LAB SESSION 3

SQL INJECTIONS

XSS

TU/e Technische Universiteit Eindhoven University of Technology

Lab Session 3 Exercises

- ***WebGoat:***

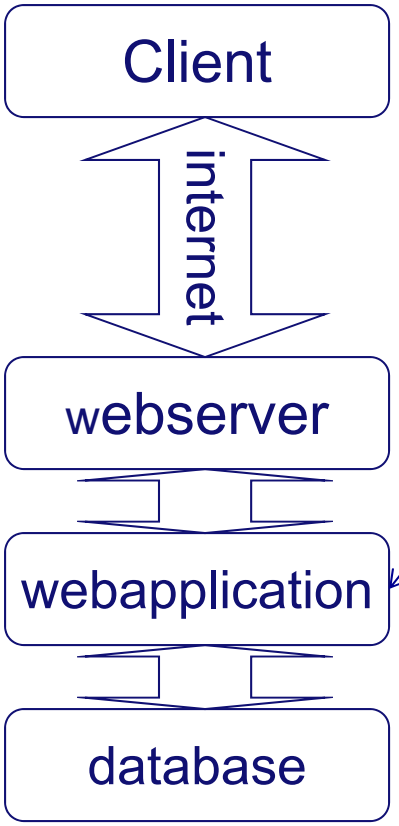
  SQL Injection
  - *Injection Flaw*
    - *Modify data with SQL injection*

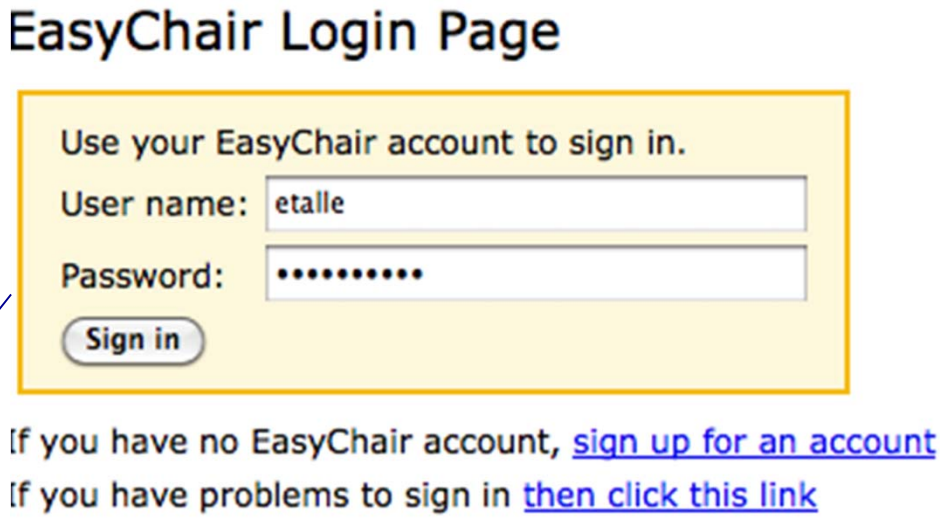
  XSS
  - *Xross-Site-Scripting (XSS)*
    - *Stage 1: Stored XSS*

If finished with those can try: attack Website (close WebGoat)

- *reconnect to network go to:*
- *sectest1.win.tue.nl:8080/SecurityCourse*
- *SQL login and password stealing:*
  - *Try to log in as dVader*
  - *Try to get the password of dVader*
  - *(Both use SQL injection, second will take multiple testing steps).*
- *XSS: use the wall*
  - *get colleague to log in as victim (see SQL 1 above) to test.*
  - *Steal cookie (or just print it to screen).*

