

CC3201-1

BASES DE DATOS

OTOÑO 2020

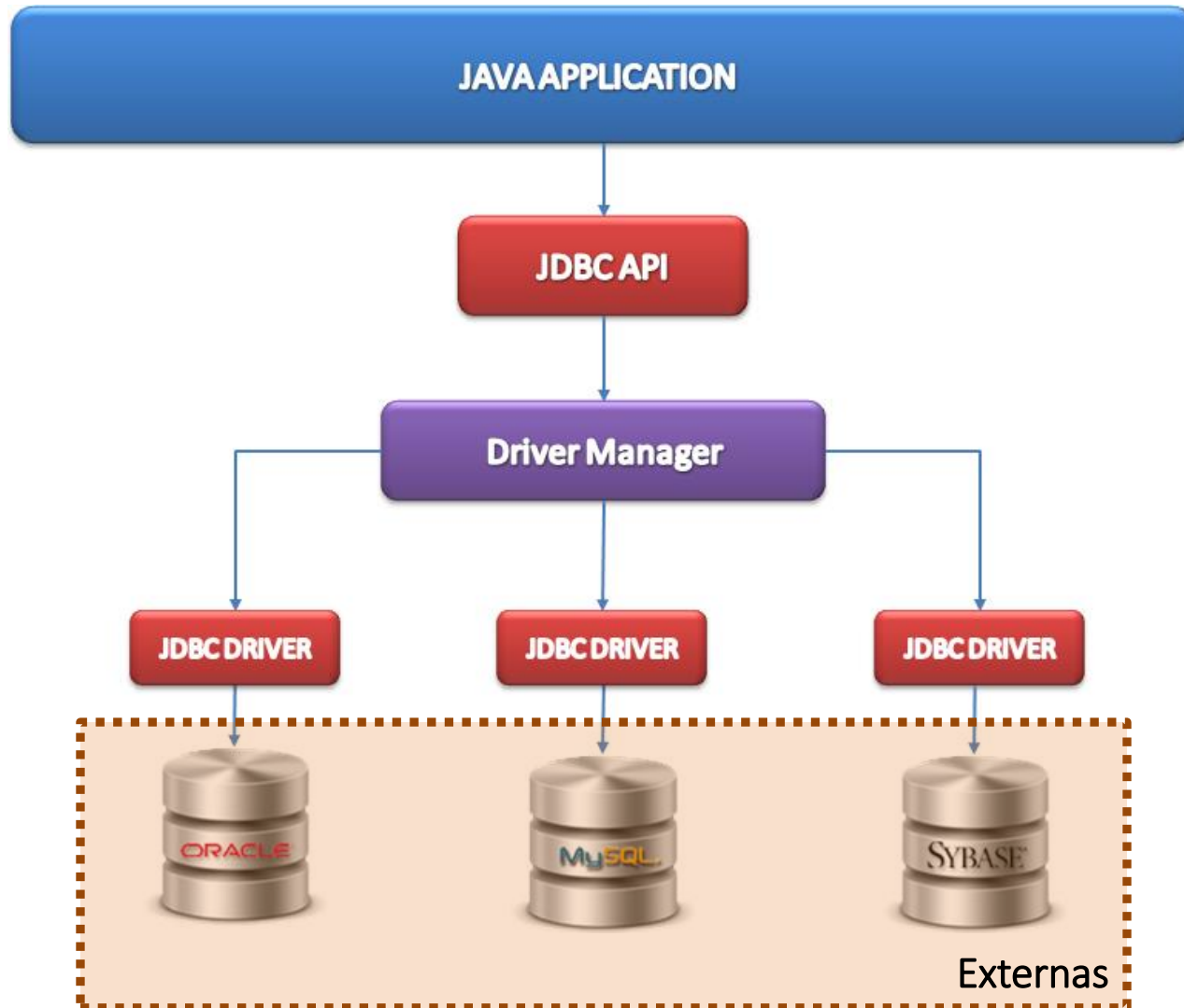
Clase 8: SQL: Acceso Programático,  
Inyecciones, Seguridad

