Communications of the IIMA

Volume 4 | Issue 2 Article 4

2004

Developing Security for E-Commerce Applications: A Teaching Case

Chang Liu
Northern Illinois University

Brian G. Mackie
Northern Illinois University

Developing Security for E-Commerce Applications: A Teaching Case

Chang Liu

Department of Operations Management and Information Systems, College of Business, Northern Illinois University,
De Kalb, IL 60115 USA
815-753-3021, cliu@niu.edu

Brian G. Mackie

Department of Operations Management and Information Systems, College of Business, Northern Illinois University,
De Kalb, IL 60115 USA
815-753-5896, bmackie@niu.edu

ABSTRACT

The number of severe computer security breaches in e-commerce applications has been on the increase over the last few years. This has become one of the biggest security problems in recent years. Although there are tools to build e-commerce application firewalls to alert and prevent intruder attacks, these tools are not trivial to install (they are not plug-and-play). Internet intruders can create havoc and produce catastrophe results by exploiting weaknesses in e-commerce applications. Therefore, developers of e-commerce web sites have to incorporate ways to systematically identify and eliminate vulnerabilities in the EC applications to enhance their security. This paper describes how Microsoft ASP.Net can be used to assist students in exploring ways to increase the security of EC applications.

INTRODUCTION

Over the past few years, Electronic Commerce (EC) has allowed organizations to enhance their economic growth, reduce barriers of market entry, improve efficiency and effectiveness, keep inventories lean, and reduce costs (Hof and Hamm, 2002; Madden and Coble-Neal, 2002). In fact, many consumers have found benefit in using the Internet for EC: convenience, more choices for products and services, vast amounts of information, and time savings. Customers are not going to let a poor economy stop them from taking advantage of the Internet for electronic commerce. More than 200 millions American who now have Web access are likely to spend more than $120 billon online this year (Economist, 2004). Business to business (B2B) online transactions now stand at more than $2.6 trillion in the US, which is double the original estimate for 2003. In addition, it is predicted that business to consumer (B2C) spending will exceed $250 billion by 2005 (Mullaney, Green, Arndt, and Hof, 2003). There is no doubt that E-Commerce will continue to grow that will change every kind of business, online as well as offline.

In order to achieve this bullish forecast, businesses need to build security into their EC web sites. Without security, customers will shy away from the electronic shopping mall and refuse to provide their personal and financial information needed for successful online transactions. According to a recent survey conducted by the Computer Security Institute (CSI), security is the top concern when conducting e-commerce activities on the web. Ninety percent of respondents, mainly from large U.S. corporations and government agencies, detected computer security breaches within the twelve month period ending in 2002 (The 2002 Computer Security Institute (CSI) report). Eighty percent of those security breaches resulted in substantial financial losses. From a business perspective, EC web site security can be divided into three categories: network, web server, and application security (Microsoft, 2004). The foregoing survey cited web connection and application as frequent points of attack. This paper describes a teaching case that emphasizes methods students can use to enhance e-commerce application security and examine ways to improve application robustness by integrating many security features within the design process.

Indeed, the number of severe computer security breaches of e-commerce applications has grown steadily and is now seen as one of the biggest security problems in recent years. Although there are tools to build e-commerce application firewalls to alert when an attack happens and/or prevent attacks, these tools are not trivial to install. Internet intruders can create havoc and produce catastrophe results by exploiting weaknesses within e-commerce applications (McClure and Scambrary, 1999). The intruder's best ally is poorly written or inadequately tested software. Unfortunately, many developers neglect this area and distribute applications containing exploitable bugs (Franklin and Wiens, 2004). Therefore, developers for EC web sites have to incorporate ways to systematically identify and eliminate vulnerabilities in the code for EC applications to enhance their security.

THE PROJECT OF SECURING E-COMMERCE APPLICATIONS

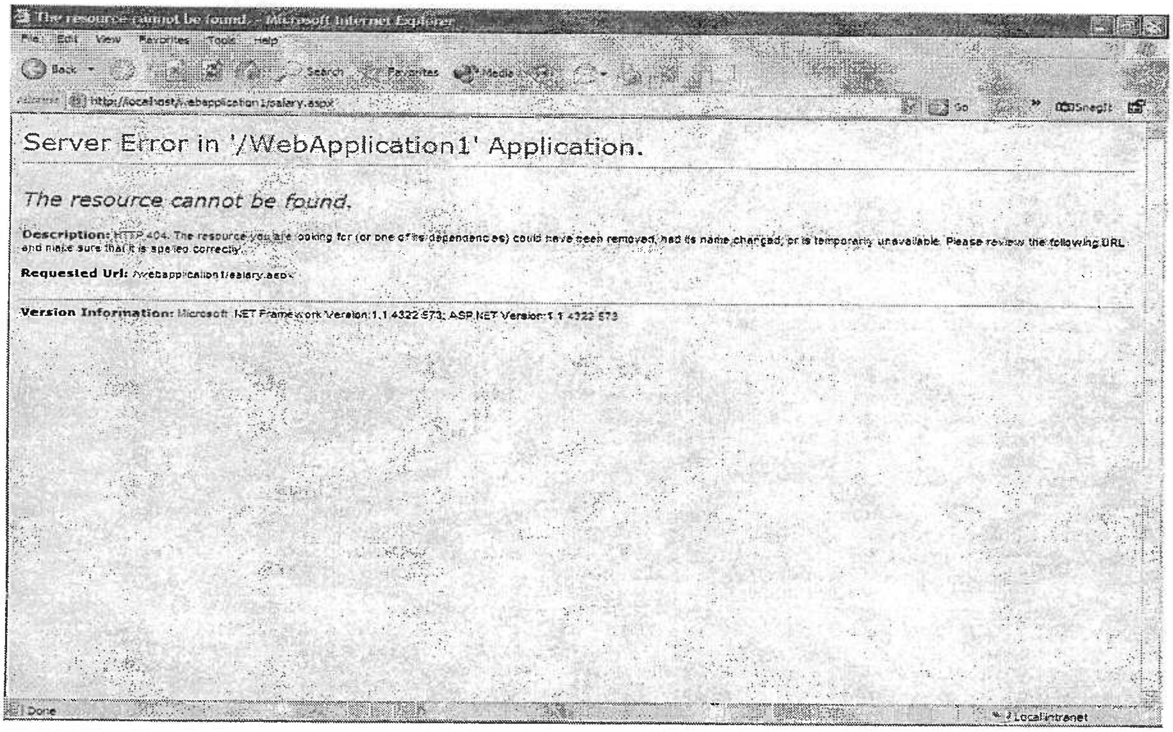
A semester long e-commerce course that one author taught in the spring 2004 at a large mid-western university included web-based application design that allowed students to use Microsoft ASP.Net to explore the EC application development process by creating an electronic shopping mall to sell a variety of products. The students, who were seniors with an Information Systems major, covered a range of e-commerce development topics, including site design, shopping cart design, input validation, web database integration, order confirmation, and incorporating security features.

