

SQL Injection

By Artem Kazanstev, ITSO and Alex Beutel, Student

SANS Priority No 2

- ▶ As of September 2009, “Web application vulnerabilities such as SQL injection and Cross-Site Scripting flaws in open-source as well as custom-built applications account for more than 80% of the vulnerabilities being discovered.”

<http://www.sans.org/top-cyber-security-risks/>



In the news..

- ▶ Recent breach: 130 million credit and debit card numbers from five companies were stolen using SQL injection attack to bypass firewalls:

“The Department of Justice issued a statement today about the indictment, which accuses Albert Gonzalez, 28, and two unnamed Russian citizens of stealing data from Heartland Payment Systems Inc., 7-Eleven Inc. and Hannaford Brothers Co. Two other companies remain unnamed because their breaches have not been made public, the DOJ said.”

<http://preview.tinyurl.com/m6cu6n> (techtarget.com)



Why are we vulnerable?

- ▶ “The risk of SQL injection exploits is on the rise because of automated tools. In the past, the danger was somewhat limited because an exploit had to be carried out manually: an attacker had to actually type their SQL statement into a text box. However, automated SQL injection programs are now available, and as a result, both the likelihood and the potential damage of an exploit has increased enormously.”

<http://preview.tinyurl.com/ycucfm4> (Techtarget.com)



SQL injections

Segment Group: inbound

Filter No.: SQL Injections

Action Type: **Blocks**

Severity: All

From: Tue Sep 01 00:00:00 EDT 2009

To: Thu Oct 01 00:00:00 EDT 2009

Filter Name	Filter No	Source IP	Dest IP	Severity
5389: HTTP: SQL Injection Evasion (MySQL Functions)	5389	81.27.33.240	152.3.140.5	Major
5389: HTTP: SQL Injection Evasion (MySQL Functions)	5389	117.104.162.225	152.3.140.5	Major
5413: HTTP: WordPress SQL Injection Vulnerability	5413	66.63.167.50	152.3.198.32	Critical
5389: HTTP: SQL Injection Evasion (MySQL Functions)	5389	85.13.135.192	152.3.196.58	Major
5389: HTTP: SQL Injection Evasion (MySQL Functions)	5389	82.150.137.70	152.3.196.58	Major
5413: HTTP: WordPress SQL Injection Vulnerability	5413	69.65.40.176	152.3.56.29	Critical
5413: HTTP: WordPress SQL Injection Vulnerability	5413	69.65.40.176	152.3.8.134	Critical
5389: HTTP: SQL Injection Evasion (MySQL Functions)	5389	74.108.14.121	152.3.58.179	Major