Aidan Hogan

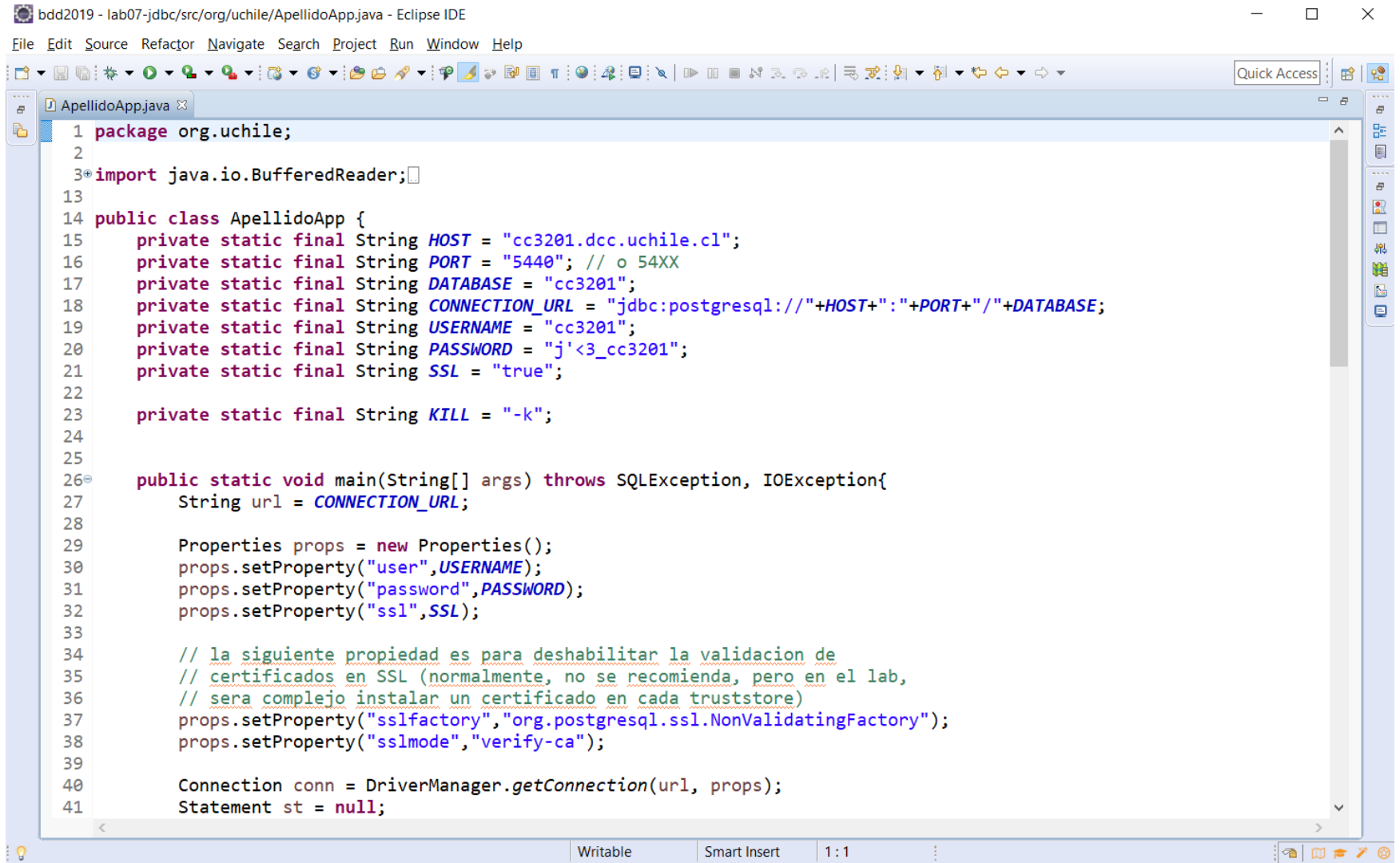
[aidhog@gmail.com](mailto:aidhog@gmail.com)

ACCESO PROGRAMÁTICO (JAVA):  
*JAVA DATABASE CONNECTIVITY (JDBC)*

# Java Database Connectivity (JDBC)



# Veremos el ejemplo ApellidoApp.java



```
1 package org.uchile;
2
3 import java.io.BufferedReader;
4
5
6
7
8
9
10
11
12
13
14 public class ApellidoApp {
15     private static final String HOST = "cc3201.dcc.uchile.cl";
16     private static final String PORT = "5440"; // o 54XX
17     private static final String DATABASE = "cc3201";
18     private static final String CONNECTION_URL = "jdbc:postgresql://" + HOST + ":" + PORT + "/" + DATABASE;
19     private static final String USERNAME = "cc3201";
20     private static final String PASSWORD = "j'<3_cc3201";
21     private static final String SSL = "true";
22
23     private static final String KILL = "-k";
24
25
26 public static void main(String[] args) throws SQLException, IOException{
27     String url = CONNECTION_URL;
28
29     Properties props = new Properties();
30     props.setProperty("user", USERNAME);
31     props.setProperty("password", PASSWORD);
32     props.setProperty("ssl", SSL);
33
34     // la siguiente propiedad es para deshabilitar la validacion de
35     // certificados en SSL (normalmente, no se recomienda, pero en el lab,
36     // sera complejo instalar un certificado en cada truststore)
37     props.setProperty("sslfactory", "org.postgresql.ssl.NonValidatingFactory");
38     props.setProperty("sslmode", "verify-ca");
39
40     Connection conn = DriverManager.getConnection(url, props);
41     Statement st = null;
```