The project of integrating security features into an e-commerce site was developed based on the three phases of marketing activities: pre-sales, sales, and after-sales (Liu, Arnett, Capella, and Beatty, 1997). Any EC activities fit within these three classifications. The pre-sales phase includes a company's efforts to attract customers by advertising, public relations, feasible site/key phrase search, new product or service announcements, and other related activities. Customers' electronic purchasing activities occur in the online phase where orders and charges are placed electronically through the web. Of course, off-line orders and charges, such as those from a mail or telephone medium, are also permitted. The after-sales phase includes customer service, problem resolution, and other issues such as handling product defects and returns and follow-up surveys to maintain or generate customer satisfaction.

SECURITY IN PRE-SALES PHASE

When discussing how to build security in the pre-sales phase of an e-commerce web site, emphasis was given on how to prevent unexpected errors, thus building robustness into the site to increase customers' trust level and confidence while browsing the site.

There is no doubt that errors will occur in e-commerce applications. Students, and even professional web developers, generally try to trap errors using a facility called TRY-CATCH statement blocks when using ASP.Net. However, this technique does not handle every possible exception. For example, when a customer tries to access a non-existent page from a site developed with ASP.Net, he or she would see a typical error message as displayed in Figure 1. Moreover, if an unexpected error occurs, the error message as displayed in Figure 2 could reveal information which includes the business logic behind the site design. Obviously, it is quite dangerous for a business organization since it can give intruders more information to use in trying to crash the application, or gain access to sensitive information, and/or execute malicious code.


The resource cannot be found. - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
http://localhost/webapplication1/salary.aspx
Go
SnagIt
Done
Local intranet

Server Error in '/WebApplication1' Application.

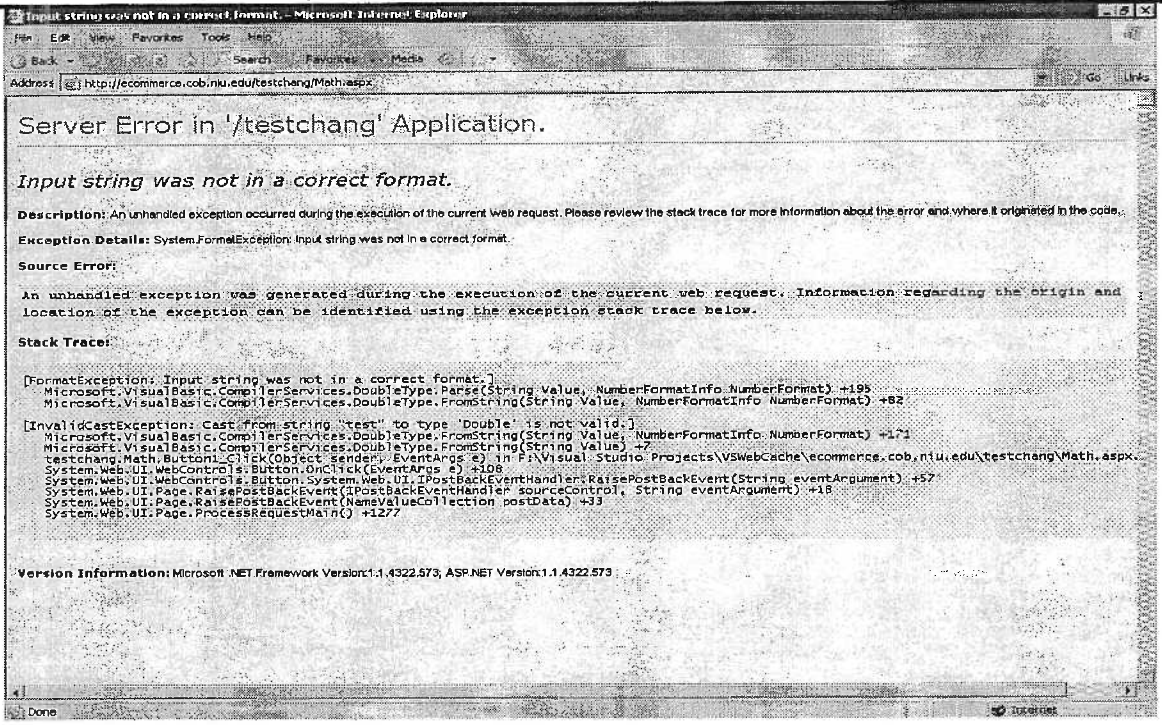
The resource cannot be found.

Description: HTTP 404. The resource you are looking for (or one of its dependencies) could have been removed, had its name changed, or is temporarily unavailable. Please review the following URL and make sure that it is spelled correctly.

Requested Url: /webapplication1/salary.aspx

Version Information: Microsoft .NET Framework Version:1.1.4322.573; ASP.NET Version:1.1.4322.573

Figure 1: ERROR MESSAGE for SEARCHING UNAVAILABLE PAGE


Input string was not in a correct format. - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://ecommerce.cob.niu.edu/testchang/Math.aspx
Go Links
Done
Internet

Server Error in '/testchang' Application.

Input string was not in a correct format.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.FormatException: Input string was not in a correct format.

Source Error:

```
An unhandled exception was generated during the execution of the current web request. Information regarding the origin and
location of the exception can be identified using the exception stack trace below.
```

Stack Trace:

```
[FormatException: Input string was not in a correct format.]
   Microsoft.VisualBasic.CompilerServices.DoubleType.Parse(String Value, NumberFormatInfo NumberFormat) +195
   Microsoft.VisualBasic.CompilerServices.DoubleType.FromString(String Value, NumberFormatInfo NumberFormat) +82

[InvalidCastException: Cast from string "test" to type 'Double' is not valid.]
   Microsoft.VisualBasic.CompilerServices.DoubleType.FromString(String Value, NumberFormatInfo NumberFormat) +171
   Microsoft.VisualBasic.CompilerServices.DoubleType.FromString(String Value) +7
   testchang.Math.Button1_Click(Object sender, EventArgs e) in F:\Visual Studio Projects\VSWebCache\ecommerce.cob.niu.edu\testchang\Math.aspx.
   System.Web.UI.WebControls.Button.OnClick(EventArgs e) +108
   System.Web.UI.WebControls.Button.System.Web.UI.IPostBackEventHandler.RaisePostBackEvent(String eventArgument) +57
   System.Web.UI.Page.RaisePostBackEvent(IPostBackEventHandler sourceControl, String eventArgument) +18
   System.Web.UI.Page.RaisePostBackEvent(NameValueCollection postData) +33
   System.Web.UI.Page.ProcessRequestMain() +1277
```

