Foundstone®
Professional Services
A DIVISION OF McAFEE

Foundstone Hacme Books™ Software Secure Training Application

User and Solution Guide

Author: David Raphael, Foundstone Professional Services

March 7, 2005

Introduction

Foundstone Hacme Books™ was built to provide a learning platform for developing secure software. The lessons offered by Hacme Books provide value to Software Developers, Application Penetration Testers, Software Architects, and anyone with a general interest in application security. As a completely functioning J2EE application, Hacme Books offers a comprehensive platform that is representative of real-world J2EE scenarios and the security problems that can potentially arise.

With the need to provide a tool that can raise general awareness and still challenge the more advanced developer, Hacme Books was built with several layers of vulnerabilities. In order to provide a one-size-fits-all application, Foundstone took a test driven pattern oriented design and implementation approach. Hacme Books is not a vanilla JSP -> Servlet type of application that are so often used when demonstrating security techniques. Hacme Books follows an MVC architecture that leverages the Inversion of Control (IOC) design pattern to drive factory configuration.

Hacme Books is offered with full source. You may use the tool and code for non-commercial use, however, the use of the tool and/or source for consulting services or "for profit" training is prohibited.

The emphasis in application security often focuses on lists of vulnerability classifications and checklists to ensure correctness. This is not an incorrect approach, but it favors auditors and testers. Hacme Books offers a different approach. During the lab sessions of Foundstone's Writing Secure Code - Java course, Hacme Books is transformed by students into a secure application by proper architectural patterns driven approach. Many companies will choose to filter all inbound http traffic through some sort of proxy that analyzes basic aspects of the protocol stream. This is cost effective, but has a severe penalty in accuracy due to lack of context. Reality shows us that applications know their business better than anyone else.

Being fully open source allows for an un-biased peer review process to provide feedback on what is good and what is not good as far as coding practices go. Hacme Books is uses best practices from a Software Engineering perspective but includes mistakes and errors that are representative of the kinds of issues that Foundstone sees daily through our consulting engagements.

For those of you who are genuinely security conscious, you will notice that Hacme Books initiates some outbound traffic. This is not to notify anyone of its usage, but it actually pulls a copy of the latest Foundstone class schedule and news. This is the only traffic that the application should initiate on its behalf.

Let's get started!

Copyright 2005 Foundstone Inc.
www.foundstone.com

Installation

- Prerequisites: Hacme Books is an all Java application that comes in 3 different formats - Windows Binary Installer, J2EE WAR file, Full Source Code. We will isolate this installation section to the Windows installer. The other 2 formats are for experienced Java developers and they are very self explanatory.
- Hacme Books was developed and tested on a Windows XP SP2 machine using Eclipse and Ant for build management. J2SDK 1.4.2_06 was the most recent 1.4.2 series of the Java 2 Standard Edition. The application should run on any modern J2SDK 1.4.x or higher. It won't run on 1.3.x.
- Jakarta Tomcat is distributed with Hacme Books with Apache Derby as the embedded RDBMS. It is possible that another instance of tomcat is installed already. If this is the case, refer to the Tomcat documentation on changing network port numbers.
- Installation Instructions: Hacme Books can be downloaded from the http://www.foundstone.com/s3i web site. For this installation procedure choose the Windows installer package.
    - **Figure 1** will be shown once you execute HacmeBooksSetup.exe. Click ***Next*** to continue the installation.
    - **Figure 2** will be shown once you have clicked next on the previous screen. You must click the ***I Agree*** button in order to continue installation. You are agreeing to the licensing terms in that dialogue box by clicking.
    - **Figure 3** displays the desired installation directory dialogue. You can edit the directory path that is highlighted in yellow. Click ***Next*** to continue.
    - **Figure 4** is a non-interactive screen you will see as the installer copies files to the target directory.
    - **Figure 5** is the last dialogue. Just click ***Finish***.
- Once Installation is Complete you can just browse your web browser to http://localhost:8080/HacmeBooks - It should look like **Figure 6**.
- Uninstalling is very easy. There is an uninstall program installed with the application in the Start Menu.

Copyright 2005 Foundstone Inc.