5389: HTTP: SQL Injectio
Major

5389: HTTP: SQL Injectio
Major

5389: HTTP: SQL Injectio
Major

5389: HTTP: SQL Injectio
Major

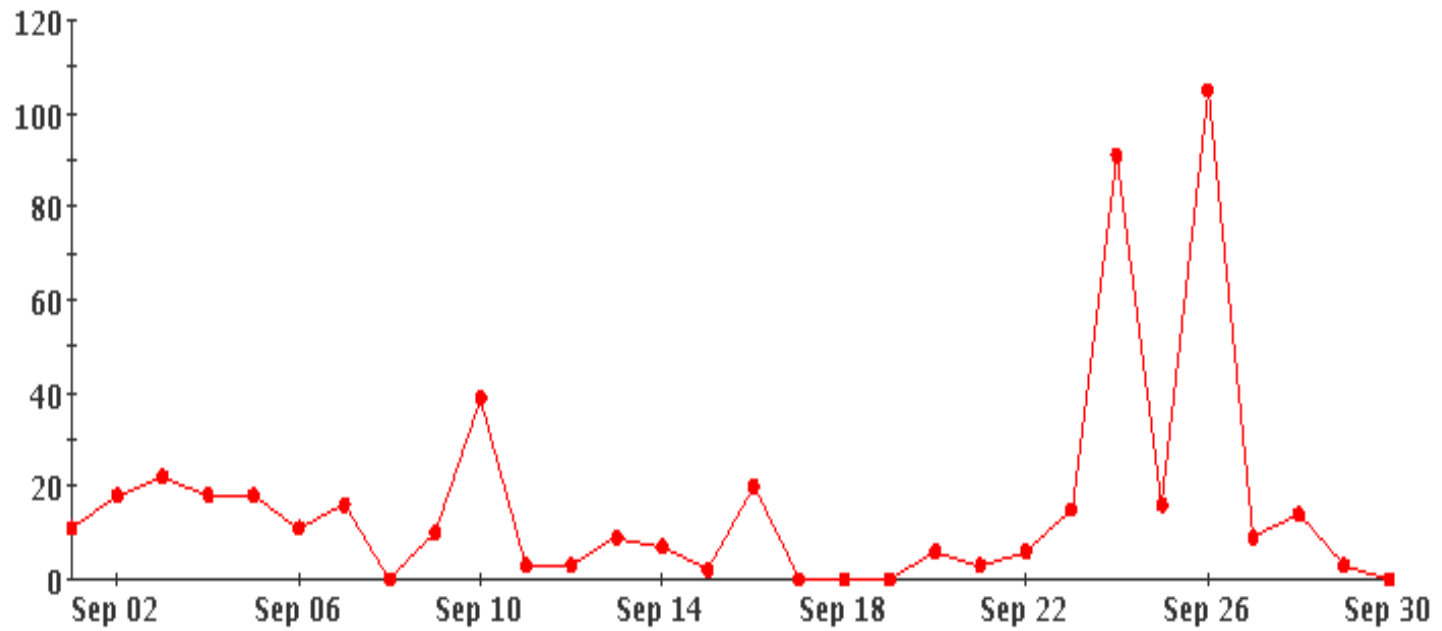
5389: HTTP: SQL Injectio
Major

5389: HTTP: SQL Injectio
Major

5389: HTTP: SQL Injectio
Major

5389: HTTP: SQL Injectio
Major

.....



The Potential Attack Surface

- ▶ The number of database-driven websites in datacenters only: around 400



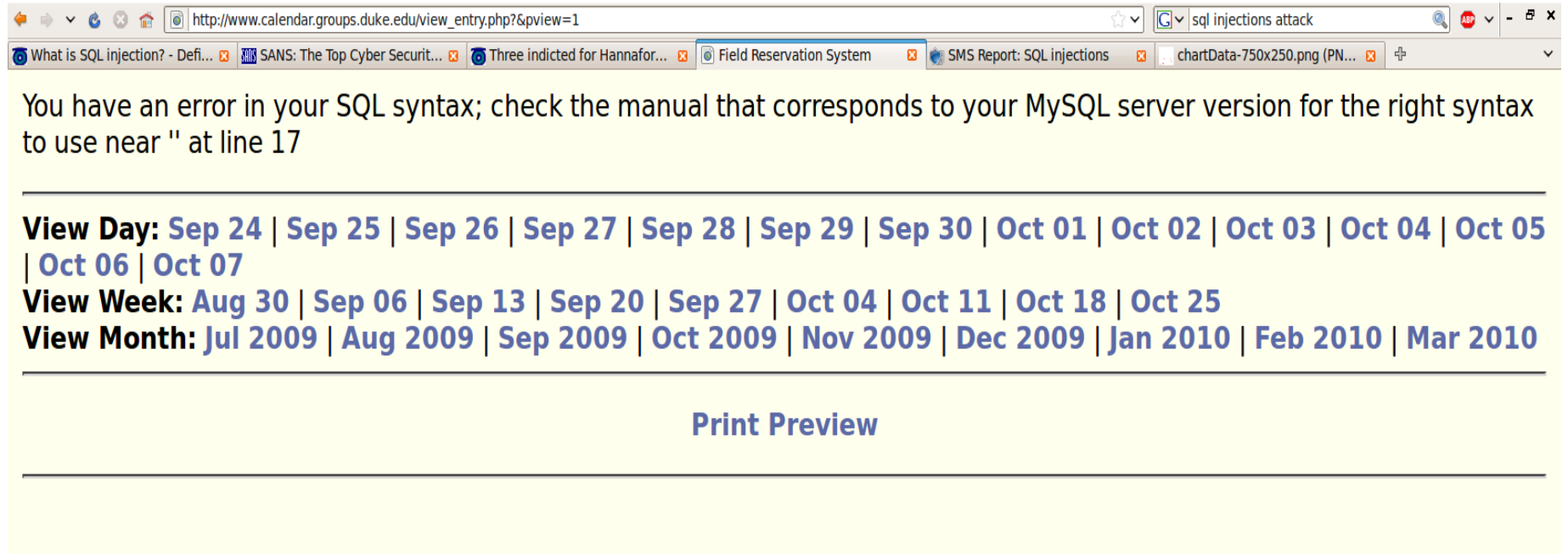
Why are we vulnerable?