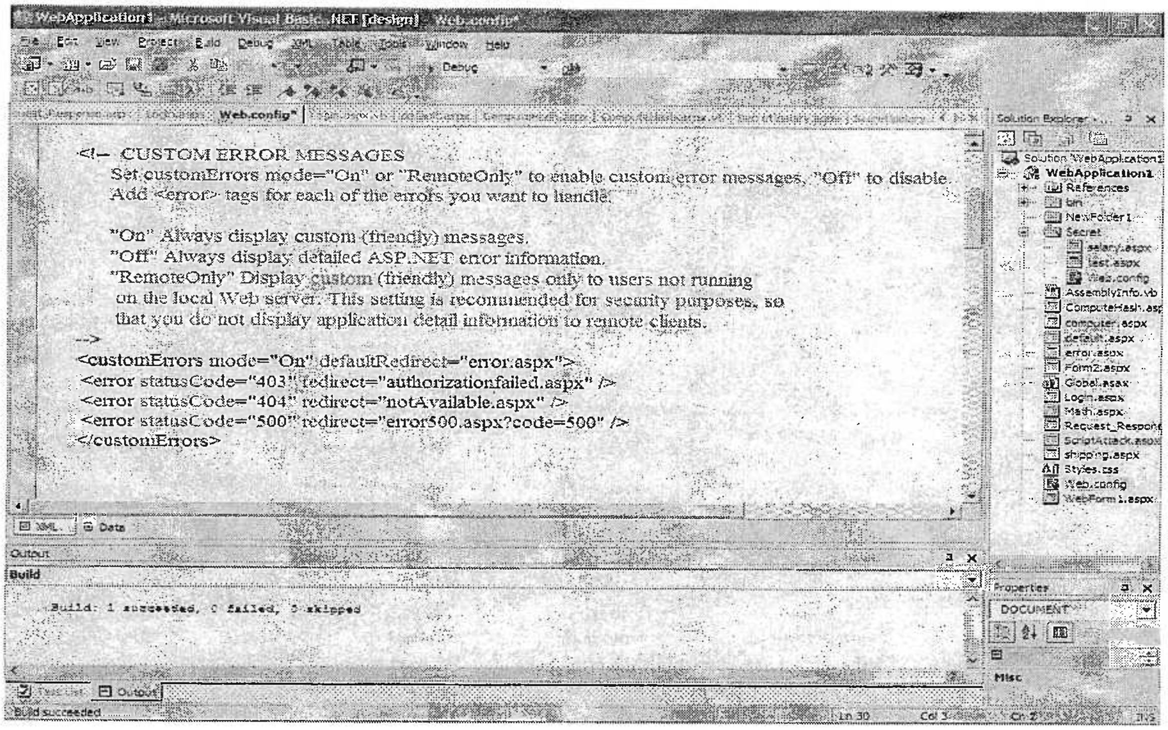
Version Information: Microsoft .NET Framework Version:1.1.4322.573; ASP.NET Version:1.1.4322.573

Figure 2: AN EXAMPLE OF AN ERROR MESSAGE FOR AN UNEXPECTED EXCEPTION

In the course, three different methods were examined to handle unexpected exceptions using ASP.Net: (1) using a feature called *customerErrors* section within the *Web.Config* file (Web.Config is the overall configuration file for a web-based application or a portion of the web-based application), (2) using a subroutine called *Application_Error* within the *Global.asax* file (Global.asax handles the web environment for a web-based application), and (3) implementing a *Page_Error* subroutine within a given page. This paper demonstrates how the authors taught error handling by using *customerErrors* within the *Web.Config* file.

The *Web.Config* file is written in XML (extensible Markup Language) format. XML is a relatively new markup language that provides a cross-platform method that can identify, or markup data. An XML document allows for complete control over the structure of the data. Subsequently, one needs only to define tags or elements within the document or file to explain the meaning of the data. Therefore before the *Web.Config* file was examined and modified, the authors taught the students the basics of XML and how to create an XML file to represent a data structure. This emphasized the popularity of XML stems from its ability to separate content with presentation, whereby the developer has the freedom to build an application having its own data structure.

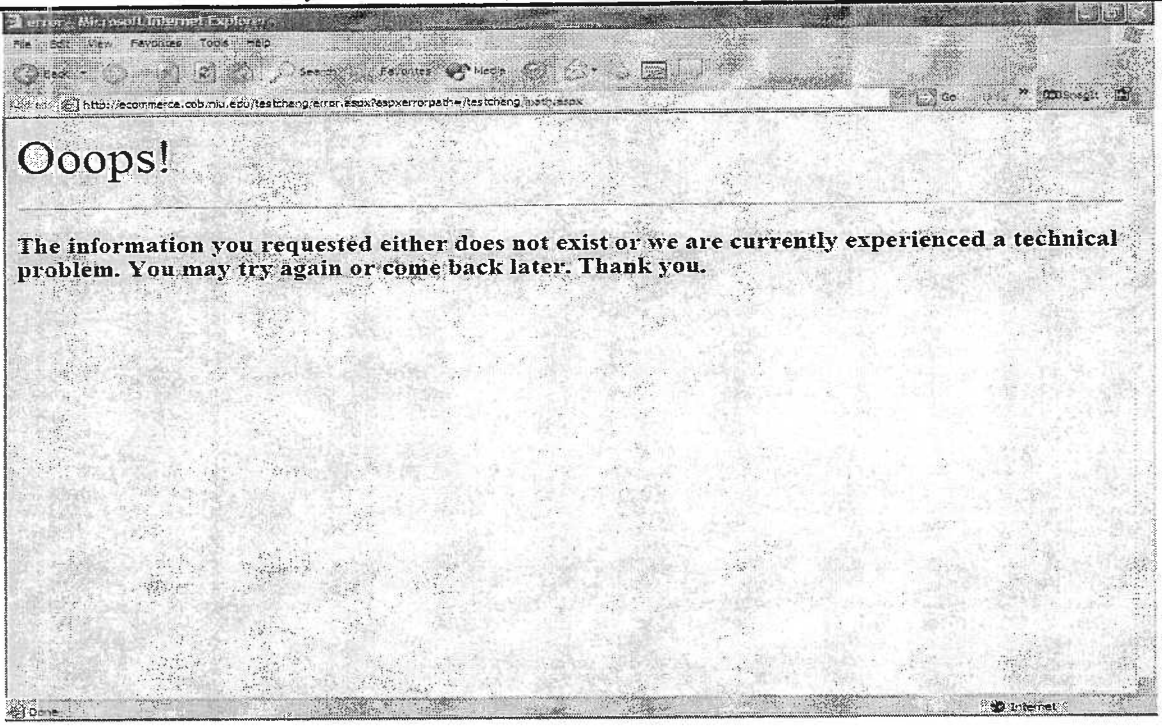
The examination and modification of the *Web.Config* file was then explained. It was explained how to handle specific errors based on the error code (such as 403, 404, 500, etc.), or use one page to handle all errors. Figure 3 how to modify the *Web.Config* file by enabling *mode = "On"* in the *customErrors* section, shows how specific errors are handled such as error 403 which is handled by redirecting to the "authorization failed page", and handling unexpected errors by specifying a default redirection page. Figure 4 presents an example of a pre-defined error handling page.



```
<!-- CUSTOM ERROR MESSAGES
     Set customErrors mode="On" or "RemoteOnly" to enable custom error messages, "Off" to disable.
     Add <error> tags for each of the errors you want to handle.

     "On" Always display custom (friendly) messages.
     "Off" Always display detailed ASP.NET error information.
     "RemoteOnly" Display custom (friendly) messages only to users not running
      on the local Web server. This setting is recommended for security purposes, so
      that you do not display application detail information to remote clients.
-->
<customErrors mode="On" defaultRedirect="error.aspx">
 <error statusCode="403" redirect="authorizationfailed.aspx" />
 <error statusCode="404" redirect="notAvailable.aspx" />
 <error statusCode="500" redirect="error500.aspx?code=500" />
</customErrors>
```

```
Build: 1 succeeded, 0 failed, 0 skipped
```

