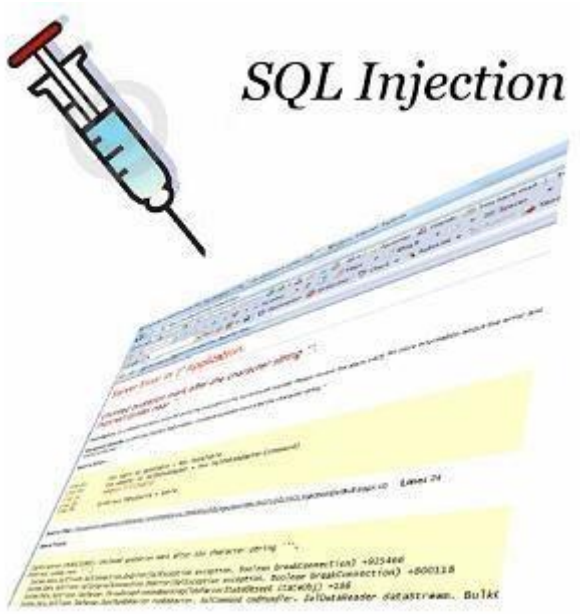
INYECCION SQL

NOMBRE SUSANA MAYTA HUANCA

YOSELIN KAREN MAMANI FLORES



OBJETIVOS

-El objetivo es aprender el concepto y manejo de forma de ataques.

- Aprender inyectar a un sistema.

- Aprender sus funciones para prevenir este tipo de ataques

INTRODUCCION

La mayoría de las aplicaciones web desarrolladas hoy en día hacen uso de una base de datos para ofrecer páginas dinámicas y almacenar información tanto de los usuarios como de la propia herramienta, datos a los que se accede por medio del lenguaje SQL, un lenguaje para interaccionar con bases de datos relacionales.

El uso de este tipo de lenguaje ha traído consigo la aparición de una de las vulnerabilidades más peligrosas a la que nos podemos enfrentar. Nos estamos refiriendo al ataque por inyección de código SQL, una vulnerabilidad que no sólo pone en riesgo la integridad de la aplicación, sino de todos los datos almacenados de los usuarios que la utilicen, y que se produce cuando no se filtra de forma correcta la información enviada por los usuarios.

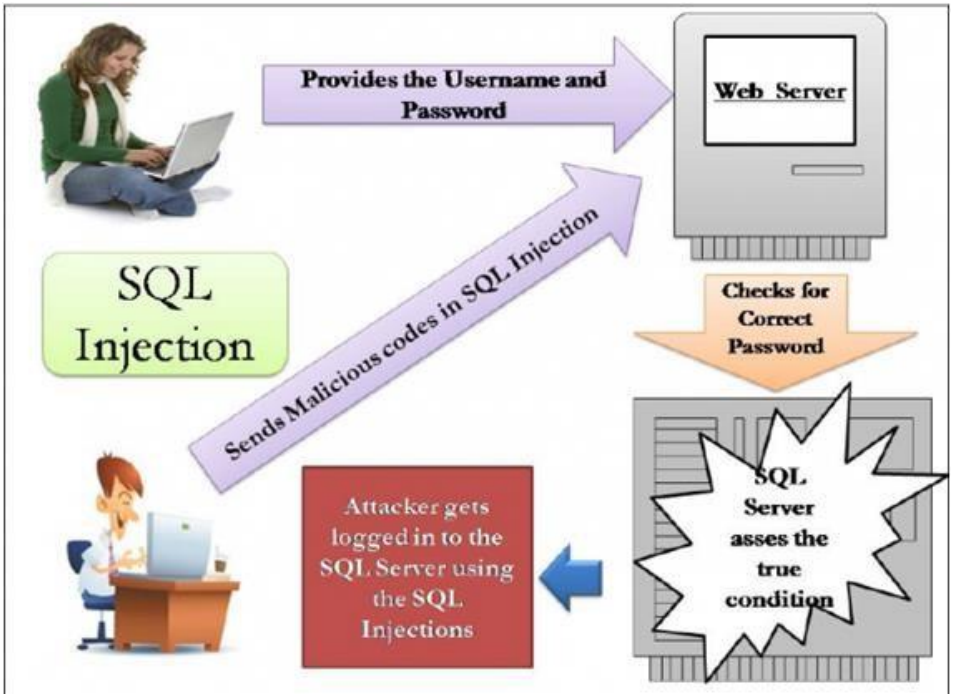
MARCO TEORICO

Consiste en la inserción de código SQL por medio de los datos de entrada desde la parte del cliente hacia la aplicación. Es decir, por medio de la inserción de este código el atacante puede modificar las consultar originales que debe realizar la aplicación y ejecutar otras totalmente distintas con la intención de acceder a

la herramienta, obtener información de alguna de las tablas o borrar los datos almacenados, entre otras muchas cosas.