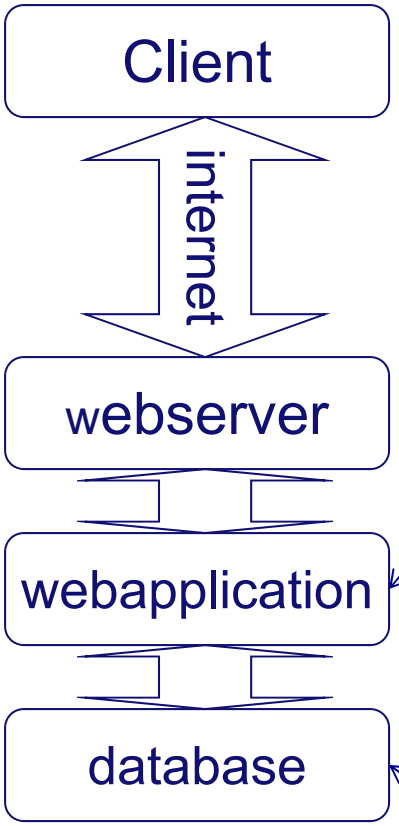
SQL injection by example

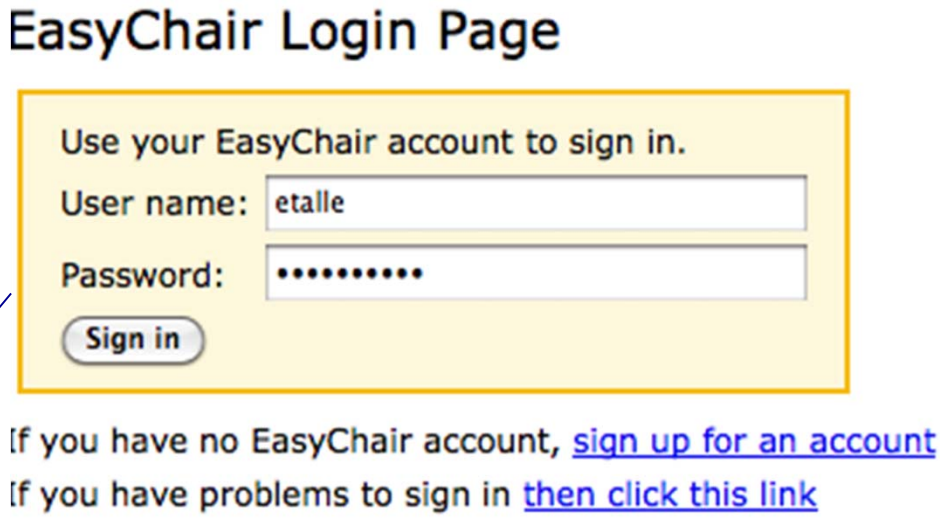




- The WEBAPPLICATION presents a form with username and password
- What happens in the database?

In the database…



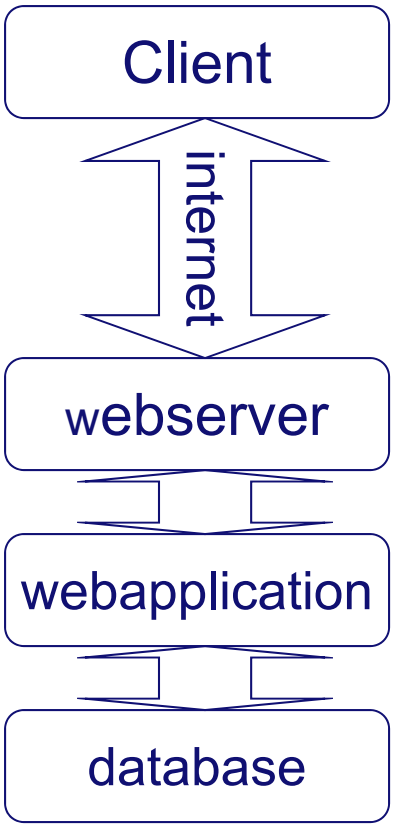


- SELECT … FROM users WHERE username = '$username' AND password = '$password'

