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.

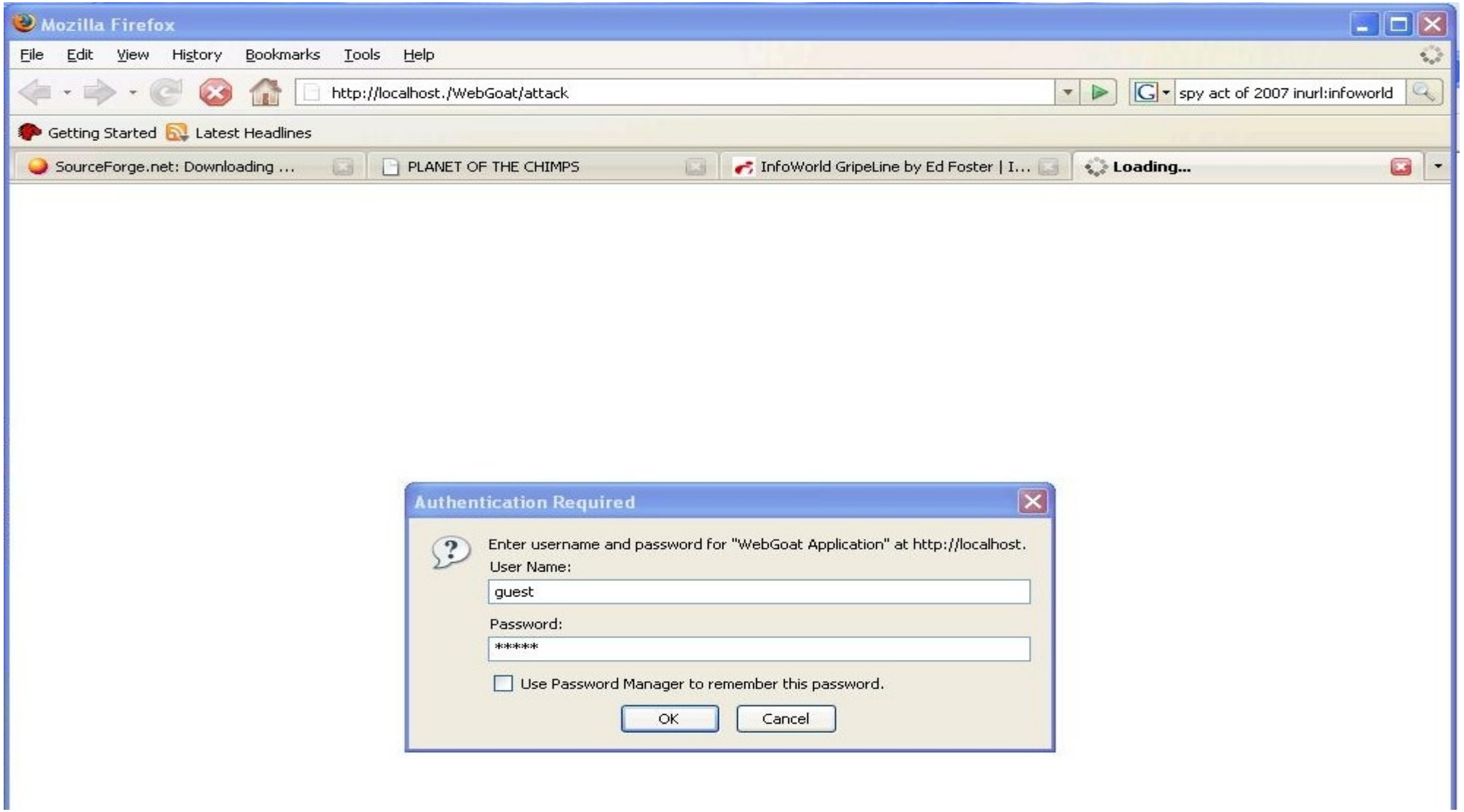
Not validating user input is one of most common mistakes in programming in general – SQL injection attacks take advantage of this flaw. They're easy to do and easy to automate.

Web Goat- A vuluneable Web Application https://owasp.org/www-project-webgoat/

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

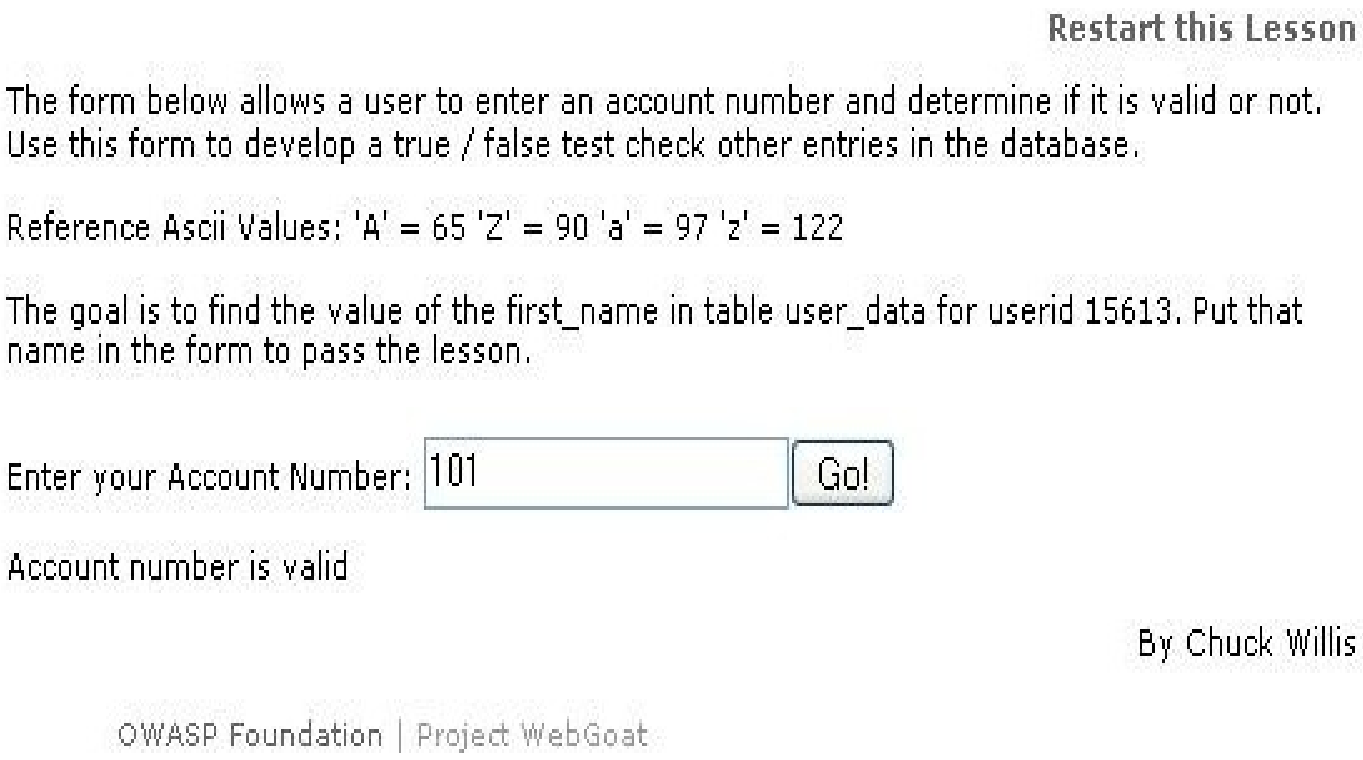
Web Goat is a Tomcat web server that's vulnerable to SQL injection attacks (among others). It's available at: http://www.owasp.org/index.php/Category:OWASP_WebGoat_Project