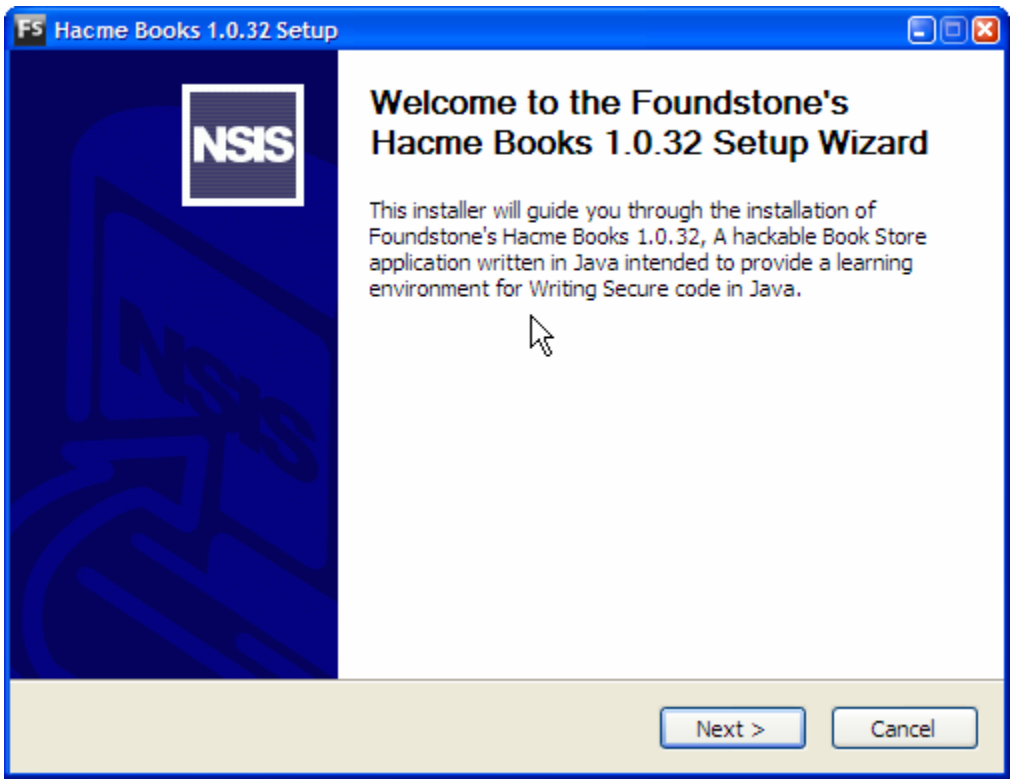


Figure 1

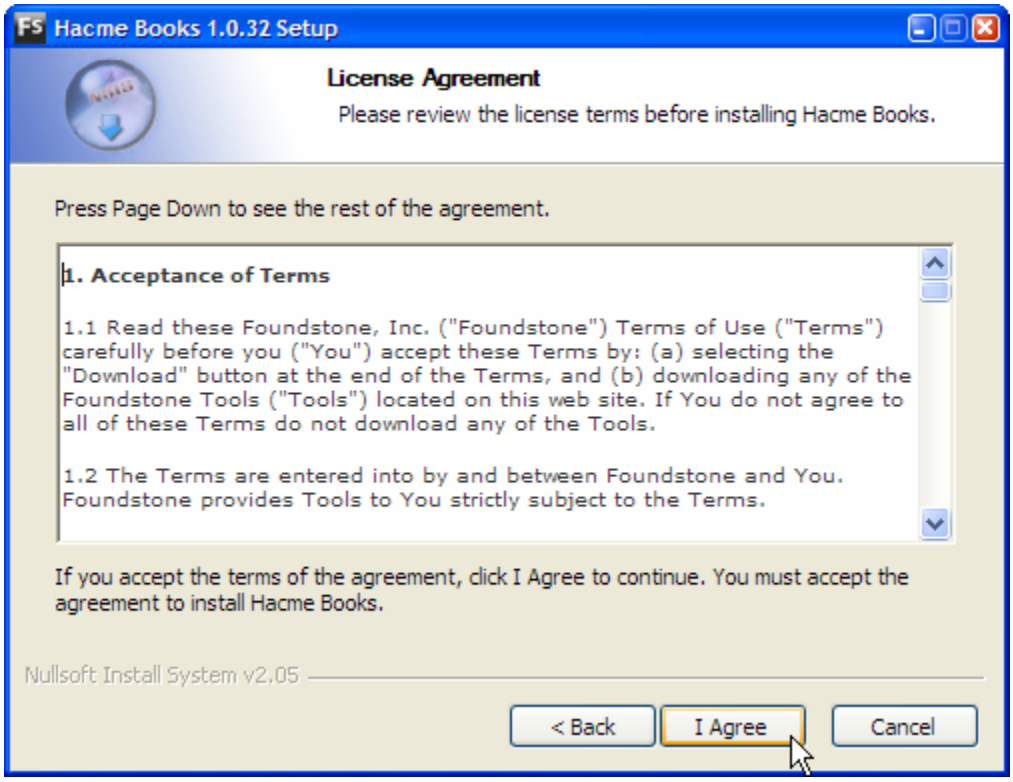


Figure 2

Copyright 2005 Foundstone Inc.
www.foundstone.com

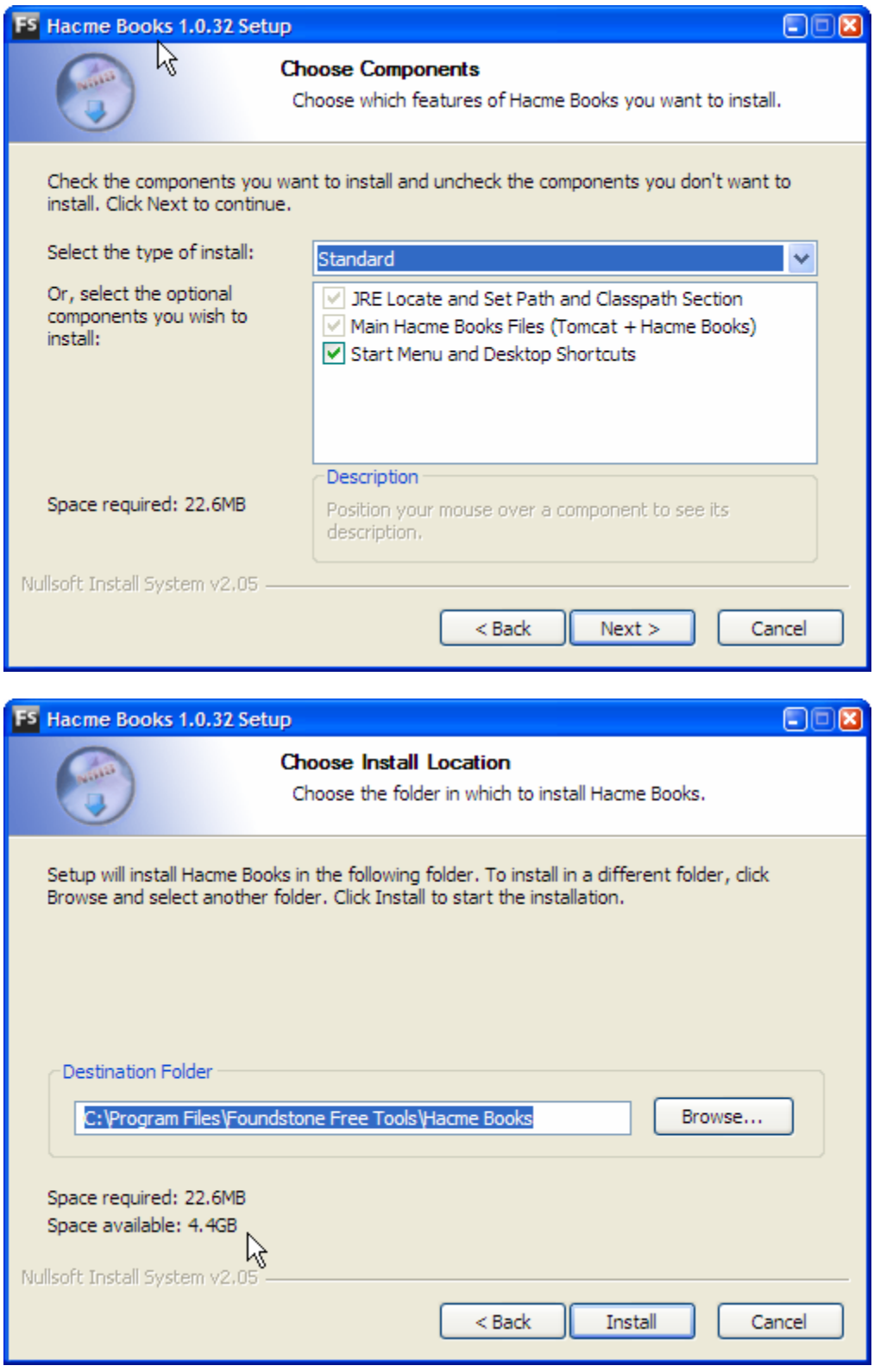


Figure 3

Copyright 2005 Foundstone Inc.
www.foundstone.com

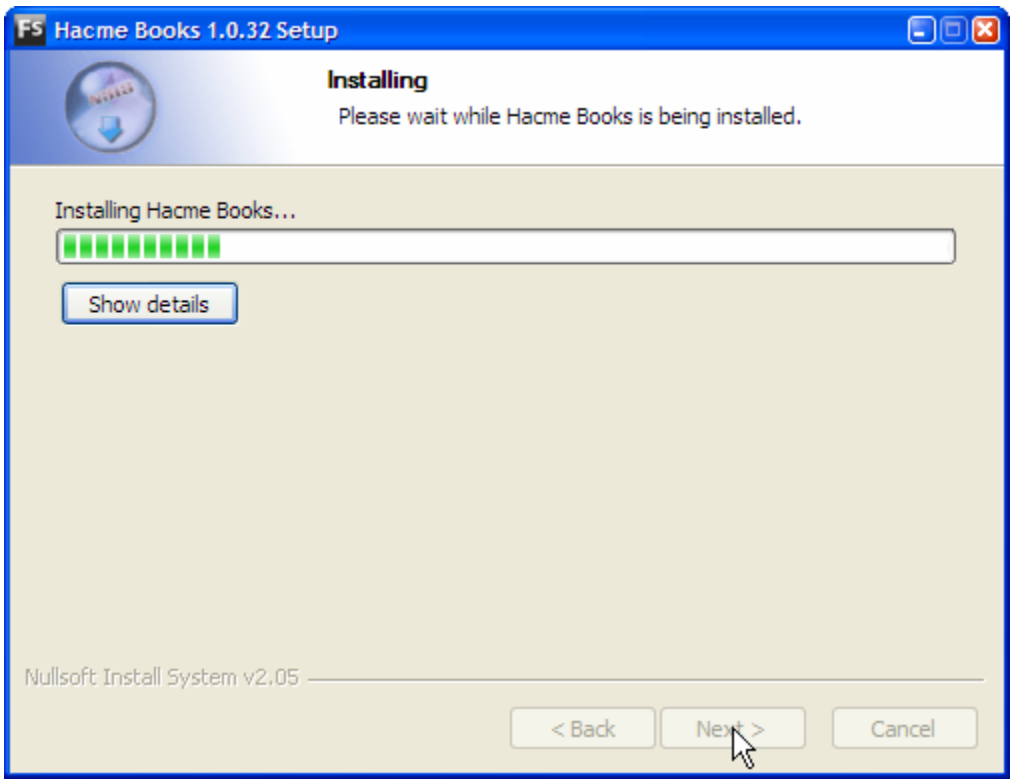


Figure 5

Copyright 2005 Foundstone Inc.
www.foundstone.com

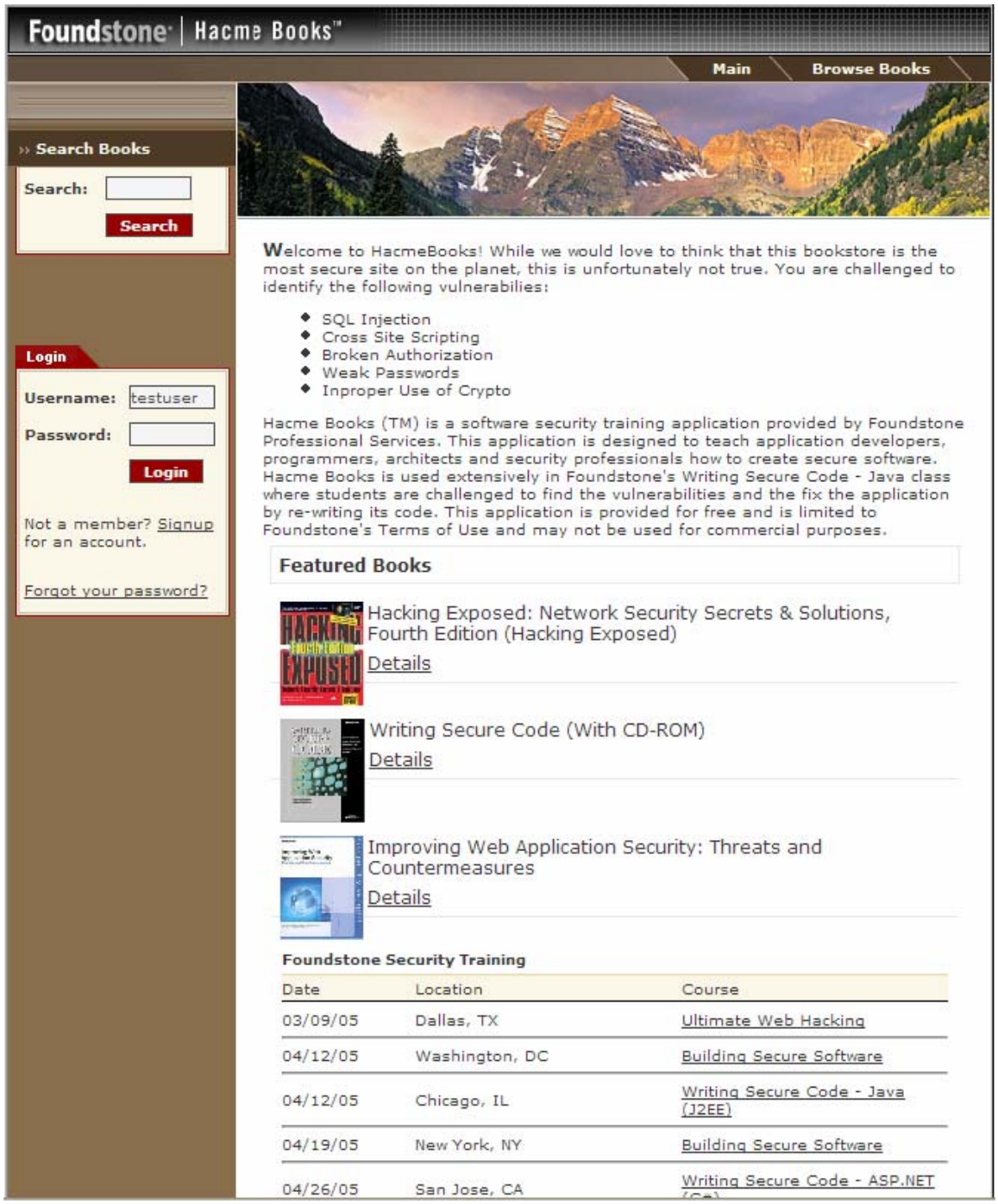


Figure 6

Copyright 2005 Foundstone Inc.
www.foundstone.com

WARNING! This application was designed as a training tool for Foundstone's Writing Secure Code – Java (J2EE) class. As such, we recommend that you attempt to identify the vulnerabilities that exist in this application before you read further.

Copyright 2005 Foundstone Inc.

Learning Guide

There are two fundamental approaches to web application security testing:

- Whitebox testing (AKA Code Review)
- Blackbox testing (AKA Penetration Test or Pen-test)

Whitebox testing is always going to produce a more accurate result based on the fact that the source code available. Testers are able to review data flows through the application from the presentation tier all the way through to the data access tier and the from data access tier through the presentation tier. Therefore, the results yielded from whitebox testing are going to be far more precise than the results gathered from blackbox testing.

For example, if there is a SQL injection vulnerability discovered in 50 different areas of a web application, a blackbox pen-tester will identify 50 vulnerabilities. However, there may be a single library that makes the database calls. In a whitebox test, the reviewer will identify the vulnerability as a single vulnerability, and will only spend a minimal amount of time on that portion of the application.

Foundstone suggests that anyone with a development background should perform a Code Review, and then perform the blackbox pen-test that is described in the rest of this document. This will validate the earlier review.

For all non-developers, the blackbox test will be most appropriate for you.

For this guide we will focus on the Pen-test approach. The Code Review methodology is a much more in depth activity that we tackle in Foundstone's Writing Secure Code classes. For more information about Foundstone's Writing Secure Code classes, go to www.foundstone.com/education.