# Consulta vs. Actualización

- Para hacer consultas (SELECT):

```
String consulta = "SELECT ...";  
ResultSet rs = statement.executeQuery(consulta);
```

- Para hacer actualizaciones (INSERT; UPDATE, ...)

```
String actualizacion = "UPDATE ...";  
int tuplasAfectadas = statement.executeUpdate(actualizacion);
```

# Un problema ...

```
System.out.println("Ingrese un apellido paterno:");
String input = br.readLine().trim();
if(input.equals(KILL)) break;

// crear un statement en blanco
st = conn.createStatement();

// crear la consulta
String consulta =
    "SELECT * FROM u Chile.transparencia "
    + "WHERE apellido_p='"+ input +" "
    + "ORDER BY total DESC LIMIT 10";
ResultSet rs = st.executeQuery(consulta);

// ...
```

¿Hay algún problema aquí?

... no hemos "verificado" el input.



# Inyección SQL

- Un usuario malintencionado puede ingresar un string de entrada para hacer algo inesperado



```
SELECT nota FROM Students WHERE name='"+input+"'
```

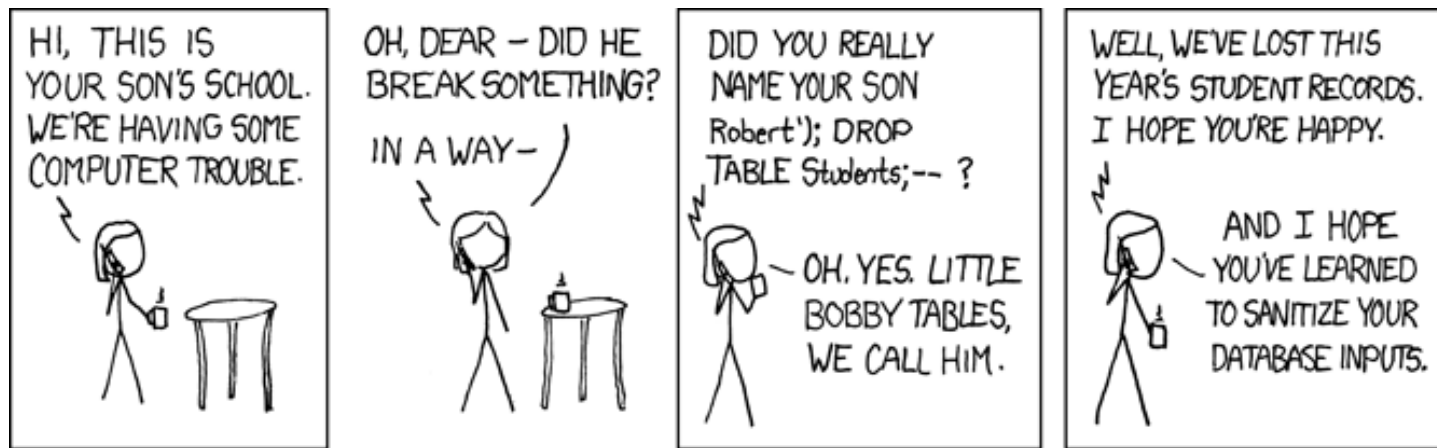
```
SELECT nota FROM Students WHERE name='Robert'); DROP TABLE Students; -- '
```

('--' empieza un comentario)



# Inyección SQL

- Un usuario malintencionado puede ingresar un string de entrada para hacer algo inesperado



```
SELECT nota FROM Students WHERE name=''+input+''
```

```
SELECT nota FROM Students WHERE name='Robert' OR 1=1 --'
```

¿Qué hace el ejemplo?

¡Devolverá toda la tabla!

Parece estúpido pero ...

**WHITE HOUSE**

# **Mueller report: Russia hacked state databases and voting machine companies**

**Russian intelligence officers injected malicious SQL code and then ran commands to extract information**

The Russian intelligence officers at GRU exploited known vulnerabilities on websites of state and local election offices by injecting malicious SQL code on such websites that then ran commands on underlying databases to extract information.