Web Goat

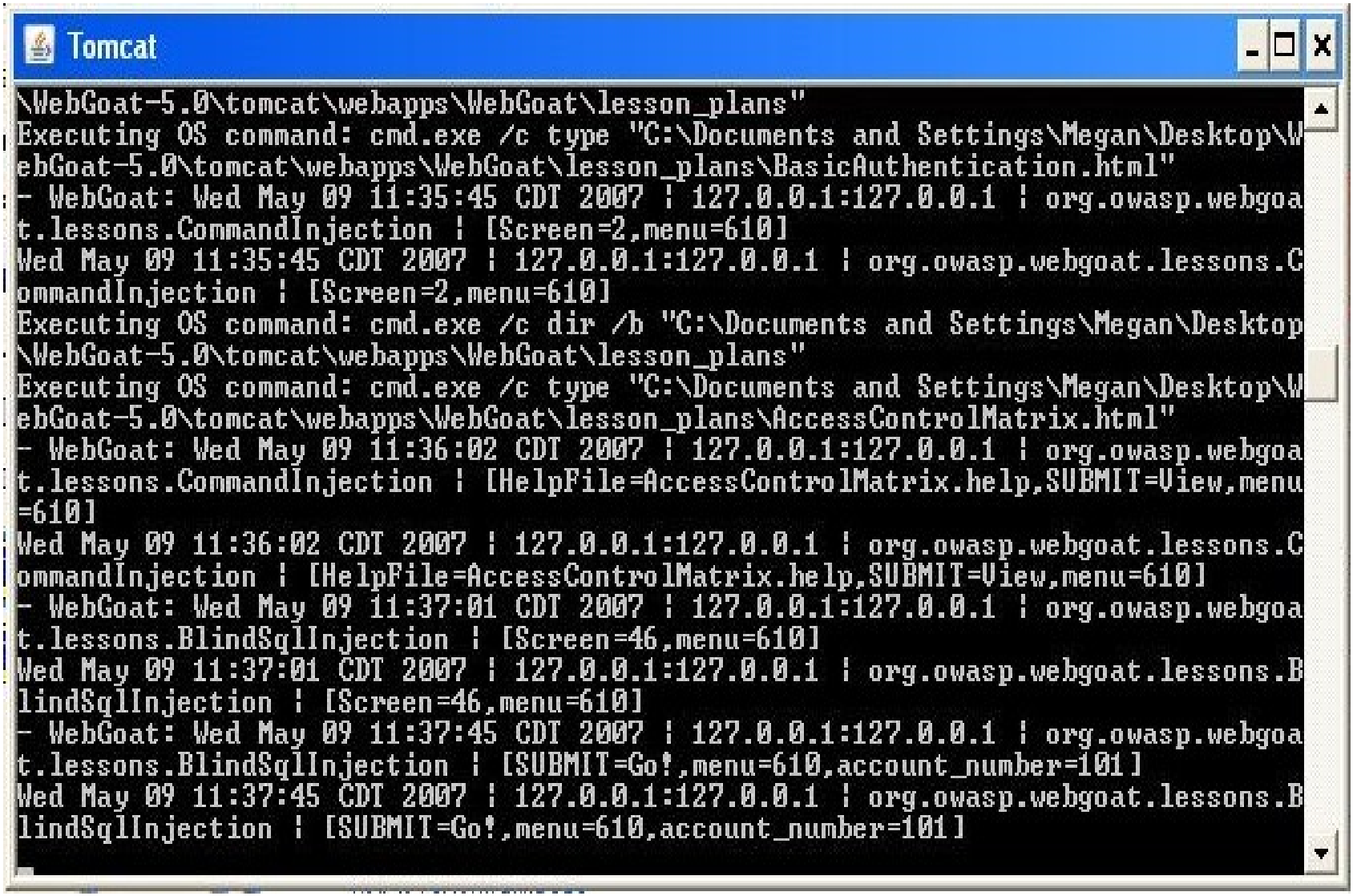

Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost./WebGoat/attack
spy act of 2007 inurl:infoworld
Getting Started Latest Headlines
SourceForge.net: Downloading ...
PLANET OF THE CHIMPS
InfoWorld GripeLine by Ed Foster | I...
Loading...
Authentication Required
Enter username and password for "WebGoat Application" at http://localhost.
User Name:
guest
Password:
*****
Use Password Manager to remember this password.
OK
Cancel

Low to Perform Blind SQL Injection

Blind SQL Injection (1)