Figure 3: THE WEB.CONFIG FILE FOR ERROR HANDLING


error - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
http://ecommerce.cob.niu.edu/testcheng/error.aspx?aspxerrorpath=/testcheng/math.aspx
Go
Done
Internet

Ooops!

The information you requested either does not exist or we are currently experienced a technical problem. You may try again or come back later. Thank you.

Figure 4: A PRE-DEFINED ERROR HANDLING PAGE

The above example demonstrates using the *Web.Config* file in an ASP.Net application to ensure all the unexpected errors will be handled in a certain way to increase the security of the site and the robustness of the pre-sales phase of the e-commerce application. These measures will increase customers trust and confidence while browsing the web site.

SECURITY IN ONLINE SALES PHASE

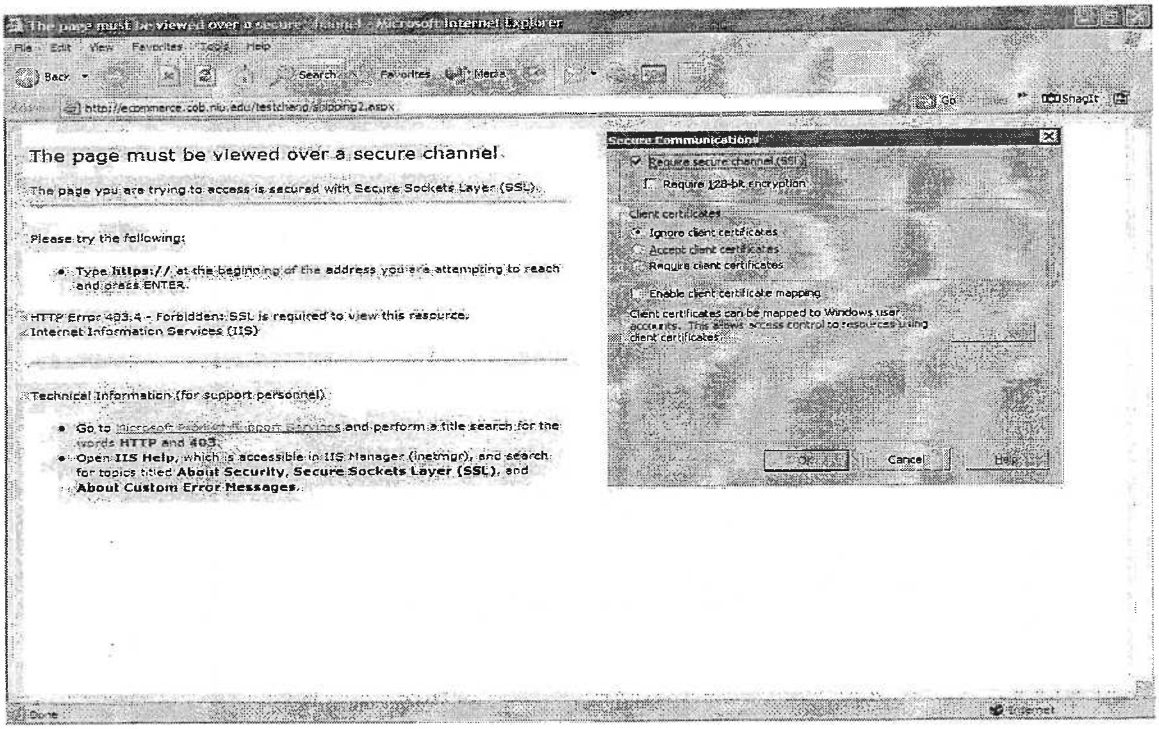
In the online sales phase the customers use the EC application to accomplish one or more electronic data transactions. The emphasis of the security design within this phrase is to demonstrate how to protect personal customer information being passed across the Internet using encryption. When customers submit their financial and personal information at an e-commerce web site, the data is transmitted from a browser to the company's web server. As the data moves through the Internet, in could be intercepted and read by unauthorized persons. The proper solution is to encrypt the data before it is sent through the Internet. Using encryption, even if the data is captured, the information is protected.

The authors explained how the Secure Socket Layer (SSL) could be used to encrypt sensitive information that would be passed back and forth between a web server and web browser. Since this course dealt with application design, the authors presented the installation process to install the Certificate Services in Microsoft Windows 2003 Server, generate a Certificate Request file, issue a Certificate, and then install a server-side Certificate by using Microsoft Internet Information Manager.

Figure 5 shows how after the server is configured to use SSL, the normal standardized *http://* address will no longer work to access the page. Any request for the page is now using a secure connection by using *https://* instead of *http://* as shown in Figure 6. Figure 6 shows information such as credit card number, customer name, and email address that can be securely collected by the EC application using SSL. Using *https* is obviously helpful for building trust because more informed consumers would notice and care about the protection on their personal information.

Building security in the online sales phase is critical for all businesses. According to Hoffman and Novak (1999), the primary reason many people have yet to shop online is the fundamental lack of trust in the security and robustness of web sites. Almost 95% of the Web users surveyed in the US have declined to provide personal information over the Internet. Moreover, 40% responded that they tend to create fake personal information when online. It appears that many customers simply do not trust Web sites enough to engage in "relationship exchanges" that involve the exchange of personal and financial information.

In addition, research has shown an increasing level of concern about privacy (Liu, Marchewka, and Ku, 2004). Obviously this concern will become even more heightened as more customers engage in e-commerce activities which collect personal and financial information. According to the Federal Trade Commission (FTC), protecting consumer's privacy is an important aspect of ensuring data security in the online sales e-commerce activities (FTC Congress Report, 2000). To emphasize the importance of trust policies the authors asked the students to research and write their own privacy policies that they felt could ease customers' concerns about online privacy,. The class was also asked to explore several seal programs such as TRUSTe (http://www.truste.org/), BBBOnLine (http://www.bbbonline.org/), Verisign


The page must be viewed over a secure channel - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Go
SnagIt
Done
Internet

http://ecommerce.cob.niu.edu/testcheng/shipping2.aspx

The page must be viewed over a secure channel

The page you are trying to access is secured with Secure Sockets Layer (SSL).