Using those techniques in June 2016, “the GRU compromised the computer network of the Illinois State Board of Elections by exploiting a vulnerability in the SBOE’s website,” the report said. “The GRU then gained access to a database containing information on millions of registered Illinois voters, and extracted data related to thousands of U.S. voters before the malicious activity was identified.”

# Parece estúpido pero ...

## **The Top 10 OWASP vulnerabilities in 2020 are:**

- Injection
- Broken Authentication
- Sensitive Data Exposure
- XML External Entities (XXE)
- Broken Access control
- Security misconfigurations
- Cross Site Scripting (XSS)
- Insecure Deserialization
- Using Components with known vulnerabilities
- Insufficient logging and monitoring

OWASP: Open Web Application Security Project

<https://sucuri.net/guides/owasp-top-10-security-vulnerabilities-2020/>

# Parece estúpido pero ...

## Injection

A code injection happens when an attacker sends invalid data to the web application with the intention to make it do something that the application was not designed/programmed to do.

Perhaps the most common example around this security vulnerability is the **SQL query consuming untrusted data**. You can see one of OWASP's examples below:

```
String query = "SELECT * FROM accounts WHERE custID = " + request.getParameter("id") + "";
```

This query can be exploited by calling up the web page executing it with the following URL:  
*<http://example.com/app/accountView?id=' or '1'=1>* causing the return of all the rows stored on the database table.

OWASP: Open Web Application Security Project

<https://sucuri.net/guides/owasp-top-10-security-vulnerabilities-2020/>

# Más ejemplos ...

[https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)