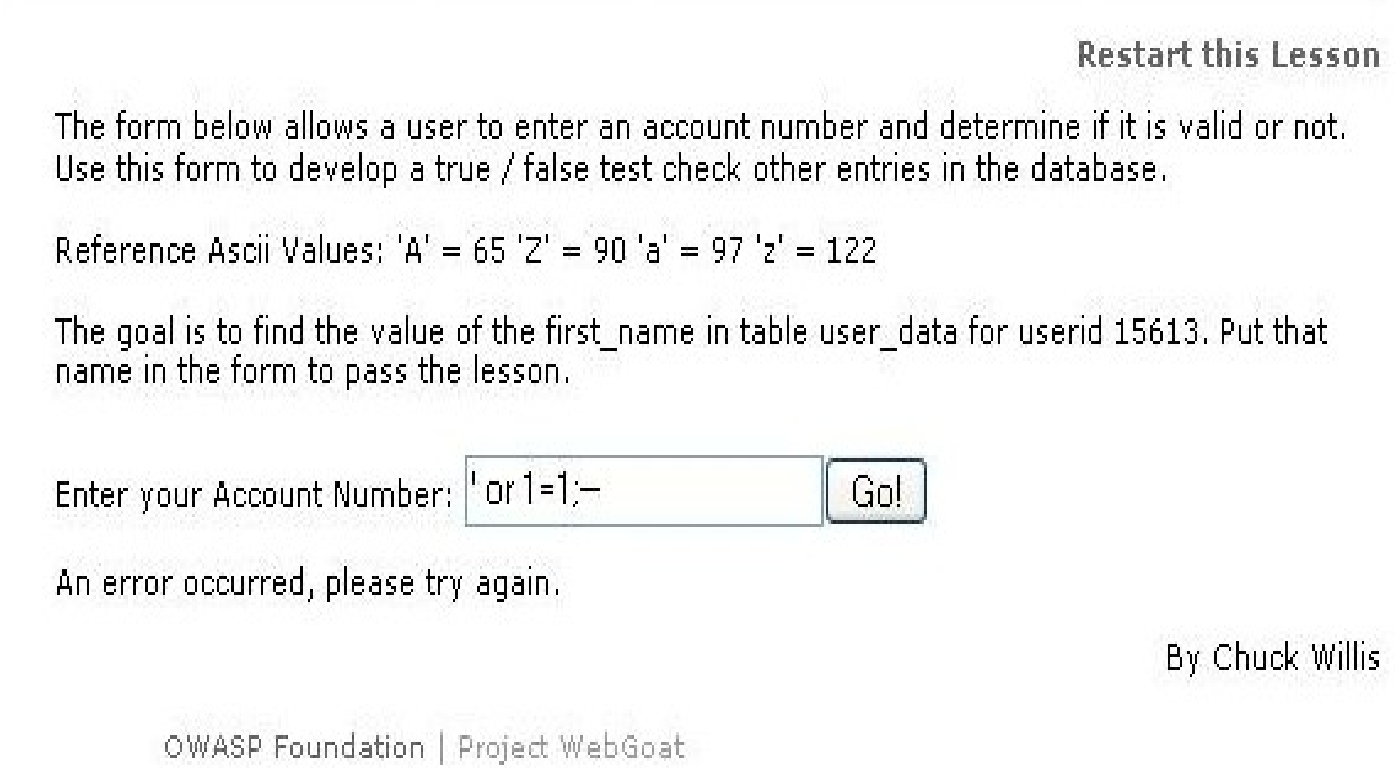


Blind SQL Injection (2)



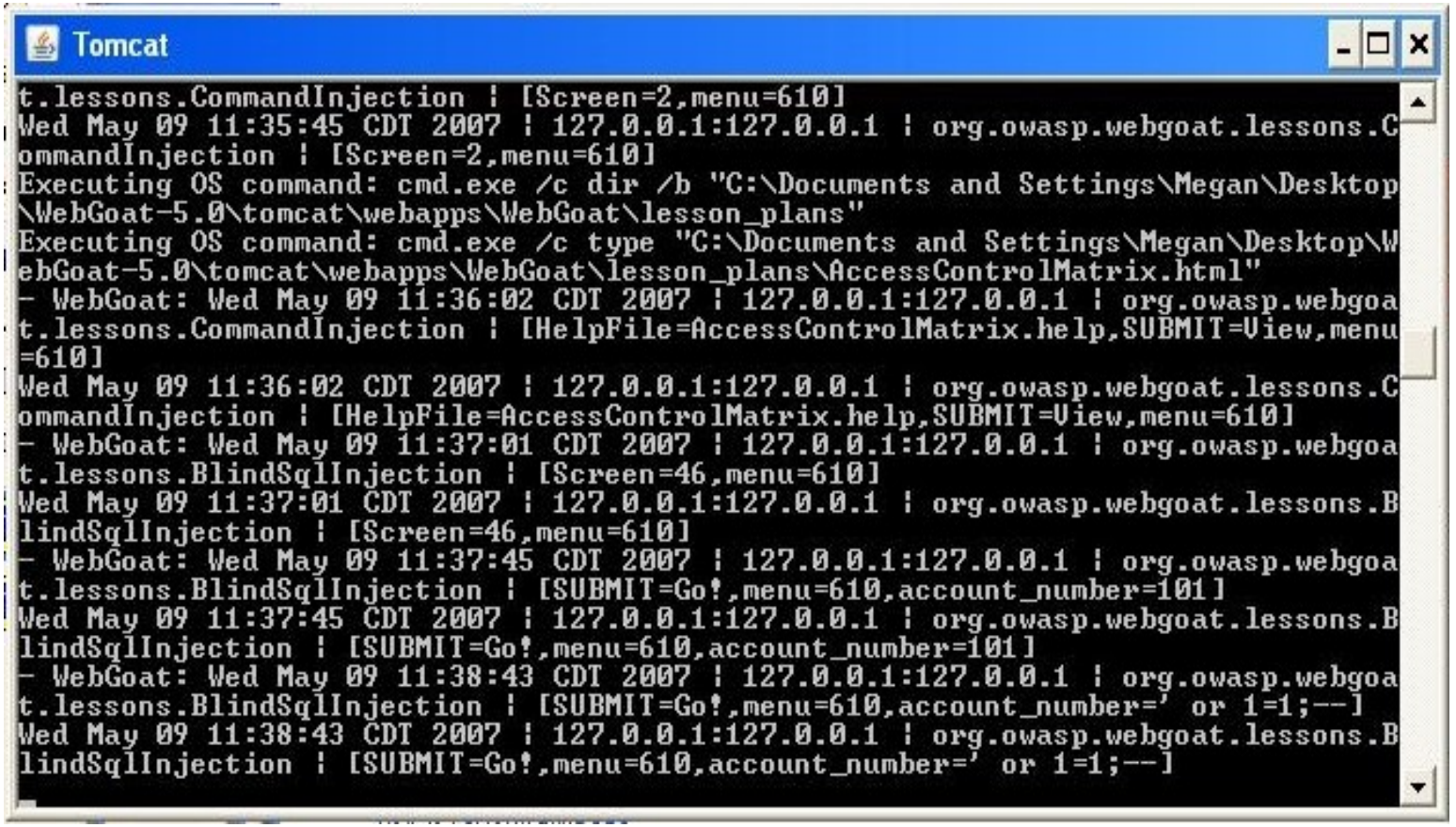
Blind SQL Injection (3)