Como consecuencias de estos ataques y dependiendo de los privilegios que tenga el usuario de la base de datos bajo el que se ejecutan las consultas, se podría acceder no sólo a las tablas relacionadas con la aplicación, sino también a otras tablas pertenecientes a otras bases de datos alojadas en ese mismo servidor.

Lo comentado anteriormente es posible gracias a que el uso de ciertos caracteres en los campos de entrada de información por parte del usuario, ya sea mediante el uso de los campos de los formularios que son enviados al servidor mediante POST o bien por medio de los datos enviados mediante GET en las urls de las páginas web, posibilitan coordinar varias consultas SQL o ignorar el resto de la consulta, permitiendo al hacker ejecutar la consulta que elija, de ahí que sea necesario realizar un filtrado de esos datos enviados para evitar problemas.



Dentro de este tipo de ataques, nos podemos encontrar el denominado **"Blind SQL Injection"**, traducido al español como "Ataque a ciegas por inyección de SQL", que se da cuando en una página web no aparece ningún mensaje de error al ejecutar una sentencia SQL errónea, por lo que el atacante va realizando pruebas hasta dar con el nombre de los campos o tablas sobre los que poder actuar.

Entre las bases de datos susceptibles a este tipo de ataques nos encontramos MySQL, Oracle, Postgres o MS SQL.

Principales problemas que causan los ataques SQL Injection

- **Confidencialidad**. De forma habitual, las bases de datos almacenan información sensible, por lo que la pérdida de confiabilidad es un problema muy frecuente en aquellos sitios que son vulnerables a este tipo de ataques.
- **Autenticación**. Si el sistema de logueo que se utiliza para acceder a una zona restringida de una web es débil, por medio de este tipo de ataques se podría acceder sin la necesidad de conocer ni el usuario ni la contraseña.
- **Integridad.** Al igual que un ataque por inyección SQL permite leer información relevante almacenada en la base de datos, también es posible realizar cambios o incluso borrar toda información mediante este tipo de vulnerabilidad.

FUNCIONAMIENTO DE ATAQUES DEL SQL INYECCION
Para explicar mejor el funcionamiento de este tipo de ataques, cogeremos el tipo formulario de logueo que existe prácticamente en toda página web y que puede ser utilizado por el atacante como punto de entrada a nuestra aplicación.

Como es lógico, en esta explicación supondremos que no se han tomado las medidas necesarias que evitarían que este tipo de ataques se pudiese llevar a cabo.

Cuando alguien rellena este formulario y pulsa el botón “Login”, “Acceder”, “Entrar” o similar, está enviando esa información a una sentencia SQL que será ejecutada contra la base de datos de la aplicación, una instrucción que podría ser similar a la siguiente.

```
$sql = SELECT * FROM usuarios WHERE usuario = '$usuario' and password =
'$pass';
```

Donde “$usuario” y “$pass” serían las variables que contendrían los valores introducidos por el visitante en cada uno de esos campos.

Un usuario común ante este tipo de formularios introduciría datos normales como “pepe” y “12345”, respectivamente, por lo que la consulta a ejecutar quedaría de la siguiente forma.

```
$sql = SELECT * FROM usuarios WHERE usuario = 'pepe' and password =
'12345';
```

Pero, ¿qué ocurriría si un atacante en el campo “password” agregara “' OR '1' = '1”? Es decir:

```
password = '12345' OR '1' = '1';
```

La respuesta es sencilla. El código marcado con el color rojo sería la inyección SQL, y ese simple código nos podría permitir acceder a aquellos sitios donde no se hubiesen tomado las medidas de seguridad necesarias. La consulta en este caso quedaría de la siguiente forma.

```
$sql= SELECT * FROM usuarios WHERE usuario = 'pepe' and password =
'12345' OR '1' = '1';
```

La instrucción anterior comprueba en la tabla usuarios si hay alguien registrado con nombre de usuario “pepe” y si su contraseña es “12345 o bien que 1 sea igual a 1”. Como esto segundo siempre es cierto, logaríamos entrar al sistema con el usuario “pepe” sin saber su verdadera contraseña.

En este punto quiero llamaros la atención sobre el papel tan importante que tienen las comillas simples. Nuestro código malicioso, el que hemos marcado con rojo, lo hemos empezado con una comilla simple. Con esto hemos cerrado el contenido de la variable “Password” y al terminar nuestro código malicioso no hemos puesto ninguna comilla simple porque suponemos que esa comilla de cierre será la que ponga por defecto el código de la aplicación.

Este ejemplo que hemos visto es uno de los más sencillos, pero la inyección de código SQL podría ser utilizada para insertar código más complejo que pudiese hacer otras funciones como listar datos o borrar datos.

