Hands-On Ethical Hacking and Network Defense, 3rd Edition

Chapter 10

Hacking Web Servers

Objectives

After completing this chapter, you will be able to:

- Describe Web applications
- Explain Web application vulnerabilities
- Describe the tools used to attack Web servers

Understanding Web Applications

- Writing a program without bugs
  - Nearly impossible
  - Some bugs create security vulnerabilities
- Web applications also have bugs
  - Larger user base than standalone applications
  - Bugs are a bigger problem

Web Applications Components

- Static Web pages
  - Created using HTML
  - Display the same information regardless of time or user
- Dynamic Web pages
  - Information varies
  - Need special components
    - `<form>` element, AJAX, Common Gateway Interface (CGI), Active Server Pages (ASP), PHP, ColdFusion, JavaScript, and database connectors

Web Forms

- Use `<form>` element or tag in HTML document
  - Allows customer to submit information to Web server
- Web servers
  - Process information from a form using a Web application
    - Easy way for attackers to intercept data users submit
  - Security testers should recognize when forms are used

Web Forms

- Web form example:

```
<html>
<body>
<form>
Enter your username:
<input type="text" name="username">
<br>
Enter your password:
<input type="text" name="password">
</form></body></html>
```

Web Forms

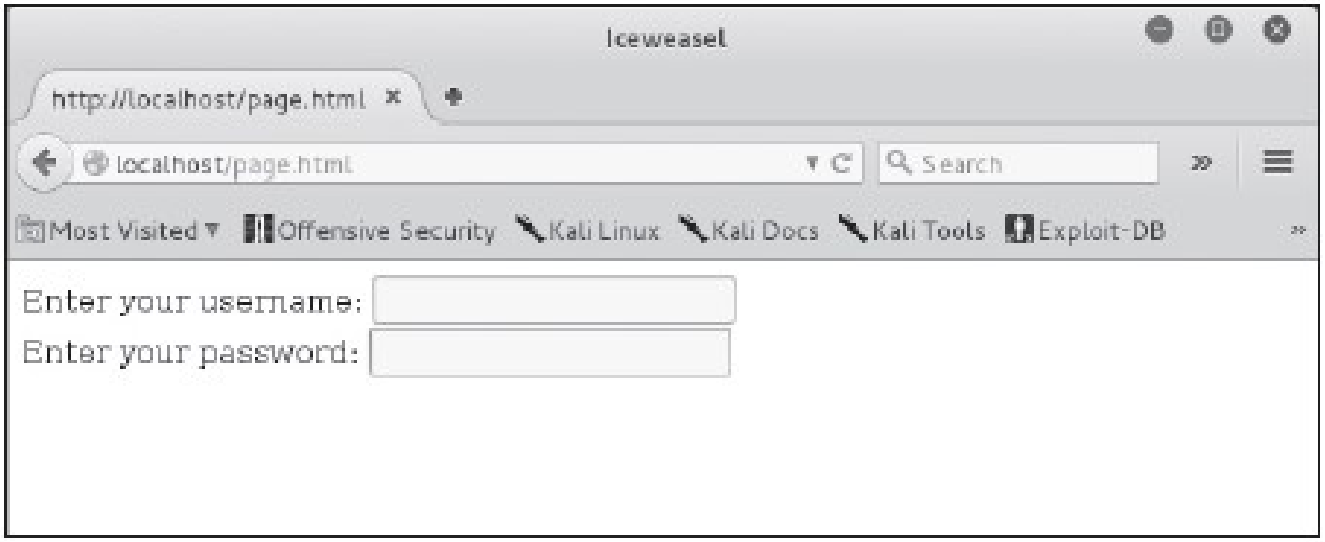

Figure 10-1 An HTML Web page with a form

Common Gateway Interface

- Handles moving data
  - From Web server to Web browser
- Dynamic Web pages
  - Many are created with CGI and scripting languages
- CGI's main goal:
  - Determines how Web server passes data to Web browser
    - Relies on C/C++, Perl or another scripting language to create dynamic Web pages
    - Programs are written in different languages

