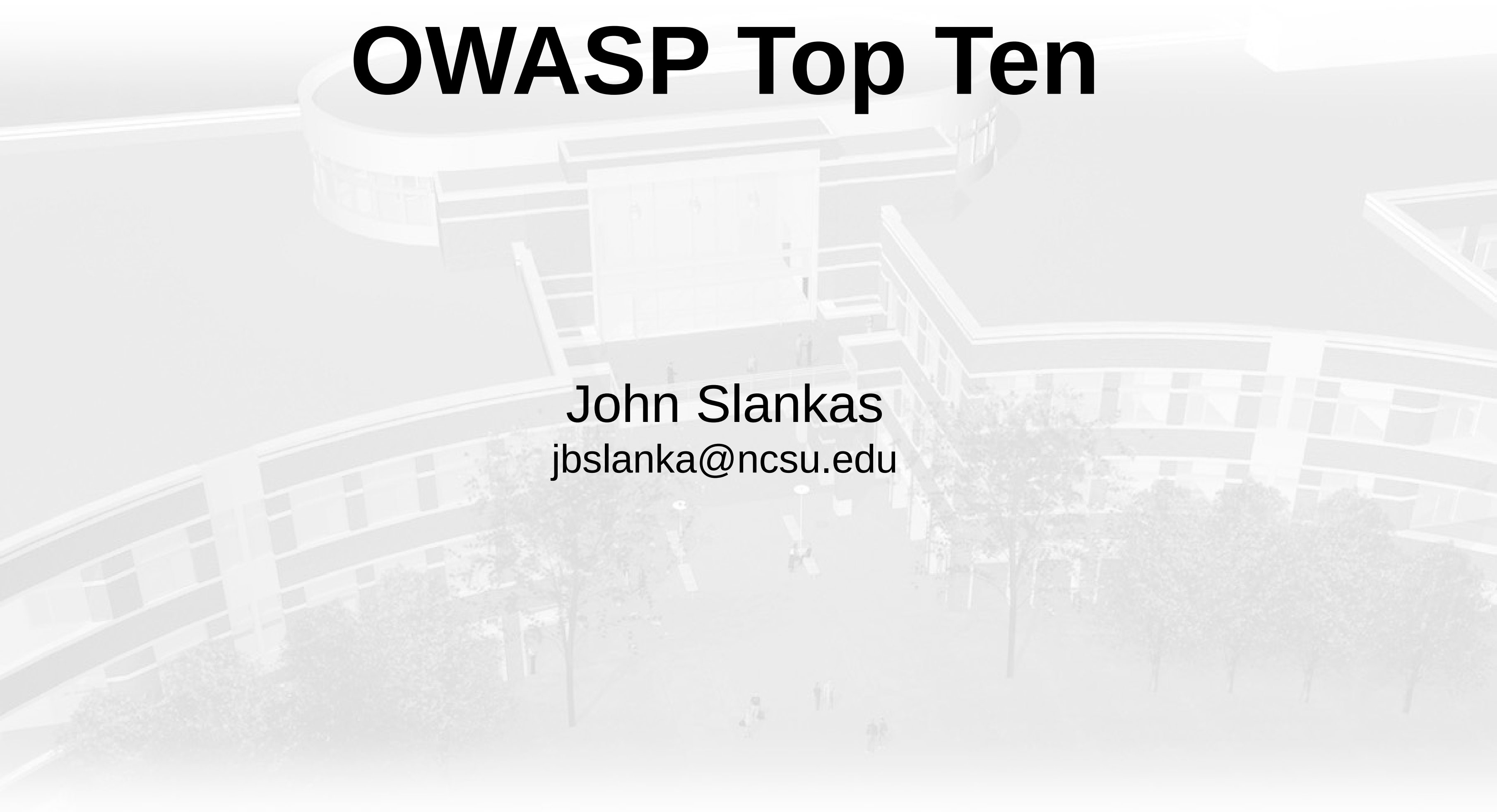


# OWASP Top Ten

John Slankas
jbslanka@ncsu.edu

Computer Science
NC STATE UNIVERSITY

Slides adopted from Laurie Williams

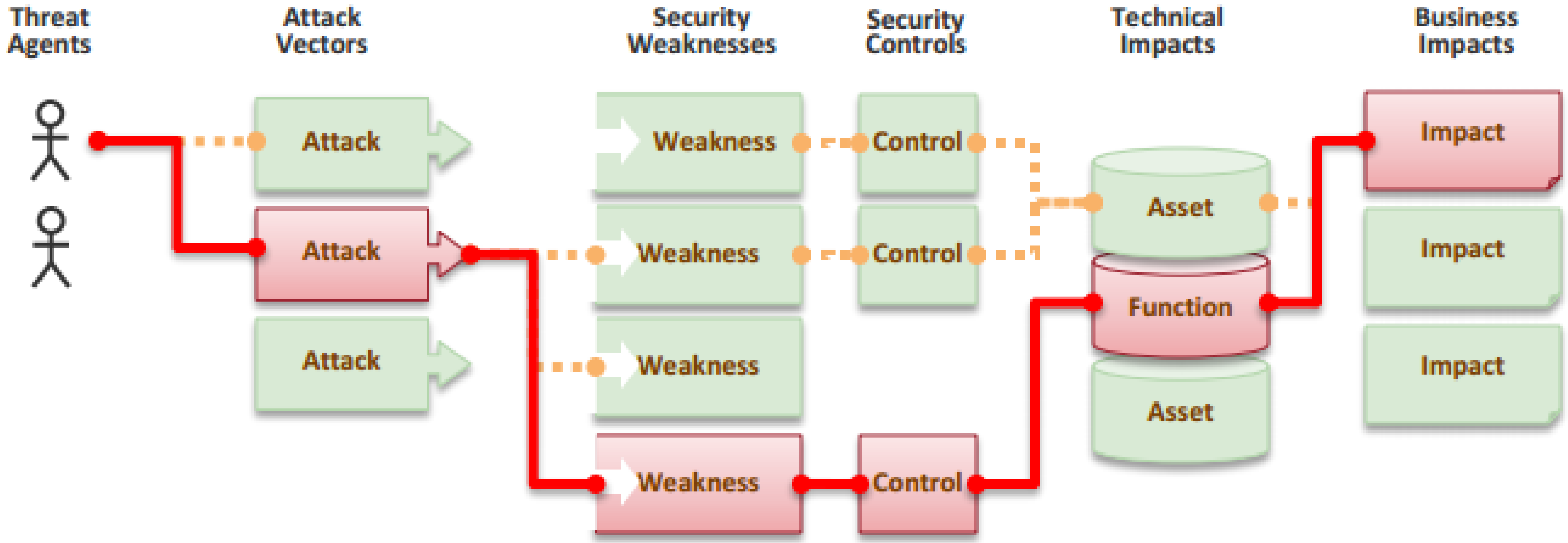
# Agenda

- Overview of the Top 10
- A1 – Injection
- A2 – Broken Authentication and Session Management
- A3 – Cross-Site Scripting
- A4 – Broken Access Control
- A5 – Security Misconfiguration

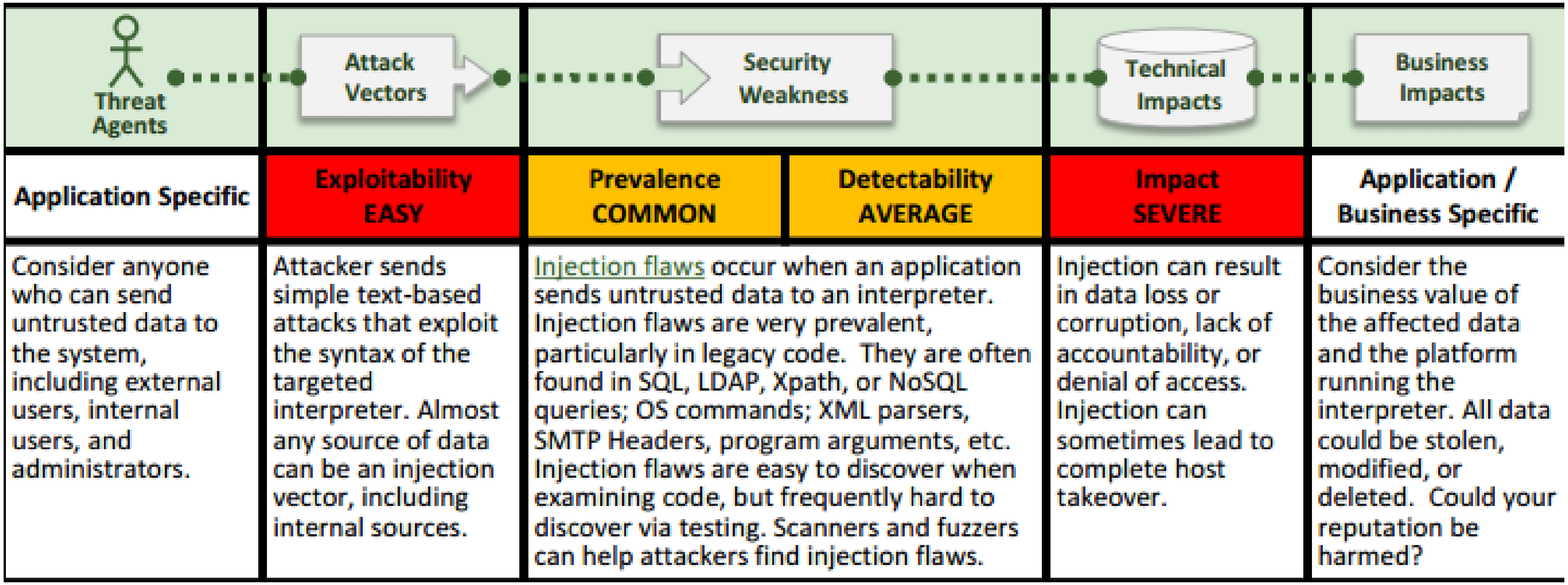
# OWASP Top 10

- Started in 2004
- Focus security efforts against the most serious vulnerabilities
- 2013 Version generated from 8 datasets from 7 companies
- 2017 Update in progress.

# OWASP: Application Security Risk



# Injection Attacks



| Threat Agents | Attack Vectors | Security Weakness | | Technical Impacts | Business Impacts |
|---|---|---|---|---|---|
| Application Specific | Exploitability<br>EASY | Prevalence<br>COMMON | Detectability<br>AVERAGE | Impact<br>SEVERE | Application / Business Specific |
| Consider anyone who can send untrusted data to the system, including external users, internal users, and administrators. | Attacker sends simple text-based attacks that exploit the syntax of the targeted interpreter. Almost any source of data can be an injection vector, including internal sources. | Injection flaws occur when an application sends untrusted data to an interpreter. Injection flaws are very prevalent, particularly in legacy code. They are often found in SQL, LDAP, Xpath, or NoSQL queries; OS commands; XML parsers, SMTP Headers, program arguments, etc. Injection flaws are easy to discover when examining code, but frequently hard to discover via testing. Scanners and fuzzers can help attackers find injection flaws. | | Injection can result in data loss or corruption, lack of accountability, or denial of access. Injection can sometimes lead to complete host takeover. | Consider the business value of the affected data and the platform running the interpreter. All data could be stolen, modified, or deleted. Could your reputation be harmed? |

Source: http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202013.pdf

# Injection Attacks

Injection attacks occur when unvalidated input is embedded in an instruction stream and cannot be distinguished from valid instructions.

If a language has a parser or an interpreter, and if the input can be confused for instructions in the language or the way the language is applied, then the language is vulnerable to an injection attack.

Consider the following fragment in a dynamic SQL query:

```
'x = x + ' + input + ';’
```

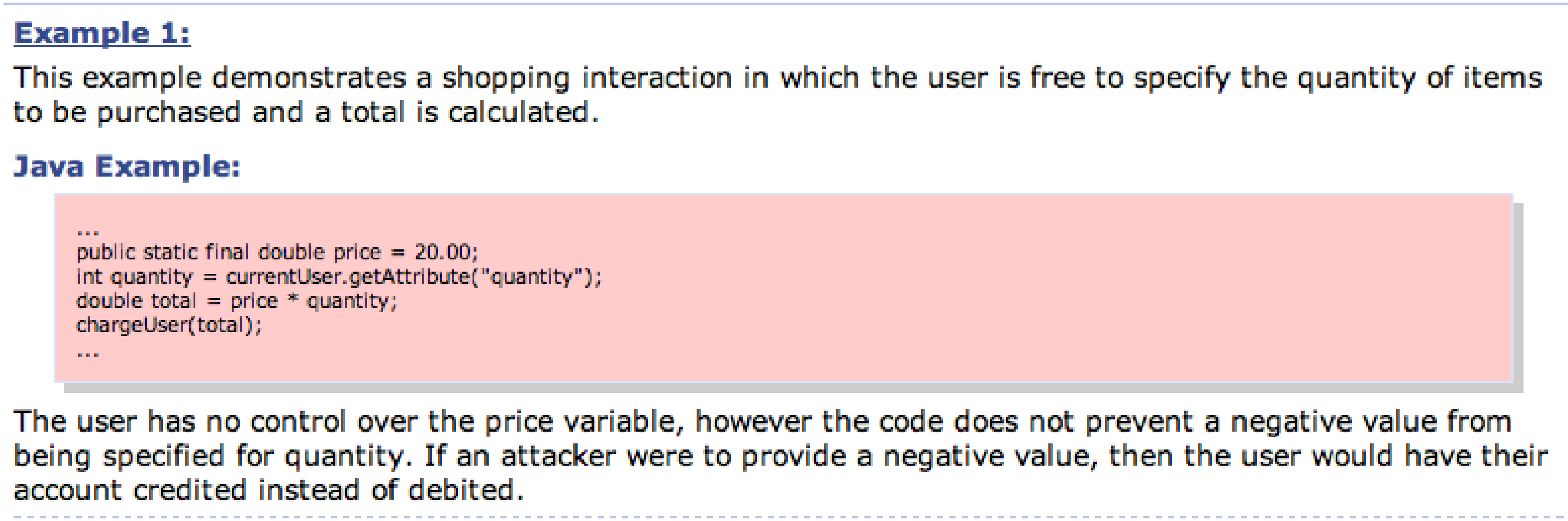
Consider the following inputs for the code:

```
input = '12'
input = '12;print("foo");'
```

# Software Security Principle

- Never trust input!
- Accept only known good input!
  - Reject input that does not conform
  - Or transform it into something that does
- Where does input come from?
  - Parameters or arguments; form fields
  - Cookies
  - Environment variables
  - Request headers
  - URL components
  - Files
  - Databases
  - ….

# Improper Input Validation Example



**Example 1:**

This example demonstrates a shopping interaction in which the user is free to specify the quantity of items to be purchased and a total is calculated.

**Java Example:**

```
...
public static final double price = 20.00;
int quantity = currentUser.getAttribute("quantity");
double total = price * quantity;
chargeUser(total);
...
```

The user has no control over the price variable, however the code does not prevent a negative value from being specified for quantity. If an attacker were to provide a negative value, then the user would have their account credited instead of debited.

http://cwe.mitre.org/data/definitions/20.html

# SQL Injection Vulnerability

- Condition: When untrusted input is used to construct dynamic SQL queries.
- Consequence: Can be used to alter the intended query logic to access, modify, and delete data in the database, possibly including execution of system commands or creating a denial of service.

http://cwe.mitre.org/top25/#CWE-89

# SQL Injection - String Type Example

| | |
|---|---|
| Login: | John |
| Password: | John1234 |

```
String query = “SELECT * FROM users
      WHERE login = ‘” + login +
      “’ AND password = ‘” + password + “’”;
```

**Expected input:**

```
SELECT * FROM users
  WHERE login = ‘John’
  AND password = ‘John1234’
```

**Result:** Returns John’s user information

# SQL Injection – Tautology

Login: `’ OR ‘1’ = ‘1`

Password: `’ OR ‘1’ = ‘1`

```
String query = “SELECT * FROM users
      WHERE login = ‘” + login +
      “’ AND password = ‘” + password + “’”;
```

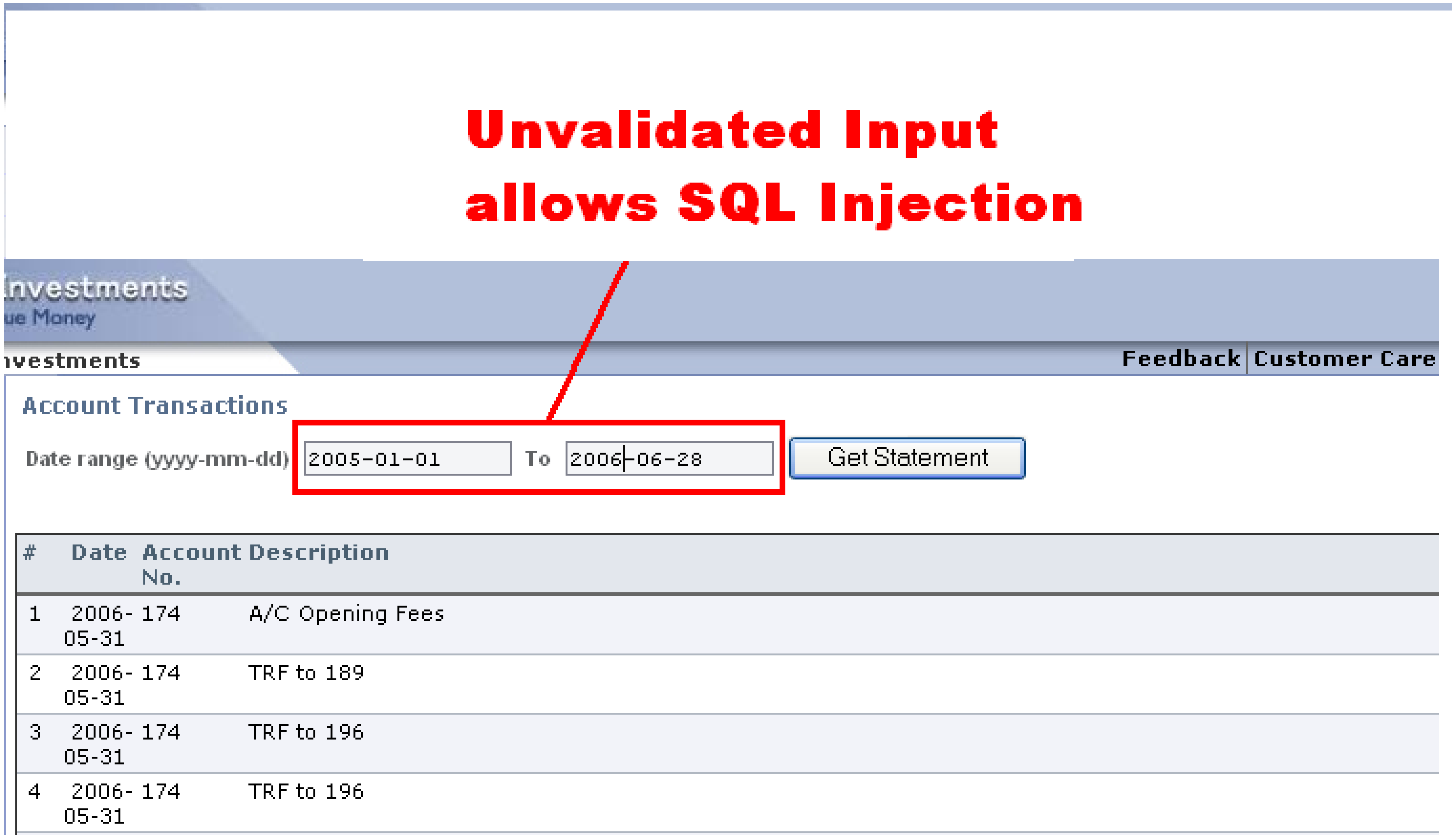
**Expected input:**

```
SELECT *FROM users
  WHERE login = ‘’ OR ‘1’=‘1’
  AND password = ‘’ OR ‘1’=‘1’
```

**Result:** Returns all user information in the users table

# SQL Injection – Date Type Example

Submitting SQL query logic instead of a valid date can expose confidential records.