Please try the following:

- Type **https://** at the beginning of the address you are attempting to reach and press ENTER.

HTTP Error 403.4 - Forbidden: SSL is required to view this resource.
Internet Information Services (IIS)

Technical Information (for support personnel)

- Go to Microsoft Product Support Services and perform a title search for the words **HTTP** and **403**.
- Open **IIS Help**, which is accessible in IIS Manager (inetmgr), and search for topics titled **About Security**, **Secure Sockets Layer (SSL)**, and **About Custom Error Messages**.

Secure Communications

☑ Require secure channel (SSL)

☐ Require 128-bit encryption

Client certificates

● Ignore client certificates

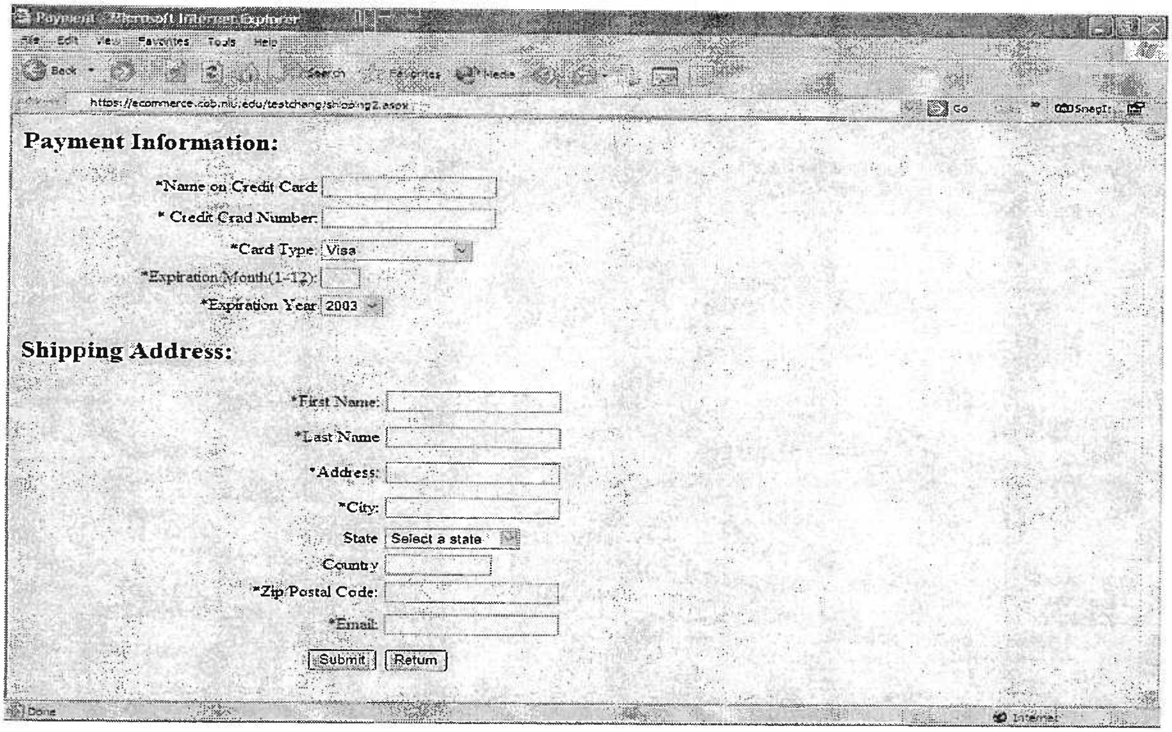
○ Accept client certificates

○ Require client certificates

☐ Enable client certificate mapping

Client certificates can be mapped to Windows user accounts. This allows access control to resources using client certificates.

Figure 5: REQUESTING A SECURE CONNECTION WHEN RETRIEVING A PAGE


Payment - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
https://ecommerce.cob.niu.edu/testchang/shipping2.aspx
Go SnagIt
Done
Internet

Payment Information:

*Name on Credit Card:
* Credit Crad Number:
*Card Type: Visa
*Expiration Month(1-12):
*Expiration Year 2003

Shipping Address:

*First Name:
*Last Name
*Address:
*City:
State Select a state
Country
*Zip/Postal Code:
*Email:

Figure 6: SECURE CONNECTION FOR COLLECTING SENSITIVE INFORMATION

(http://www.verisign.com), and CyberTrust (http://www.cybertrust.gte.com). They found that the seal programs require their licensees to abide by posted privacy and security policies and various types of compliance monitoring in order to display a seal of trust on their web sites. They also found that the use of seals represents an industry-wide effort to use self-enforcement mechanisms to make the Internet a safe and comfortable forum for customers to exchange accurate information and conduct online transactions. The above were important features that have to be incorporated into the security design for the online sales phase.

Encryption methods were also explained. These included using .Net encryption classes to encrypt sensitive information. Encryption refers to the process by which data is transformed into a format that is unreadable. The intended recipients, however, have the necessary keys to unlock the format (Mackey, 2003). Encryption offers four major services to protect data: confidentiality, integrity, authenticity, and non-repudiation. Confidentiality is the protection of data. Integrity is the ability to ensure that data is not modified by unauthorized persons. Authenticity is the validation of the identity of the sender. Finally, the ability to provide non-repudiation ensures that the sender cannot deny sending the message.

Obviously, using encryption can enhance the security build up for an e-commerce application. One method was to use Hash algorithms to encrypt credit card information before it was passed from the browser to the server and then decrypted before it was stored in a database table. By using methods such as the above, the students demonstrated how to ensure data security in the online sales phase of ecommerce activities.