Copyright 2005 Foundstone Inc.
www.foundstone.com

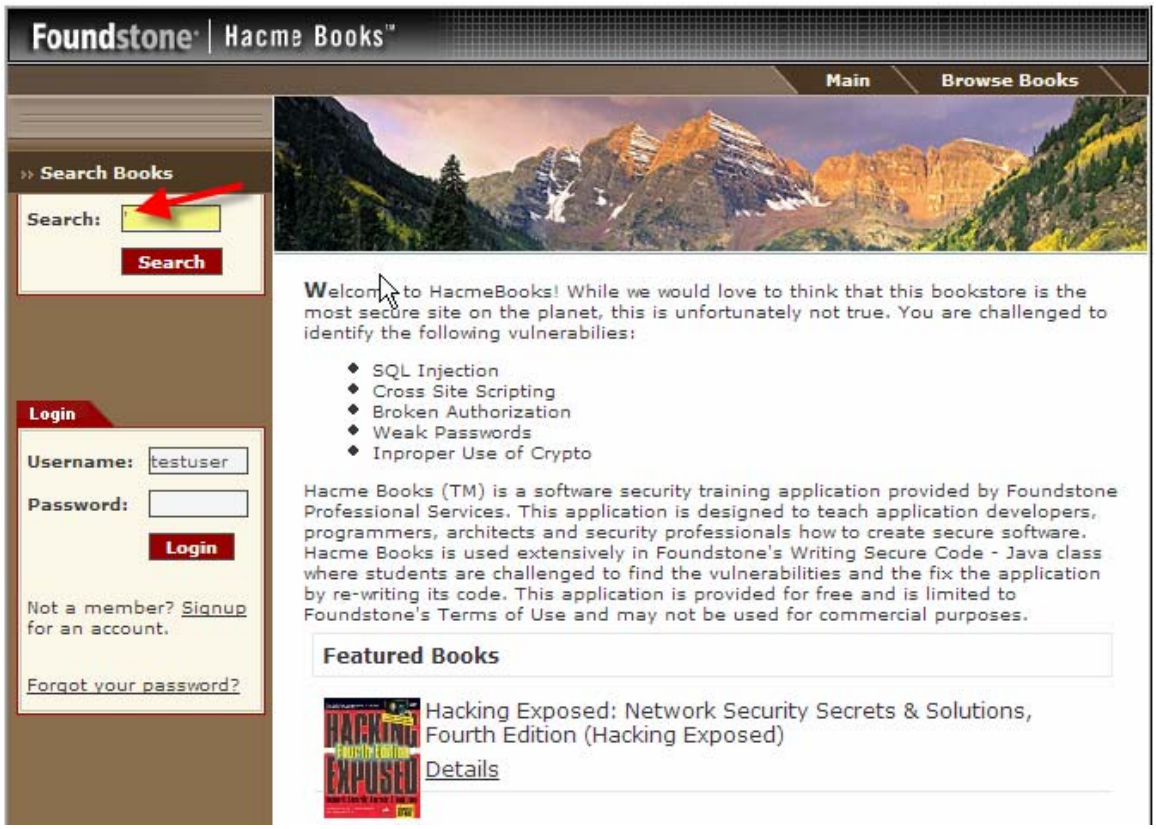
Attack Scenario 1 - Error Generation

When a malicious user is trying to gain access to a specific system, that user is going to be able to launch the best attacks when he/she has the most information. Knowledge is power when it comes to launching attacks on applications.

One of the best things you can do as a developer is to limit the amount of system information that a user can gather from the application by creating fault conditions. Unfortunately, the developers of Hacme Books have decided that the most verbose error reporting is the best error reporting. Because of this, any faults that the attacker generates in the system are reported back to them with full Java stack traces!

This vulnerability can be demonstrated as follows:

1. From the home page enter some malicious input into the search box. One of the best tests to determine if an application is performing any input validation is to enter a single apostrophe into the search input field:



Copyright 2005 Foundstone Inc.

2. After entering the apostrophe in the search box, the attacker can see that there is quite a bit of information being returned. From this single fault condition the attacker now has the following information:

- Database Type – Hypersonic SQL (just Google org.hsqldb !)
- Application Server Type – Apache Tomcat
- Other filters that are running in the Servlet stack
- The fact that this is a J2EE application – Identified by the Java namespaces

3. The attacker now has quite a bit of information to begin launching their first attacks.

Copyright 2005 Foundstone Inc.

Attack Scenario 2 - SQL Injection

General SQL Injection Overview

A Malicious user is trying to gain access or create some other mishap on your system. The way that a hacker will go about doing SQL Injection attack is through trial and error. SQL Injection is the process of placing special SQL character(s) into the input flow of an application. Frequently this is accomplished via http request parameters. Here is a comprehensive overview of SQL injection: http://www.nextgenss.com/papers/advanced_sql_injection.pdf.

Based on the fact that the attacker knows the database server they are working with, they can analyze the various SQL constructs that are supported. This will provide them with their 'arsenal' necessary to successfully attack the system.

Variation 1 (Denial of Service via SQL Injection):

Security should provide CIA (Confidentiality, Integrity and Availability). A successful attack will break one or more of these attributes. In this scenario, the attacker will start by breaking availability.

A common attack pattern is referred to as DoS (Denial of Service). The attacker will use SQL injection as their attack technique in this example.

Since the attacker did their Google search on how to shutdown Hypersonic SQL Server, they know that this command is *SHUTDOWN.*

Since most SQL injection vulnerabilities are a result of concatenating SQL statements including input directly from the user:

```
String query = "select * from products where "
                        + createCriteria(keywords);
```

This particular code snippet takes the input from the user and processes the keywords into SQL criteria via some method that iterates over the tokenized input. This allows the users to maliciously insert extra SQL statements. In this particular case the attacker will insert a SHUTDOWN command.

A healthy SQL statement that the code will generate should look like this (user input is highlighted in red:

```
select * from products where title like '%someuserkeyword%' and like
'%someotheruserinputtedkeyword%'
```

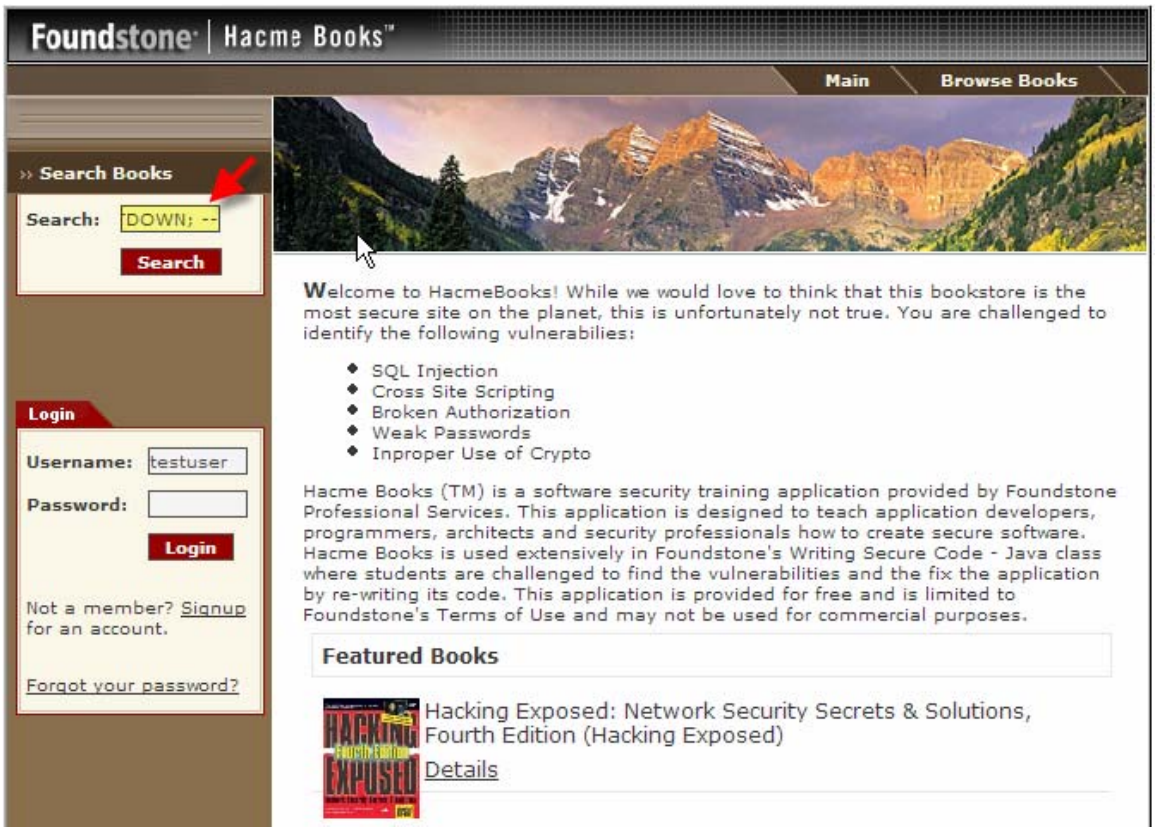
Now that the attacker understands how typical SQL statements are composed in vulnerable applications, they will start to plan their SQL injection attack.