Account Transactions

Date range (yyyy-mm-dd) 2005-01-01 To 2006-06-28

| # | Date | Account No. | Description |
|---|---|---|---|
| 1 | 2006-05-31 | 174 | A/C Opening Fees |
| 2 | 2006-05-31 | 174 | TRF to 189 |
| 3 | 2006-05-31 | 174 | TRF to 196 |
| 4 | 2006-05-31 | 174 | TRF to 196 |

From: www.itsa.ufl.edu/2006/presentations/malpani.ppt

Computer Science
NC STATE UNIVERSITY

# SQL Injection – Date Type Example

```
String query = “SELECT * FROM accounts
      WHERE username = ‘” + strUName + “’
      AND tran_date >= ‘” + strSDate + “’
      AND tran_date <= ‘” + strEDate + “’”;
```

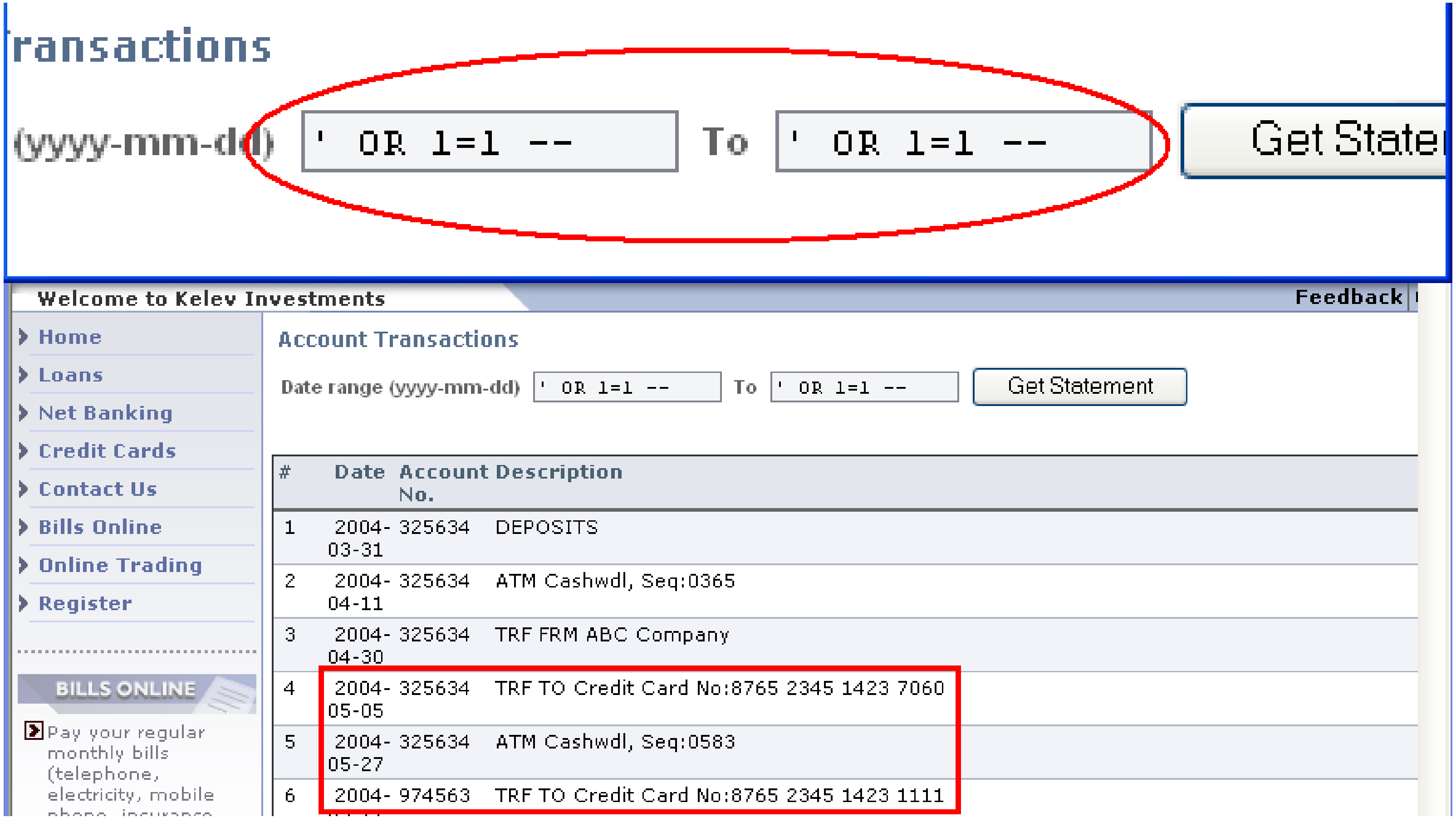
**Expected input:**

```
SELECT *FROM accounts
  WHERE username = ‘John’
  AND tran_date >= ‘2005-01-01’
  AND tran_date <= ‘2006-06-28’
```

**Result:** Returns John’s transactions between given dates

# SQL Injection – Date Type Example

Submitting SQL query logic instead of a valid date can expose confidential records.



(yyyy-mm-dd) ' OR 1=1 -- To ' OR 1=1 --

Date range (yyyy-mm-dd) ' OR 1=1 -- To ' OR 1=1 --

| # | Date | Account No. | Description |
|---|---|---|---|
| 1 | 2004-03-31 | 325634 | DEPOSITS |
| 2 | 2004-04-11 | 325634 | ATM Cashwdl, Seq:0365 |
| 3 | 2004-04-30 | 325634 | TRF FRM ABC Company |
| 4 | 2004-05-05 | 325634 | TRF TO Credit Card No:8765 2345 1423 7060 |
| 5 | 2004-05-27 | 325634 | ATM Cashwdl, Seq:0583 |
| 6 | 2004- | 974563 | TRF TO Credit Card No:8765 2345 1423 1111 |

Modified from: www.itsa.ufl.edu/2006/presentations/malpani.ppt

# SQL Injection – Date Type Example

```
String query = “SELECT * FROM accounts
      WHERE username = ‘” + strUName + “’
      AND tran_date >= ‘” + strSDate + “’
      AND tran_date <= ‘” + strEDate + “’”;
```

**Expected input:**

```
SELECT * FROM accounts
  WHERE username = ‘John’
  AND tran_date >= ‘’ OR 1=1 --’
  AND tran_date <= ‘’ OR 1=1 --’
```

**Result:** Returns all saved transactions

# SQL Injection – Drop Table

Could be any SQL command; add data; delete rows, etc.

- What if the attacker had instead entered:
  - **blah‘; DROP TABLE prodinfo; --**
- Results in the following SQL:
  - SELECT prodinfo FROM prodtable WHERE prodname = ‘**blah’; DROP TABLE prodinfo; --**’
  - Note how comment (--) consumes the final quote
- Causes the entire database to be deleted
  - Depends on knowledge of table name
  - This is sometimes exposed to the user in debug code called during a database error
  - Use non-obvious table names, and never expose them to user
- Usually data destruction is not your worst fear, as there is low economic motivation

Jim Whitehead http://www.soe.ucsc.edu/classes/cmps183/Spring06/lectures/SQL%20Injection%20Attacks.pdf

# Union Injections

## Return records from other tables or functions

```
SELECT header, txt FROM news
UNION ALL
SELECT name, pass FROM members
```

This will combine results from both news table and members table and return all of them.

```
$id = “%' or 0=0 union select null, version() #”

$getid = "SELECT first_name, last_name
               FROM users WHERE user_id = '$id'";
$result = mysql_query($getid);
```

http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/#UnionInjections
http://www.computersecuritystudent.com/SECURITY_TOOLS/DVWA/DVWAv107/lesson6/

# Mitigation -1

- Any security checks that are performed on the client side, ensure that these checks are duplicated on the server side. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely.
- Even though client-side checks provide minimal benefits with respect to server-side security, they are still useful.
  - They can support intrusion detection. If the server receives input that should have been rejected by the client, then it may be an indication of an attack.
  - Client-side error-checking can provide helpful feedback to the user about the expectations for valid input.
  - Reduction in server-side processing time for accidental input errors.

http://cwe.mitre.org/data/definitions/602.html

# Mitigation -2

- Blacklist – list that specifies the format of input that should be rejected ("foes")
  - Example: ' < > -- ; script
  - Enumerate the bad stuff
  - Don't allow anything on the blacklist
  - Drawback: infinite, easy to get around
  - Benefit: react quickly (often no re-compilation), straightforward
- Whitelist – list that specifies the format of input that should be accepted ("friends")
  - Only accept known good input
  - Often done with regex's
  - Drawbacks:
    - Sometimes not possible to block certain characters
    - Often requires re-compilation and patches
  - Benefit: finite

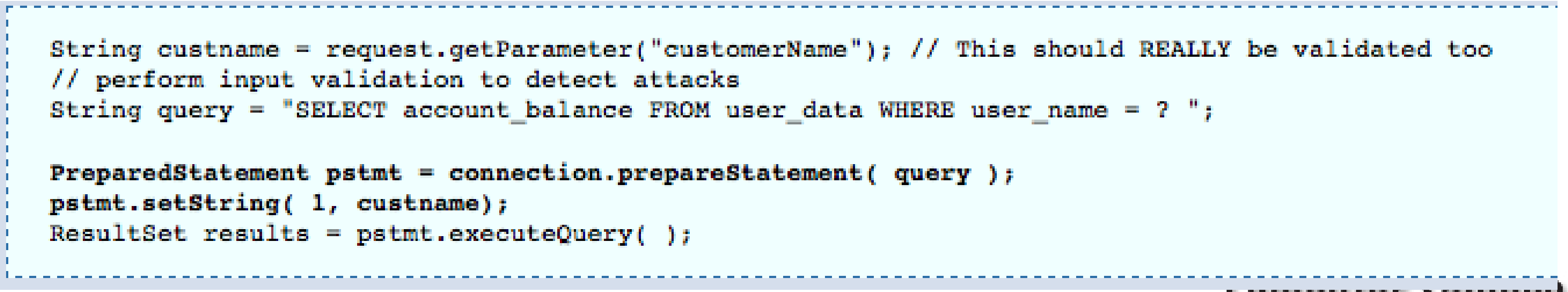
# Mitigation - 3

- For blacklists and whitelists:
  - Considerations: length, type, syntax, business rules, special characters, checksum
  - Usually expressed as regular expressions
- Instead of blocking input, *sanitize* it
  - All input comes in, but it's manipulated
  - Convert it to something that won't be interpreted as code
  - Usually utilizes escape characters
    - e.g. HTML
      - < is &lt;
    - e.g. Java
      - " is \"

https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

# Mitigation - 4

- Prepared Statements: Pre-compiled parameterized SQL queries
  - A setter method sets a value to a bind variable as well as performs strong type checking and will nullify the effect of invalid characters, such as single quotes in the middle of a string.
  - setString(index, input), sets the bind variable in the SQL structure indicated by the index to input



```
String custname = request.getParameter("customerName"); // This should REALLY be validated too
// perform input validation to detect attacks
String query = "SELECT account_balance FROM user_data WHERE user_name = ? ";

PreparedStatement pstmt = connection.prepareStatement( query );
pstmt.setString( 1, custname);
ResultSet results = pstmt.executeQuery( );
```

https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

# Mitigation - 5

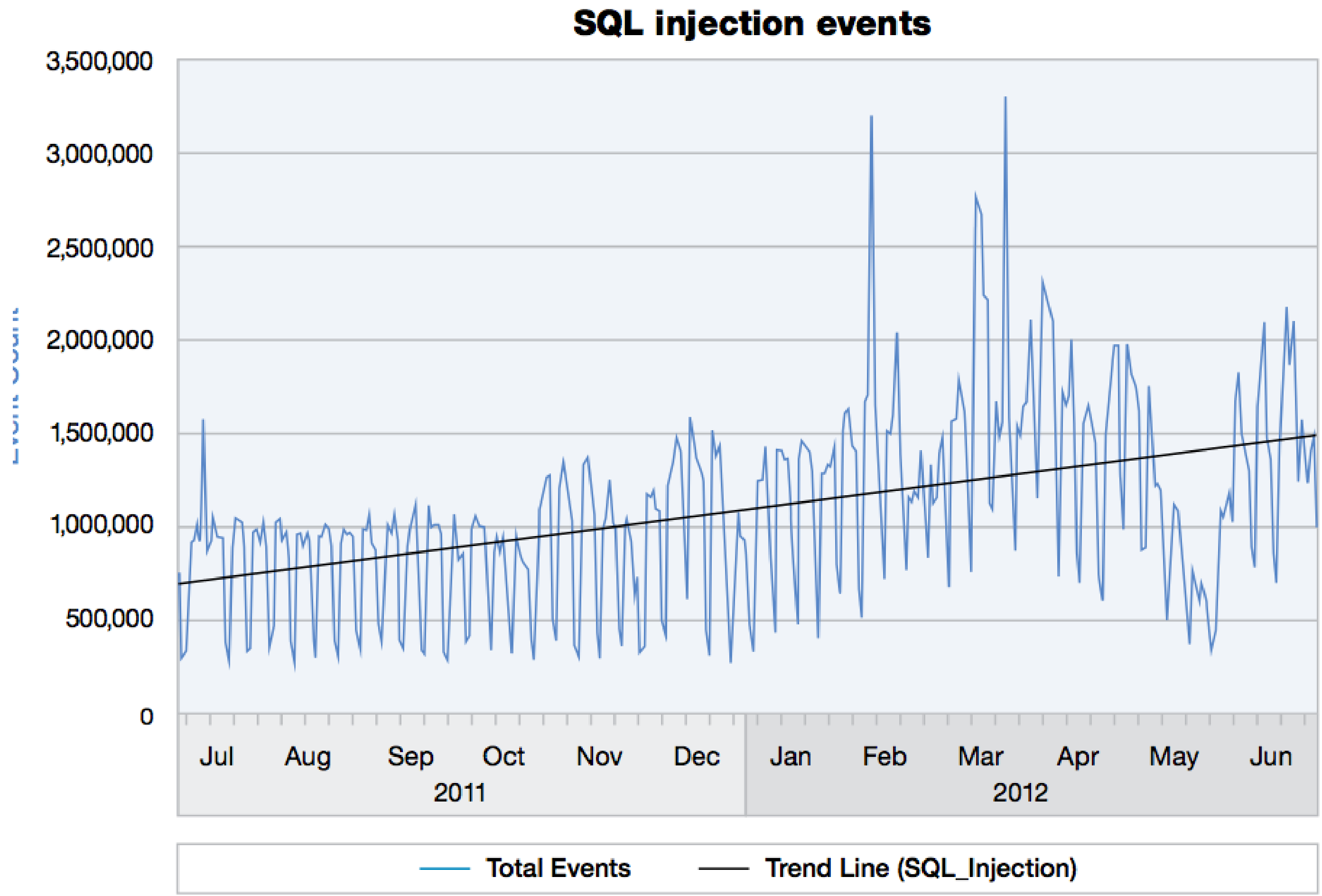
- Database Frameworks
  - e.g. Hibernate framework … use createQuery()
- Use stored procedures
- Limit database permissions
- Use static analysis tools which can be trained to find injection vulnerabilities

- All mitigation methods have holes or can be misused.
- Defense in depth … use several!