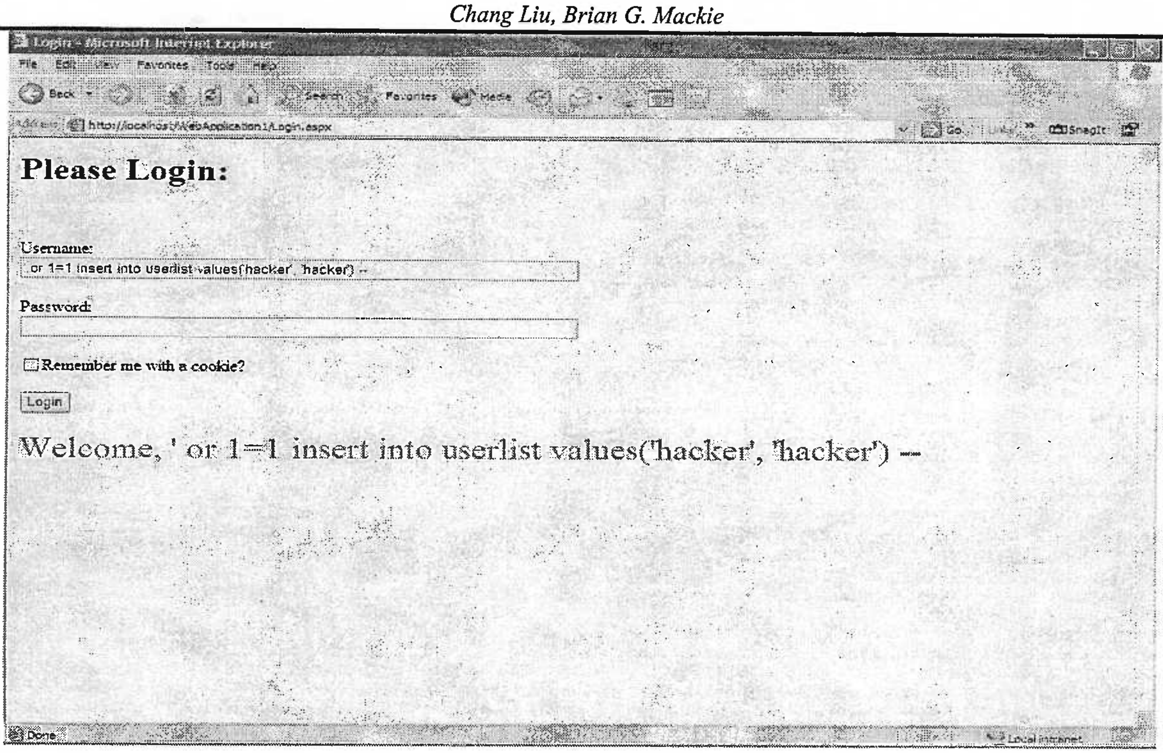
SECURITY IN AFTER-SALES PHASE

The focus of the security design for the after-sales function in an e-commerce application is on providing customers secure access to the data collected from and about them. It was demonstrated that the application should allow customers to view and update /correct/delete their personal and financial information submitted to the business site.

Most often, application designers would create a login page as shown in Figure 7 to verify a user's identify against a backend user database table. Using ADO.Net database access from within ASP.Net, it was demonstrated that one could easily drag the database connection to the login page and then incorporate structured query language (SQL) queries to authenticate a user using the information stored in the backend database. Figure 8 shows the code associated with the login button displayed in the login page displayed in Figure 7.

Once the students could successfully implement the login page, and verification, the authors introduced a technique that intruders could use to access and compromise the backend database. The technique is called SQL Injection. In this type of attack the intruder attempts to pass malicious SQL code into the application in an attempt to determine rights, passwords or information about the data and structure of the backend database. This malicious code is typically appended to the legitimate SQL statement contained within the application. This can cause the malicious SQL code to be executed against the database, so the intruder can get additional information needed to gain more access into the backend database or to be granted complete access to the database because the SQL command was altered through the injected SQL code. For example, the students were required to conduct an anatomy of a SQL Injection Attack by typing the following in the user name textbox of the login page as shown in Figure 7:

' or 1=1; insert into userlist values('hacker', 'hacker') - -


Login - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
http://localhost/WebApplication1/Login.aspx
Go
Done
Local intranet

Please Login:

Username:
' or 1=1 insert into userlist values('hacker', 'hacker') --

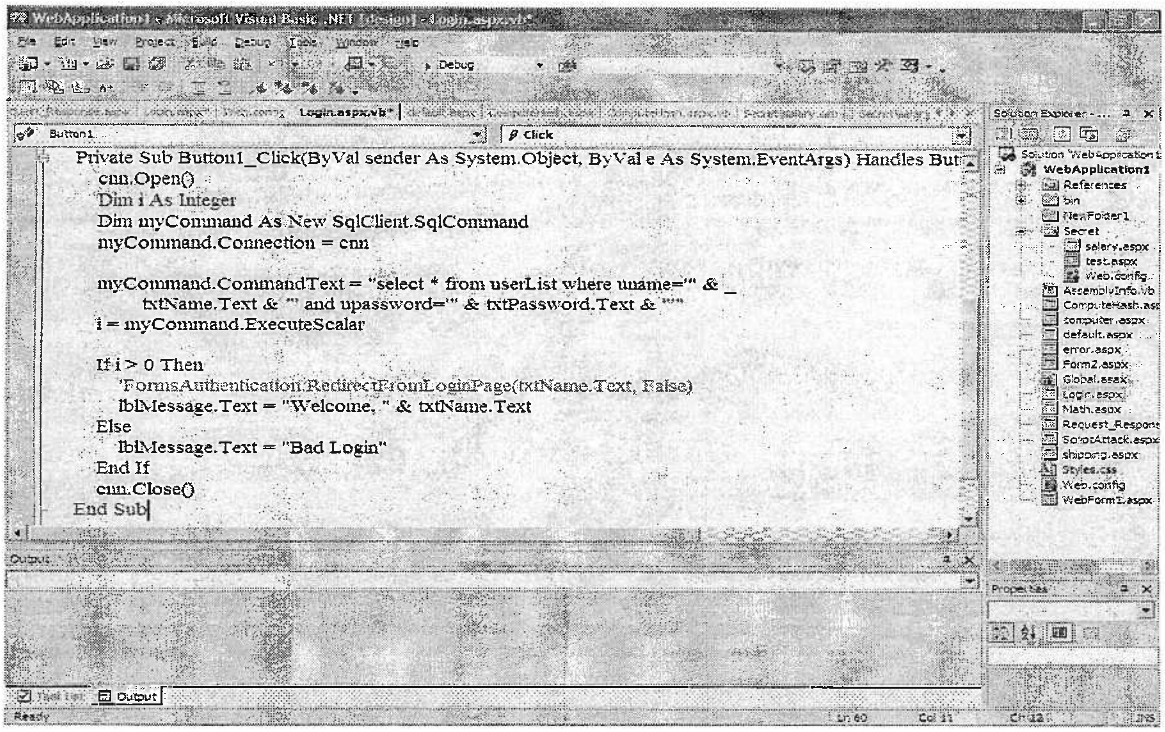
Password:

□ Remember me with a cookie?