Copyright 2005 Foundstone Inc.
www.foundstone.com

Since the input in red is the area the user has control over, the attacker knows that an effective SHUTDOWN should look like this:

```
select * from products where title like '%'; SHUTDOWN; --%' and like
'%someotheruserinputtedkeyword%'
```

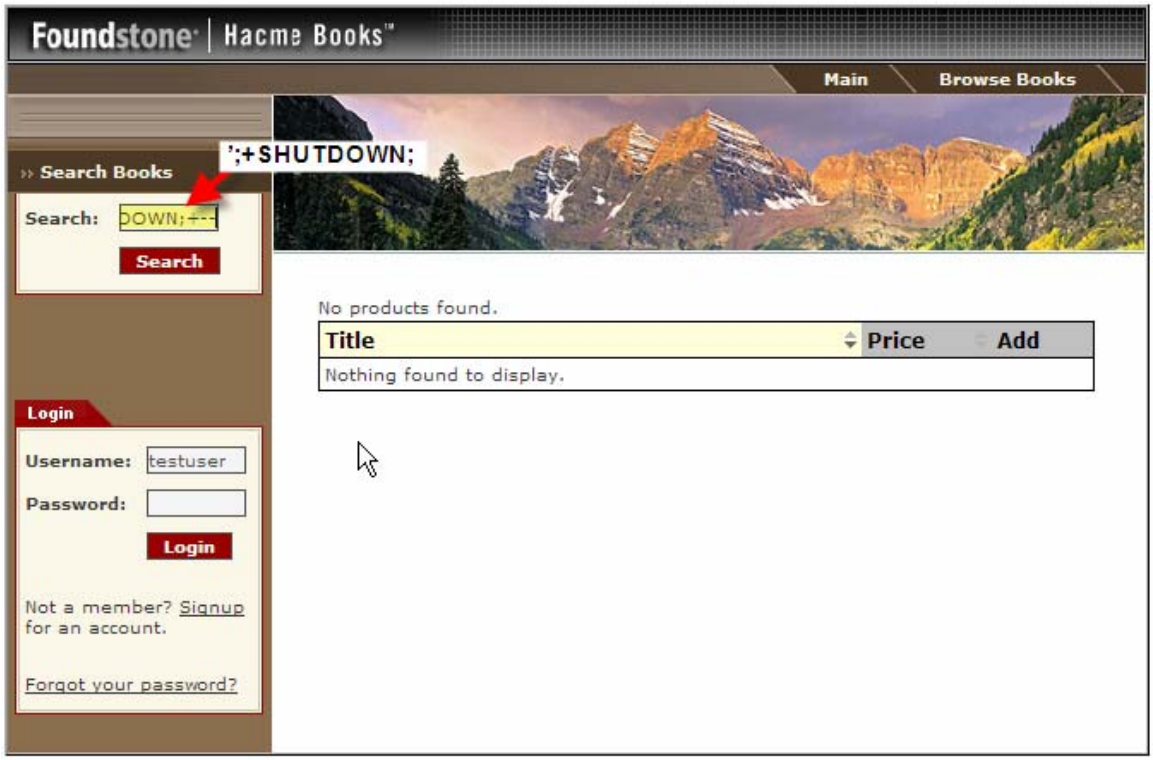
Because the – is a comment in SQL, the rest of the SQL statement has been ignored by the SQL interpreter, and it should be a successful attack. Here is a demo of the attack:



However, this attack failed because it did not cause a DoS. But the attacker can get around this problem.

Search engines generally tokenize input into separate pieces for methods like the createCriteria() method which was seen earlier. Since the attacker knows that the '+' character forces a search engine to treat the input as one keyword, they will just tweak the attack:

Copyright 2005 Foundstone Inc.



By utilizing the '+' functionality of the search engine, the attack was successful:

Copyright 2005 Foundstone Inc.
www.foundstone.com

Variation 2 (Data Tampering via SQL Injection):