Computer Science

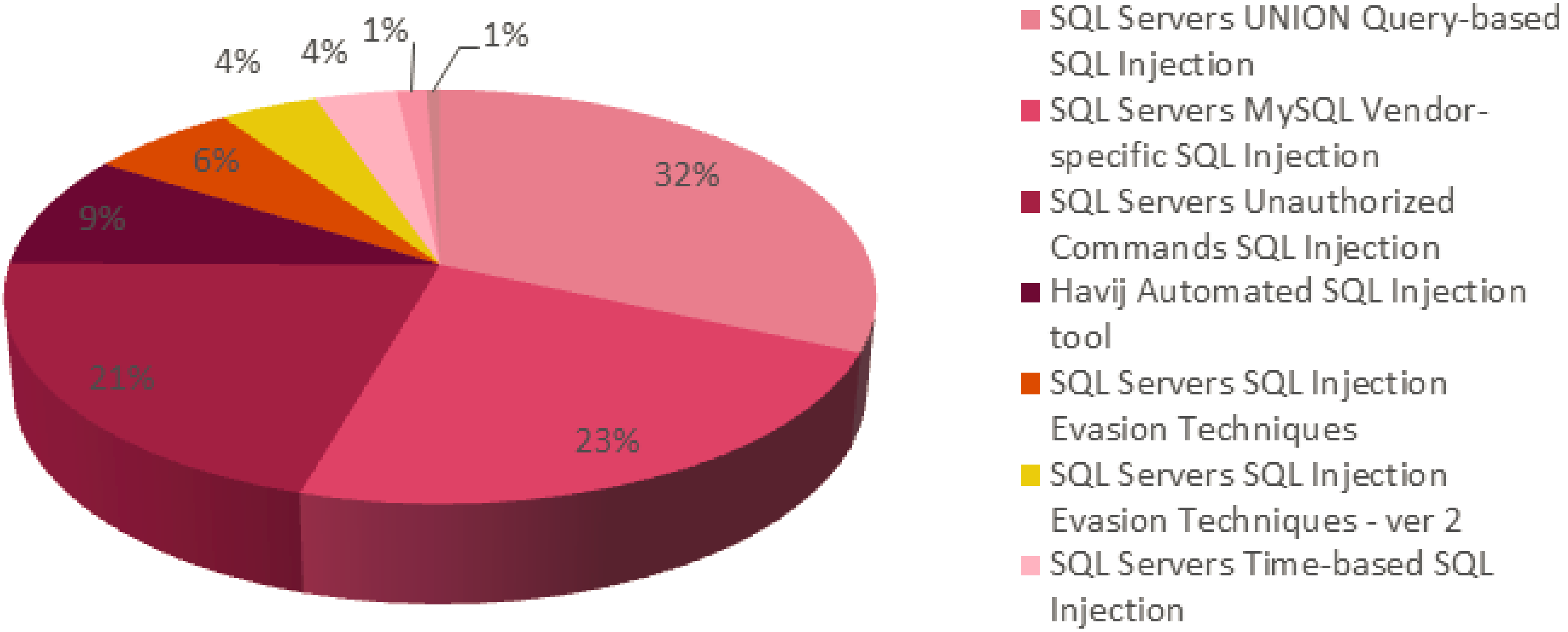
NC STATE UNIVERSITY

# Prevent and Mitigate



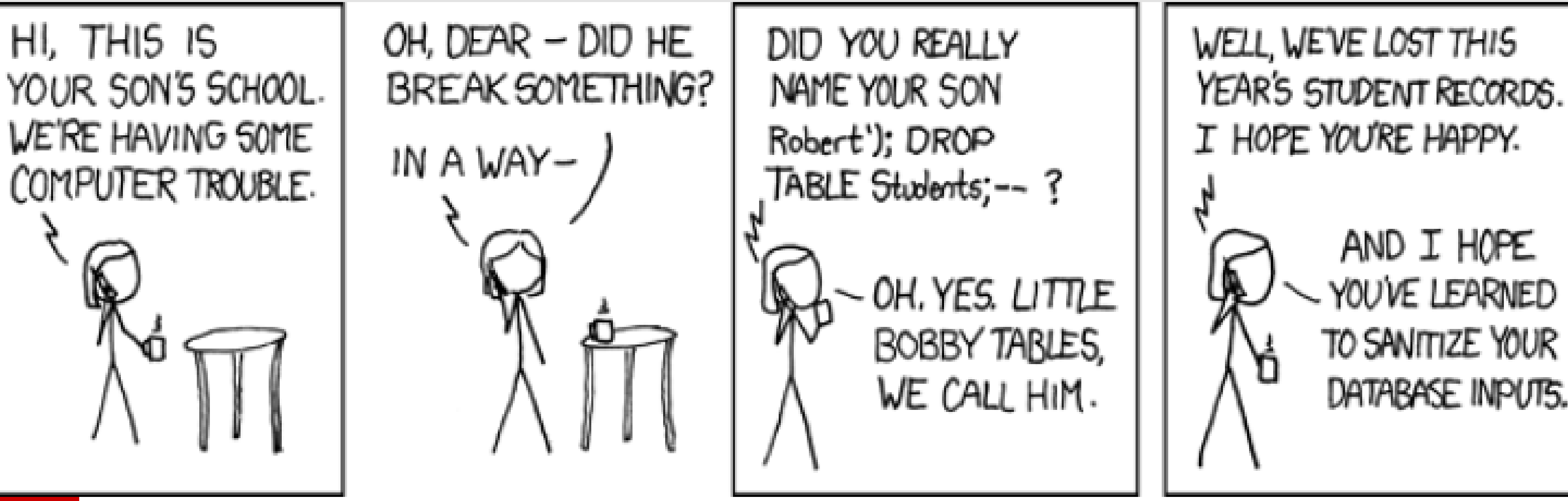
IBM X-force Report September 2012

# SQL Injection Trends



Source: http://blog.checkpoint.com/2015/05/07/latest-sql-injection-trends/

# Bobby Tables


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

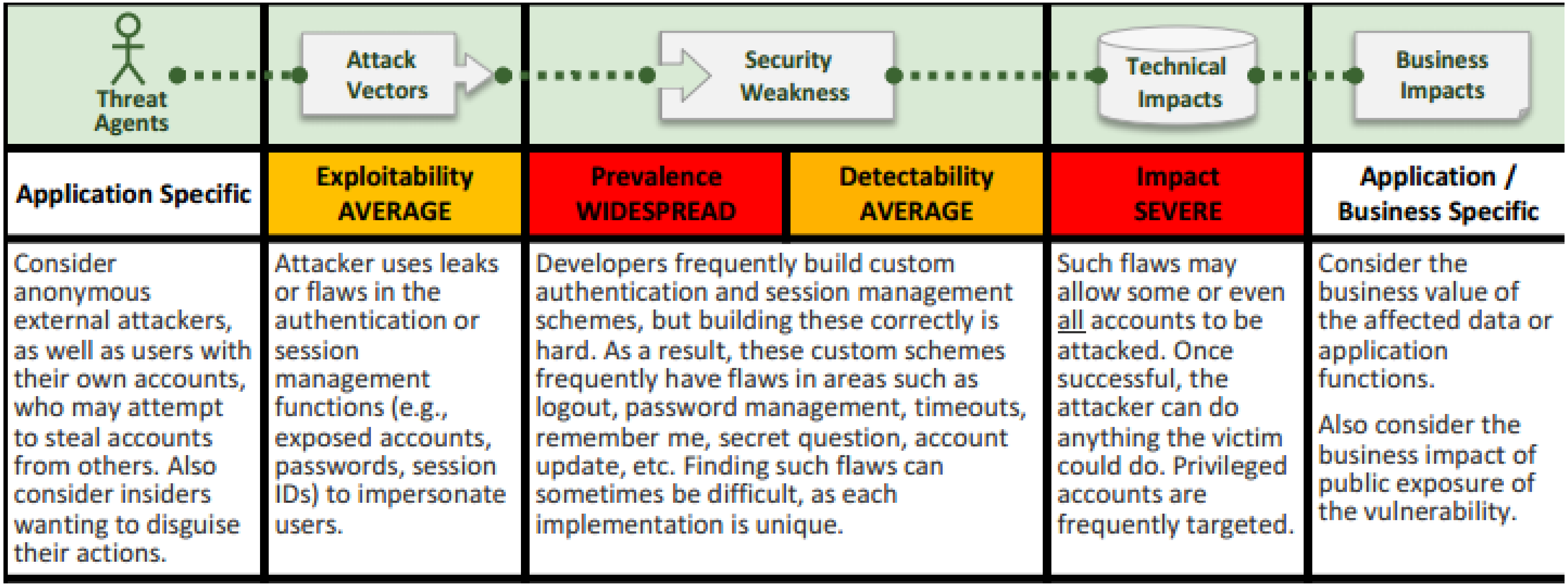
http://xkcd.com/327/

# Class Exercise

- Use DVWA /SQL Injection (Blind Injunction)
- Use “View Source” to see the PHP serverside script
- Goal: Find the passwords

# A2 – Broken Authentication and Session Management

# A2 Overview



| Threat Agents | Attack Vectors | Security Weakness | | Technical Impacts | Business Impacts |
|---|---|---|---|---|---|
| Application Specific | Exploitability<br>AVERAGE | Prevalence<br>WIDESPREAD | Detectability<br>AVERAGE | Impact<br>SEVERE | Application / Business Specific |
| Consider anonymous external attackers, as well as users with their own accounts, who may attempt to steal accounts from others. Also consider insiders wanting to disguise their actions. | Attacker uses leaks or flaws in the authentication or session management functions (e.g., exposed accounts, passwords, session IDs) to impersonate users. | Developers frequently build custom authentication and session management schemes, but building these correctly is hard. As a result, these custom schemes frequently have flaws in areas such as logout, password management, timeouts, remember me, secret question, account update, etc. Finding such flaws can sometimes be difficult, as each implementation is unique. | | Such flaws may allow some or even all accounts to be attacked. Once successful, the attacker can do anything the victim could do. Privileged accounts are frequently targeted. | Consider the business value of the affected data or application functions.<br>Also consider the business impact of public exposure of the vulnerability. |

https://www.owasp.org/index.php/Top_10_2013-A2-Broken_Authentication_and_Session_Management

# Session ID

- Session ID = a piece of data that is used in network communications to identify a session
  - Session = a series of related message exchanges
- Session ID is often a long, hashed, randomly generated string to decrease the probability of obtaining a valid one by means of a brute-force search
- Session ID become necessary when the communications infrastructure uses a stateless protocol, such as HTTP
- Session ID typically granted when a user first enters a site and are short lived
  - May expire after a preset time of inactivity
  - May become invalid after a certain goal has been met (for example, once the buyer has finalized his order, he cannot use the same session ID to add more items).
- As session IDs are often used to identify a user that has logged into a website, they can be used by an attacker to hijack a session and obtain potential privileges.

http://en.wikipedia.org/wiki/Session_ID

# You may be vulnerable if:

- User authentication credentials aren't protected when stored using hashing or encryption.
- Credentials can be guessed or overwritten through weak account management functions (e.g., account creation, change password, recover password, weak session IDs).
- Session IDs are exposed in the URL (e.g., URL rewriting) or hidden field.
- Session IDs don't timeout, or user sessions or authentication tokens, particularly single sign-on (SSO) tokens, aren't properly invalidated during logout.
- Passwords, session IDs, and other credentials are sent over unencrypted connections.
- Session ID can be obtained through XSS attack

https://www.owasp.org/index.php/Top_10_2013-A2-Broken_Authentication_and_Session_Management

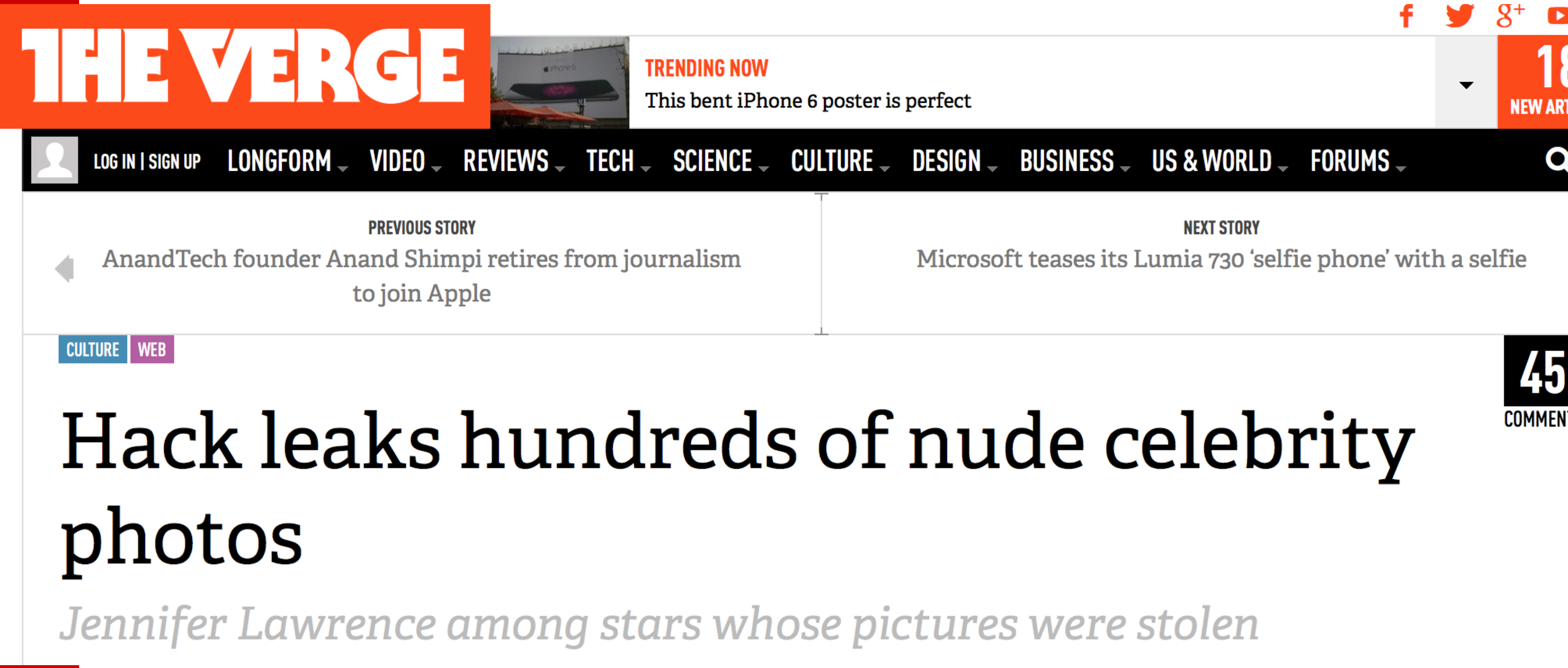
# Example Attacks

- **Scenario #1:** Airline reservations application supports URL rewriting, putting session IDs in the URL:
  - http://example.com/sale/saleitems jsessionid=2P0OC2JSNDLPSKHCJUN2JV ?dest=Hawaii

    An authenticated user of the site wants to let his friends know about the sale. He e-mails the above link without knowing he is also giving away his session ID. When his friends use the link they will use his session and credit card.
- **Scenario #2:** Application's timeouts aren't set properly. User uses a public computer to access site. Instead of selecting "logout" the user simply closes the browser tab and walks away. Attacker uses the same browser an hour later, and that browser is still authenticated.
- **Scenario #3:** Insider or external attacker gains access to the system's password database. User passwords are not properly hashed, exposing every users' password to the attacker.

https://www.owasp.org/index.php/Top_10_2013-A2-Broken_Authentication_and_Session_Management

# Bad password management …



TRENDING NOW
This bent iPhone 6 poster is perfect

PREVIOUS STORY
AnandTech founder Anand Shimpi retires from journalism to join Apple

NEXT STORY
Microsoft teases its Lumia 730 'selfie phone' with a selfie

CULTURE WEB

## Hack leaks hundreds of nude celebrity photos

*Jennifer Lawrence among stars whose pictures were stolen*

# OWASP Password Guidelines

- Typical guidelines:
  - Password must meet at least 3 out of the following 4 complexity rules
    - at least 1 uppercase character (A-Z)
    - at least 1 lowercase character (a-z)
    - at least 1 digit (0-9)
    - at least 1 special character
- at least 10 characters
- at most 128 characters … but enable passphrases
- Require the current credentials for an account before updating sensitive account information such as the user's password, user's email, or before sensitive transactions, such as shipping a purchase to a new address