Login

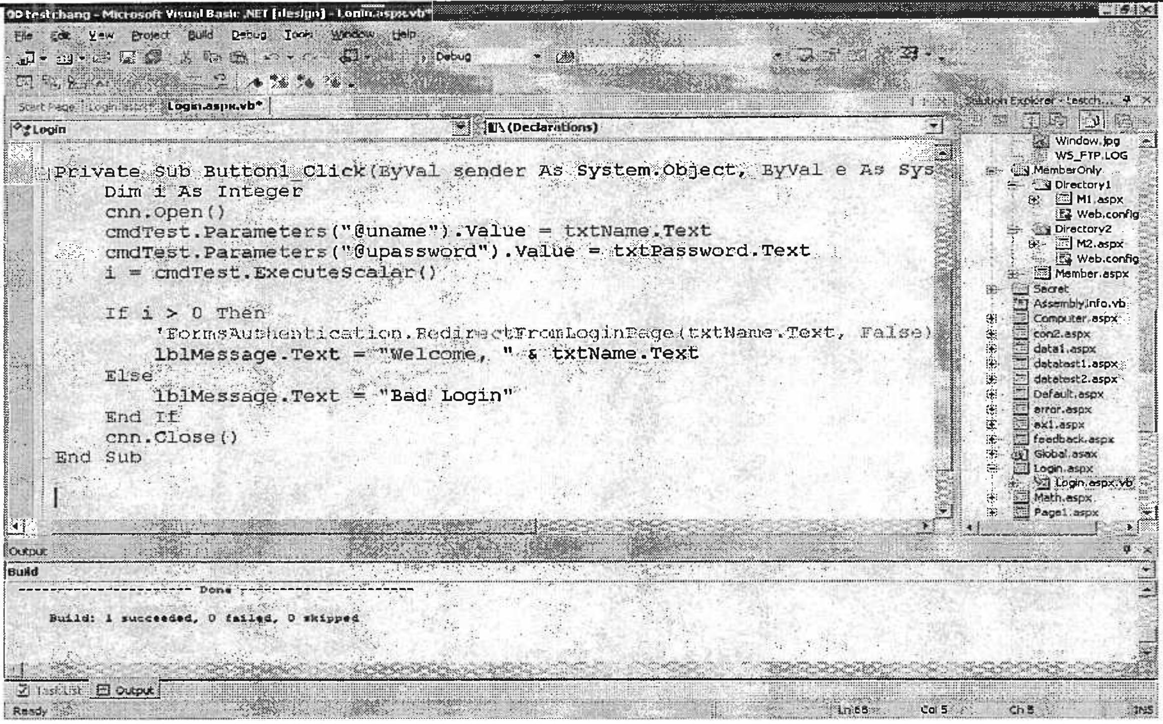
Welcome, ' or 1=1 insert into userlist values('hacker', 'hacker') --

Figure7: AN EXAMPLE OF A LOGIN PAGE TO ACCESS CUSTOMER INFORMATION



```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles But
    cnn.Open()
    Dim i As Integer
    Dim myCommand As New SqlClient.SqlCommand
    myCommand.Connection = cnn

    myCommand.CommandText = "select * from userList where uname='" & _
         txtName.Text & "' and upassword='" & txtPassword.Text & "'"
    i = myCommand.ExecuteScalar

    If i > 0 Then
      'FormsAuthentication.RedirectFromLoginPage(txtName.Text, False)
      lblMessage.Text = "Welcome, " & txtName.Text
    Else
      lblMessage.Text = "Bad Login"
    End If
    cnn.Close()
End Sub
```

Figure 8: CODE OF USING A SQL STATEMENT FOR THE LOGIN PAGE



```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As Sys
    Dim i As Integer
    cnn.Open()
    cmdTest.Parameters("@uname").Value = txtName.Text
    cmdTest.Parameters("@upassword").Value = txtPassword.Text
    i = cmdTest.ExecuteScalar()

    If i > 0 Then
        'FormsAuthentication.RedirectFromLoginPage(txtName.Text, False)
        lblMessage.Text = "Welcome, " & txtName.Text
    Else
        lblMessage.Text = "Bad Login"
    End If
    cnn.Close()
End Sub
```

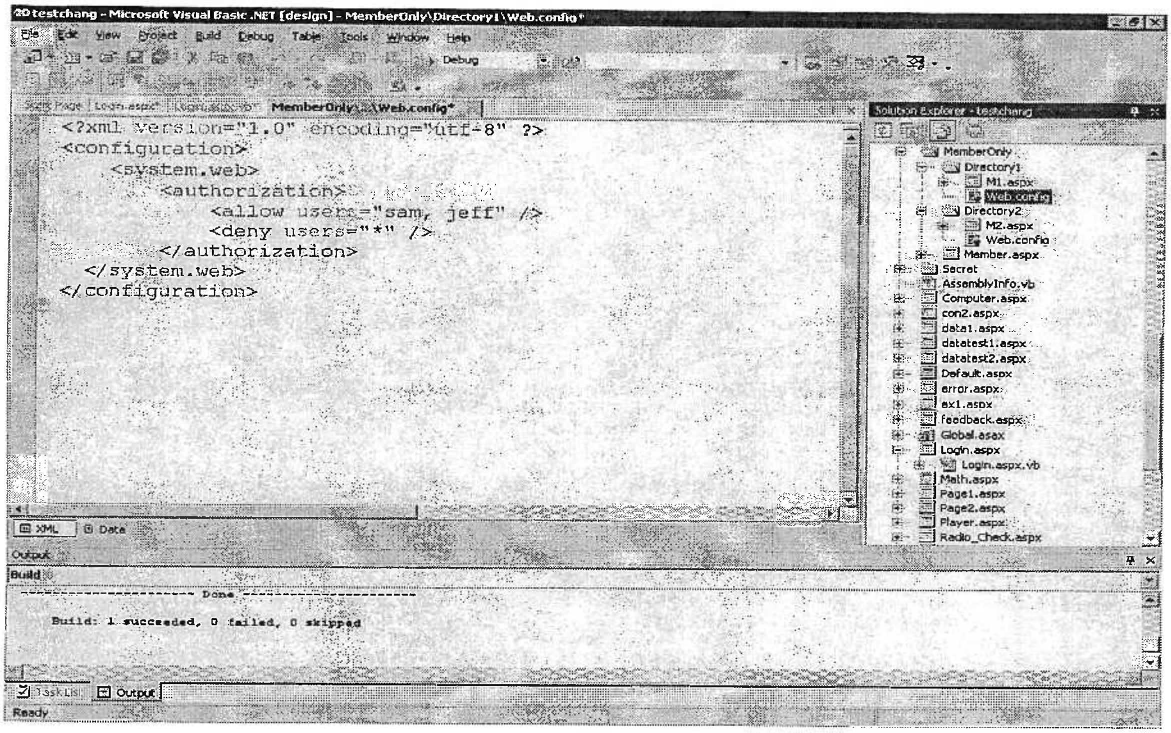
```
---------------------- Done ----------------------

    Build: 1 succeeded, 0 failed, 0 skipped
```

Figure 9: A SOLUTION OF USING STORED PROCEDURE FOR INJECTION ATTACK

This demonstrated that the intruder could insert any data into the database and also remove existing user information by issuing a delete command, or change an existing user's password by using an update command. Students were shocked that they could even delete an entire table within the database by using the drop command.

After the students understood how SQL Injection Attacks were produced, the authors then demonstrated several solutions to avoid these security beaches. These included adding validation controls to the input textboxes to verify user input by constraining certain characters including "- -", and requiring queries to be implemented as a SQL stored procedure instead of using the input textboxes to dynamically generate a SQL query. Figure 9 shows an implementation of the stored procedure technique replaces a dynamically generated SQL command. This drastically decreases the vulnerability to SQL Injection Attacks.

