarse en texto plano en la base de datos. En su lugar, deben almacenarse en forma de hash. Esto hace que sea más difícil para un atacante descifrar las contraseñas si logra robarlas.

Las funciones hash convierten una contraseña en un valor único que es difícil de descifrar. Cuando un usuario inicia sesión, la contraseña ingresada se compara con el hash almacenado en la base de datos. Si los hashes coinciden, el usuario se autentica.
Utilizar procedimientos almacenados
Los procedimientos almacenados son funciones que se ejecutan en el servidor de base de datos. Estos procedimientos pueden utilizarse para validar las entradas de usuario y ejecutar consultas SQL de forma segura.

3.Utilizar procedimientos almacenados. Los procedimientos almacenados son funciones que se ejecutan en el servidor de base de datos. Estos procedimientos pueden utilizarse para validar las entradas de usuario y ejecutar consultas SQL de forma segura.

Los procedimientos almacenados pueden ayudar a prevenir la inyección SQL de las siguientes maneras:

- Validación de entradas: Los procedimientos almacenados pueden utilizarse para validar las entradas de usuario antes de ejecutar una consulta SQL. Esto ayuda a garantizar que las entradas no contengan código malicioso.
- Consultas seguras: Los procedimientos almacenados pueden utilizarse para ejecutar consultas SQL de forma segura. Esto ayuda a evitar que un atacante utilice una consulta SQL para inyectar código malicioso en la base de datos.

4.Utilizar cifrado para proteger los datos. Los datos sensibles pueden cifrarse para protegerlos de la exposición. Esto puede hacerse en el servidor de base de datos o en el cliente.

Los datos sensibles pueden cifrarse para protegerlos de la exposición. Esto puede hacerse en el servidor de base de datos o en el cliente.
El cifrado en el servidor de base de datos se realiza antes de que los datos se almacenen en la base de datos. El cifrado en el cliente se realiza antes de que los datos se envíen al servidor de base de datos.
El cifrado puede ayudar a proteger los datos sensibles de los ataques de inyección SQL. Si un atacante logra inyectar código malicioso en la base de datos, los datos cifrados estarán protegidos.

5. Marco legal y cumplimiento normativo (Sami)

corregido:

La inyección SQL como hemos visto hasta ahora, es una gran vulnerabilidad de seguridad en organizaciones o ambientes de desarrollo de software. Los “crakers” o los atacantes pueden llegar a aprovecharse de estas vulnerabilidades, llegando a robar datos datos o información crítica. Esto nos da paso a nuestro siguiente punto el cual es la importancia de comprender y cumplir con el marco legal y las regulaciones relacionadas con la inyección SQL.

En el contexto Español, la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales (LOPDGDD), establece las bases legales para abordar la inyección SQL. Dicha ley establece las responsabilidades que deben de manejar las entidades que influyen o manejan datos personales, para que así estas contengan las medadidas de seguridad correspondientes para proteger la información.

Además el artículo 32 de la LOPDGDD registra que aquellos que son responsables del tratamiento de datos deben adoptar necesarias para garantizar un nivel de seguridad adecuado, incluyendo:

La aplicación de medidas técnicas para proteger los datos personales contra el acceso no autorizado, la alteración, la divulgación, la destrucción o la pérdida accidental, ilícita o maliciosa.

La implementación de medidas organizativas adecuadas para asegurar que solo personas autorizadas tengan acceso a los datos personales.

La realización de copias de seguridad de los datos personales como precaución adicional.

Cumplimiento normativo

Para cumplir con las disposiciones de la “LOPDGDD” y mitigar la amenaza de la inyección SQL, las organizaciones deben llevar a cabo las siguientes medidas:

> Utilizar parámetros en lugar de consultas concatenadas para evitar la exposición a ataques SQL. Esta práctica permite separar los datos del código y previene la inyección de comandos maliciosos en las consultas.

Validar de manera rigurosa todas las entradas de usuario para garantizar que no contienen datos maliciosos. Esto incluye la validación del formato, la longitud y el contenido de las entradas para detectar cualquier anomalía.

Implementar filtros de salida para prevenir la exposición de datos confidenciales, ya que dichos filtros de salida son útiles para poder eliminar caracteretes indebidos y así poder garantizar datos seguros.

Además de estas medidas, las organizaciones deben adoptar otras prácticas para proteger eficazmente los datos personales:

- Se debe incorporar cifrado para salvaguardar la confidencialidad de la información sensible. El cifrado asegura que, aunque un atacante logre acceder a los datos, no podrá comprender su contenido sin la clave correspondiente.
- Se deben establecer controles de acceso rigurosos para limitar el acceso a los datos al personal autorizado.
- Se deben realizar auditorías de seguridad de forma periódica para identificar posibles vulnerabilidades y realizar correcciones oportunas. Las auditorías permiten evaluar el cumplimiento de las medidas de seguridad y detectar cualquier amenaza potencial.