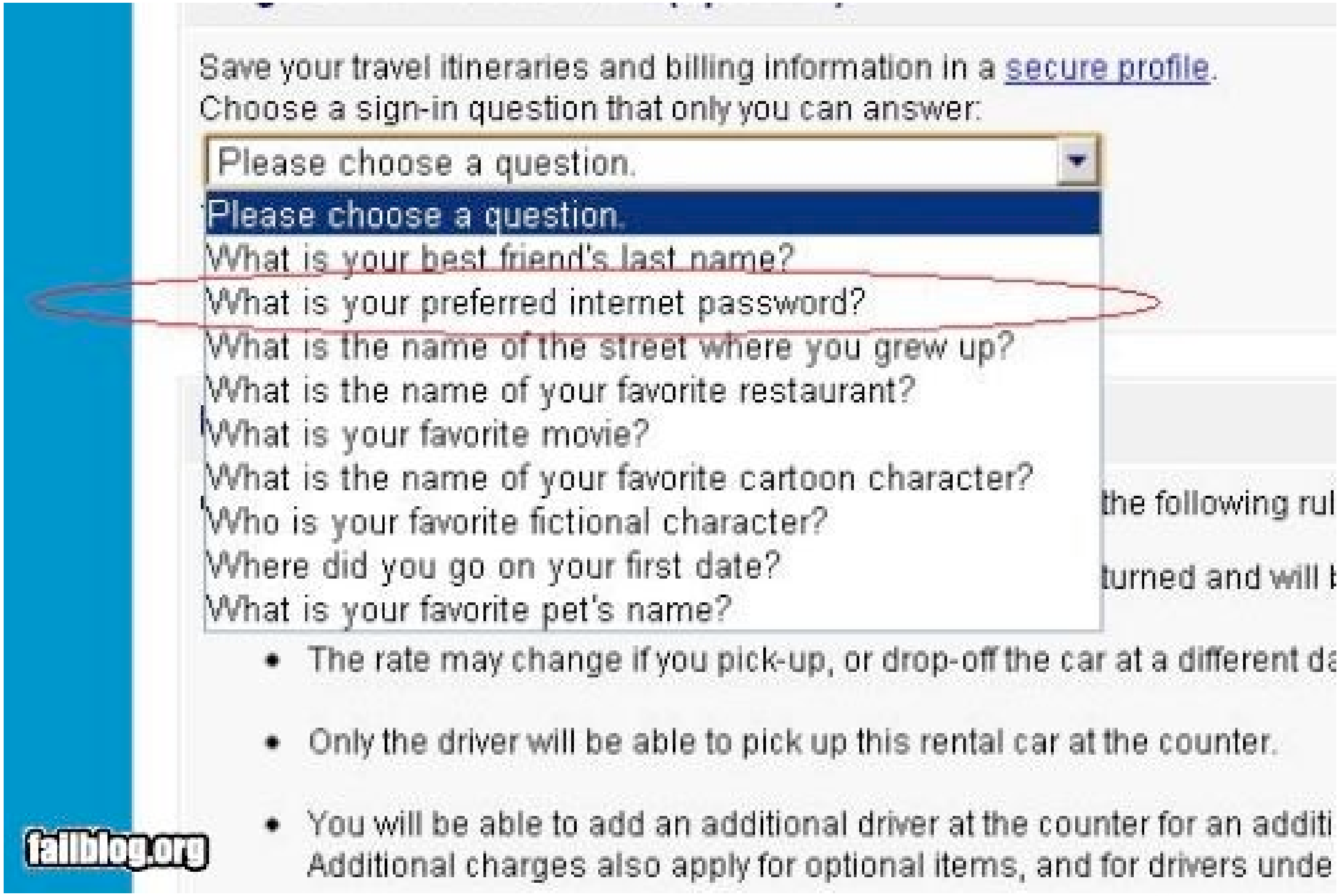
https://www.owasp.org/index.php/Authentication_Cheat_Sheet

# Forgot password

1. Gather Identity Data or 1-2 Security Questions
   - May be available through organization already
   - Answers to questions should not be easily available through web searching or social engineering
   - Answers should be stable
2. Verify Security Questions
   - All or nothing on answering questions correctly
   - Limit number of tries
3. Send a Token Over a Side-Channel (multi-factor)
   - SMS or email
   - Limited lifetime (~ 20 minutes)
4. Allow user to change password in the existing session

https://www.owasp.org/index.php/Forgot_Password_Cheat_Sheet

# Security Questions



Save your travel itineraries and billing information in a secure profile.
Choose a sign-in question that only you can answer:

- Please choose a question.
- What is your best friend's last name?
- What is your preferred internet password?
- What is the name of the street where you grew up?
- What is the name of your favorite restaurant?
- What is your favorite movie?
- What is the name of your favorite cartoon character?
- Who is your favorite fictional character?
- Where did you go on your first date?
- What is your favorite pet's name?

the following rul

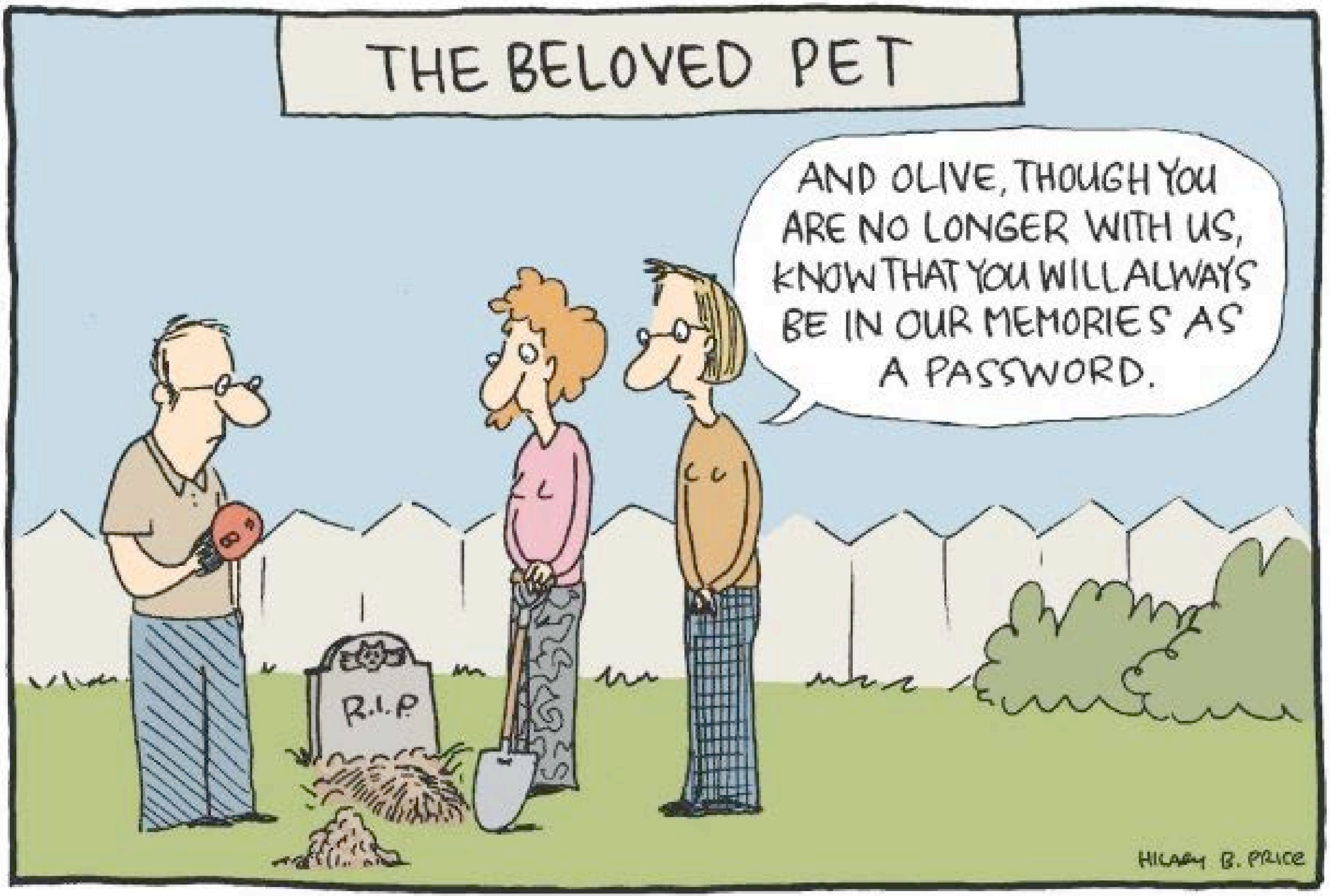
turned and will b

- The rate may change if you pick-up, or drop-off the car at a different da
- Only the driver will be able to pick up this rental car at the counter.
- You will be able to add an additional driver at the counter for an additi Additional charges also apply for optional items, and for drivers unde

Computer Science
NC STATE UNIVERSITY

# Good Practices

- Don’t pass SessionIDs in URL. Instead use non-persistent, httpOnly (so it is unavailable from client code) cookies instead
- Check that all requests in a session originate from the same address
- Session rotation (reassign session ID periodically) and Session timeout
  - Balance between security and usability
- Hash passwords
- Encrypt essential data, such as security questions
- Balance convenience and the ability to “remember me”
  - Consider bank versus Facebook
- Use a CAPTCHA to prevent automated attacks
- Consider multi-factor authentication (MFA): using more than one authentication factor to logon or process a transaction:
  - Something you <u>know</u> (account details or passwords)
  - Something you <u>have</u> (tokens or mobile phones)
  - Something you <u>are</u> (biometrics)

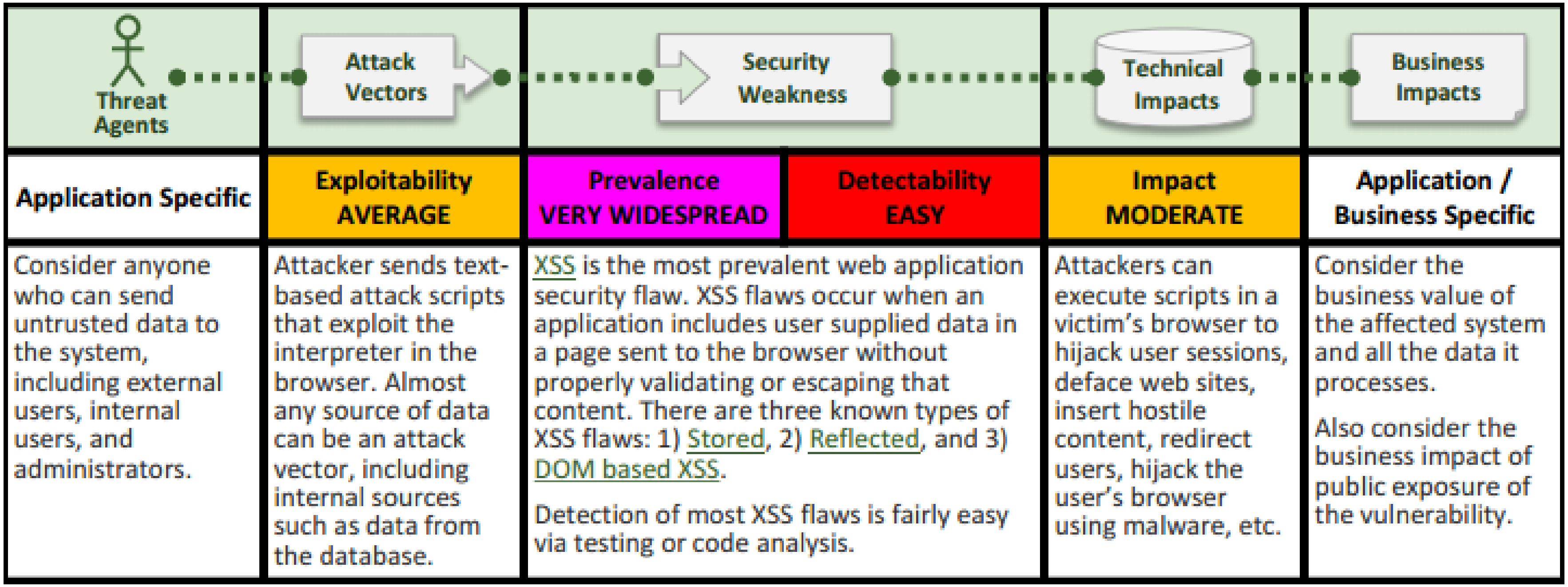

THE BELOVED PET
AND OLIVE, THOUGH YOU ARE NO LONGER WITH US, KNOW THAT YOU WILL ALWAYS BE IN OUR MEMORIES AS A PASSWORD.
R.I.P.
HILARY B. PRICE

http://www.weeklystorybook.com/comic_strip_of_the_daycom/2012/12/grooming-your-semi-public-image.html

Computer Science
NC STATE UNIVERSITY

# Class Exercise

# A3 – Cross-Site Scripting (XSS)



| Threat Agents | Attack Vectors | Security Weakness | | Technical Impacts | Business Impacts |
|---|---|---|---|---|---|
| **Application Specific** | **Exploitability**<br>**AVERAGE** | **Prevalence**<br>**VERY WIDESPREAD** | **Detectability**<br>**EASY** | **Impact**<br>**MODERATE** | **Application / Business Specific** |
| Consider anyone who can send untrusted data to the system, including external users, internal users, and administrators. | Attacker sends text-based attack scripts that exploit the interpreter in the browser. Almost any source of data can be an attack vector, including internal sources such as data from the database. | XSS is the most prevalent web application security flaw. XSS flaws occur when an application includes user supplied data in a page sent to the browser without properly validating or escaping that content. There are three known types of XSS flaws: 1) Stored, 2) Reflected, and 3) DOM based XSS.<br>Detection of most XSS flaws is fairly easy via testing or code analysis. | | Attackers can execute scripts in a victim's browser to hijack user sessions, deface web sites, insert hostile content, redirect users, hijack the user's browser using malware, etc. | Consider the business value of the affected system and all the data it processes.<br>Also consider the business impact of public exposure of the vulnerability. |

# Cross-site Scripting (XSS)

- Condition: When untrusted input from the server or client is used to construct dynamic HTML web pages.

- Consequence: An attack can control the appearance of a web site, transfer sensitive data, and hijack the session to take control of the user's account.

- The vulnerability results because some data (input) is interpreted to be instructions (code) by the browser. In all XSS attacks, this execution of code is performed in the context of the vulnerable server.

# Cross Site Scripting (XSS)

- Web application takes input from a user but fails to validate the input
- Input is echoed directly in a web page.
- Input could be malicious JavaScript, when echoed and interpreted in the destination browser any number of issues could result

Possible attack:

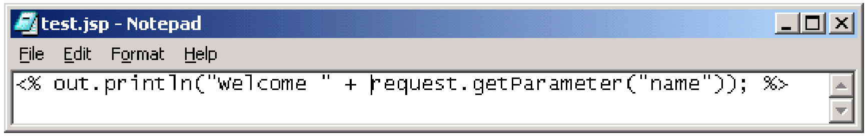
- When the victim is tricked to click on a crafted link (via web server or email), he is referred to the host in the URL
- The host processes the query string and echoes it to the victim's browser,
- The victim's browser executes the malicious script.

[1] "Build Security In" https://buildsecurityin.us-cert.gov/daisy/bsi/articles/knowledge/guidelines/342.html

# Cross Site Scripting

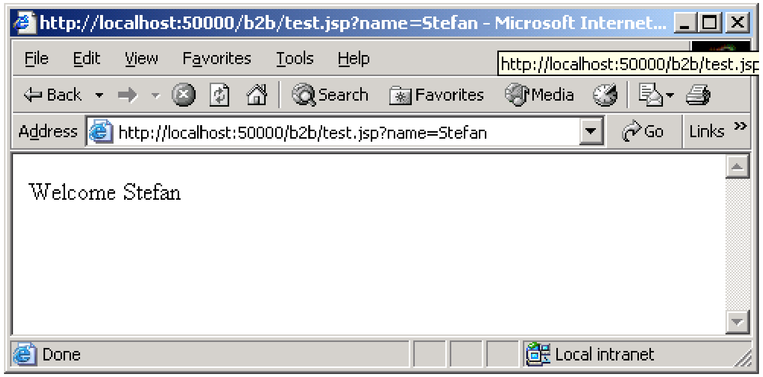
- Reflected XSS
  - Attacker-provided script is embedded in the web page generated by the server as an immediate response of an HTTP request or in an email.  The client executes code in the context of the current user.
- Stored XSS
  - Attacker-provided script is stored to a database or other persistent storage.  Later, the script is retrieved and embedded in the web page generated by the server.

# Cross Site Scripting – Reflected XSS



```
<% out.println("welcome " + request.getParameter("name")); %>
```

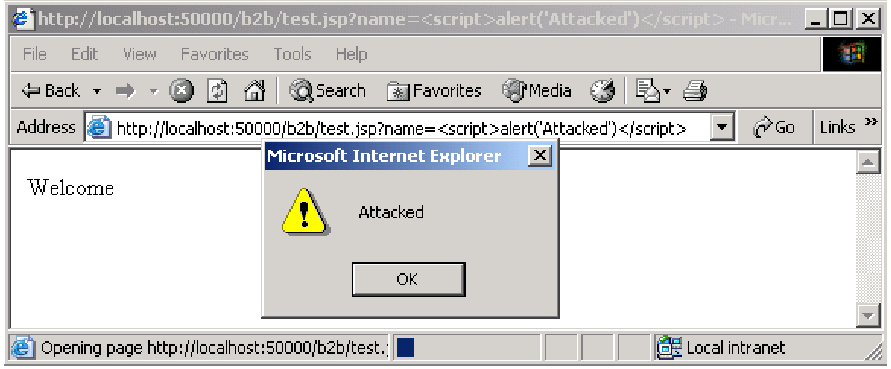
http://myserver.com/test.jsp?name=Stefan


http://localhost:50000/b2b/test.jsp?name=Stefan - Microsoft Internet...
File Edit View Favorites Tools Help
http://localhost:50000/b2b/test.jsp
Back Search Favorites Media
Address http://localhost:50000/b2b/test.jsp?name=Stefan
Go Links
Done
Local intranet

Welcome Stefan

```
<HTML>
<Body>
Welcome Stefan
</Body>
</HTML>
```

http://myserver.com/welcome.jsp?name=<script>alert("Attacked")</script>


http://localhost:50000/b2b/test.jsp?name=<script>alert('Attacked')</script> - Micr...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:50000/b2b/test.jsp?name=<script>alert('Attacked')</script>
Go Links
Microsoft Internet Explorer
OK
Opening page http://localhost:50000/b2b/test.;
Local intranet