What happens if …

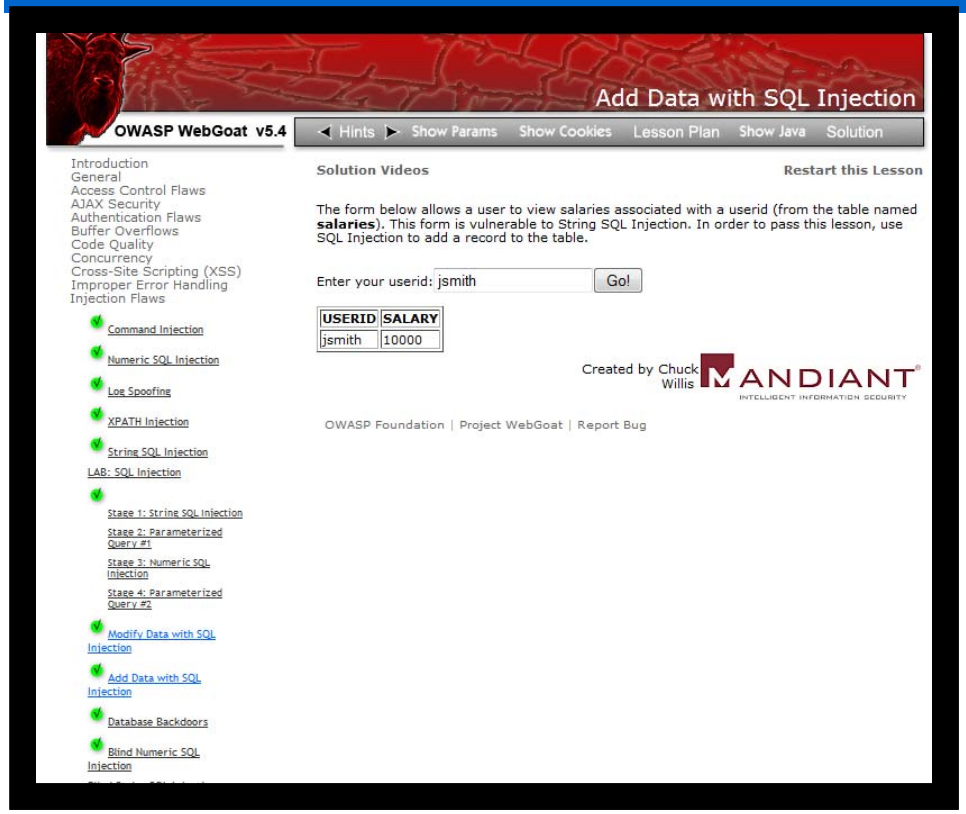
- Username: just any legal user name
- Password: anything' OR 'x'='x
- Recall the MySQL command is:
  - SELECT … FROM users WHERE username = ‘$username’ AND password = ‘$password’
- It becomes:
  - SELECT … FROM users WHERE username = ‘elisa’ AND password = ‘**anything’ OR ‘x’=‘x**’
- And it gets parsed the wrong way.
- Just to be clear: the above query selects all users, so the reply is the list of all users, together with all the parameters that are in the …

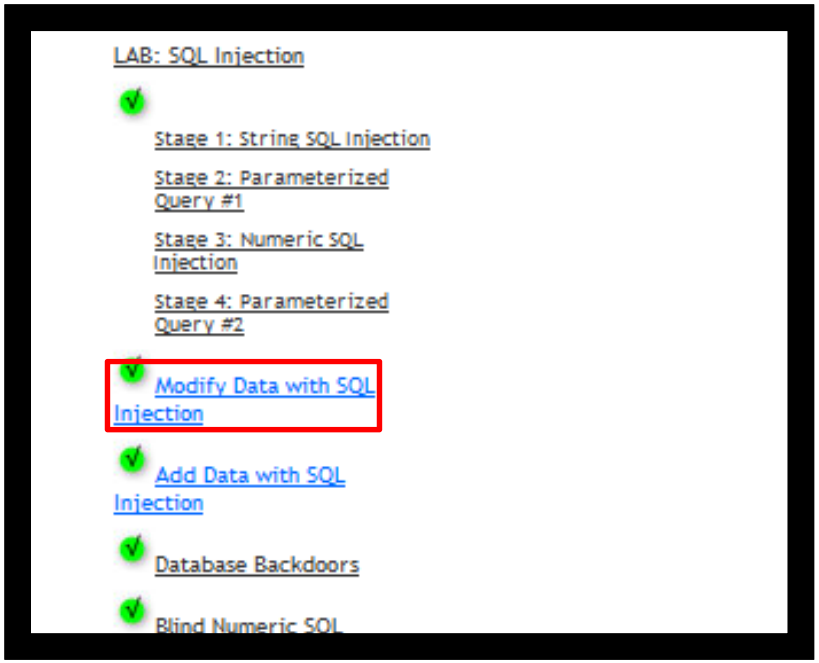
What do you try to achieve with a SQL injection



- Have the database do something that the programmer didn't think of.
- Particularly useful for data extrusion (stealing of data like passwords).
- But it can also be used for something else …
- … like having the database **modify** the data it has.
- Let's see an example