Además de lo comentado, con el objetivo de ayudar a las organizaciones a cumplir con la LOPDGDD y prevenir la inyección SQL, sugieren las siguientes acciones:

- Crear y documentar una política de seguridad de la información que contemple medidas específicas para prevenir la inyección SQL, adaptada a las necesidades y el entorno de la organización. La política debe ser un documento vivo que se actualiza conforme cambian las amenazas y las necesidades de seguridad.

- Utilizar herramientas de seguridad SQL. Estas herramientas pueden incluir firewalls de aplicaciones web, sistemas de detección de intrusiones y análisis de vulnerabilidades.

La inyección SQL representa un riesgo considerable para la seguridad de las organizaciones, y su mitigación es fundamental. Al tomar medidas proactivas y cumplir con las leyes y regulaciones, las organizaciones pueden salvaguardar la integridad de sus datos y mantener la confianza de sus usuarios y clientes.

Además de la LOPDGDD, existen otras leyes y regulaciones relevantes en España, como la Ley 34/2002 de servicios de la sociedad de la información y de comercio electrónico (LSSI) y el Reglamento General de Protección de Datos (RGPD) de la Unión Europea. Cada una de estas regulaciones tiene requisitos específicos que las organizaciones deben cumplir, lo que enfatiza aún más la importancia de abordar la inyección SQL de manera efectiva en el entorno digital actual.

La gestión adecuada de la seguridad de la información y la prevención de la inyección SQL son evidentemente fundamentales en el desarrollo de software, tanto es así que las organizaciones que así lo implementen se encontrarán mejor preparadas para una posible vulnerabilidad.

6. Futuro de los ataques de SQL Injection + BBDD(Javi)

Con el rápido avance de la tecnología, se vislumbra un panorama complejo en la seguridad de sistemas y aplicaciones. Los ataques de inferencia surgen como un desafío crítico en el contexto de la Inteligencia Artificial (IA) y el Aprendizaje Automático (Machine Learning, ML).

Los sistemas de ML operan a través de la asimilación progresiva de datos de entrenamiento para perfeccionar sus capacidades de toma de decisiones y predicciones. Es bien sabido que cuantos más datos se suministren, mayor será la precisión en las decisiones resultantes. Una vez en producción, estos sistemas despliegan sus interfaces de programación de aplicaciones (API) públicas, las cuales son accesibles tanto para usuarios como para otras aplicaciones que buscan aprovechar su funcionalidad.

Sin embargo, esta apertura no viene sin sus preocupaciones. Los modelos de ML, en muchos casos, operan con datos de una naturaleza extremadamente confidencial, tales como detalles de tarjetas de crédito o información personal identificable, los cuales representan auténticos tesoros para posibles atacantes.

En este contexto, los **ataques de inferencia**, específicamente los de membresía (MI), emergen como una amenaza latente. En un ataque de este tipo, el objetivo del atacante radica en obtener los datos subyacentes que sirvieron para el entrenamiento del modelo o incluso comprender su funcionamiento interno. Estos ataques aprovechan el modo en que los modelos de ML emiten respuestas, basándose en la concordancia entre los datos de entrada y aquellos utilizados en el proceso de entrenamiento.

La proliferación del Aprendizaje Automático como Servicio (Machine Learning as a Service, MaaS) en los últimos años ha expandido la superficie de ataque. Empresas que desean implementar modelos sin tener que construirlos desde cero ahora pueden hacerlo mediante servicios en la nube. No obstante, esta comodidad también implica que los atacantes pueden explotar posibles vulnerabilidades en los modelos de ML, potencialmente llevando a cabo ataques de inferencia contra múltiples empresas si se descubren fallos en la implementación.

Para abordar estos desafíos emergentes, es crucial reconocer que los ataques a los modelos de IA y ML demandan enfoques de seguridad distintos. No existe un simple "parche" para solucionar un algoritmo, ya que el problema reside en los datos subyacentes utilizados en el entrenamiento.

Por tanto, se deben implementar controles específicos para mitigar estos riesgos:

- Los modelos de ML deben tener la capacidad de generalizar los datos de entrenamiento, asegurando que las puntuaciones sean coherentes entre datos previamente vistos y no vistos.
- Las API de modelos de ML deben contar con características de limitación para detectar posibles consultas maliciosas, siguiendo el modelo de alerta ante múltiples "inicios de sesión fallidos" en una dirección IP, como ejemplo análogo en ciberseguridad.
- Las pruebas de penetración en modelos de IA y ML deben integrarse como parte fundamental de las actividades regulares de seguridad, desarrollando casos de ataque específicos y detallados.