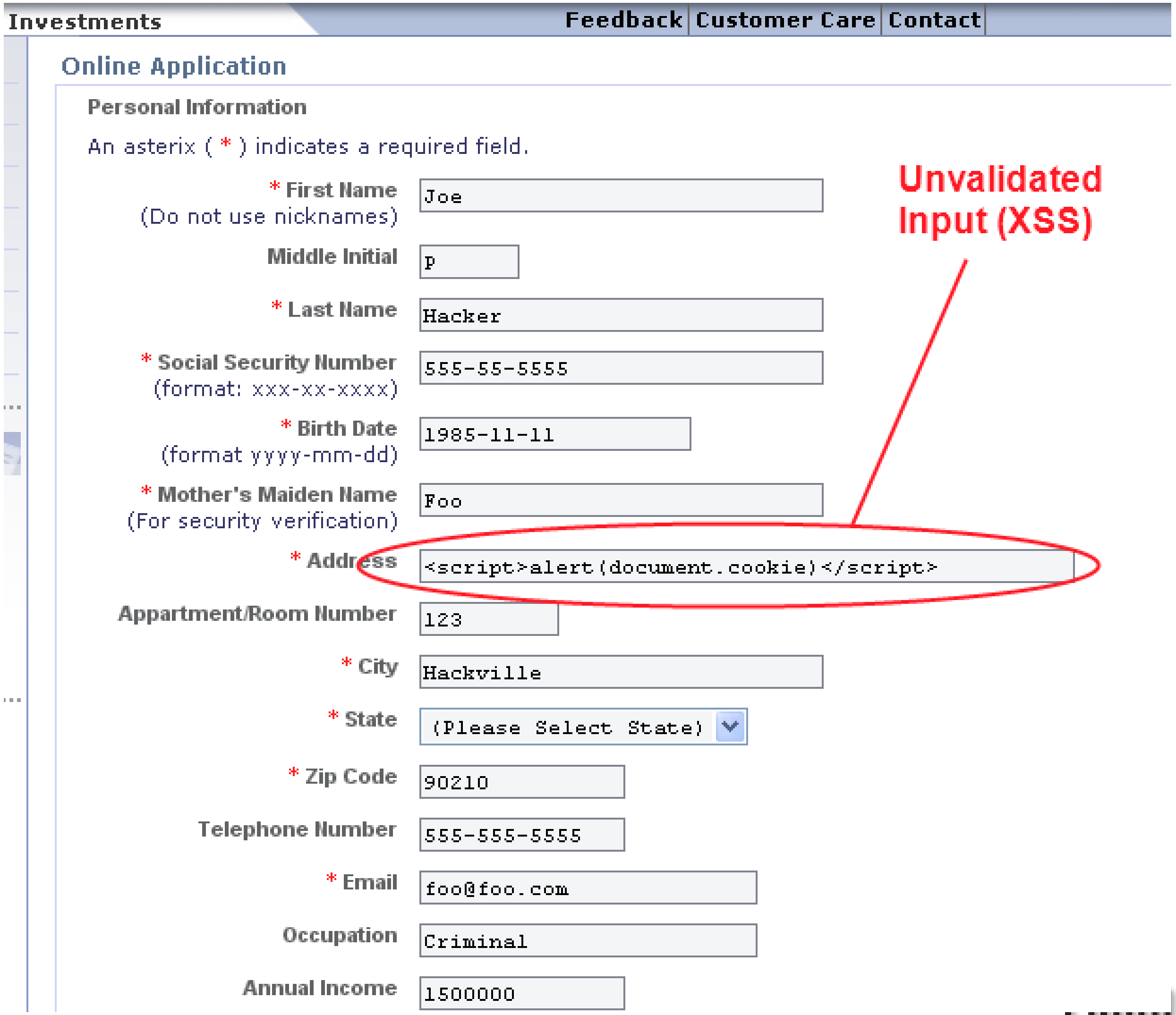
Welcome

Attacked

```
<HTML>
<Body>
Welcome <script>alert("Attacked")</script>
</Body>
</HTML>
```

Computer Science
NC STATE UNIVERSITY

# Cross Site Scripting – Stored XSS



Online Application

Personal Information

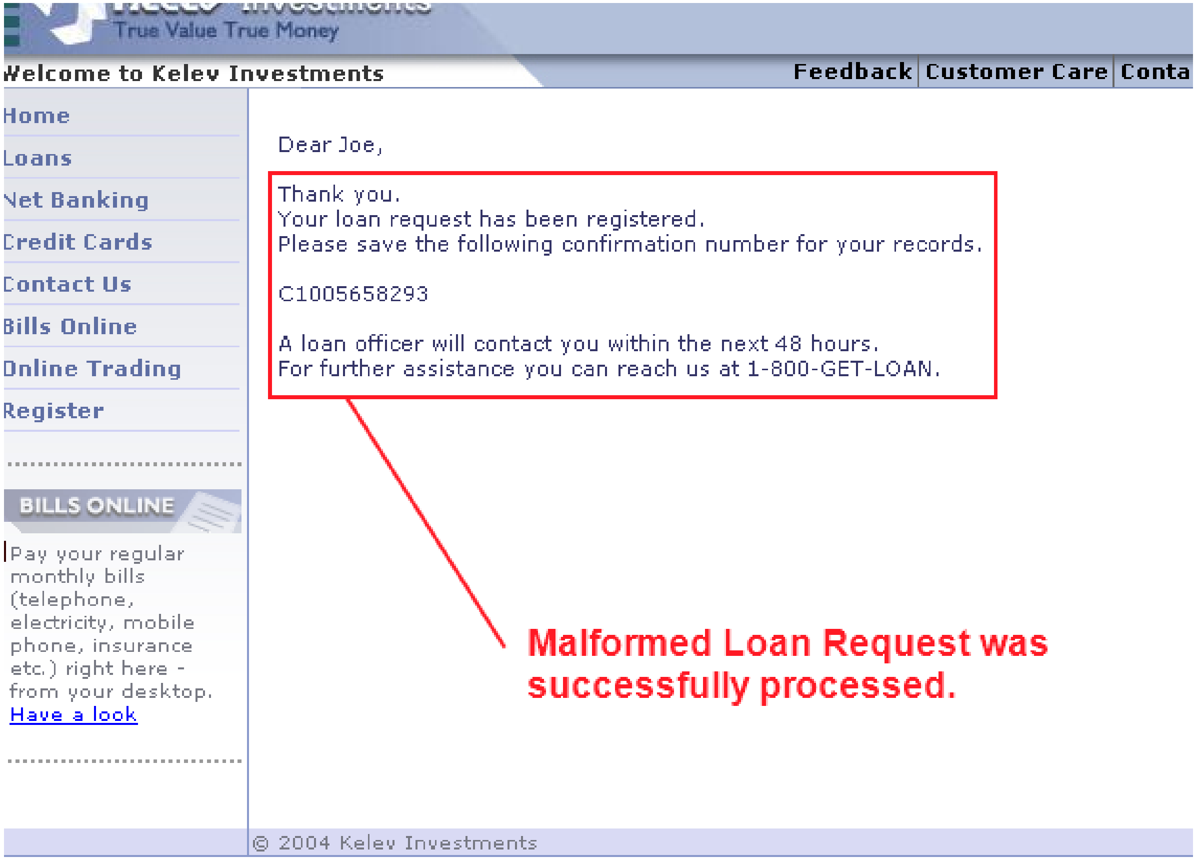
An asterix ( * ) indicates a required field.

| | |
|---|---|
| * First Name<br>(Do not use nicknames) | Joe |
| Middle Initial | P |
| * Last Name | Hacker |
| * Social Security Number<br>(format: xxx-xx-xxxx) | 555-55-5555 |
| * Birth Date<br>(format yyyy-mm-dd) | 1985-11-11 |
| * Mother's Maiden Name<br>(For security verification) | Foo |
| * Address | <script>alert(document.cookie)</script> |
| Appartment/Room Number | 123 |
| * City | Hackville |
| * State | (Please Select State) |
| * Zip Code | 90210 |
| Telephone Number | 555-555-5555 |
| * Email | foo@foo.com |
| Occupation | Criminal |
| Annual Income | 1500000 |

Unvalidated Input (XSS)

From: www.itsa.ufl.edu/2006/presentations/malpani.ppt

# Cross Site Scripting – Stored XSS



Dear Joe,

Thank you.
Your loan request has been registered.
Please save the following confirmation number for your records.

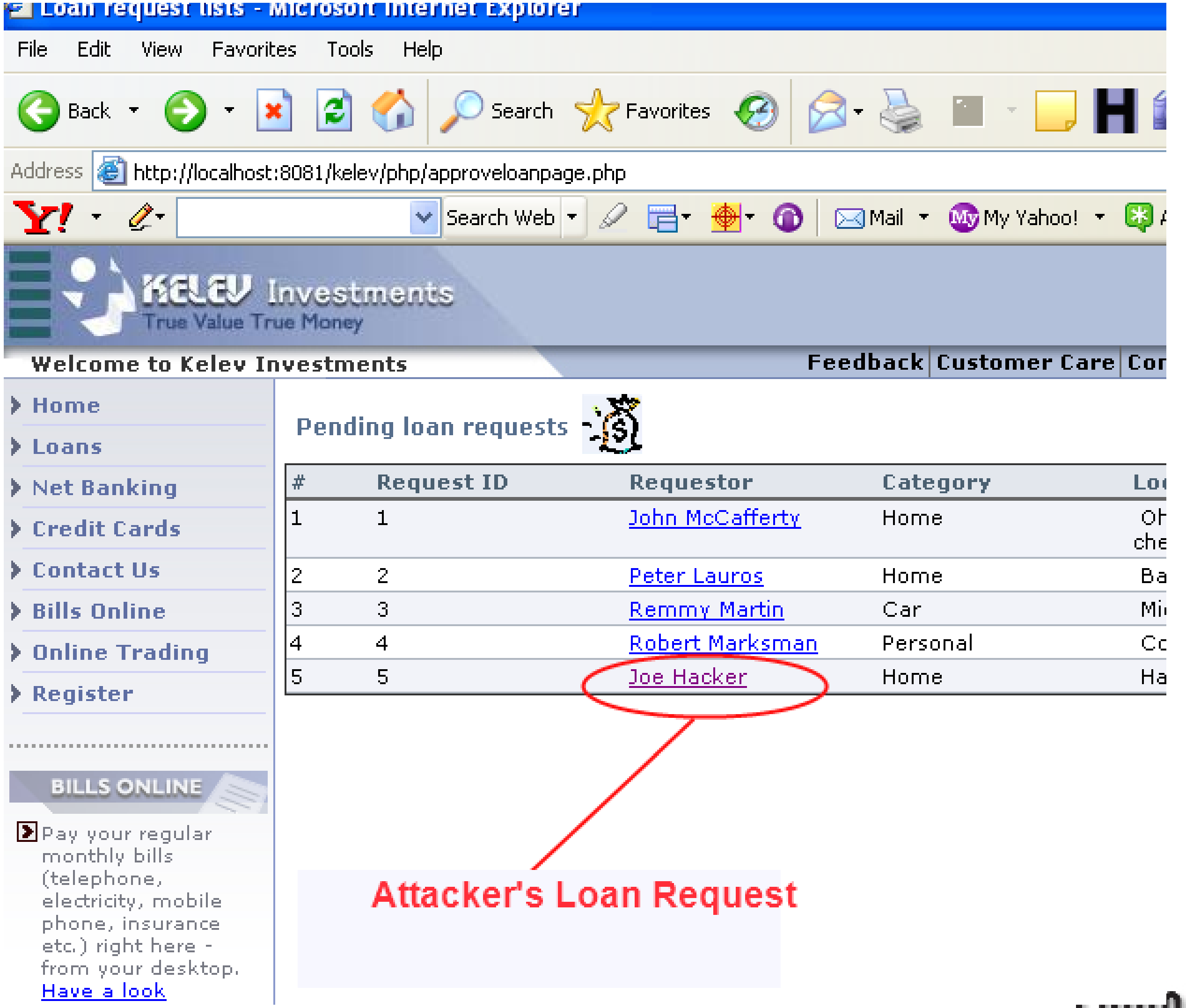
C1005658293

A loan officer will contact you within the next 48 hours.
For further assistance you can reach us at 1-800-GET-LOAN.

Malformed Loan Request was successfully processed.

From: www.itsa.ufl.edu/2006/presentations/malpani.ppt

# Cross Site Scripting – Stored XSS



http://localhost:8081/kelev/php/approveloanpage.php

Pending loan requests

| # | Request ID | Requestor | Category | Lo |
|---|---|---|---|---|
| 1 | 1 | John McCafferty | Home | Ol<br>che |
| 2 | 2 | Peter Lauros | Home | Ba |
| 3 | 3 | Remmy Martin | Car | Mi |
| 4 | 4 | Robert Marksman | Personal | Cc |
| 5 | 5 | Joe Hacker | Home | Ha |

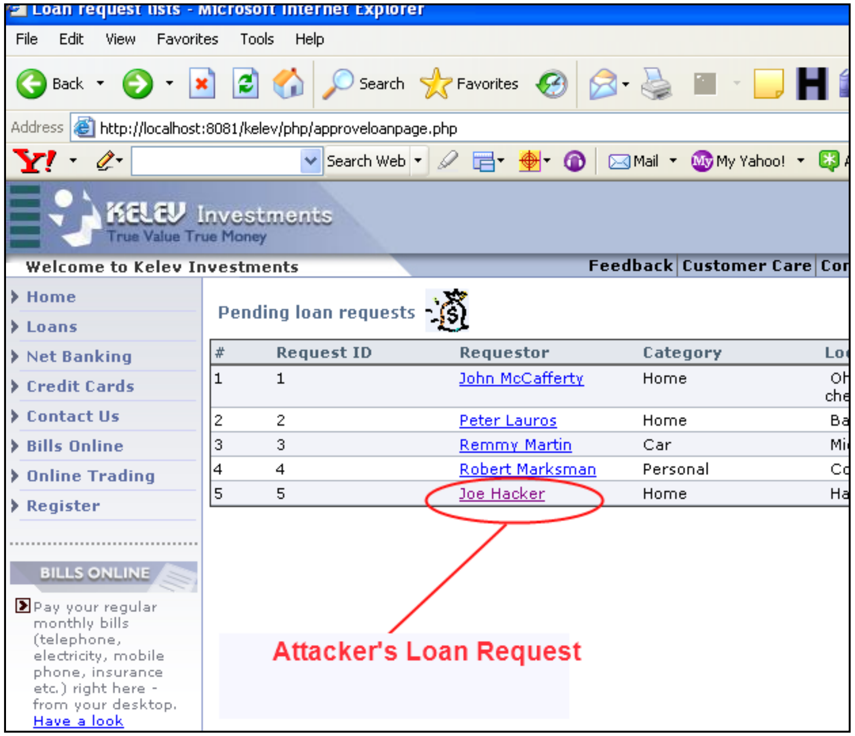
Pay your regular monthly bills (telephone, electricity, mobile phone, insurance etc.) right here - from your desktop. Have a look

Attacker's Loan Request

From: www.itsa.ufl.edu/2006/presentations/malpani.ppt

Computer Science
NC STATE UNIVERSITY

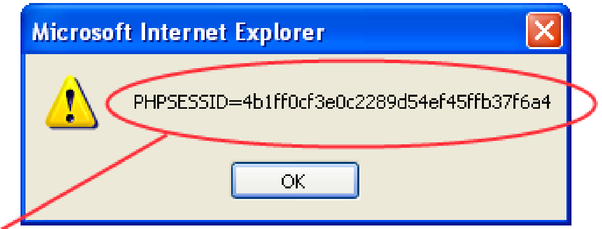
# Cross Site Scripting – Stored XSS


Loan request lists - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8081/kelev/php/approveloanpage.php
Search Web Mail My Yahoo!
KELEV Investments
True Value True Money
Welcome to Kelev Investments
Feedback Customer Care Cor
Home
Loans
Net Banking
Credit Cards
Contact Us
Bills Online
Online Trading
Register
BILLS ONLINE
Pay your regular monthly bills (telephone, electricity, mobile phone, insurance etc.) right here - from your desktop.
Have a look

Pending loan requests

| # | Request ID | Requestor | Category | Lo |
|---|---|---|---|---|
| 1 | 1 | John McCafferty | Home | Oh<br>che |
| 2 | 2 | Peter Lauros | Home | Ba |
| 3 | 3 | Remmy Martin | Car | Mi |
| 4 | 4 | Robert Marksman | Personal | Co |
| 5 | 5 | Joe Hacker | Home | Ha |

Attacker's Loan Request



PHPSESSID=4b1ff0cf3e0c2289d54ef45ffb37f6a4

Unvalidated Input resulted in a Cross-Site Scripting Attack and the theft of the Administrator's Cookie

From: www.itsa.ufl.edu/2006/presentations/malpani.ppt

# A script that causes a denial of service …

```
article.php?title=<meta%20http-
equiv="refresh"%20content="0;">
```

# Mitigation

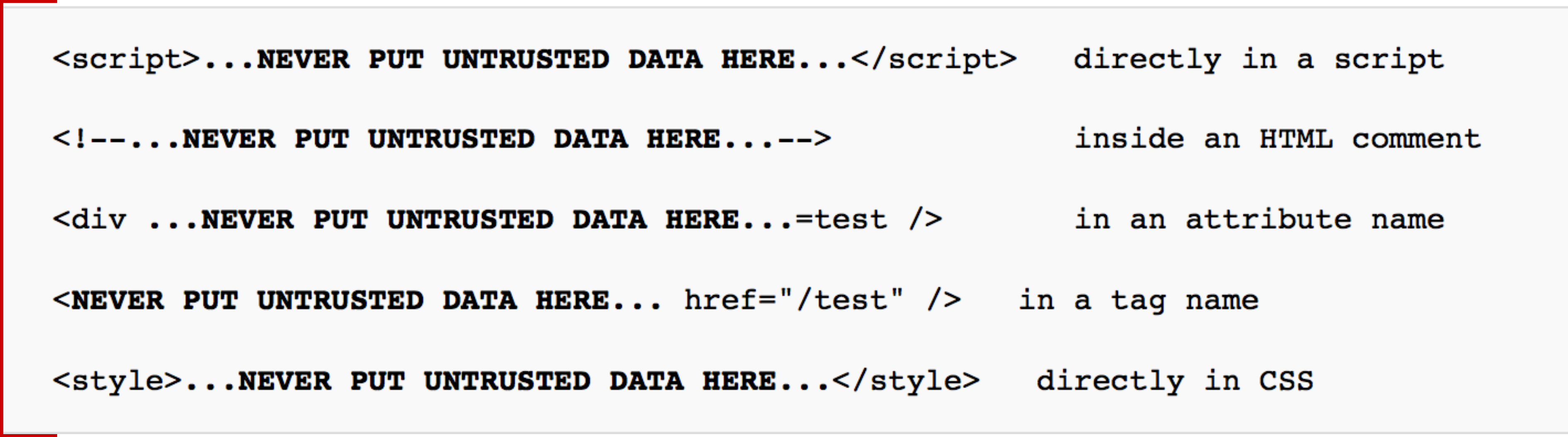
- Input filtering (blacklist/whitelist)
- Output filtering (blacklist/whitelist)
- Available … output encoding libraries
  - Microsoft's Anti-XSS library
  - OWASP ESAPI Encoding module
  - Encoding rules depend on context (HTML, HTML attribute, Script, URL query string, etc)
- The HttpOnly flag indicates that the cookie should not be available to the client script. Secure cookies indicate that the cookie holds sensitive data and should be sent only over an encrypted channel.