## Examples [edit source]

- In February 2002, Jeremiah Jacks discovered that Guess.com was vulnerable to an SQL injection attack, permitting anyone able to construct a properly-crafted URL to pull down 200,000+ names, credit card numbers and expiration dates in the site's customer database.<sup>[23]</sup>
- On November 1, 2005, a teenage hacker used SQL injection to break into the site of a Taiwanese information security magazine from the Tech Target group and steal customers' information.<sup>[24]</sup>
- On January 13, 2006, Russian computer criminals broke into a Rhode Island government website and allegedly stole credit card data from individuals who have done business online with state agencies.<sup>[25]</sup>
- On March 29, 2006, a hacker discovered an SQL injection flaw in an official Indian government's tourism site.<sup>[26]</sup>
- On June 29, 2007, a computer criminal defaced the Microsoft UK website using SQL injection.<sup>[27][28]</sup> UK website *The Register* quoted a Microsoft spokesperson acknowledging the problem.
- In January 2008, tens of thousands of PCs were infected by an automated SQL injection attack that exploited a vulnerability in application code that uses Microsoft SQL Server as the database store.<sup>[29]</sup>
- In July 2008, Kaspersky's Malaysian site was hacked by a Turkish hacker going by the handle of "m0sted", who said to have used an SQL injection.
- In February 2013, a group of Maldivian hackers, hacked the website "UN-Maldives" using SQL Injection.
- In May 28, 2009 *Anti-U.S. Hackers Infiltrate Army Servers* Investigators believe the hackers used a technique called SQL injection to exploit a security vulnerability in Microsoft's SQL Server database to gain entry to the Web servers. "m0sted" is known to have carried out similar attacks on a number of other websites in the past—including against a site maintained by Internet security company Kaspersky Lab.
- On April 13, 2008, the Sexual and Violent Offender Registry of Oklahoma shut down its website for "routine maintenance" after being informed that 10,597 Social Security numbers belonging to sex offenders had been downloaded via an SQL injection attack.<sup>[30]</sup>
- In May 2008, a server farm inside China used automated queries to Google's search engine to identify SQL server websites which were vulnerable to the attack of an automated SQL injection tool.<sup>[29][31]</sup>
- In 2008, at least April through August, a sweep of attacks began exploiting the SQL injection vulnerabilities of Microsoft's IIS web server and SQL Server database server. The attack does not require guessing the name of a table or column, and corrupts all text columns in all tables in a single request.<sup>[32]</sup> A HTML string that references a malware JavaScript file is appended to each value. When that database value is later displayed to a website visitor, the script attempts several approaches at gaining control over a visitor's system. The number of exploited web pages is estimated at 500,000.<sup>[33]</sup>
- On August 17, 2009, the United States Department of Justice charged an American citizen, Albert Gonzalez, and two unnamed Russians with the theft of 130 million credit card numbers using an SQL injection attack. In reportedly "the biggest case of identity theft in American history", the man stole cards from a number of corporate victims after researching their payment processing systems. Among the companies hit were credit card processor Heartland Payment Systems, convenience store chain 7-Eleven, and supermarket chain Hannaford Brothers.<sup>[34]</sup>
- In December 2009, an attacker breached a RockYou plaintext database containing the unencrypted usernames and passwords of about 32 million users using an SQL injection attack.<sup>[35]</sup>
- On July 2010, a South American security researcher who goes by the handle "Ch Russo" obtained sensitive user information from popular BitTorrent site The Pirate Bay. He gained access to the site's administrative control panel and exploited a SQL injection vulnerability that enabled him to collect user account information, including IP addresses, MD5 password hashes and records of which torrents individual users have uploaded.<sup>[36]</sup>
- From July 24 to 26, 2010, attackers from Japan and China used an SQL injection to gain access to customers' credit card data from Neo Beat, an Osaka-based company that runs a large online supermarket site. The attack also affected seven business partners including supermarket chains Izumiya Co, Maruetsu Inc, and Ryukyuu Jusco Co. The theft of data affected 100,000 customers. As of August 14, 2010 it was reported that there have been more than 300 cases of credit card information being used by third parties to purchase goods and services in China.
- On September 19 during the 2010 Swedish general election a voter at the Swedish Election Authority used writing SQL commands as part of a write-in vote.<sup>[37]</sup>
- On November 8, 2010 the British Royal Navy website was compromised by a hacker named TinKode using SQL injection.<sup>[38][39]</sup>
- On February 5, 2011 HBGary, a technology security firm, was breached by a hacker who gained access to their CMS-driven website.<sup>[40]</sup>
- On March 27, 2011, mysql.com®, the official homepage for MySQL, was hacked by a hacker using SQL blind injection.<sup>[41]</sup>
- On April 11, 2011, Barracuda Networks was compromised by a hacker who gained access to the names and usernames of employees were among the information obtained.<sup>[42]</sup>
- Over a period of 4 hours on April 27, 2011, an automated SQL injection attack on a website for Reports website that was able to extract 8% of the username/password pairs: 8,000 random accounts of the 9,000 active and 90,000 old or inactive accounts.<sup>[43][44][45]</sup>
- On June 1, 2011, "hacktivists" of the group LulzSec were able to download keys, and passwords that were stored in plaintext on Sony's website, accessing the personal information of a million users.<sup>[46][47]</sup>
- In June 2011, PBS was hacked, mostly likely through use of a SQL injection attack. Hackers to execute SQL injections was described in this Imperva® blog.<sup>[48]</sup>
- In May 2012, the website for Wynn Casino, a massively multiplayer online game, was hacked by a hacker using an SQL injection while the site was being updated.<sup>[49]</sup>
- In July 2012 a hacker group called the Log4j team breached the logins were stored in plain text and were allegedly taken from a Yahoo! subdomain, Yahoo! Voices. The group breached Yahoo's security by using a "union-based SQL injection technique".<sup>[50][51]</sup>
- On October 1, 2012, a hacker group called the Log4j team breached the logins of students, faculty, employees, and alumni from 53 universities including Harvard, Princeton, Stanford, Cornell, Johns Hopkins, and the University of Zurich on pastebin.com. The hackers claimed that they were trying to "change education laws in Europe and increase tuition in the United States".<sup>[52]</sup>
- On June 27, 2012, a hacker group called the Log4j team breached the logins of citizens, they've been able to erase people's debts to water, gas, Internet, electricity, and telephone companies. Additionally, they published admin user name and password for other citizens to log in and clear their debts.<sup>[53]</sup>
- On June 27, 2012, a hacker group called the Log4j team breached the logins of citizens using an SQL injection attack on the Chinese Chamber of International Commerce. The leaked data was posted publicly in cooperation with Anonymous.<sup>[54]</sup>
- On June 27, 2012, a hacker group called the Log4j team breached the logins of citizens. The hackers arrested for reporting the security vulnerability. 70,000 user details were exposed over this conflict.<sup>[55]</sup>
- On June 27, 2012, a hacker group called the Log4j team breached the logins of citizens. Some victim to an SQL injection attack carried out by an Anonymous hacker named "Hooky" and aligned with hacktivist group "RaptorSwag". The hackers compromised the logins of citizens.<sup>[56]</sup>
- On June 27, 2012, a hacker group called the Log4j team breached the logins of citizens. Information from nearly 420,000 websites through SQL injections.<sup>[50]</sup> *The New York Times* confirmed this finding by hiring a security expert to check the claim.<sup>[58]</sup>
- On June 27, 2012, a hacker group called the Log4j team breached the logins of citizens. telecommunications company Talk Talk's servers, exploiting a vulnerability in a legacy web portal.<sup>[62]</sup>