Another technique demonstrated was to use separate *Web.Config* files in subdirectories of an EC application. These *Web.Config* files were used to limit user access to ensure security in the after-sales phrase of the e-commerce activities. A scenario was developed in which one had to create a MemberOnly directory within their EC application to serve returning customers in the after-sales phase of an e-commerce activity. The MemberOnly directory has two subdirectories: Directory1 and Directory 2. Each directory has its own authorization rules declared in the *Web.Config* file residing in that directory. Figure 10 shows how by specifying the Form-based authentication this can be accomplished. Using Figure 10 users "*sam*" and "*jeff*" can gain access but no one else can gain access to the portion of the EC application in the subdirectory. Therefore, access is determined by your identity.



```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <system.web>
        <authorization>
            <allow users="sam, jeff" />
            <deny users="*" />
        </authorization>
    </system.web>
</configuration>
```

```
---------------------- Done ----------------------

    Build: 1 succeeded, 0 failed, 0 skipped
```

Figure 10: An Example of Authorization in Sub-Directories

CONCLUSIONS

In today's ecommerce, business web sites gather personal and financial information about their customers in order to gain competitive advantages. However, it appears to be imperative that the web site ensures that security concerns are adequately addressed.

Figure 11 is an application security model that the author proposed in the course to guide the students to build security into the three stages of e-commerce activities. Based on relevant literature review, this paper shows a teaching case that demonstrates many of the methods used in handling security issues for an EC application. In the teaching case all methods were implemented using the Microsoft ASP.Net development system. Using the three phases of marketing: During Pre-Sales: techniques used include anticipating errors and handling them appropriately. During the Online-Sales phase: methods used include validating input, creating secure data connections, using data encryption and implementing privacy seal and protection.

During the After-Sales phase: methods used include implementing authentication, authorization and avoiding vulnerabilities.

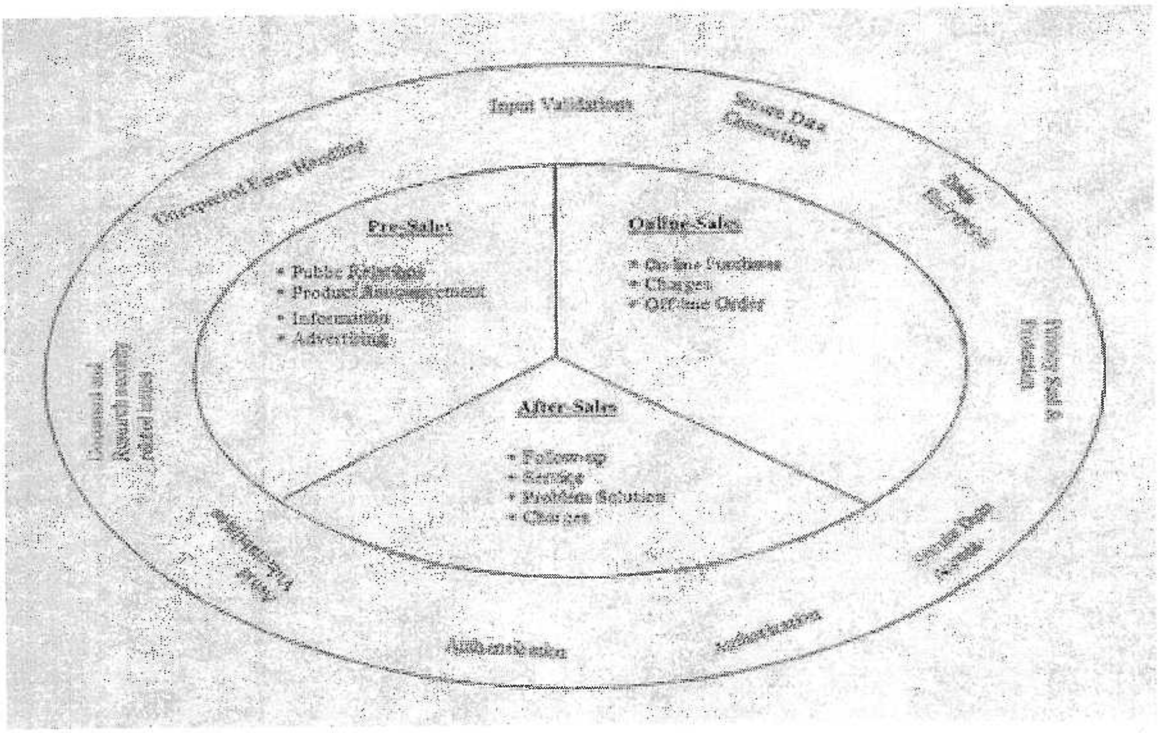


Figure 31. SECURITY FOR EC APPLICATION MODEL USING THE THREE PHASES OF MARKETING

Using the Security for EC Application model each method was developed into an exercise for the students. Each exercise included allowing the student to see the problem and implement the method to protect against or correct the security issue. The students walked away surprised at the many security issues involved and excited that they could protect against these security vulnerabilities. One student even went to another faculty to let them know that there was vulnerability in an online product the faculty was using. Overall, the students had a good feedback which reflected on the teaching evaluation at the end of the semester on those materials covered in class.

The authors definitely feel that both the students and the authors benefited from using the teaching case. They plan to expand the teaching case to incorporate more methods and to implement them both in ASP.NET and PHP.

REFERENCES

Economist (2004), "Leaders: E-commerce takes off – E-commerce takes off; To come," 371(8375), May 5, p. 9.

Franklin, C. and Wiens, J. (2004). "Are your web apps secure?" *Inforworld,* February 9, pp. 35-40.

FTC Report to Congress: Privacy online: fair information practices in the electronic marketplace, http://www.ftc.gov/os/2000/05/index.htm#22, May, 2001.

Hoffman, D. L. and Novak, T. (1999), "Building consumer trust online", *Communications of the ACM,* 42(4): 80-85.

Hof, R.D. and Hamm, S. (2002), "How e-biz rose, fell, and will rise anew", *BusinessWeek,* May 13, pp. 64-72.

Liu, C., Arnett, K.P., Capella, L., and Beatty, R.C. (1997), "Web Sites of the Fortune 500: Facing Customers through Home Pages," *Information & Management* (I&M) Vol. 31, No. 1, pp. 335-345.

Liu, C., Marchewka, J.T., and Ku, C. (2004), "American and Taiwanese PerceptionsConcerning Privacy, Trust, and Behavioral Intentions in Electronic Commerce," *Journal of Global Information Management,* 12(1), pp. 18-40.

Madden, G. and Coble-Neal, G. (2002), "Internet economic and policy: an Australian perspective", *Economic Record,* 78(242), pp. 343-357.

McClure, S. and Scambrary, J. (1999). "Scanned your web applications lately for security holes? Try these free audit tools", *Inforworld,* November 29, p. 58.

Mullaney, T.J., Green, H., Arndt, M, and Hof, R.D. (2003), "The E-biz surprise", *BusinessWeek,* May 12, pp. 60-68.

The 2002 Computer Security Institute (CSI) report, "Cyber crime bleeds U.S. corporations", http://www.gocsi.com/press/20020407.html.

Yager, T. (2003). "Where security belongs," *Inforworld,* February 17, p. 34.