# Testing for XSS

- Step 1: Identify where untrusted input can be used as output
  - Welcome message, error message, etc.
- Step 2: Test whether the input is not validated and valid HTML and script code can be executed

```
<script>alert(“XSS”)</script>
```

# XSS Prevention



```
<script>...NEVER PUT UNTRUSTED DATA HERE...</script>   directly in a script

<!--...NEVER PUT UNTRUSTED DATA HERE...-->             inside an HTML comment

<div ...NEVER PUT UNTRUSTED DATA HERE...=test />       in an attribute name

<NEVER PUT UNTRUSTED DATA HERE... href="/test" />   in a tag name

<style>...NEVER PUT UNTRUSTED DATA HERE...</style>   directly in CSS
```

https://www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet

# XSS Resources

OWASP:

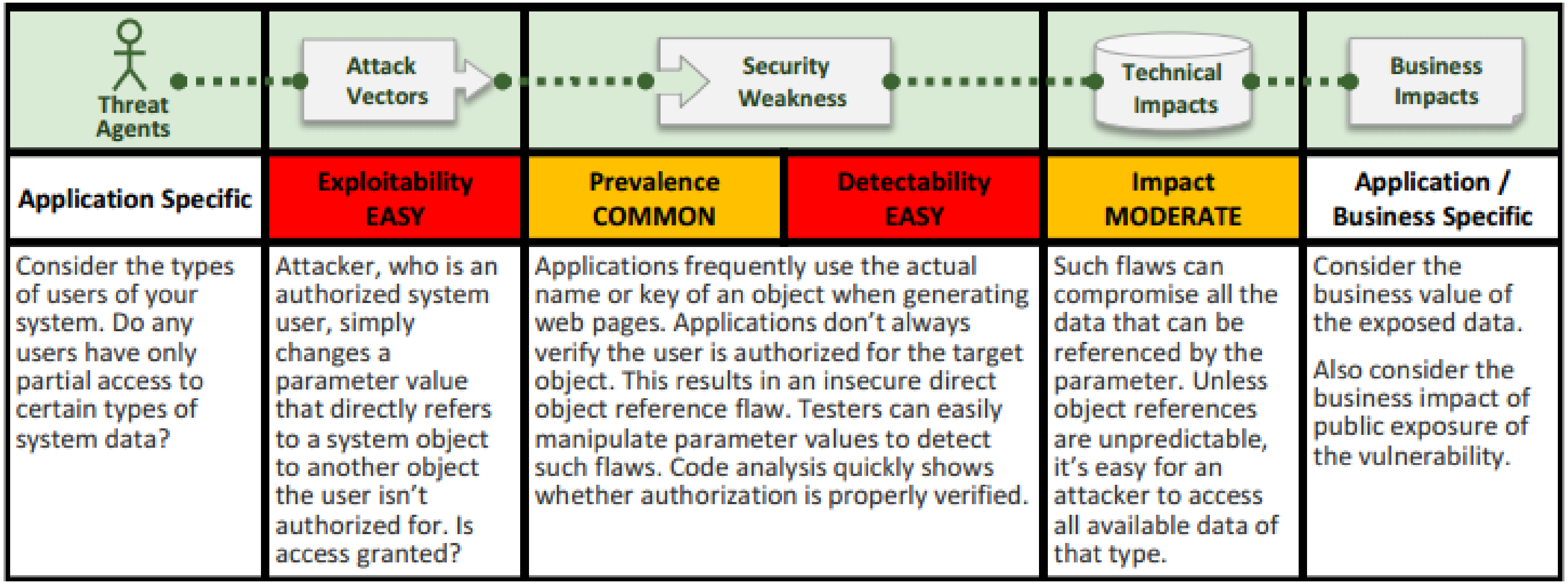
- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

Google Application Security:
https://www.google.com/about/appsecurity/learning/xss/

# Class Exercise

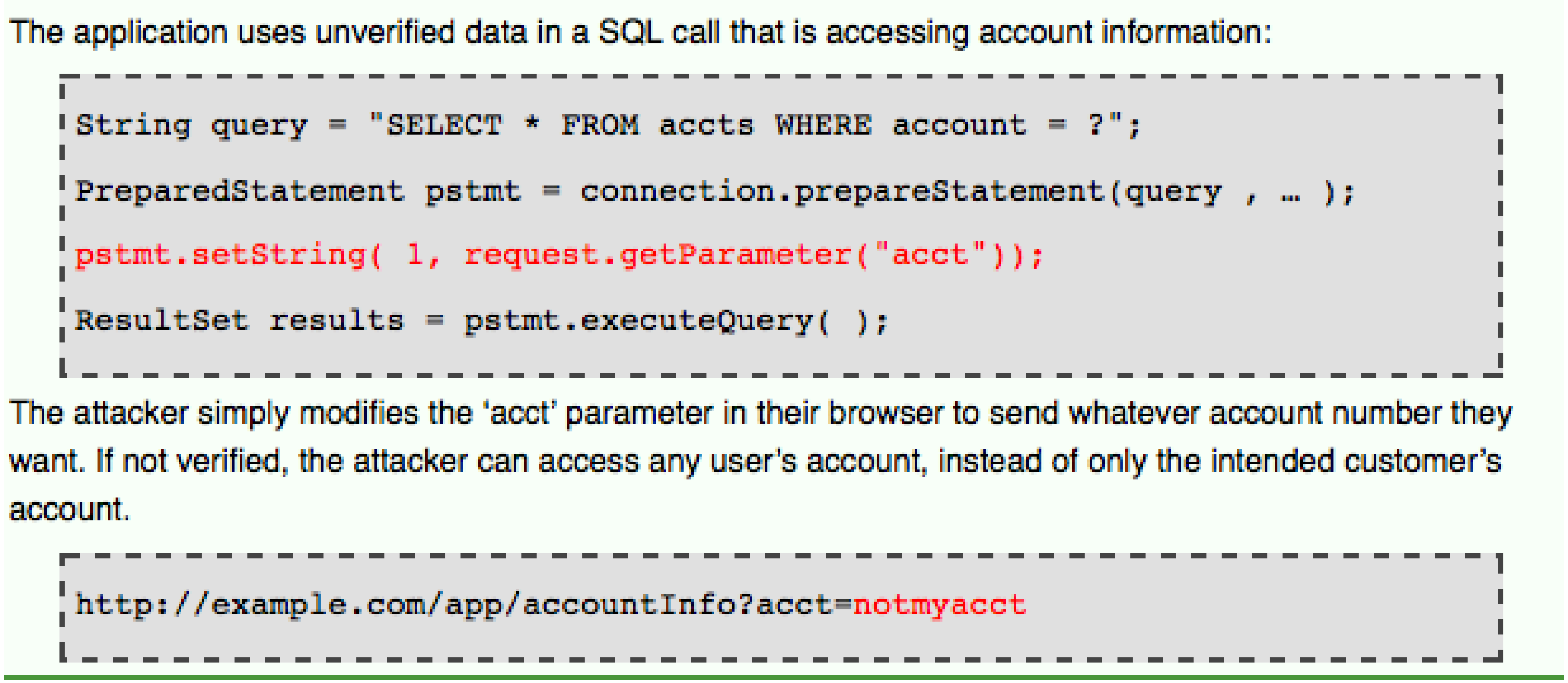
- Use Gruyere to find 2 XSS vulnerabilities
  - Profile page (stored)
  - URL (reflected)

# A4 – Broken Access Control



| Threat Agents | Attack Vectors | Security Weakness | | Technical Impacts | Business Impacts |
|---|---|---|---|---|---|
| Application Specific | Exploitability<br>EASY | Prevalence<br>COMMON | Detectability<br>EASY | Impact<br>MODERATE | Application / Business Specific |
| Consider the types of users of your system. Do any users have only partial access to certain types of system data? | Attacker, who is an authorized system user, simply changes a parameter value that directly refers to a system object to another object the user isn't authorized for. Is access granted? | Applications frequently use the actual name or key of an object when generating web pages. Applications don't always verify the user is authorized for the target object. This results in an insecure direct object reference flaw. Testers can easily manipulate parameter values to detect such flaws. Code analysis quickly shows whether authorization is properly verified. | | Such flaws can compromise all the data that can be referenced by the parameter. Unless object references are unpredictable, it's easy for an attacker to access all available data of that type. | Consider the business value of the exposed data.<br>Also consider the business impact of public exposure of the vulnerability. |

# Example Attacks



The application uses unverified data in a SQL call that is accessing account information:

```
String query = "SELECT * FROM accts WHERE account = ?";
PreparedStatement pstmt = connection.prepareStatement(query , … );
pstmt.setString( 1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery( );
```

The attacker simply modifies the 'acct' parameter in their browser to send whatever account number they want. If not verified, the attacker can access any user's account, instead of only the intended customer's account.

```
http://example.com/app/accountInfo?acct=notmyacct
```

https://www.owasp.org/index.php/Top_10_2013-A4-Insecure_Direct_Object_References

# Example Direct Objects

- The following direct objects are susceptible to an insecure direct object reference vulnerability:
  - Files and filenames
  - Registry keys
  - Ports and network resources
  - Tables, columns, and rows in a database

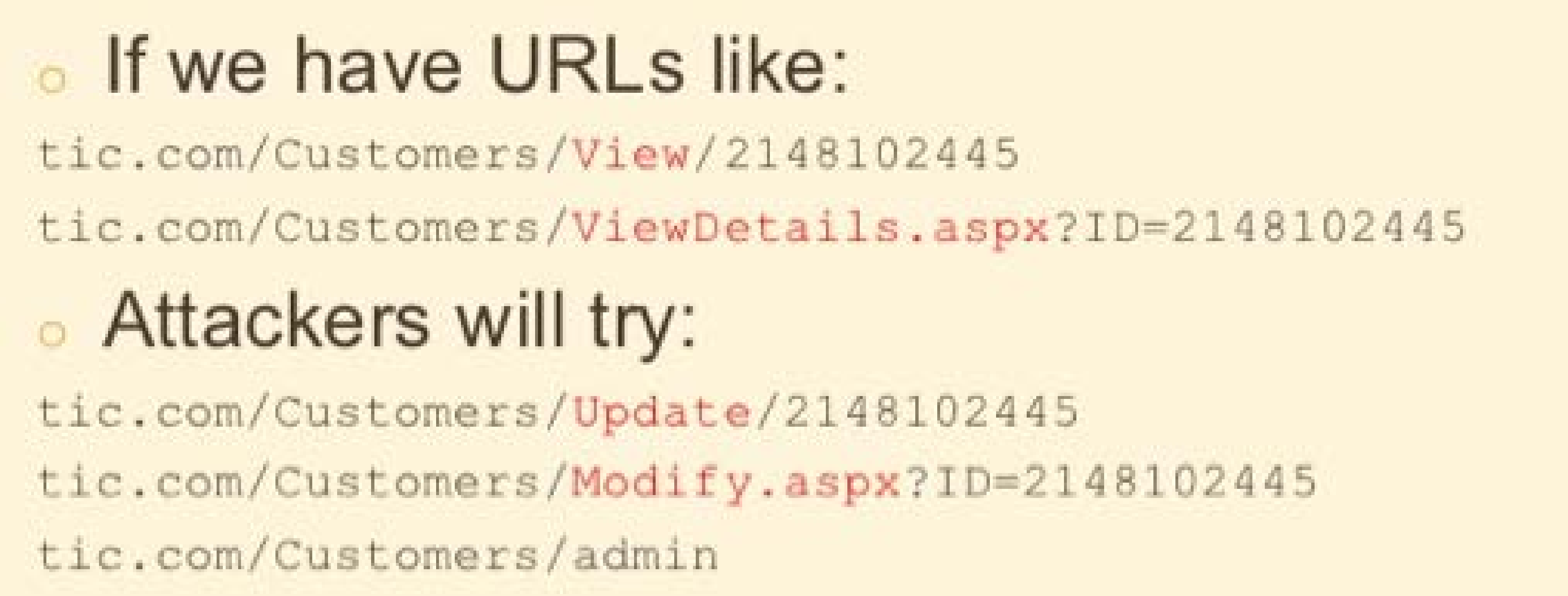
http://course.klocwork.com/sites/default/files/SCORM/15/CourseFiles/Player.htm

# How to change parameter values

- URL parameters
- Cookies
- Form fields (including hidden variables)

# There is no security through obscurity

- Assume the attackers will find our internal conventions



- If we have URLs like:

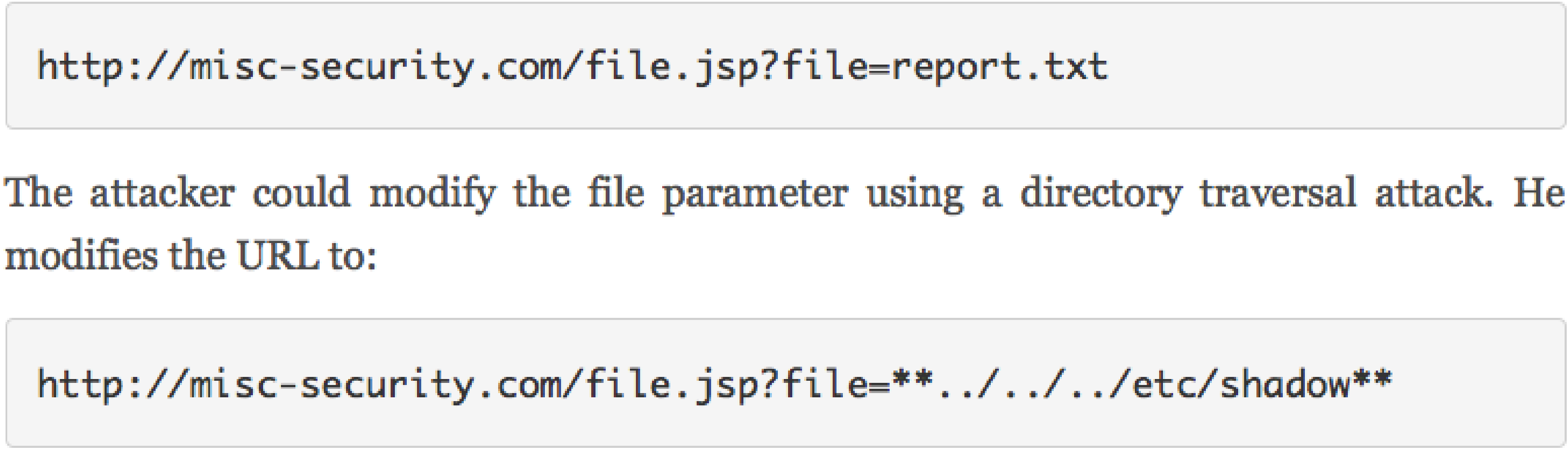
```
tic.com/Customers/View/2148102445
tic.com/Customers/ViewDetails.aspx?ID=2148102445
```