FUTURO DE LAS BASES DE DATOS

La migración hacia entornos en la nube se ha establecido como un tema crucial en la gestión de bases de datos en la era actual de la tecnología. Este cambio fundamental en la infraestructura de TI ha generado un impacto significativo en la forma en que las organizaciones gestionan y acceden a sus datos. Figuras influyentes en el ámbito de las bases de datos, como Michael Stonebraker, cuyo legado incluye el desarrollo de PostgreSQL y otras tecnologías vanguardistas, proporcionan una visión valiosa sobre esta transición hacia la nube.

El Rol de la Nube en el Futuro

La afirmación contundente de Stonebraker de que "La Nube está en Tu Futuro" resuena con la estrategia de titanes tecnológicos como Amazon, Google y Microsoft, que han invertido recursos sustanciales en el desarrollo de soluciones en la nube. Este movimiento hacia la nube refleja una transformación en la forma en que las organizaciones abordan el almacenamiento y procesamiento de datos a escala global.

Ventajas y Desafíos

La adopción de la nube abre una serie de oportunidades y desafíos para las organizaciones. En el lado positivo, la reducción de costos es un factor determinante. Stonebraker destaca que, si bien los precios actuales ya son más asequibles, se espera que la creación de un mayor número de centros de datos y la optimización de la infraestructura en la nube conduzcan a una reducción aún mayor en los costos de servidores y contenedores.

Además de los beneficios económicos, la elasticidad en el uso de recursos es una ventaja distintiva de la infraestructura en la nube. Las organizaciones pueden pagar solo por los recursos que utilizan y escalarlos dinámicamente en respuesta a la demanda, lo que ofrece una flexibilidad operativa sin precedentes.

Sin embargo, la transición a la nube no está exenta de desafíos. Consideraciones de seguridad, restricciones geográficas y legales, así como políticas internas de la organización, requieren una planificación meticulosa. Es fundamental abordar estos aspectos para garantizar una migración exitosa y segura.

Estrategias de Implementación

Para maximizar los beneficios de la migración a la nube, Stonebraker subraya la importancia de evitar la dependencia de características propietarias de los proveedores de

servicios en la nube. Se recomienda evaluar opciones como Platform as a Service (PaaS), donde se alquila la plataforma y se gestiona la instalación de instancias, o Database as a Service (DBaaS), donde se alquila el uso de la base de datos como servicio.

En cuanto a la selección de instancias en la nube, es crucial considerar cuidadosamente el tamaño y el tipo para satisfacer los requisitos específicos de cada proyecto. Se sugiere la posibilidad de iniciar con instancias de gran tamaño y, si es necesario, ajustar dinámicamente los recursos para optimizar el rendimiento.

Consideraciones de Rendimiento y Estrategias

Para mejorar el rendimiento en bases de datos alojadas en la nube, Stonebraker propone una serie de estrategias detalladas. Estas incluyen:

- Minimizar la interacción humana con la base de datos para evitar posibles errores humanos y garantizar la eficiencia en el acceso a los datos.
- Utilizar interfaces basadas en procedimientos almacenados para evitar el uso de cursores, lo que puede resultar en un acceso más eficiente y un procesamiento de datos más rápido.
- Modificar la aplicación para evitar la contención de control de concurrencia, lo que puede mejorar la eficiencia en situaciones de alta concurrencia.

Además de estas estrategias, es importante considerar optimizaciones a nivel de implementación del código, como la minimización de modificaciones de datos y la optimización de consultas para mejorar la eficiencia de la base de datos.

7. Bibliografía.

Cheuk:
Seguridad de bases de datos:

- 5411 - INTRODUCCIN A LA TECNOLOGA DE LAS B.D. (uv.es)
- Base de datos: qué es, para qué sirve, tipos, ejemplos... (ccm.net)
- Seguridad de las bases de datos: guía básica | IBM
- ¿Qué es la seguridad de datos? | Oracle España

Inyección SQL:

- Ataques de inyección SQL: tipos y consecuencias - Red Seguridad
- Qué es una inyección de SQL? | NordPass

Javi

- https://keepcoding.io/blog/tipos-de-inyeccion-sql/
- https://softwarelab.org/es/blog/que-es-la-inyeccion-sql/
- https://altenwald.org/2022/03/07/el-futuro-de-las-bases-de-datos/
- https://nordpass.com/es/blog/what-is-sql-injection/