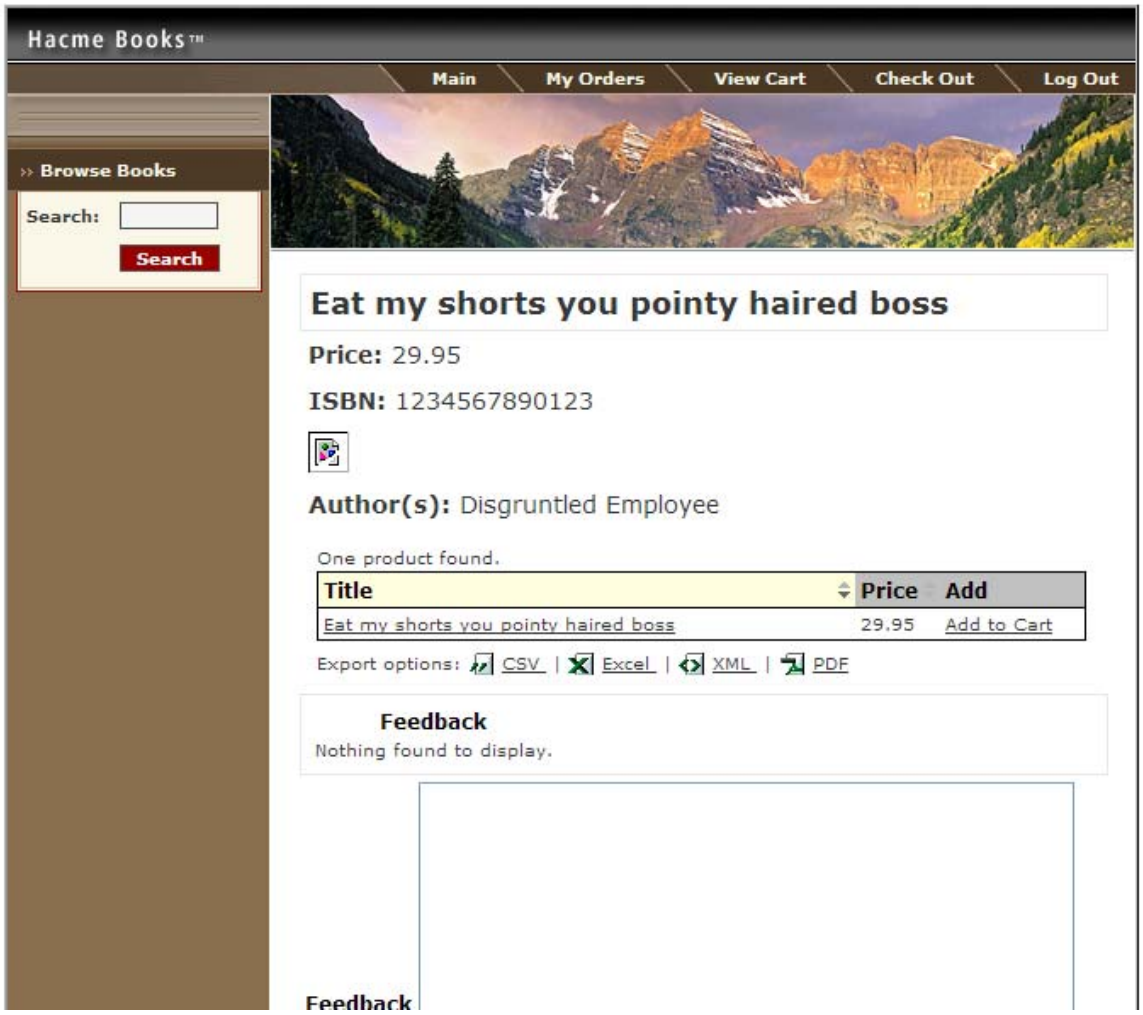
The malicious user will now attack the 'I' in CIA; Integrity. Let's say the attacker is a bitter Hacme Books employee that wants to get back at his/her employer for unfair treatment.

In this scenario, the attacker wants to add a book to the company's database that will be embarrassing.

There are ways to determine the schema of the target database. These techniques are covered thoroughly in the aforementioned SQL injection whitepaper. For this scenario, it is assumed that the attacker understands the data schema.

Using SQL Injection, the attacker can do a search on their newly created title and it should be in the database. Here is the details page:

Copyright 2005 Foundstone Inc.
www.foundstone.com

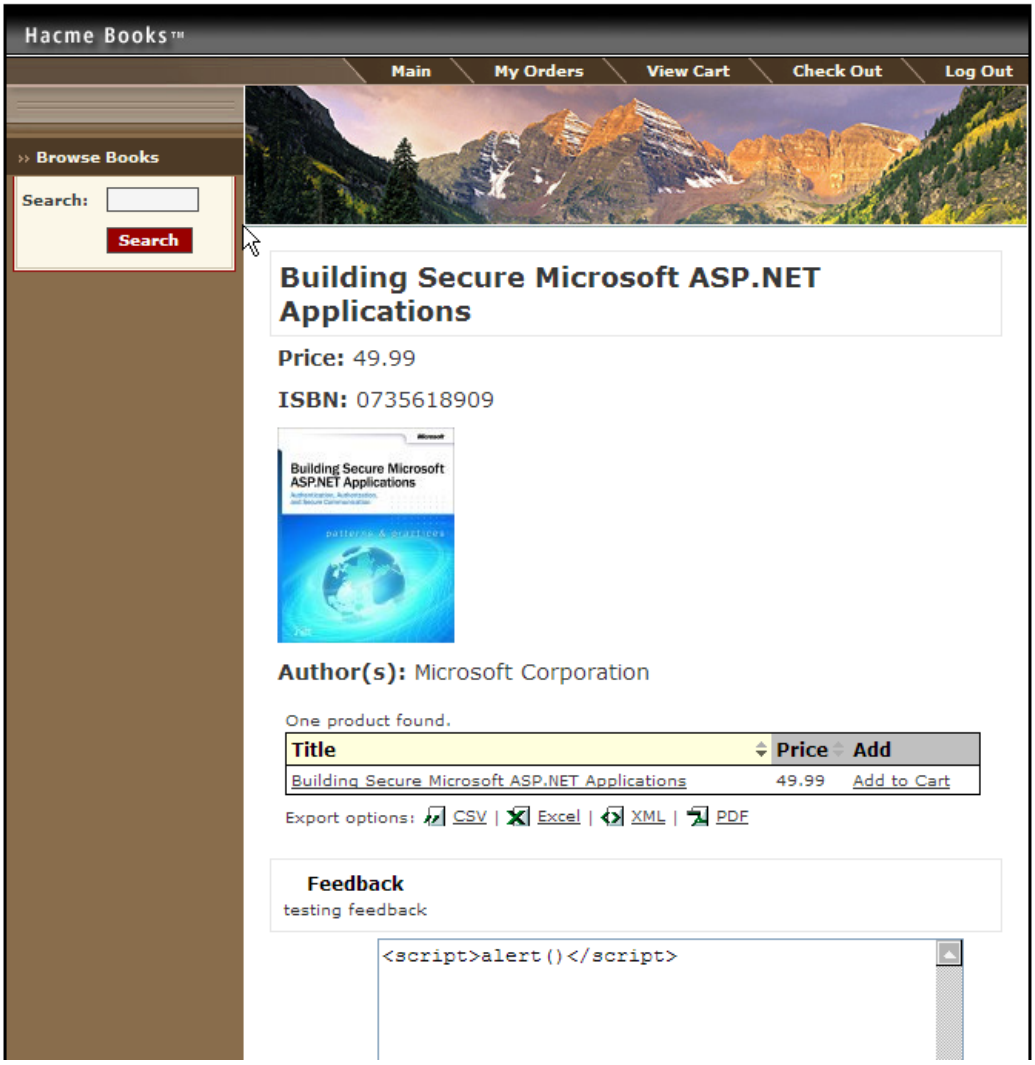


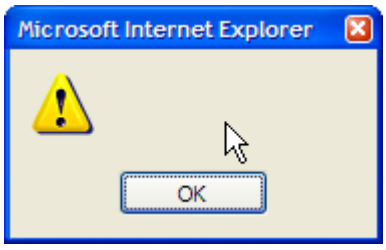
Copyright 2005 Foundstone Inc.

Attack Scenario 3 - XSS – Cross Site Scripting

Cross site scripting is one of the most common attacks seen on the public internet. It is quite commonly used for luring attacks.

It is performed by entering <script>alert()</script>.





This particular vulnerability could send sensitive information to another person's website if the user were to provide that information. This is often considered a social engineering attack.

Copyright 2005 Foundstone Inc.
www.foundstone.com

Attack Scenario 4 - Crypto Wannabe

Many times developers believe that obscuring data makes it protected. The wonderful developers of Hacme Books believe in generosity. Because they are so generous, they frequently hand out coupons and advertise promotions on the radio. In this scenario, the attacker heard that Hacme Books is offering a 15% discount on all books for a limited time only. The hacker has written down several different codes from different times.