- Attackers will try:

```
tic.com/Customers/Update/2148102445
tic.com/Customers/Modify.aspx?ID=2148102445
tic.com/Customers/admin
```

http://www.slideshare.net/RapPayne/a4-insecure-direct-object-referencepptx

# Insecure File Access through Directory Traversal



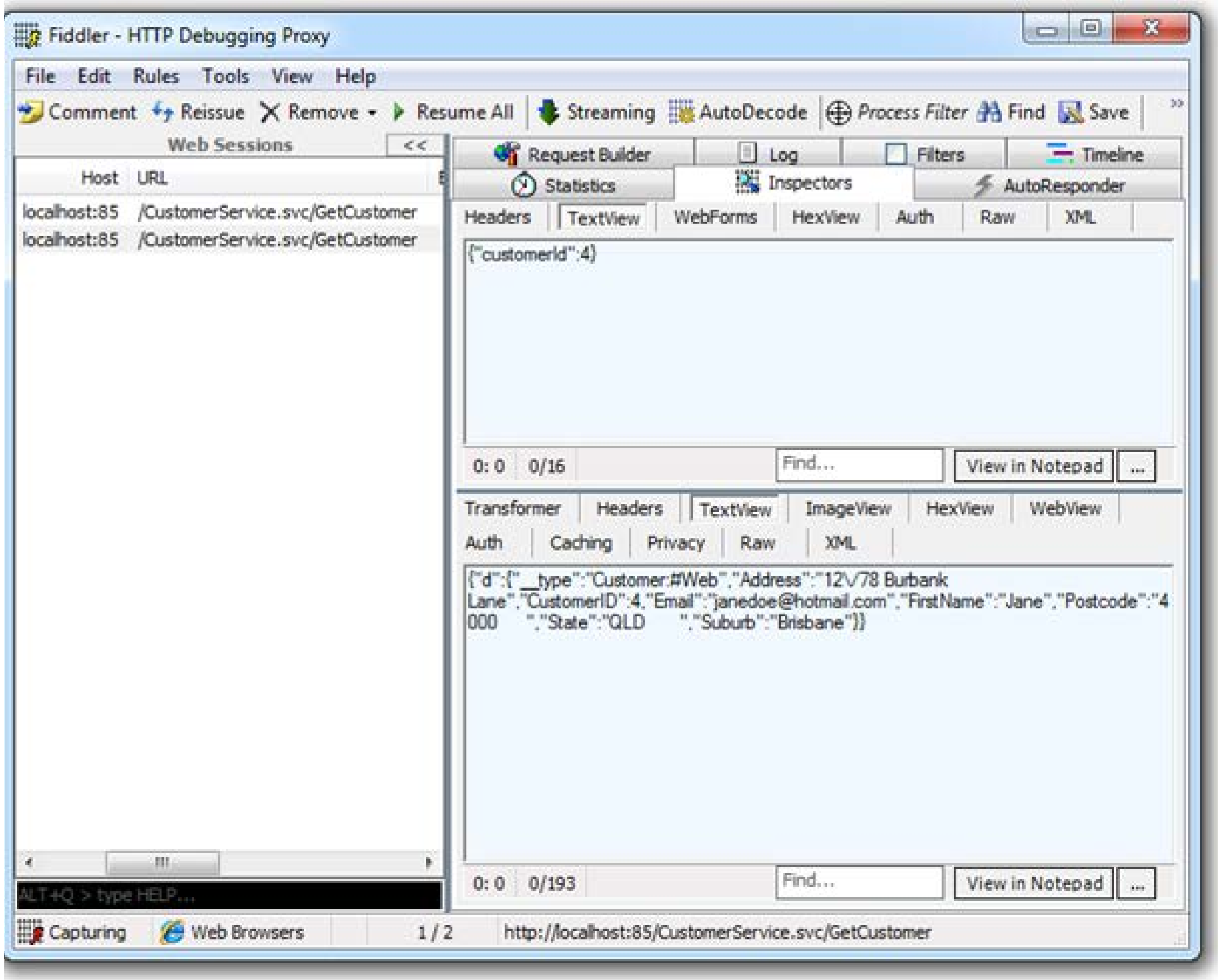
```
http://misc-security.com/file.jsp?file=report.txt
```

The attacker could modify the file parameter using a directory traversal attack. He modifies the URL to:

```
http://misc-security.com/file.jsp?file=**../../../etc/shadow**
```

http://bretthard.in/2009/07/insecure-direct-object-reference/

# Bypassing Client-side Checking …

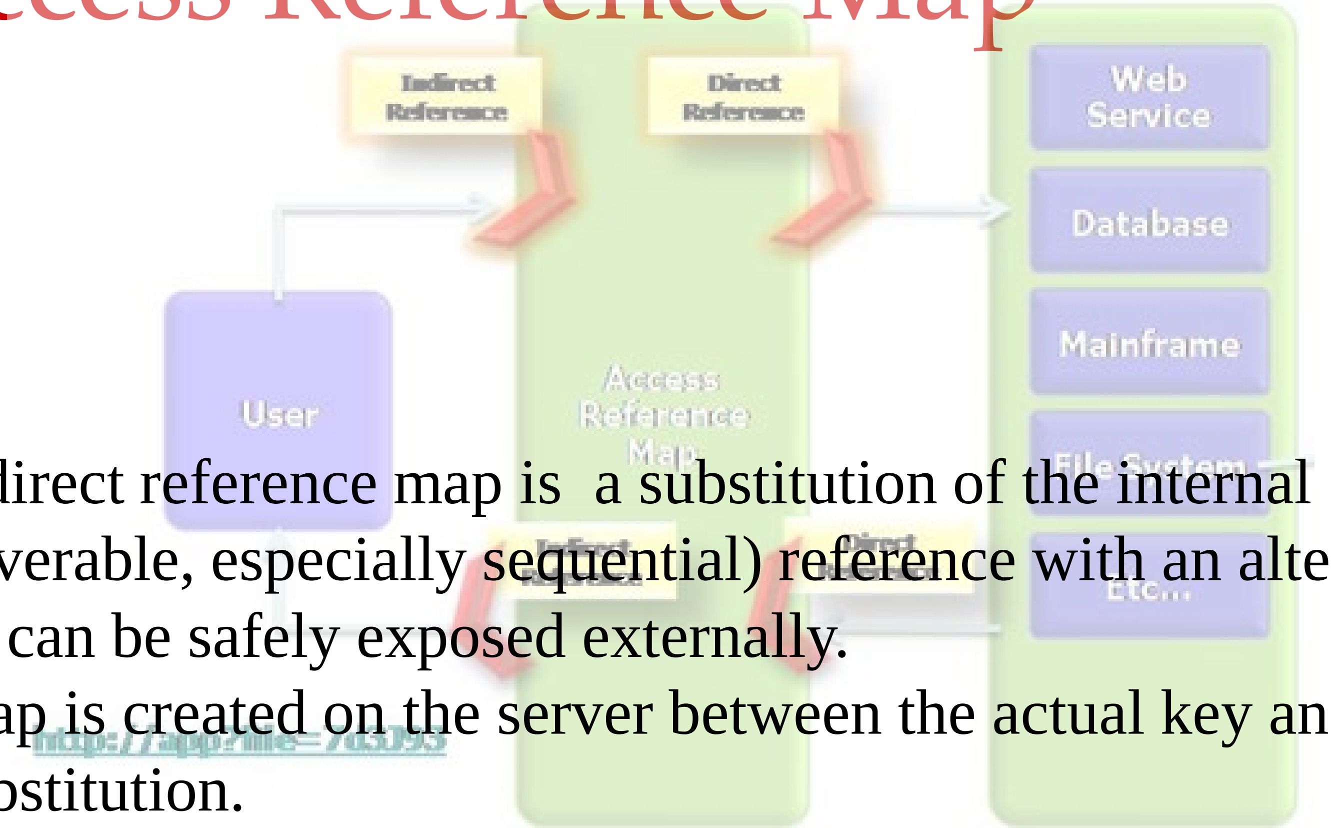

Fiddler - HTTP Debugging Proxy
File Edit Rules Tools View Help
Comment Reissue Remove Resume All Streaming AutoDecode Process Filter Find Save
Web Sessions
Host URL
localhost:85 /CustomerService.svc/GetCustomer
localhost:85 /CustomerService.svc/GetCustomer
Request Builder Log Filters Timeline
Statistics Inspectors AutoResponder
Headers TextView WebForms HexView Auth Raw XML
0: 0 0/16 Find... View in Notepad ...
Transformer Headers TextView ImageView HexView WebView
Auth Caching Privacy Raw XML
0: 0 0/193 Find... View in Notepad ...
ALT+Q > type HELP...
Capturing Web Browsers 1 / 2 http://localhost:85/CustomerService.svc/GetCustomer

```
{"customerId":4}
```

```
{"d":{"__type":"Customer:#Web","Address":"12\/78 Burbank
Lane","CustomerID":4,"Email":"janedoe@hotmail.com","FirstName":"Jane","Postcode":"4
000    ","State":"QLD       ","Suburb":"Brisbane"}}
```

http://www.troyhunt.com/2010/09/owasp-top-10-for-net-developers-part-4.html

# Access Reference Map



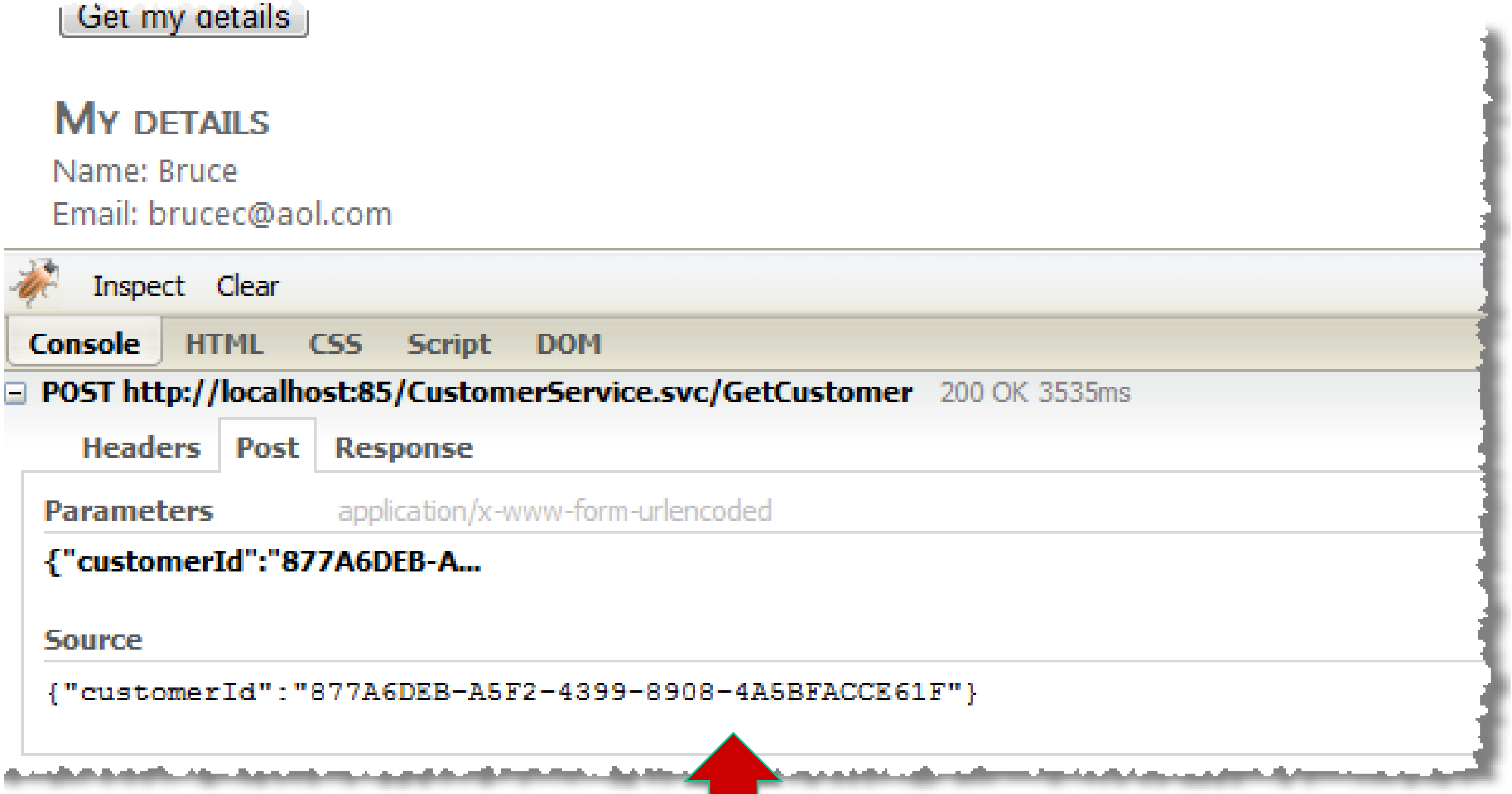
An indirect reference map is a substitution of the internal (discoverable, especially sequential) reference with an alternate ID which can be safely exposed externally.

1. Map is created on the server between the actual key and the substitution.
2. The key is translated to its substitution before being exposed to the UI.
3. After the substituted key is returned to the server, it’s translated back to the original before the data is retrieved.

http://www.troyhunt.com/2010/09/owasp-top-10-for-net-developers-part-4.html

Computer Science
NC STATE UNIVERSITY

# Defense in Depth …



Get my details

MY DETAILS

Name: Bruce

Email: brucec@aol.com

Inspect Clear

Console HTML CSS Script DOM

⊟ POST http://localhost:85/CustomerService.svc/GetCustomer 200 OK 3535ms

Headers Post Response

Parameters application/x-www-form-urlencoded

{"customerId":"877A6DEB-A...

Source

```
{"customerId":"877A6DEB-A5F2-4399-8908-4A5BFACCE61F"}
```

*http://myserver/index.jsp?getfileByID=2*

… still, authentication checks are also absolutely essential!

http://www.troyhunt.com/2010/09/owasp-top-10-for-net-developers-part-4.html
http://martin-white.blogspot.com/2011/05/a4-insecure-direct-object-references.html

# Prevention

- **Validate input before using it.** Evil input is the root cause of this threat. Check server side.
- **Use per user or session indirect object references.** This prevents attackers from directly targeting unauthorized resources. For example, instead of using the resource’s database key, a drop down list of six resources authorized for the current user could use the numbers 1 to 6 to indicate which value the user selected. The application has to map the per-user indirect reference back to the actual database key on the server
- **Check access.** Each use of a direct object reference from an untrusted source must include an access control check to ensure the user is authorized for the requested object. Check at time of use!

https://www.owasp.org/index.php/Top_10_2013-A4-Insecure_Direct_Object_References

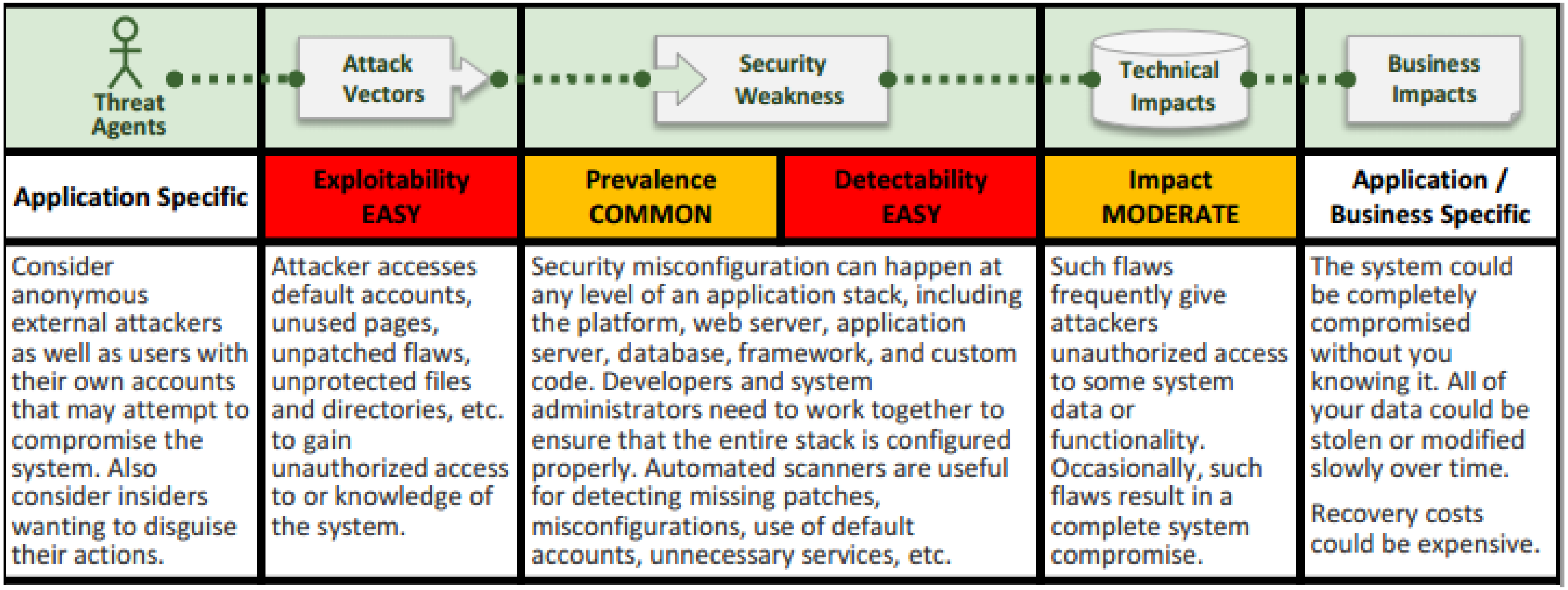
MORDAC, THE PREVENTER OF INFORMATION SERVICES
DO YOU KNOW HOW THE SYSTEM DISTINGUISHES BETWEEN AUTHORIZED AND ILLEGAL ACCESS?
ZAP!
IT DOESN'T.
HA HA!!!
www.dilbert.com scottadams@aol.com
4-4-08 © 2008 Scott Adams, Inc./Dist. by UFS, Inc.

http://search.dilbert.com/comic/Computers%20And%20It%20Security

# Class Exercise

- Answer questions
- Find direct object references

# A5 – Security Misconfiguration



| Threat Agents | Attack Vectors | Security Weakness | | Technical Impacts | Business Impacts |
|---|---|---|---|---|---|
| Application Specific | Exploitability<br>EASY | Prevalence<br>COMMON | Detectability<br>EASY | Impact<br>MODERATE | Application / Business Specific |
| Consider anonymous external attackers as well as users with their own accounts that may attempt to compromise the system. Also consider insiders wanting to disguise their actions. | Attacker accesses default accounts, unused pages, unpatched flaws, unprotected files and directories, etc. to gain unauthorized access to or knowledge of the system. | Security misconfiguration can happen at any level of an application stack, including the platform, web server, application server, database, framework, and custom code. Developers and system administrators need to work together to ensure that the entire stack is configured properly. Automated scanners are useful for detecting missing patches, misconfigurations, use of default accounts, unnecessary services, etc. | | Such flaws frequently give attackers unauthorized access to some system data or functionality. Occasionally, such flaws result in a complete system compromise. | The system could be completely compromised without you knowing it. All of your data could be stolen or modified slowly over time.<br>Recovery costs could be expensive. |