SQL Injection - Exercise





- *Goal:* use SQL injection to change database entries
- *Exercise:*
  - Go to **Injection Flaws→Modify Data with SQL Injection**
  - increase the salary of *jsmith* from 10000$ to 30000$

SQL Injection – Solution (1)

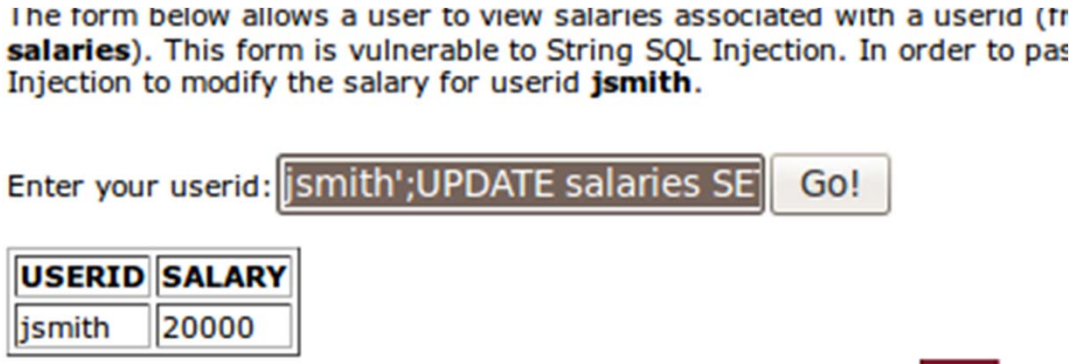
- Underlying database SQL query is likely to be:

SELECT userid, salary FROM users where userid='jsmith'

  - red part is our input

- Use stacked query by giving input:

jsmith';UPDATE salaries SET salary =30000
WHERE userid = 'jsmith'

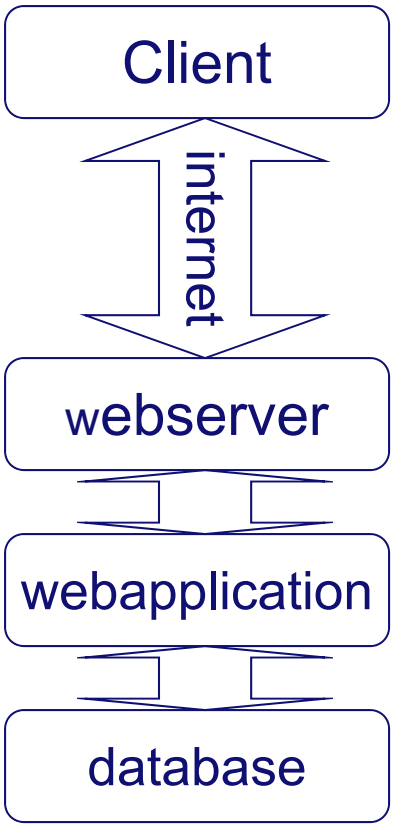


Created by Chuck Willis

SQL Injection - Lesson Learned

- An attacker can use SQL Injection to read or modify data of a database
  - An attacker could, for example, read the account numbers of all the costumer of a bank
- Web servers should use input sanitation to avoid SQL injection
  - i.e. recognizing the user is inserting SQL or script commands and not accept such string as input