- 15 %
  - AE0DB0B00E
- 25%
  - BEAAABB00E
  - BE0ABDB00E

The attacker now stacks all of those codes on top of each other and sees:

AE0DB0B00E
BEAAABB00E
BE0ABDB00E

Notice that there some common text with just these 3 different examples. The attacker knows that 2 of them share the 25% discount, but there are other common points.

BEAAABB00E
BE0ABDB00E

The attacker can then deduce that a discount code is based on some sort of substitution algorithm. Since the last four characters of the coupons are the same, and there are four digits in a year…they can deduce that the last 4 characters of the coupon should be the year.

BEAAAB2005
BE0ABD2005

And it is able to deduce that the letter ‘B’ is the representation of the number ‘2’.

2EAAAB2005
2E0ABD2005

It now appears that the developer has decided to substitute numbers with letters.

1504202005
2511122005
2501242005

Ok. It now appears that this is a combination of percentage and expiration!

Percentage	Month	Day	Year
15	04	20	2005
25	11	12	2005
25	01	24	2005

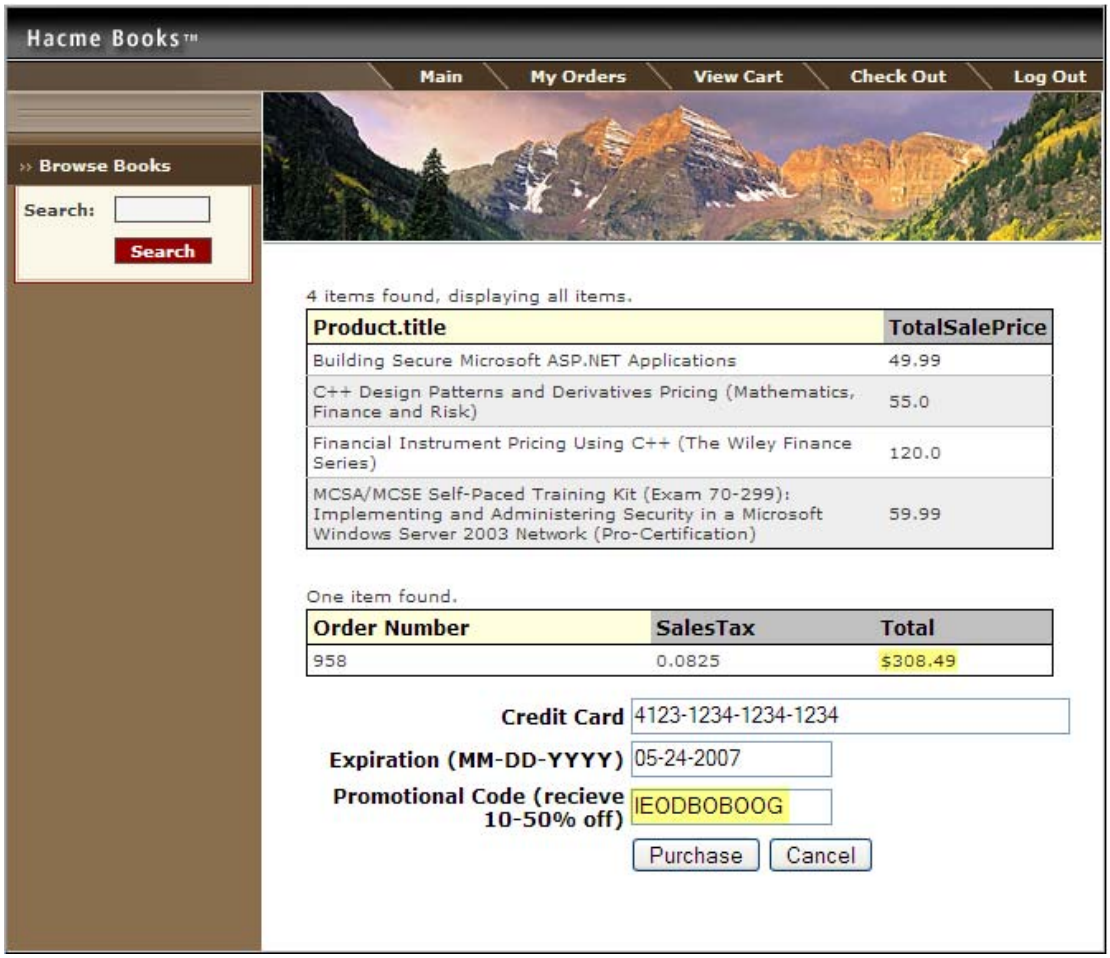
Now the attacker can test this theory. The attacker creates their own coupon and executes it against a purchase

Copyright 2005 Foundstone Inc.

How about a 95% off for the next couple of years!

```
95          04    20    2007
IE          OD    BO    BOOG
```

```
Test this coupon: IEODBOBOOG
```



Copyright 2005 Foundstone Inc.
www.foundstone.com

Attack Scenario 5 - Broken Access Control

It is often easy to overlook access control scenarios that are horizontal in nature. We tend to be very conscious of elevated privilege needs. Most developers effectively check for administrator privileges within the escalated code blocks.

But what happens when an application does not check if the user is crossing a horizontal privilege boundary?

In this scenario, the attacker will look at their profile or previous orders. The attacker begins by looking for certain patterns in application behavior. A particular feature raised a red flag during the application review. The password hint page doesn't ask for any particular information! It just asks the user what their username is. In a real application this might be harmless since we would just email that info out-of-band. But it highlights the fact that the application is doing database queries on username based criteria.

It is common for developers to leave comments in HTML code that is in JSPs. Often times they confuse HTML comments for JSP comments!

The attacker discovered the following in the HTML source code for the order browser:

```
            <!-- Remove -->
<!-- For testing purposes this page accepts a userId parameter -->
<span class="pagebanner">2 products found, displaying all products.</span>
<table cellpadding="0" class="list userList" cellspacing="0" id="parent">
<thead>
<tr>
```

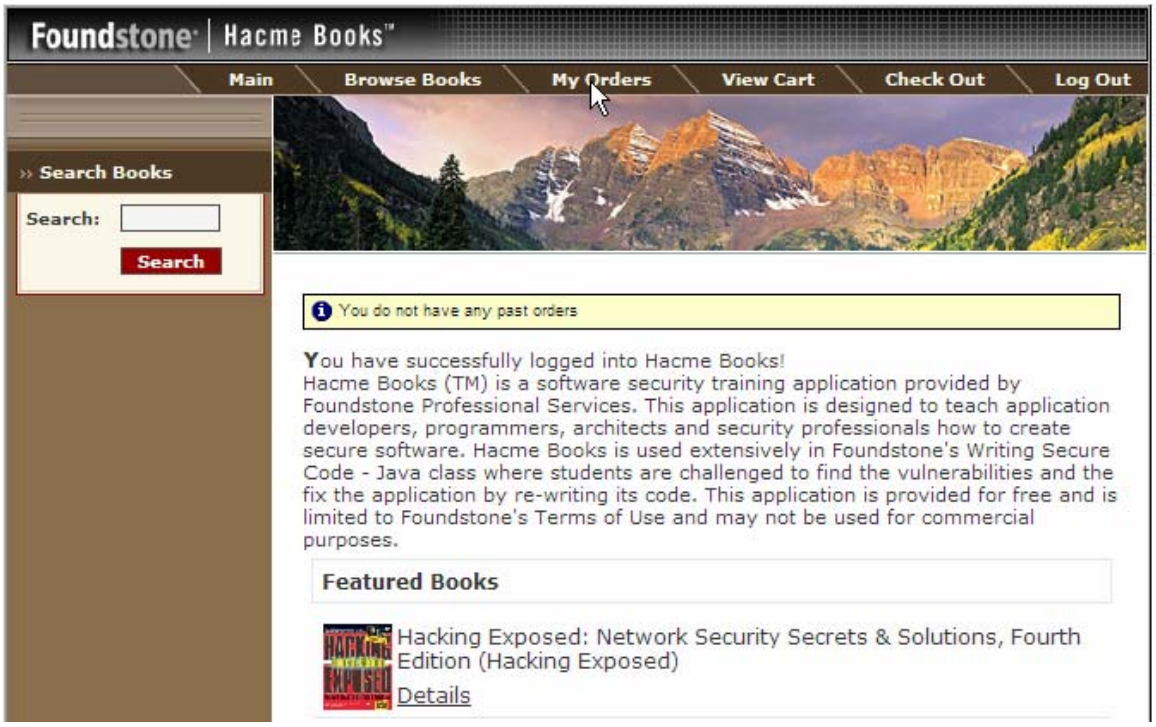
Hmmm…looks like a developer forgot to remove something from the functionality of the code!