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



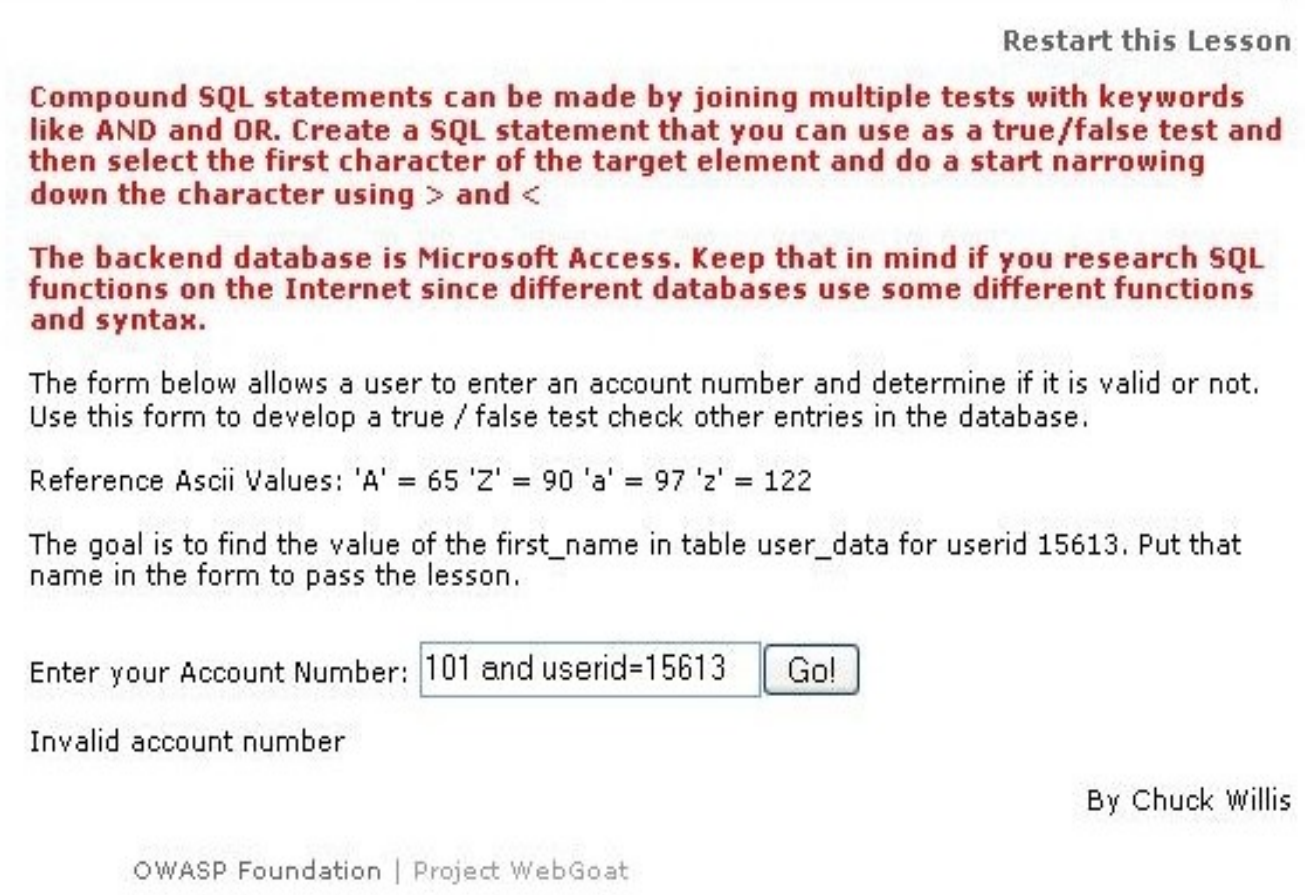
Blind SQL Injection (4)

What will the admin see?



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: 101 and userid=15613 Go!