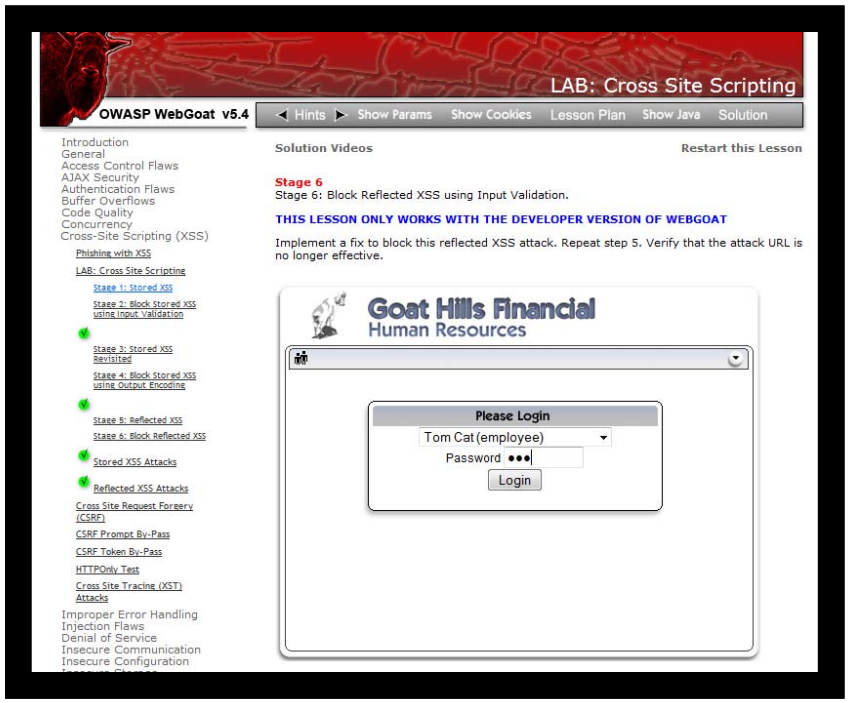
XSS In a nutshell

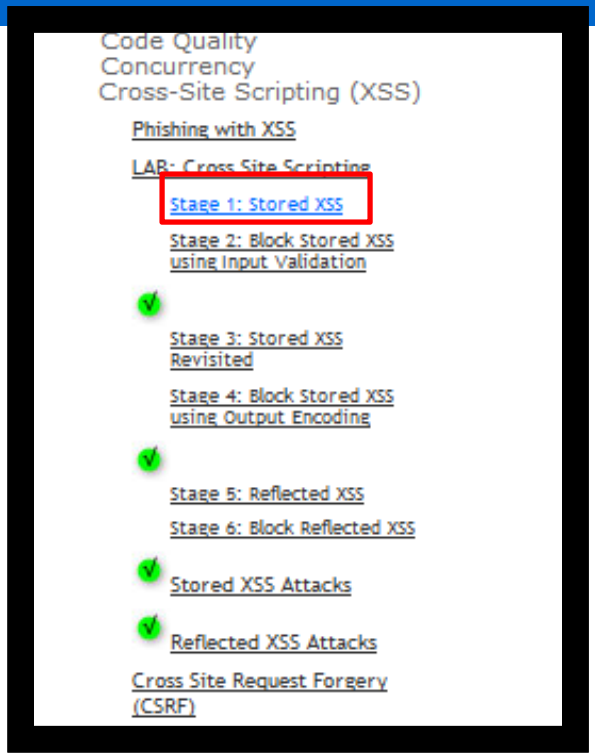


- While with SQL injection the goal was to hack the DB …
- … now the goal is to hack the client of your victim …
- … for instance by storing something in the webserver …
- … that will trigger something in your victim's client when he will look at the page you tampered with.
- … let's see an example.


http://seclists.org/fulldisclosure/2011/Sep/50
Security
Volo
HH
Home
Admission Inter...
Google
Bing
ITEA project
Sentinels II
Full Disclosure mailing list archives
By Date
By Thread
Search
20 Famous websites vulnerable to Cross Site Scripting (XSS) Attack
From: Mohit Kumar <thehackernews () gmail com>
Date: Wed, 7 Sep 2011 04:58:21 +0530
Most of the biggest and Famous sites are found to be Vulnerable to XSS attack . Cross-site scripting (XSS) is a type of computer security vulnerability typically found in web applications which allow code injection by malicious web users into the web pages viewed by other users. Examples of such code include HTML code and client-side scripts. An exploited cross-site scripting vulnerability can be used by attackers to bypass access controls such as the same origin policy. Recently, vulnerabilities of this kind have been exploited to craft powerful phishing attacks and browser exploits. Cross-site scripting was originally referred to as CSS, although this usage has been largely discontinued.
Hacker with code name "*Invectus*" list some such famous sites with XSS vulnerability as listed below :
1.)
http://video.state.gov/en/search/img-srchttp-i55tinypiccom-witu7dpng-height650-width1000/Ij48aW
2.)
http://www.telegraph.co.uk/search/?queryText=%22%3E%3Cimg%20src=%22http://i55.tinypic.com/witu7
3.)
http://www.dsm.com/en_US/cworld/public/home/pages/searchResults.jsp?search-site=%22%3E%3Cimg+src