The attacker will test this theory and see what they can find.

In order to prepare for this attack, the attacker needs to create some bogus data including at least 2 user accounts and complete some book purchase transactions. This will provide the seed data needed to launch the attack. The attacker created the users 'testuser' and 'hacker'.

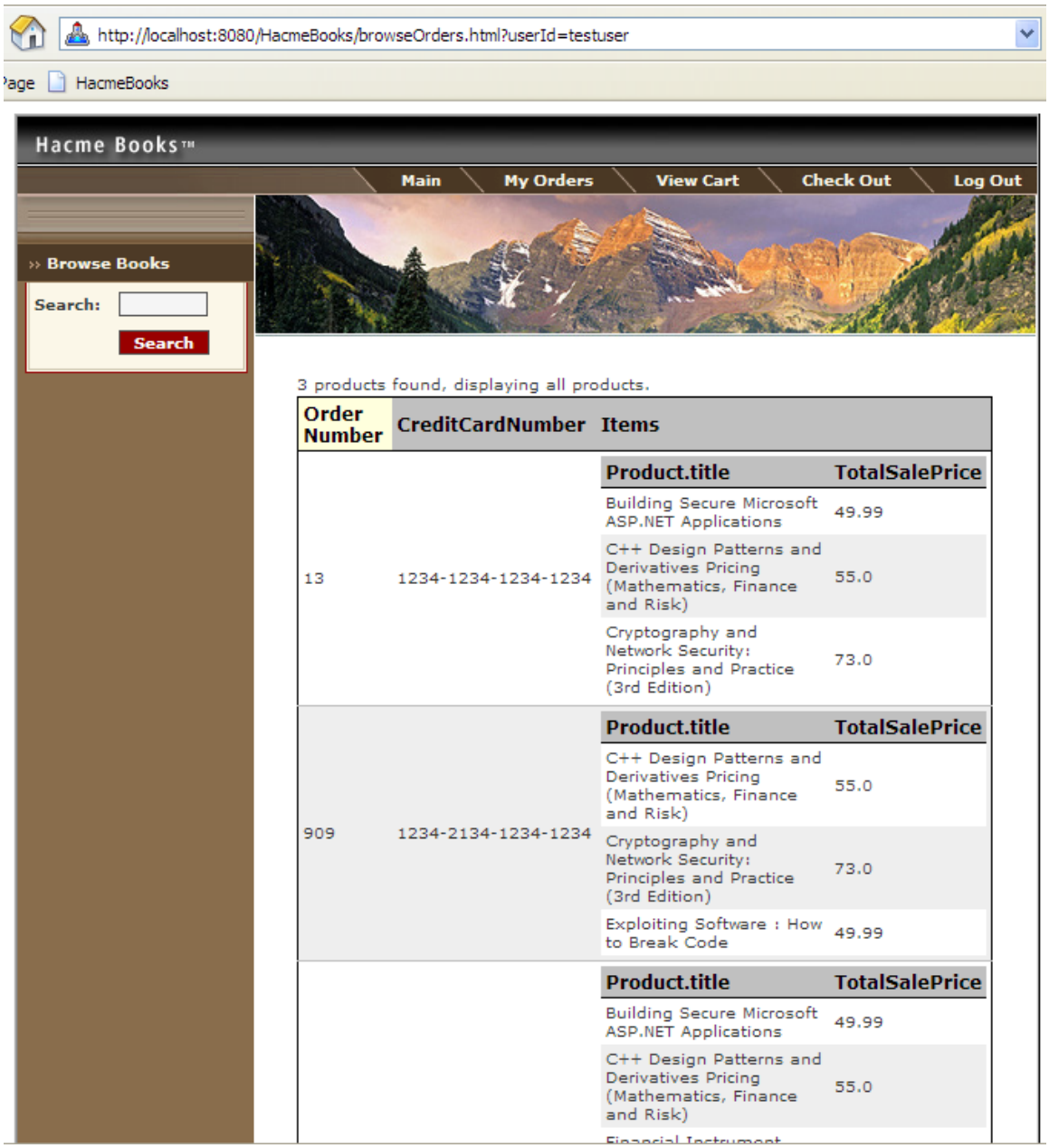
Copyright 2005 Foundstone Inc.

While logged in as ‘hacker’, they have never purchased any books. If they browse to ‘My Orders’ they will see the following:



The application displays an information message that explains that the user has never bought anything. Based on the source code comment that was seen earlier, the attacker will try to access information from ‘testuser’.

Copyright 2005 Foundstone Inc.
www.foundstone.com



The attacker's theory is correct. Now the attacker is free to launch future attacks on other user accounts.

This attack scenario points out two problems. First, the developer left comments in the source code that provided the attacker with the clues necessary to launch the attack. Second, the developer is not authorizing the action. There should be a horizontal privilege check to ensure that this information is only available to the user that made the purchases.

Copyright 2005 Foundstone Inc.
www.foundstone.com

About Foundstone Professional Services

Foundstone Professional Services, a division of McAfee, offers a unique combination of services and education to help organizations continuously and measurably protect the most important assets from the most critical threats. Through a strategic approach to security, Foundstone identifies, recommends, and implements the right balance of technology, people, and process to manage digital risk and leverage security investments more effectively.

Foundstone's Secure Software Security Initiative (S3i™) services help organizations design and engineer secure software. By building in security throughout the Software Development Lifecycle, organizations can significantly reduce their risk of malicious attacks and minimize costly remediation efforts. Services include:

- Source Code Audits
- Software Design and Architecture Reviews
- Threat Modeling
- Web Application Penetration Testing
- Software Security Metrics and Measurement

For more information about Foundstone S3i services, go to www.foundstone.com/s3i.

Foundstone S3i training is designed to teach programmers and application developers how to build secure software and to write secure code. Classes include:

- Building Secure Software
- Writing Secure Code – Java (J2EE)
- Writing Secure Code – ASP.NET (C#)
- Ultimate Web Hacking

For the latest course schedule, go to www.foundstone.com/education.

Copyright 2005 Foundstone Inc.