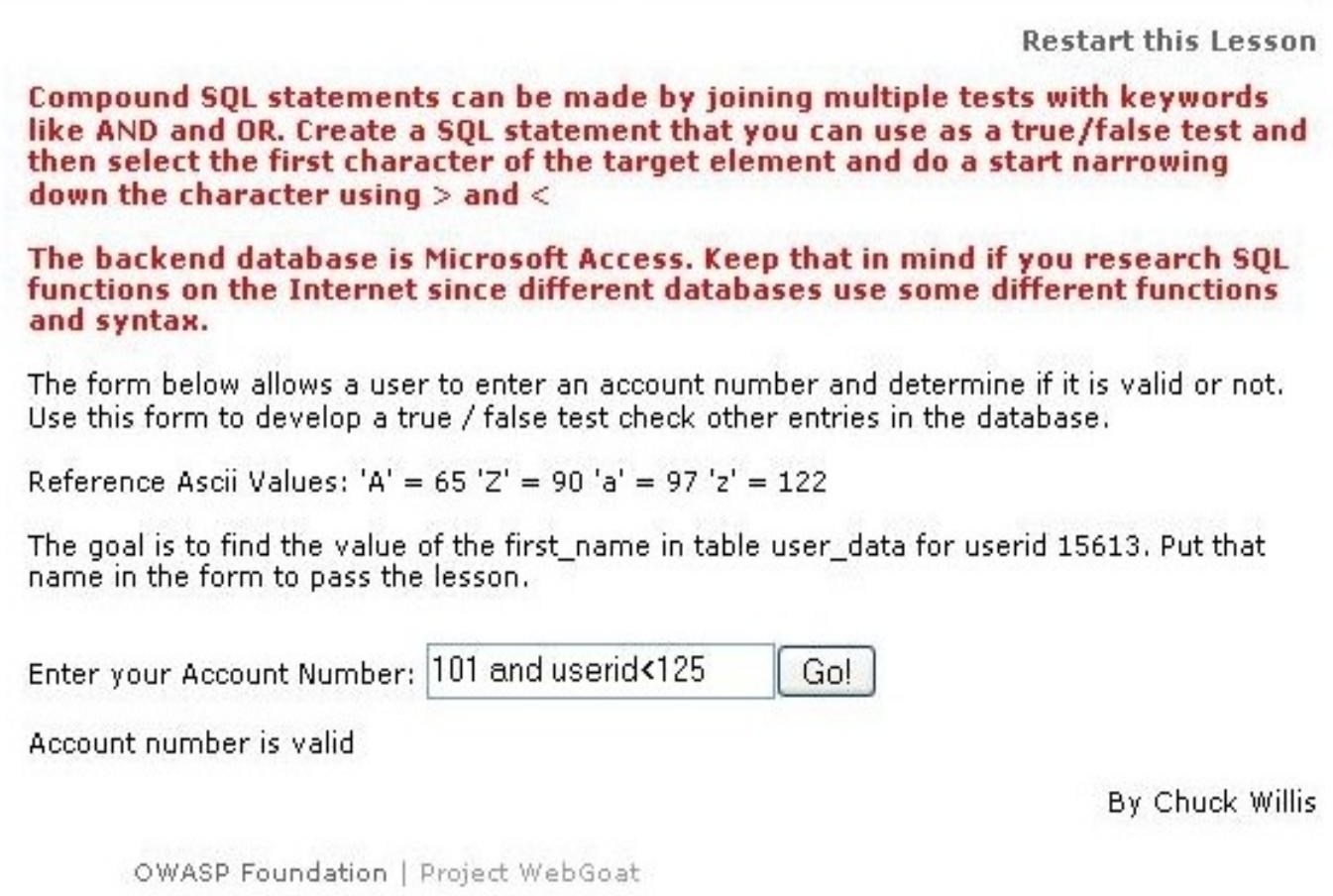
Invalid account number

By Chuck Willis

OWASP Foundation | Project WebGoat

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.

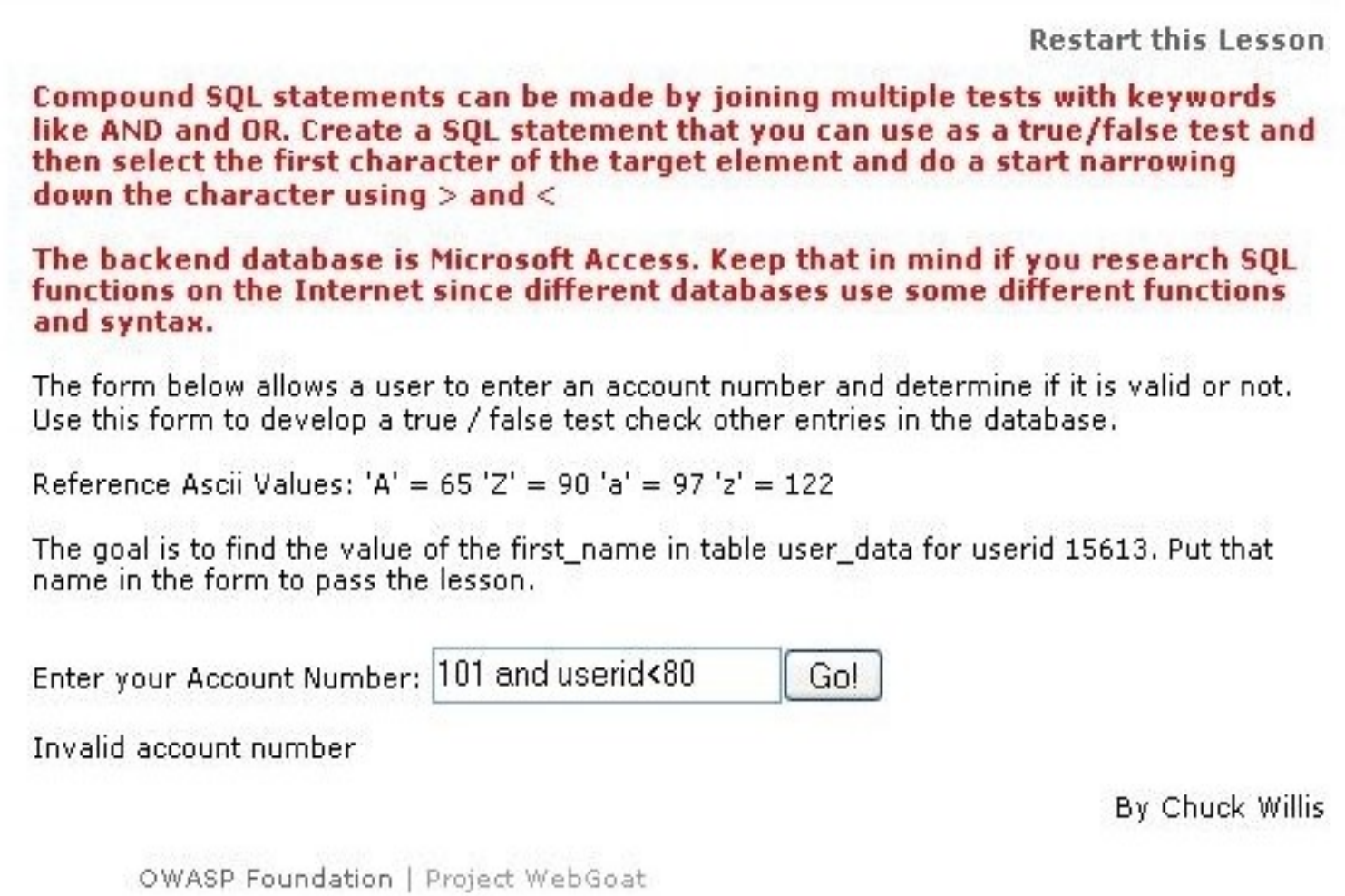
Enter your Account Number: 101 and userid<125 Go!