XSS – Exercise





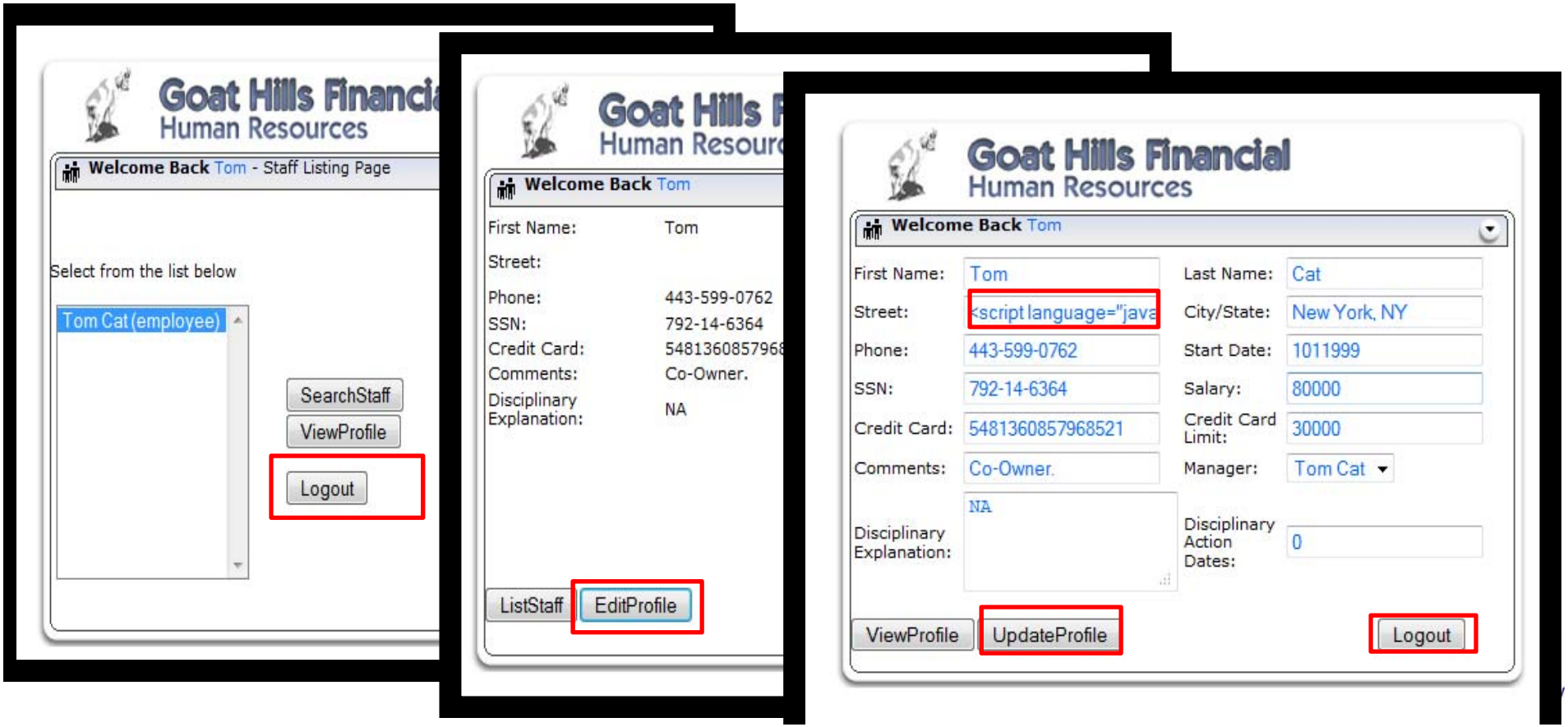
- Goal: Add a script to your (Tom's) profile to infect visitors
- Exercise:
  - You are Tom Cat (your password is tom)
  - Go to **Cross Site Scripting (XSS)→Stage 1: Stored XSS**
  - Insert the script
  - Test: log in as other user (guess password) and visit toms profile

XSS – Solution (1)

- Tom Cat can view his own profile, and he cannot see the profiles of his colleagues.
- On the other hand, David and Jerry can see the profiles of a few people.
- In particular Jerry can see Tom's profile.
- Now, Tom can try to attack Jerry by storing something a "kind of virus" on his profile.
- In the moment Jerry will look at Tom's profile, he will be infected.