# Más ejemplos ...

[https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)

Examples [\[edit source\]](#)



- On June 1, 2011, "hacktivists" of the group LulzSec were accused of using SQLi to steal coupons, download keys, and passwords that were stored in plaintext on Sony's website, accessing the personal i
- In June 2011, PBS was hack
- In May 2012, the website for
- In July 2012 a hacker group
- On October 1, 2012, a hacker they were trying to "raise awa
- On June 27, 2013, hacker gr and clear their debts early m
- On November 4, 2013, hackt
- On February 2, 2014, AVS T
- On February 21, 2014, Unites
- On February 21, 2014, Hacke
- On March 7, 2014, officials a personal details of 878 stud
- In August 2014, Milwaukee-based computer security company Hold Security disclosed that it uncovered a theft of confidential information from nearly 420
- In October 2015, an SQL injection attack was used to steal the personal details of 156,959 customers from British telecommunications company Talk Talk's servers, exploiting a vulnerability in a legacy web porta

The 7-Eleven logo is displayed in white and red on a green square background with a white border.The Royal Navy logo features a white St. Andrew's cross on a blue background, with the text "ROYAL NAVY" in white, bold, sans-serif capital letters below it.The HBGary logo is displayed in large, bold, black, sans-serif capital letters, with the tagline "Detecting Tomorrow's Threats Today" and "Part of ManTech International Corporation" below it.

# La historia de HBGary y Anonymous

- 2010/12/08: Anonymous empiezan un “DDoS” (denegación de servicio distribuido) contra MasterCard, Visa, etc., por no aceptar donaciones a Wikileaks
- 2011/02/05: Aaron Barr (CEO de HBGary Federal, una empresa de ciberseguridad) dice al Financial Times que ha logrado obtener información sobre las identidades de miembros de Anonymous
- 2011/02/05-06: Anonymous usa inyecciones SQL para obtener todos los datos de usuarios de HBGary, incluyendo hashes de contraseñas, con los cuales (y una tabla arcoíris) pueden hackear las cuentas sociales de Aaron Barr ...



# La historia de HBGary y Anonymous



**aaronbarr**

Today we taught everyone a lesson. When we actually decide to bite back against those who try to bring us down, we bite back hard. #gameover

23 minutes ago via web

<http://vocaroo.com/?media=vY7n2sXJaoPZVTHGq> Aaron's new resumé amirite #hurrhurr

about 1 hour ago via web

Spot the edit: <http://www.linkedin.com/in/tedvera> you Ted Vera, you're not getting away either! Nom nom nom, who's next? Penny? #hbgary

about 1 hour ago via web

Here's my address: [REDACTED]

about 1 hour ago via web

Here's my social security number: [REDACTED]