Account number is valid

By Chuck Willis

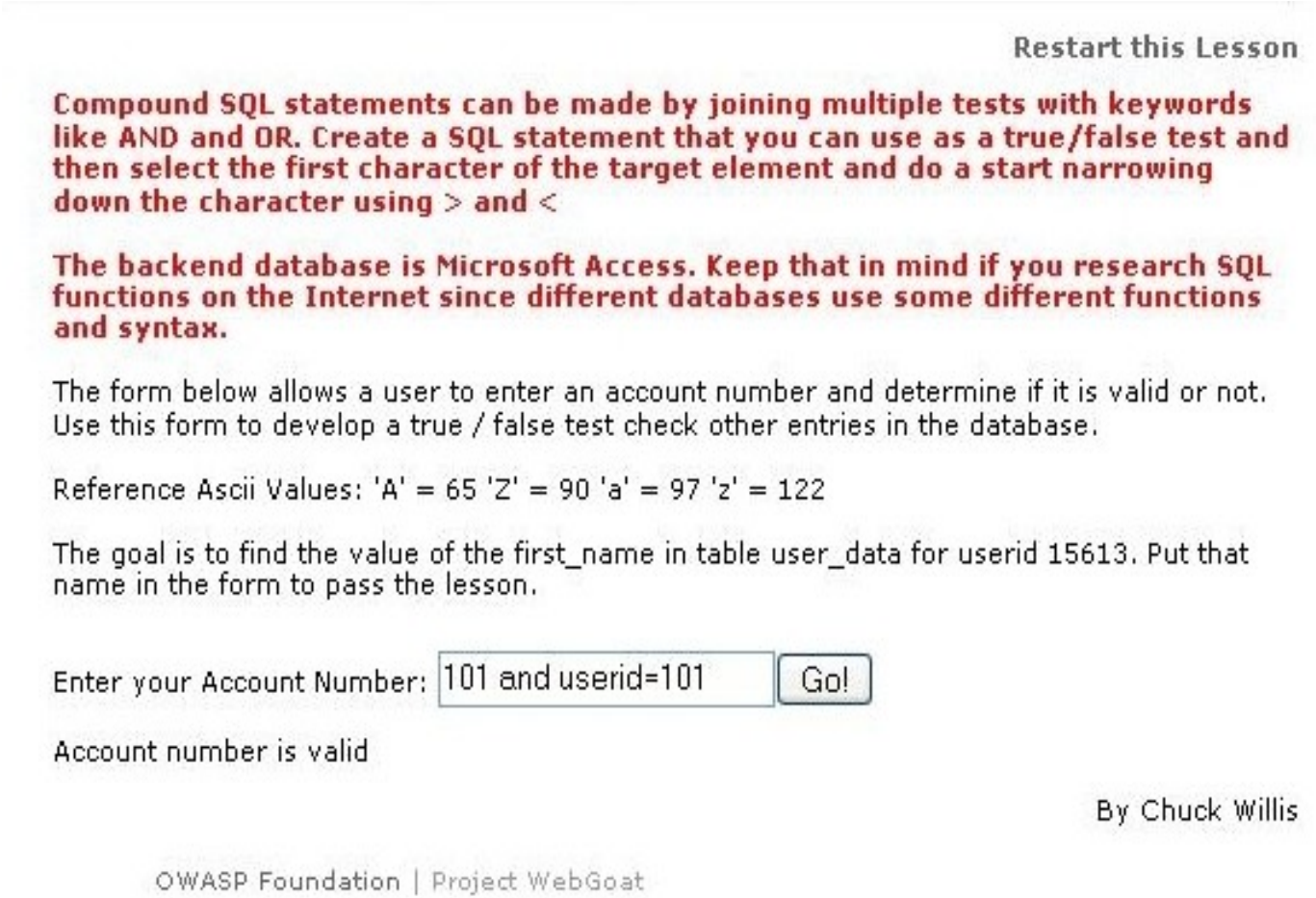
OWASP Foundation | Project WebGoat

Blind SQL Injection (7)



Blind SQL Injection (8)

A couple guesses later. . .



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: and first_name LIKE '%

Account number is valid

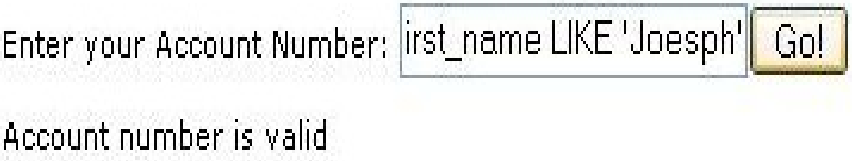
OWASP Foundation | Project WebGoat

Tomcat

```
=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.BlindSqlInjection | [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.BlindSqlInjection | [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.BlindSqlInjection | [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 userid=15613]
- WebGoat: Wed May 09 12:12:19 CDT 2007 | 127.0.0.1:127.0.0.1 | org.owasp.webgoa
t.lessons.BlindSqlInjection | [SUBMIT=Go!,menu=610,account_number=15613 and user
id=15613 and first_name LIKE '%']
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 userid=15613 an
d first_name LIKE '%']
```

Blind SQL Injection (10)

Guessing some more. . .