- ▶ “Programmers who aren't security savvy are coding SQL injection as a feature in some Web applications, putting users at risk when an application goes live or is distributed to affiliates of online advertising networks.”

<http://preview.tinyurl.com/mv5tw8>



Outdated and orphan apps



The screenshot shows a web browser window with the address bar containing `http://www.calendar.groups.duke.edu/view_entry.php?&pview=1`. The browser has several tabs open, including "What is SQL injection?", "SANS: The Top Cyber Securit...", "Three indicted for Hannafor...", "Field Reservation System", "SMS Report: SQL injections", and "chartData-750x250.png (PN...". The main content area displays an error message: "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 17". Below the error message, there are navigation links for "View Day", "View Week", and "View Month", each followed by a list of dates. At the bottom of the content area, there is a "Print Preview" link.

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 17

View Day: [Sep 24](#) | [Sep 25](#) | [Sep 26](#) | [Sep 27](#) | [Sep 28](#) | [Sep 29](#) | [Sep 30](#) | [Oct 01](#) | [Oct 02](#) | [Oct 03](#) | [Oct 04](#) | [Oct 05](#) | [Oct 06](#) | [Oct 07](#)

View Week: [Aug 30](#) | [Sep 06](#) | [Sep 13](#) | [Sep 20](#) | [Sep 27](#) | [Oct 04](#) | [Oct 11](#) | [Oct 18](#) | [Oct 25](#)

View Month: [Jul 2009](#) | [Aug 2009](#) | [Sep 2009](#) | [Oct 2009](#) | [Nov 2009](#) | [Dec 2009](#) | [Jan 2010](#) | [Feb 2010](#) | [Mar 2010](#)

[Print Preview](#)



SANS Courses

- ▶ 2-4 Day Developer Courses
- ▶ 538 Web Application Pentesting Hands-On Immersion
- ▶ 536 Secure Coding for PCI Compliance
- ▶ 530 Essential Secure Coding in Java/JEE
- ▶ 548 Secure Coding in C: Developing Defensible Applications
- ▶ 544 Secure Coding in .NET: Developing Defensible Applications
- ▶ 545 Secure Coding in PHP: Developing Defensible Applications
- ▶ 422 Defending Web Applications Security Essentials
- ▶ 541 Secure Coding in Java/JEE: Developing Defensible Applications

<http://www.sans.org/security-training/courses.php#developer>



What is SQL?

- ▶ Structured Query Language
- ▶ Almost all modern web applications use a database backend to store data
- ▶ Majority of databases use a variant of SQL as a querying language to retrieve information from the database
 - ▶ MySQL, MSSQL, Postgresql, DB 2
- ▶ Flexible and robust but if not carefully implemented can make a server extremely vulnerable.