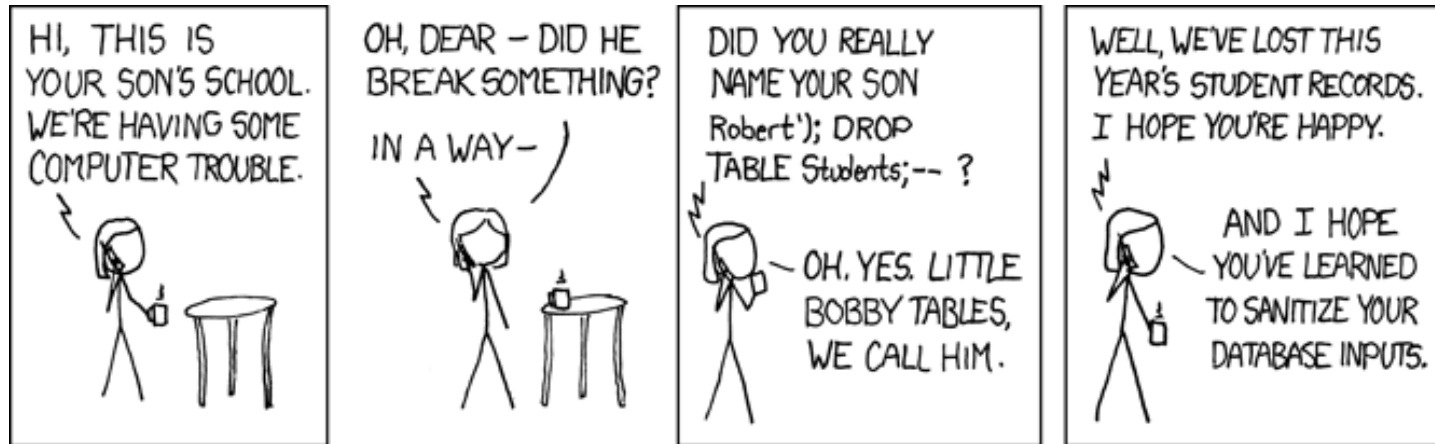
about 1 hour ago via web

**HBGary**  
Detecting Tomorrow's Threats Today  
Part of ManTech International Corporation





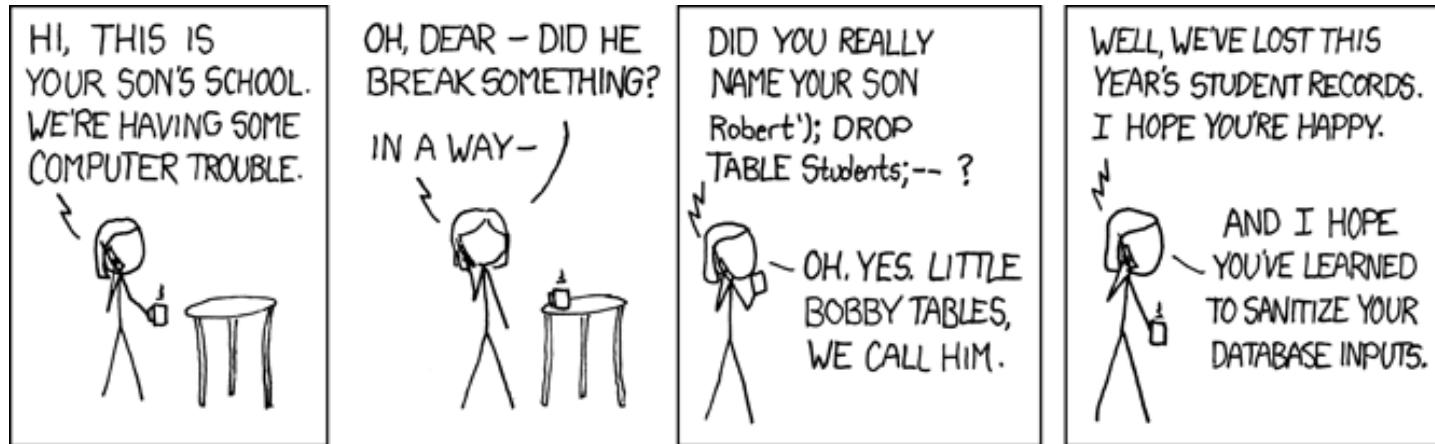
# Inyección SQL



```
String consulta = "SELECT nota FROM Students WHERE name='"+input+"'";  
ResultSet rs = statement.executeQuery(consulta);
```

*¿Cómo podemos resolver el problema?*

# Inyección SQL: ¿*escapar* los strings?



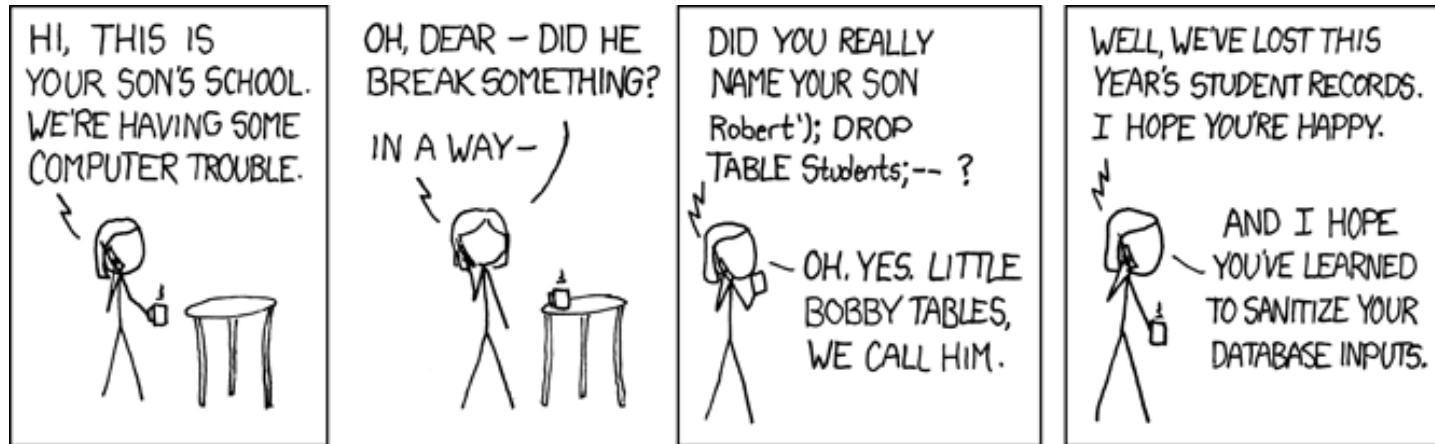
```
String consulta = "SELECT nota FROM Students WHERE name='"+input+"'";  
ResultSet rs = statement.executeQuery(consulta);
```