'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

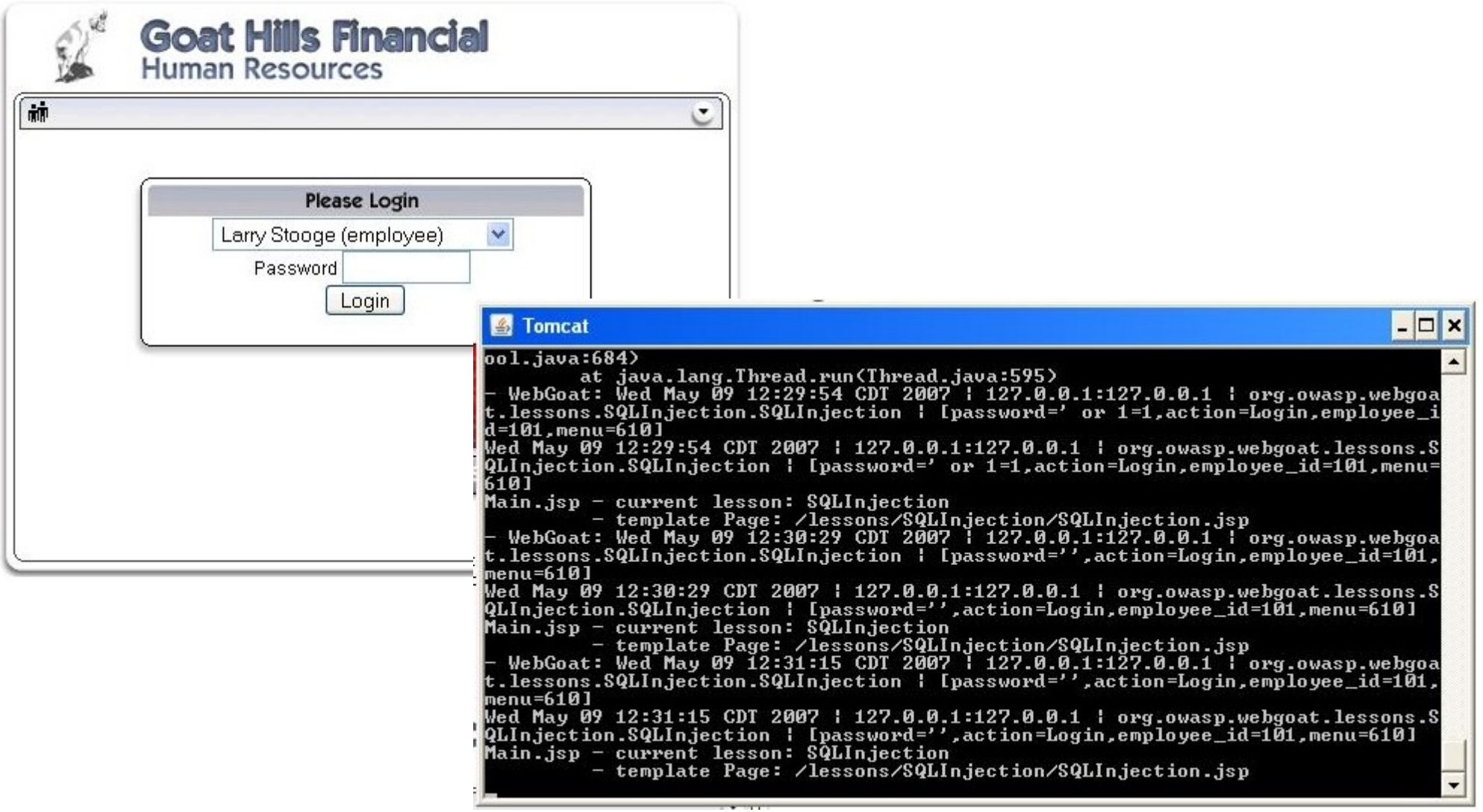
(Magic words vary by application and database.)