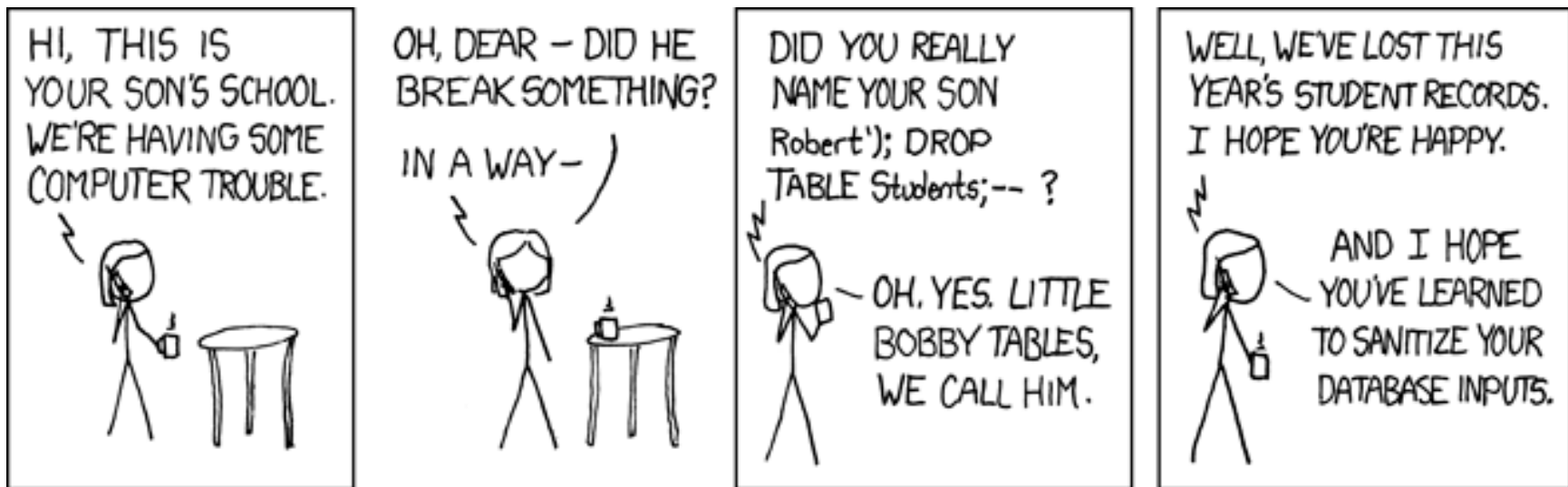


SQL Injection

- ▶ One common way to execute a query command is to pass a SQL string to the Database Management System (DBMS)
- ▶ In many cases, this string is dynamically crafted based on user input
- ▶ “Tell me everything you know about”
 - ▶ “Phil Beutel”
 - ▶ “Phil Beutel or Phillip Beutel”
 - ▶ “Phil Beutel. If you cant find him just burn up all your records.”
- ▶ “SELECT * FROM Users WHERE username=’ ’. \$_GET[’name’] .” ;”



Little Bobby Tables



Courtesy of xkcd.com



SQL capabilities

- ▶ Normally just used for SELECT, INSERT, UPDATE
- ▶ Many other commands exist that can be manipulated:
 - ▶ DELETE, CREATE, ALTER, DROP, UNION, JOIN, INTO
- ▶ Stacked queries
 - ▶ Allow for multiple SQL queries in one string



Fingerprinting

- ▶ An important element of SQL injection is fingerprinting
- ▶ Detect a vulnerability by forcing an error in the SQL script
- ▶ Errors if not appropriately caught can give detailed information to users about database structure such as table names, column names, and the structure of a given SQL request.

```
query_error_report(SELECT clicked FROM  
user_article WHERE user_id = 68 AND article_id  
= INJECTION;) called at [/home/content/full/  
address/html/dbms_caller.php:10]
```





"You have an error in your SQL syntax" site:duke.edu - Search Advanced Search

Web Show options... Results 1 - 7 of 7 from duke.edu OR cs.duke.edu for "You have an error in your SQL syntax". (0.44 seconds)