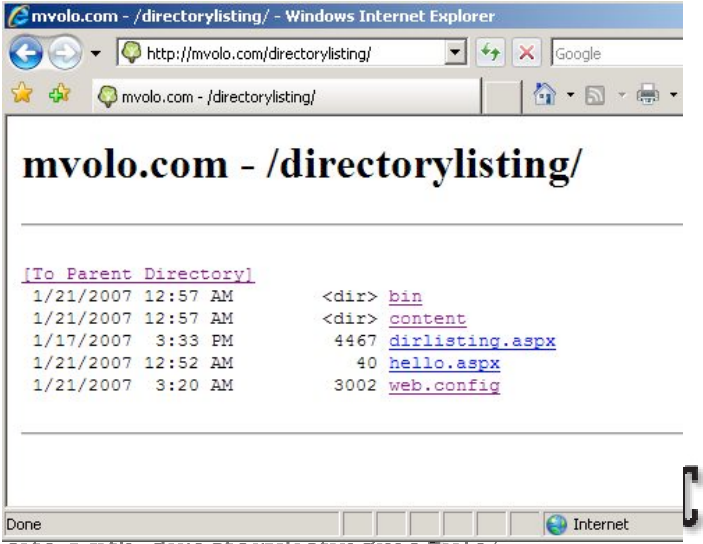
# Misconfiguration Types

- **Missing patches:** Patches, hotfixes, service packs, and updates contain the latest security fixes and need to be applied when they are available.
- **Misconfigured or disabled security features**: If a security feature is disabled or not configured, it cannot provide protection.
- **Default accounts:** Default accounts may allow a malicious user to automatically login with the credentials published in product documentation.
- **Unnecessary/unused services or features:** These represent an increased risk for security defects. Bugs exist even in the best-written code, By disabling unused and unnecessary services, code, and DLLs, you limit the amount of code that needs to be maintained and patched. When in doubt, turn features off, and turn them back on only if you need them.
- **Administrative back doors:** Administrative back doors are known as front doors in the hacking community.

http://course.klocwork.com/sites/default/files/SCORM/15/CourseFiles/Player.htm

# Example Attack Scenarios

- **Scenario #1:** The app server admin console is automatically installed and not removed. Default accounts aren't changed. Attacker discovers the standard admin pages are on your server, logs in with default passwords, and takes over.
- **Scenario #2:** Directory listing is not disabled on your server. Attacker discovers she can simply list directories to find any file. Attacker finds and downloads all your compiled Java classes, which she decompiles and reverse engineers to get all your custom code. She then finds a serious access control flaw in your application.


mvolo.com - /directorylisting/ - Windows Internet Explorer
http://mvolo.com/directorylisting/
Google
mvolo.com - /directorylisting/
Done
Internet

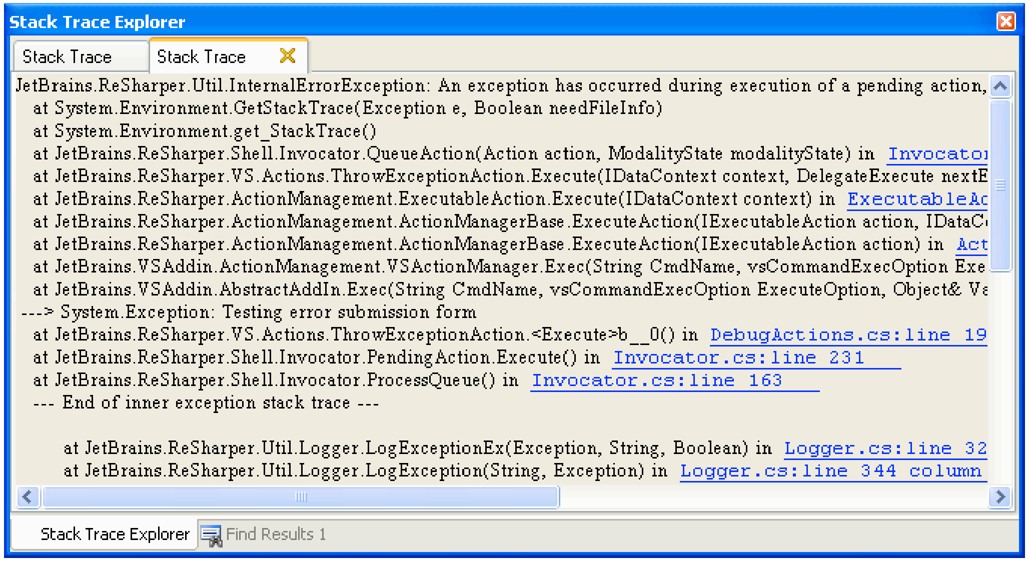
**mvolo.com - /directorylisting/**

```
[To Parent Directory]
 1/21/2007 12:57 AM        <dir> bin
 1/21/2007 12:57 AM        <dir> content
 1/17/2007  3:33 PM         4467 dirlisting.aspx
 1/21/2007 12:52 AM           40 hello.aspx
 1/21/2007  3:20 AM         3002 web.config
```

http://mvolo.com/get-nice-looking-directory-listings-for-your-iis-website-with-directorylistingmodule/
https://www.owasp.org/index.php/Top_10_2013-A5-Security_Misconfiguration

# Example Attack Scenarios -2

- **Scenario #3:** App server configuration allows stack traces to be returned to users, potentially exposing underlying flaws. Attackers love the extra information error messages provide.
- **Scenario #4:** App server comes with sample applications that are not removed from your production server. Said sample applications have well known security flaws attackers can use to compromise your server.

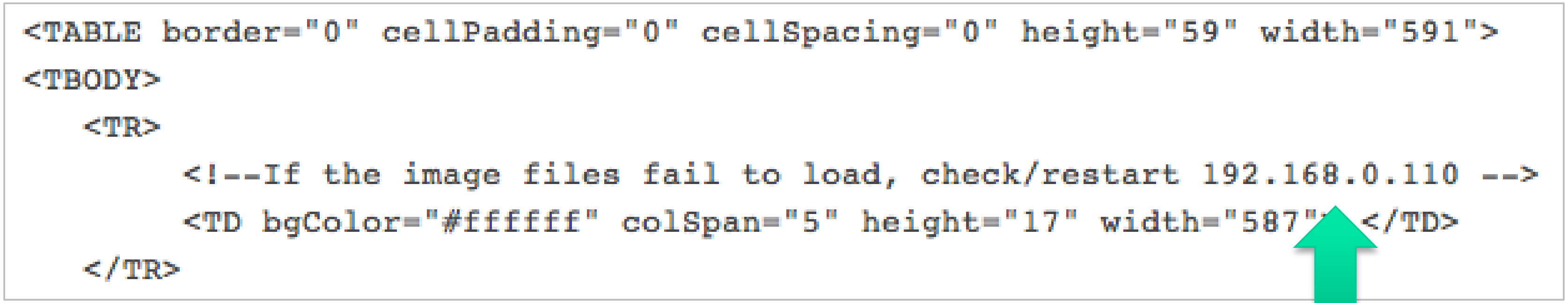


```
JetBrains.ReSharper.Util.InternalErrorException: An exception has occurred during execution of a pending action,
  at System.Environment.GetStackTrace(Exception e, Boolean needFileInfo)
  at System.Environment.get_StackTrace()
  at JetBrains.ReSharper.Shell.Invocator.QueueAction(Action action, ModalityState modalityState) in Invocato
  at JetBrains.ReSharper.VS.Actions.ThrowExceptionAction.Execute(IDataContext context, DelegateExecute nextE
  at JetBrains.ReSharper.ActionManagement.ExecutableAction.Execute(IDataContext context) in ExecutableAc
  at JetBrains.ReSharper.ActionManagement.ActionManagerBase.ExecuteAction(IExecutableAction action, IDataC
  at JetBrains.ReSharper.ActionManagement.ActionManagerBase.ExecuteAction(IExecutableAction action) in Act
  at JetBrains.VSAddin.ActionManagement.VSActionManager.Exec(String CmdName, vsCommandExecOption Exe
  at JetBrains.VSAddin.AbstractAddIn.Exec(String CmdName, vsCommandExecOption ExecuteOption, Object& Va
---> System.Exception: Testing error submission form
  at JetBrains.ReSharper.VS.Actions.ThrowExceptionAction.<Execute>b__0() in DebugActions.cs:line 19
  at JetBrains.ReSharper.Shell.Invocator.PendingAction.Execute() in Invocator.cs:line 231
  at JetBrains.ReSharper.Shell.Invocator.ProcessQueue() in Invocator.cs:line 163
  --- End of inner exception stack trace ---

     at JetBrains.ReSharper.Util.Logger.LogExceptionEx(Exception, String, Boolean) in Logger.cs:line 32
     at JetBrains.ReSharper.Util.Logger.LogException(String, Exception) in Logger.cs:line 344 column
```

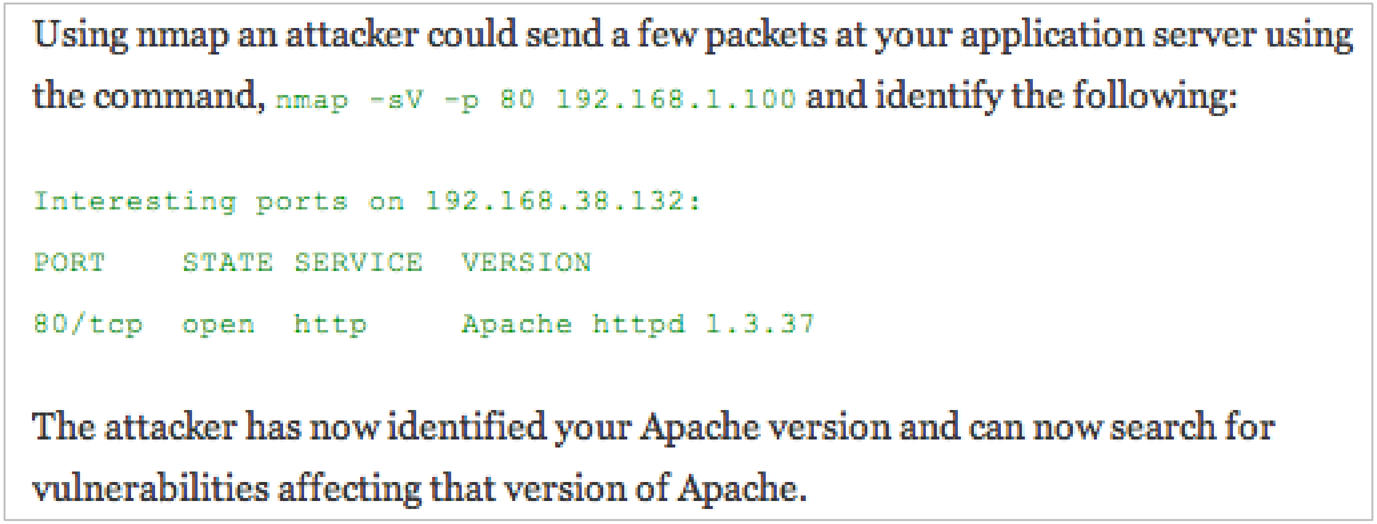
http://www.jetbrains.com/resharper/documentation/help20/Navigation/stackTraceExplorer.html
https://www.owasp.org/index.php/Top_10_2013-A5-Security_Misconfiguration

# Information Leakage via Comments



```
<TABLE border="0" cellPadding="0" cellSpacing="0" height="59" width="591">
<TBODY>
   <TR>
         <!--If the image files fail to load, check/restart 192.168.0.110 -->
         <TD bgColor="#ffffff" colSpan="5" height="17" width="587"> </TD>
   </TR>
```

- Provides host IP Address



Using nmap an attacker could send a few packets at your application server using the command, `nmap -sV -p 80 192.168.1.100` and identify the following:

```
Interesting ports on 192.168.38.132:
PORT     STATE SERVICE  VERSION
80/tcp   open  http     Apache httpd 1.3.37
```

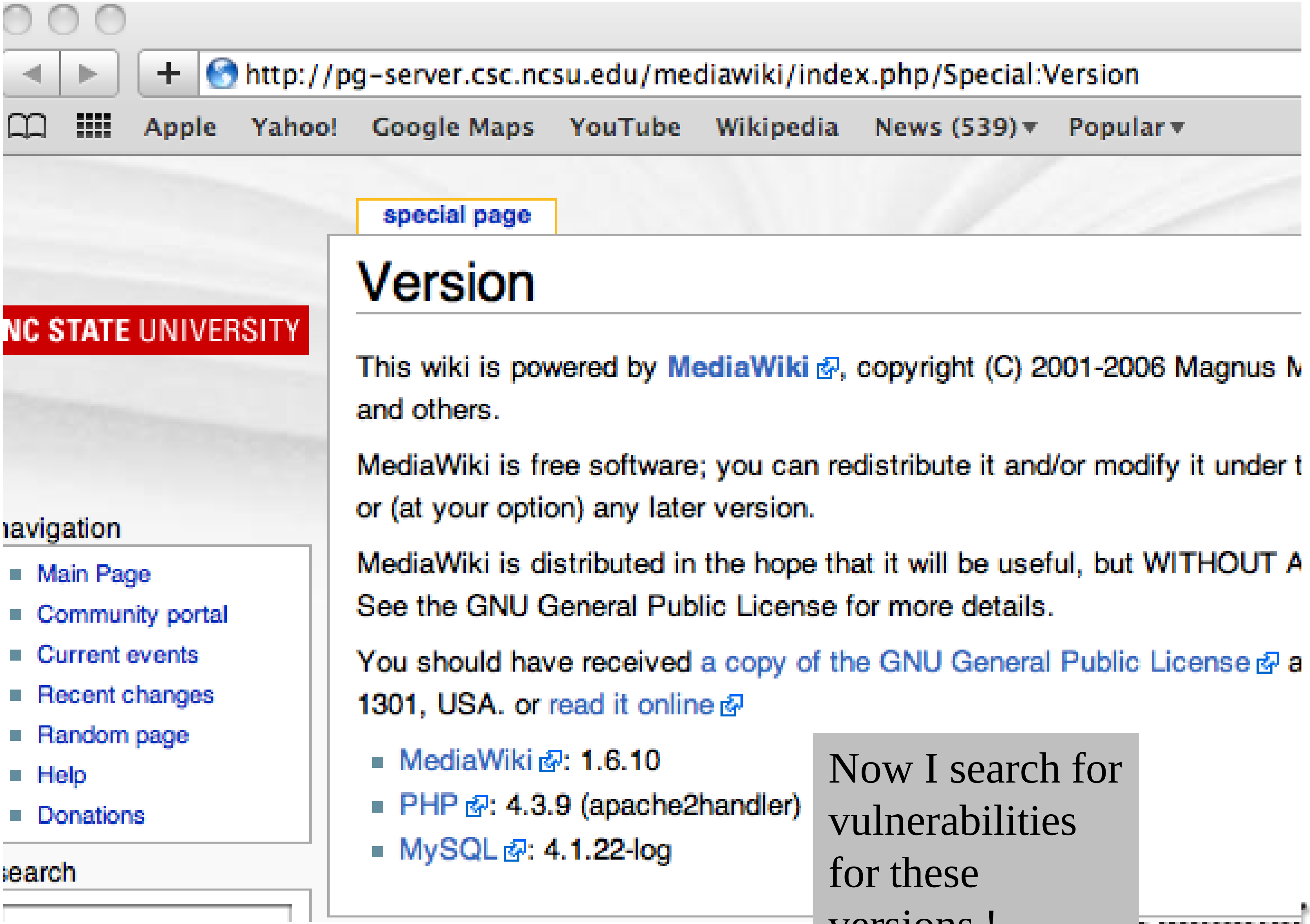
The attacker has now identified your Apache version and can now search for vulnerabilities affecting that version of Apache.

http://projects.webappsec.org/Information-Leakage
http://misc-security.com/2009/08/12/information-leakage-and-improper-error-handling/

# Google hacking

- Find Microsoft Excel files that contain login names and passwords
  - "login: *" "password: *" filetype:xls
- Locate passwords in plain text found in exposed log files
  - "your password is" filetype:log
- Discover insecure instances of the phpMyAdmin database front end
  - “Welcome to phpMyAdmin” “Create new database”
  - intitle:PhpMyAdmin "Welcome to phpMyAdmin***" running on * as root@*"
- Find human resources web sites from the internal intranet which are accessible to external users
  - intitle:intranet inurl:intranet +intext:"human resources”
- Search for private contact lists that have been synced up from PDA’s or cell phones
  - contacts ext:wml

http://johnny.ihackstuff.com
http://netsecurity.about.com/od/perimetersecurity/a/4_leakage.htmß
http://www.go4expert.com/forums/showthread.php?t=12242



http://pg-server.csc.ncsu.edu/mediawiki/index.php/Special:Version

# Version

This wiki is powered by MediaWiki, copyright (C) 2001-2006 Magnus M
and others.

MediaWiki is free software; you can redistribute it and/or modify it under t
or (at your option) any later version.

MediaWiki is distributed in the hope that it will be useful, but WITHOUT A
See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License a
1301, USA. or read it online

- MediaWiki: 1.6.10
- PHP: 4.3.9 (apache2handler)
- MySQL: 4.1.22-log

Now I search for vulnerabilities for these versions !

# Prevention - 1

- A repeatable hardening process that makes it fast and easy to deploy another environment that is properly locked down. Development, QA, and production environments should all be configured identically (with different passwords used in each environment). This process should be automated to minimize the effort required to setup a new secure environment.
- A process for keeping abreast of and deploying all new software updates and patches in a timely manner to each deployed environment. This needs to include all code libraries as well. Vulnerability information on old versions is readily available. **Threats change constantly**!
- A strong application architecture that provides effective, secure separation between components.

https://www.owasp.org/index.php/Top_10_2013-A5-Security_Misconfiguration

# Prevention - 2

- Running scans and doing audits periodically to help detect future misconfigurations or missing patches.
- Remove/change default credentials
- Disable or removed untrusted and unnecessary components or services
- Remove all unused pages and user accounts

# Resources

- http://checklists.nist.gov
- http://scap.nist.gov/

# Class Exercise

- Answer questions