SQL Injection

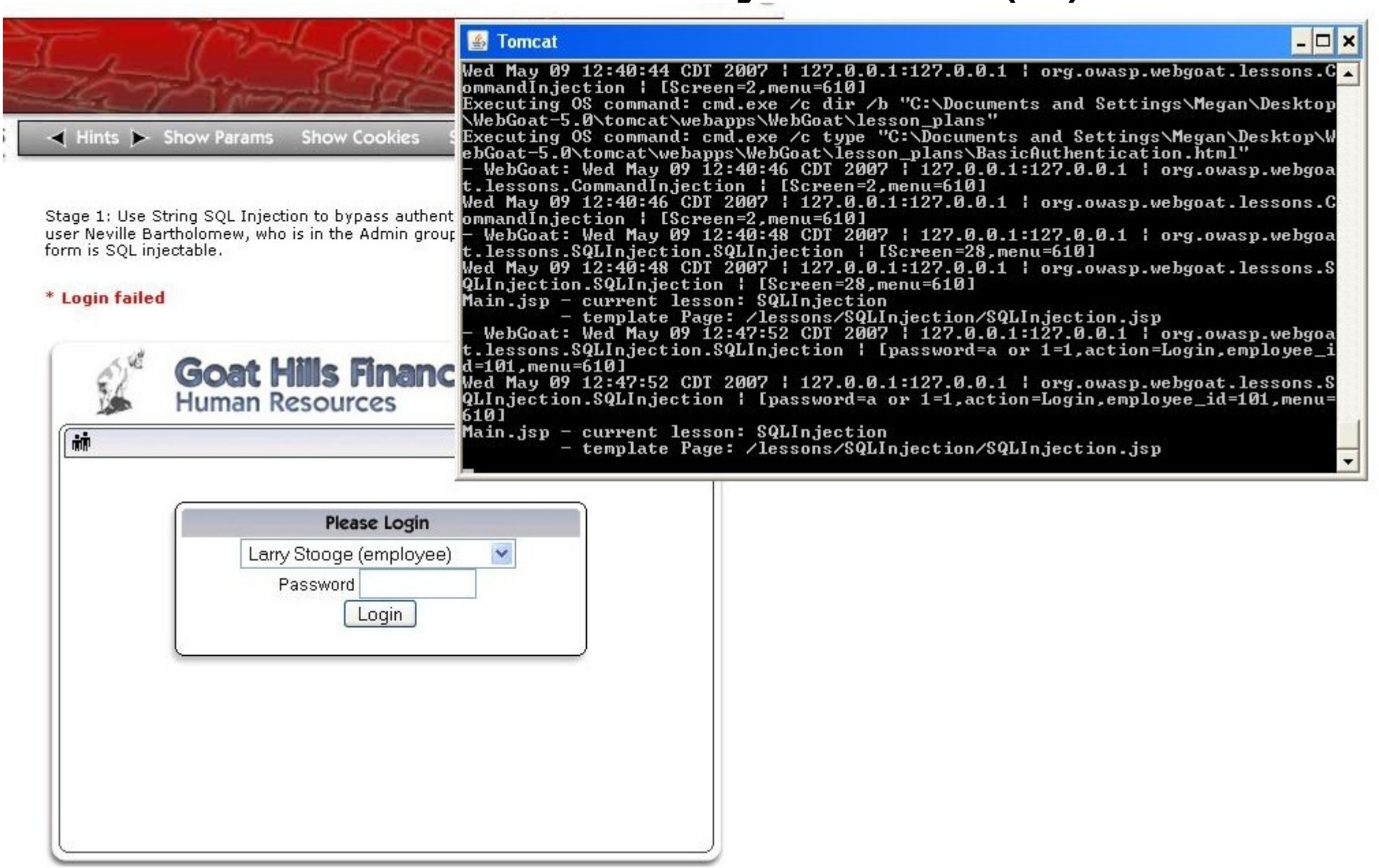
LAB: SQL Injection (1)

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**



LAB: SQL Injection (2)

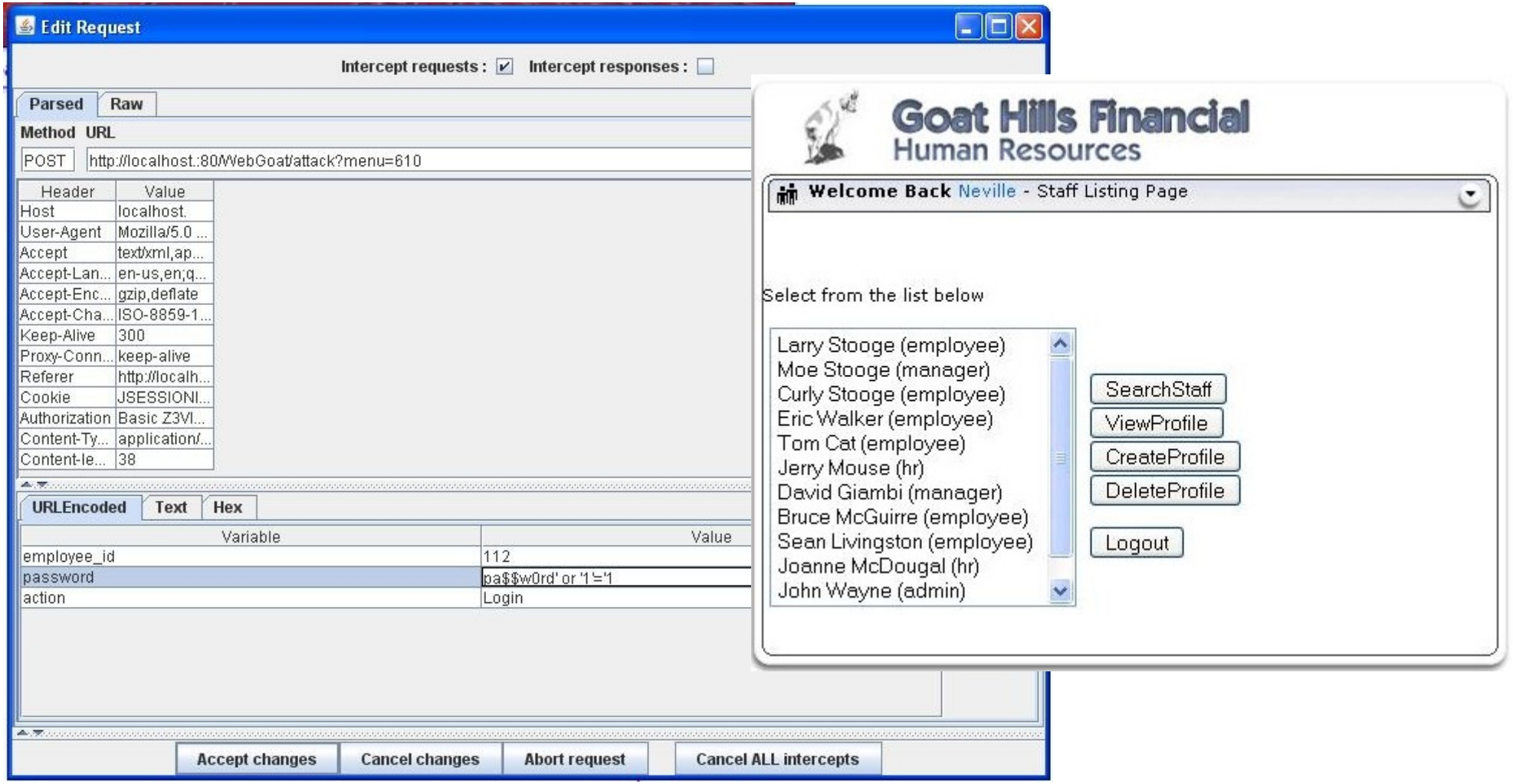


LAB: SQL Injection (3)

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

LAB: SQL Injection (4)

Larry isn't all that important. So what?



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.