DIAL Web Calendar: Event Display

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 'AND m = AND d = AND ...
camrd4.mc.duke.edu/Webcalendar/eventdisplay.php?id... - [Cached](#) - [Similar](#)

Field Reservation System

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 17 ...
www.calendar.groups.duke.edu/view_entry.php?&pview=1 - [Cached](#) - [Similar](#)

Duke University: Search Results

Failed: **You have an error in your SQL syntax;** check the manual that corresponds to your MySQL server version for the right
www.duke.edu/search/?q=biomedical&start=380 - [Cached](#) - [Similar](#)

Duke University: Search Results

Failed: **You have an error in your SQL syntax;** check the manual that corresponds to your MySQL server version for the right
www.duke.edu/search/?q=biomedical&start=540 - [Similar](#)

S. cerevisiae Gene AAP1' (YHR047C) Description and Page Index

mySQL error 1064: **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 ...
genome-mirror.duhs.duke.edu/cgi-bin/hgGene?org... - [Similar](#)

DIAL Web Calendar: Event Display

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 'AND m = AND d = AND ...
camrd4.mc.duke.edu/Webcalendar/eventdisplay.php?id... - [Cached](#) - [Similar](#)

You have an error in your SQL syntax; check the manual that ...

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 17 ...
www.ski.imsports.duke.edu/schedule/view_entry.php?&...1 - [Cached](#) - [Similar](#)

"You have an error in your SQL syntax" site:duke.edu - Search



Example Users Table

username	password	isAdmin	SSN	Zip	Credit
Artem	f9f16d97c	0	123-45-6789	12345	5555555555
Alex	b6d1f1992	1	987-65-4321	12345	2222222222
Patti	8c6f2cab3	0	555-55-5555	54321	7777777777
Phil	648f3a03	0	112-35-8132	13455	8914423337
Stephen	d6a6bc0d	0	761-09-8715	97258	4418167651
Tucker	a47cbe66	1	094-61-7711	28657	4636875025
Cooper	7c6ef401	0	121-39-3196	41831	7811514229



Example of a simple SELECT manipulation

- ▶ Without proper sanitization, users can manipulate log in scripts:

```
SELECT * FROM Users WHERE username = '$_GET['name']'
```

- ▶ What if name is set to my name' OR 1=1 /*
 - ▶ The apostrophe mark after my name ends that string being compared to the name column.
 - ▶ By doing OR 1=1 suddenly every entry in the table will be valid
 - ▶ Need to ignore all other input as a comment so we use /*
- ▶ Rather than being a valid user, the script will most likely take the first returned result from the table, which will be the first entry, and now the person has gained control of someone else's account
- ▶ Of course this can be manipulated even more by appending AND isAdmin=1 which will then only return administrative users.



How to steal information with SELECT

- ▶ What if you had a page that said who was from a given location:
 - ▶ `example.com/who.php?zip=12345`
 - ▶ `SELECT username FROM Users WHERE Zip = $_GET['zip']`
 - ▶ Page returned looks like:

Users from 12345:

- Alex
- Artem



Stealing the “Private” information

- ▶ Now what if the query were this:
 - ▶ `example.com/who.php?zip=12345 UNION SELECT SSN as username FROM Users`
 - ▶ `SELECT username FROM Users WHERE Zip=12345 UNION SELECT SSN as username FROM Users`
 - ▶ Page returned now looks like:

Users from 12345:

- Alex
- Artem
- 123-45-6789
- 987-65-4321
- 555-55-5555
- 112-35-8132
- 761-09-8715
- Etc.