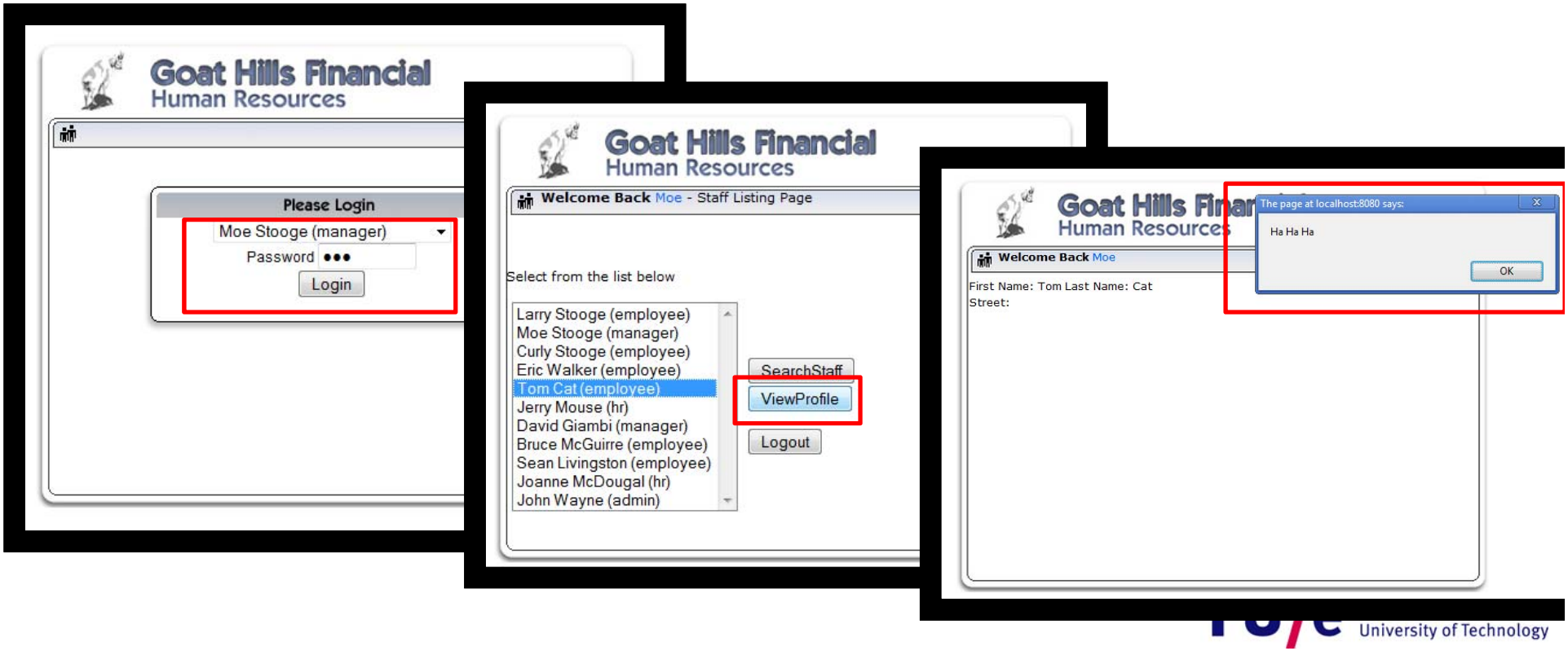
XSS - Solution (2)

- Login as Tom Cat (password: tom)
- Go to ViewProfile and then to EditProfile
- Add the script to one of the profile field (e.g. Street)
- **<script language="javascript" type="text/javascript">alert("Ha Ha Ha");</script>**
  - This is our little “virus”
- UpdateProfile and Logout



XSS - Solution (3)

- Now login as *Moe Stooge* (password: moe)
- Select *Tom Cat* and view his profile
- At this point you should see the alert message, resulting from the script being run



XSS - Lesson Learned

- As well as for SQL injection, XSS attack are possible if no input sanitation takes place
- The application should have checked that the street you inserted was not a legal street name.
- In other words, the application should have **sanitized** your input. But it failed to do so.
- Your input was a script in javascript, and when Jerry looked at the page, Jerry's browser happily executed the script.
- That script could have done much more harm than just laughing.
  - Could have stolen some information (cookies)
  - Could have executed code at the client side