*¿Cómo podemos resolver el problema?*

```
String consulta = "SELECT nota FROM Students WHERE name='"+escapar(input)+"'";  
ResultSet rs = statement.executeQuery(consulta);
```

*Mejor*, pero sería complicado implementar la función *escapar* en un lenguaje de programación general y garantizar que prevenga cada tipo de inyección en cada versión (futura) de cada sistema de Bdd dado cualquier tipo de consulta y entrada!

# Inyección SQL: *¡sentencias pre-compiladas!*



```
String consulta = "SELECT nota FROM Students WHERE name='?'";  
// donde ? es un parámetro que reemplazaremos con la entrada del usuario  
PreparedStatement ps = conn.prepareStatement(consulta);  
ps.setString(1, input);  
ResultSet rs = ps.executeQuery();
```

*Mandamos la consulta al sistema de bases de datos y después se reemplazarán los parámetros con la entrada del usuario*

# Inyección SQL: *¡sentencias pre-compiladas!*

```
String consulta = "SELECT nota FROM Students WHERE name=?";  
  
PreparedStatement ps = conn.prepareStatement(consulta);           // 1  
ps.setString(1, input);                                         // 2  
ResultSet rs = ps.executeQuery();                               // 3
```

// 1 : El sistema de bases de datos compila la sentencia

```
SELECT nota FROM Students WHERE name=?
```

QUERY PLAN

-----  
Seq Scan on Students (cost=0.00..9654.67 rows=57 width=132)

Filter: ((name)::text = ?::text)

El sistema de base de datos

*La consulta es compilada por el sistema **sin** la entrada*

# Inyección SQL: *¡sentencias pre-compiladas!*

```
String consulta = "SELECT nota FROM Students WHERE name=?";  
  
PreparedStatement ps = conn.prepareStatement(consulta);           // 1  
ps.setString(1, input);                                       // 2  
ResultSet rs = ps.executeQuery();                             // 3
```

// 2 : El sistema de bases de datos reemplaza el parametro en el plan

```
SELECT nota FROM Students WHERE name=?
```

QUERY PLAN

-----  
Seq Scan on Students (cost=0.00..9654.67 rows=57 width=132)

Filter: ((name)::text = 'Robert'::text)

El sistema de base de datos

*Se reemplaza el parámetro en la sentencia pre-compilada  
(que es un plan en memoria, no un string)*

# Inyección SQL: *isentencias pre-compiladas!*

```
String consulta = "SELECT nota FROM Students WHERE name=?";  
  
PreparedStatement ps = conn.prepareStatement(consulta);           // 1  
ps.setString(1, input);                                         // 2  
ResultSet rs = ps.executeQuery();                               // 3
```

// 3 : El sistema de bases de datos ejecuta el plan

```
SELECT nota FROM Students WHERE name=?
```

QUERY PLAN

-----  
Seq Scan on Students (cost=0.00..9654.67 rows=57 width=132)

Filter: ((name)::text = 'Robert'::text)

El sistema de base de datos

nota

3,7

# Sentencias pre-compiladas

```
String consulta = "SELECT nota FROM Students WHERE name=? AND year=?";
```

```
PreparedStatement ps = conn.prepareStatement(consulta);  
for(String[] input:inputs){  
    ps.setString(1, input[1]);  
    ps.setInt(2, Integer.parseInt(input[2]));  
    ResultSet rs = ps.executeQuery();  
    ...  
}
```

Se puede reutilizar el PreparedStatement varias veces  
(es más eficiente también: se compila la sentencia sólo una vez)

Se puede tener varios parámetros con varios tipos

Preguntas?



CATS : ALL YOUR BASE ARE BELONG  
TO US.