Stacked Queries

- ▶ Some DBMS's and programming languages support stacked queries.
- ▶ Allow for multiple, distinct queries within one query string:
 - ▶ `INSERT INTO Users VALUES ('Alex'); SELECT * FROM Users;`

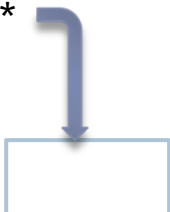
	ASP.NET	ASP	PHP
MySQL	Supported	Not Supported	Not Supported
MSSQL	Supported	Supported	Supported
Postgresql	Supported	Supported	Supported

-
- ▶ Table from Muhaimin Dzulfakar at BlackHat USA 2009

Stacked Query Risks

- ▶ Increased functionality, increased risk
- ▶ Same vulnerabilities can be exploited, but with this users have much more power because now they can use other commands that normally require their own query.

```
ME',NOW()); DELETE * FROM Users; /*
```



```
INSERT INTO Users VALUES ('', NOW());
```



```
INSERT INTO Users Values ('ME',NOW()); DELETE *  
FROM Users; /*',NOW());
```



Advanced Hacks

- ▶ Can even use SQL injection (with or without stacked queries) to write to the file system
- ▶ Some DBMS have functions for logging information or loading in files.
- ▶ Example injection string:

```
some_id UNION SELECT 0x0123ABCD, 0x00  
INTO OUTFILE '/var/www/  
meterpreter.exe' /*
```

-
- ▶ Hack from Muhaimin Dzulfakar at BlackHat USA 2009

Consequences

- ▶ Even basic hacks can be used to fingerprint databases and get password hashes or other sensitive user information
- ▶ A more stealthy hacker may simply update data such as HTML that will be displayed to users so that he can perform a XSS or other attack
- ▶ Vulnerabilities can result in data loss through deleting individual records or whole databases
- ▶ Advanced techniques described earlier can give users a back door to the server, and thus much more access to the system



Preventing SQL Injection

- ▶ **Good news:** There are multiple ways to prevent these issues.
- ▶ **Bad news:** Slightly different in each language
- ▶ **Two main solutions:**
 - ▶ Check the form of the input correctly in each case and escape inappropriate characters
 - ▶ Keep instructions separate from input through special methods and well written stored procedures



Sanitizing Input (in PHP)

- ▶ Two main types of sanitization

- ▶ Strings

- ▶ `SELECT * FROM Users WHERE username=' [INPUT] ' ;`

- ▶ Input ended by single quote

- ▶ Other types of input such as numbers

- ▶ `SELECT * FROM Users WHERE Zip=[INPUT] ;`

- ▶ No single quote required



Sanitizing Strings

- ▶ **Escape characters lets the DBMS know that a given character is input, not instruction.**
 - ▶ Common in other parts of programming such as `\n` for new line
- ▶ **Escaping certain characters makes it impossible for input to corrupt the request.**
 - ▶ `$sInput = mysql_real_escape_string($input);`
 - ▶ Usually there are other characters other than you realize that are dangerous so don't make your own escape method, use the database's.



Sanitizing other types of inputs

- ▶ For other types of inputs, need to be more specific in checking based on what value you are expecting.
- ▶ Language will still give you a string, but must check if it matches expected form
- ▶ If you're expecting a number, check that the input is numeric.
 - ▶ `is_numeric($input)`
- ▶ Regular expressions or other methods may be necessary for other inputs



Separate instructions from input

▶ Stored Procedures

- ▶ Typically used for large, repeated functions
- ▶ By construction, all instructions reside completely separately from input – Instructions in DBMS, input passed to the DBMS separately
- ▶ Still requires that stored procedures are written to avoid SQL injection as there are cases that simply execute a SQL string:
 - EXEC, similar to eval in other languages

▶ Parameterized input

- ▶ Pass SQL string with “blanks” for input and input passed as extra parameters:

```
cursor.execute( "SELECT * FROM Users WHERE email  
= '%s' AND pw = '%s' " , ( email , password ) )
```



Designer logic

- ▶ Always important to think: “Can my code, if used in an unintended way, do things I don’t want it to do?”
- ▶ Be careful about what you consider to be *trusted* input. Hackers can modify some values that you may expect are safe (example: user-agent)
- ▶ Hide errors from users
- ▶ Try to make sure input matches what you expect as closely as possible
- ▶ Always sanitize input