PROTECCION O LA EVITACION DE ATAQUE INYECCION DE SQL

A la hora de desarrollar una aplicación, es muy complicado crear una herramienta totalmente segura a las primeras de cambio. La falta de tiempo y la intervención de varios programadores para su desarrollo, son factores que juegan en contra de la seguridad. A pesar de estos inconvenientes, siempre se pueden tomar medidas de seguridad que nos ayuden a desarrollar aplicaciones más robustas, ajenas a este tipo de problemas.

Veamos a continuación algunos consejos para evitar sufrir el ataque por inyección de código SQL en nuestros desarrollos:

a) Escapar los caracteres especiales utilizados en las consultas SQL

Al hablar de "escapar caracteres" estamos haciendo referencia a añadir la barra invertida "\" delante de las cadenas utilizadas en las consultas SQL para evitar que estas corrompan la consulta. Algunos de estos caracteres especiales que es aconsejable escapar son las comillas dobles ("), las comillas simples (') o los caracteres \x00 o \x1a ya que son considerados como peligrosos pues pueden ser utilizados durante los ataques.

Los distintos lenguajes de programación ofrecen mecanismos para lograr escapar estos caracteres. En el caso de PHP podemos optar por la función mysql_real_scape_string(), que toma como parámetro una cadena y la modifica evitando todos los caracteres especiales, devolviéndola totalmente segura para ser ejecutada dentro de la instrucción SQL.

b) Delimitar los valores de las consultas

Aunque el valor de la consulta sea un entero, es aconsejable delimitarlo siempre entre comillas simples. Una instrucción SQL del tipo:

```
SELECT nombre FROM usuarios WHERE id_user = $id
```

Será mucho más fácilmente inyectable que:

```
SELECT nombre FROM usuarios WHERE id_user = '$id'
```

Donde $id es un número entero.

c) Verificar siempre los datos que introduce el usuario

Si en una consulta estamos a la espera de recibir un entero, no confiemos en que sea así, sino que es aconsejable tomar medidas de seguridad y realizar la comprobación de que realmente se trata del tipo de dato que estamos esperando. Para realizar esto, los lenguajes de programación ofrecen funciones que realizan esta acción, como pueden ser `ctype_digit()` para saber si es un número o `ctype_alpha ()` para saber si se trata de una cadena de texto en el caso del lenguaje PHP.

También es aconsejable comprobar la longitud de los datos para descartar posibles técnicas de inyección SQL, ya que si por ejemplo estamos esperando un nombre, una cadena extremadamente larga puede suponer que estén intentando atacarnos por este método. En el caso de PHP, podemos utilizar la función `strlen()` para ver el tamaño de la cadena.

d) Asignar mínimos privilegios al usuario que conectará con la base de datos

El usuario que utilicemos para conectarnos a la base de datos desde nuestro código debe tener los privilegios justos para realizar las acciones que necesitemos. No utilizar nunca un usuario root con acceso a todas las bases de datos ya que de esta forma estaremos dando facilidades a los hackers para que puedan acceder a toda la información.

e) Programar bien

Aunque pueda parecer una tontería, no hay mejor medida para evitar este tipo de ataques que realizar una buena programación, poniendo en práctica las necesidades básicas y el interés para desarrollar una aplicación totalmente segura.

Además de las medidas que podemos tomar a la hora de implementar el código, siempre podemos acudir a auditorías de código para asegurarnos de que no hemos dejado ningún tipo de puertas abiertas, aunque suelen ser procesos caros realizados por terceras empresas.

HERRAMIENTA PARA SQL INYECION

Otra de las opciones que podemos utilizar para realizar un análisis de nuestro código es el uso de herramientas que testen nuestras aplicaciones en busca de vulnerabilidades por inyección SQL. Algunas de estas herramientas son:

- **SQLiHelper 2.7 SQL Injection**: Se trata de una aplicación cuyo objetivo es facilitar la extracción de información procedente de bases de datos utilizando para ello técnicas de inyección SQL. Una vez indicada la url que queremos analizar, la aplicación realizará peticiones inyectando código SQL con el fin de comprobar si es realmente vulnerable.

- **Pangolin**: Se trata de una herramienta de pago que ofrece más posibilidades que la vista en el punto anterior y que está destinada a descubrir vulnerabilidades tanto del tipo inyección SQL como inyección SQL ciego.

- **SQLMap**: Se trata de una herramienta de pruebas de código abierto que automatiza el proceso de detectar y explorar los errores de inyección SQL

Para finalizar, os dejamos esta tira gráfica donde se representa en clave de humor los problemas que puede llegar a provocar la aaaaqa de código SQL, una tira gráfica que hemos visto en el portal XKCD :) Y unos enlaces de ayuda en inglés con más información sobre este tipo de problema.

CONCLUSION

En conclusión aprendimos a inyectar a un sistemas mediantes el sql que es el código a otro código pero todo es conectado con el base de datos tuvimos problemas para poner las funciones que evitamos como el string para los caracteres cambiara para que el procedimiento de la inyecion no se ejecute en fin aprendimos mucho sobre la inyeccion sql.