Common Gateway Interface

- CGI example written in Perl:

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "Hello Security Testers!";
```

Third Party Frameworks and Libraries

- A few of the hundreds of frameworks designed to make programming easier:
  - Spring
  - JSF
  - AngularJS
  - Yeoman
  - Sass
  - Vaadin
- As third-party libraries grow in popularity
  - Keeping them current and secure is important

Active Server Pages

- Main difference from HTML pages
  - HTML documents can be displayed on the fly
    - User requests a Web page, one is created
- Uses scripting languages
  - JScript
  - VBScript
- Has evolved
  - Largely replaced by ASP.NET
- Not all Web servers support ASP

Active Server Pages

- ASP example:

```
<HTML>
<HEAD><TITLE> My First ASP Web Page </TITLE></HEAD>
<BODY>
  <H1>Hello, security professionals</H1>
  The time is <% = Time %>.
</BODY>
</HTML>
```

- Microsoft does not want users to be able to view an ASP Web page's source code
  - Makes ASP more secure

Apache Web Server

- Apache
  - Another Web Server program
  - Said to run on more than twice as many Web servers as IIS
- Advantages
  - Works on just about any *nix and Windows platform
  - Free
- Apache Web Server daemon (httpd) 2.4 is included on the Kali DVD

Using Scripting Languages

- Web pages
  - Developed using several scripting languages
    - VBScript
    - JavaScript
- Many security-testing tools are written with scripting languages
- Macro viruses and worms may take advantage of cross-site scripting vulnerabilities
  - Most are based on scripting language

PHP Hypertext Processor

- PHP
  - Enables creation of dynamic Web pages
  - Similar to ASP
- Open-source server-side scripting language
  - Embedded in an HTML Web page
    - Using PHP tags <?php and ?> browsers
  - Users cannot see PHP code on their Web browser
- Originally used mainly on UNIX systems
  - More widely used now
    - Macintosh and Windows

PHP Hypertext Processor

- PHP example:

```
<html>
<head>
<title>My First PHP Program </title>
</head>
<body>
<?php echo '<h1>Hello, Security Testers!</h1>'; ?
>
</body>
</html>
```

Cold Fusion

- Server-side scripting language
  - Used to develop dynamic Web pages
  - Created by the Allaire Corporation
- Uses proprietary tags
  - Written in ColdFusion Markup Language (CFML)
- CFML Web applications
  - Can contain other technologies (e.g., HTML or JavaScript)

Cold Fusion

- CFML example:

```
<html>
<head>
<title>Using CFML</title>
</head>
<body>
<CFLOCATION URL="www.isecom.org/cf/index.htm"
  ADDTOKEN="NO">
</body>
</html>
```

VBScript

- Visual Basic Script
  - A scripting language developed by Microsoft
  - Converts static Web pages into dynamic Web pages
- Advantage:
  - Powerful programming language features
- The Microsoft Security Bulletin
  - Starting point for investigating VBScript vulnerabilities

VBScript

- VBScript example:

```
<html>
<body>
<script type="text/vbscript">
document.write("<h1>Hello Security Testers!
  </h1>")
document.write("Date Activated: " & date())
</script>
</body>
</html>
```

VBScript

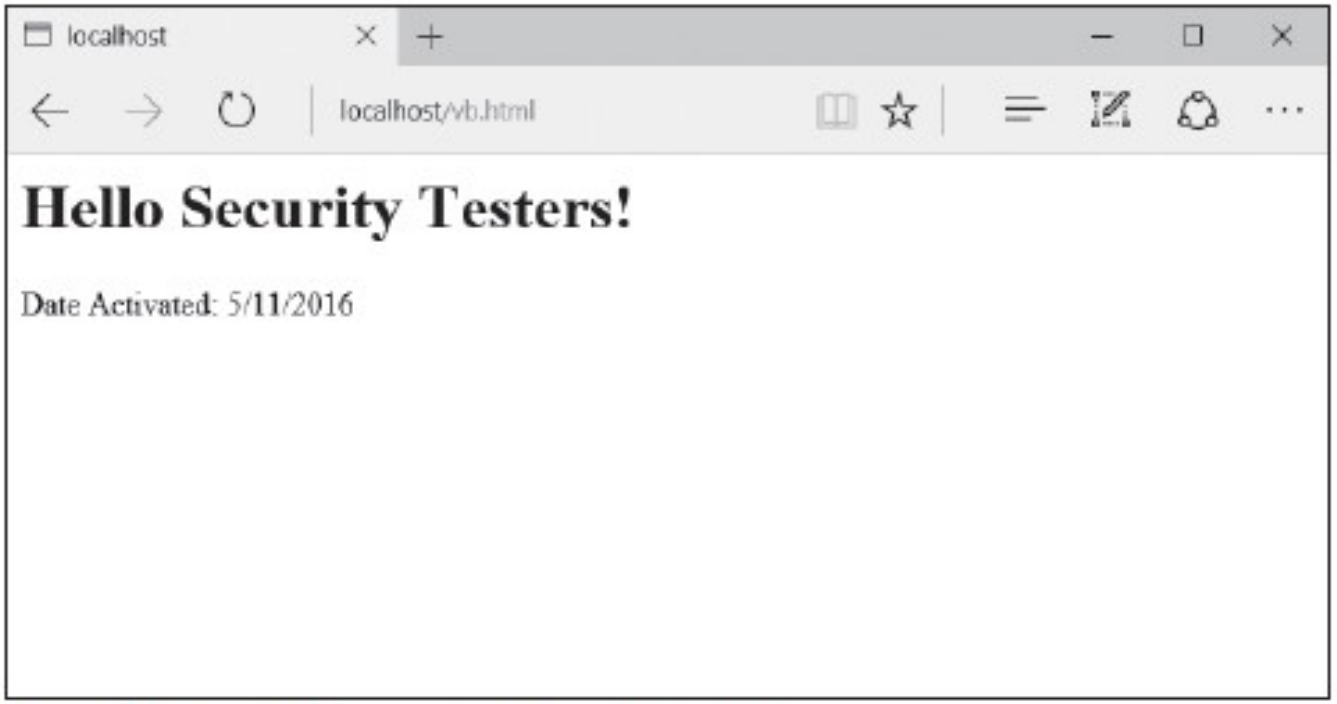

Figure 10-6 A Web page created with VBScript

Source: © 2016 Microsoft

JavaScript

- Popular scripting language used for creating dynamic Web pages
- Has power of programming language
  - Branching
  - Looping
  - Testing
- Widely used
- Variety of vulnerabilities
  - Exploited in older Web browsers

JavaScript

- JavaScript example:

```
<html>
<head>
<script type="text/javascript">
function chastise_user()
{
alert("So, you like breaking rules?")
document.getElementByld("cmdButton").focus()
}
</script>
</head>
<body>
```

JavaScript

- JavaScript example (cont'd.):

```
<h3>"If you are a Security Tester, please do not click
  the command button below!"</h3>
<form>
<input type="button" value="Don't Click!"
  name="cmdButton"
onClick="chastise_user()" />
</form>
</body>
</html>
```

JavaScript

Figure 10-7 A command button created with JavaScript

Source: © 2016 Microsoft

Figure 10-8 An alert message created with JavaScript

Source: © 2016 Microsoft

Connecting to Databases

- Most Web pages can display information stored on a database server
- The technology used to connect Web applications to database servers
    - Depends on the OS
    - Theory is the same

Open Database Connectivity

- Open Database Connectivity (ODBC)
  - A standard database access method
- ODBC interface
  - Allows application to access data stored in a database management system, or any system that can understand and issue ODBC commands
- Interoperability is accomplished by defining:
  - Standardized representation for data types
  - Library of ODBC function calls
  - Standard method of connecting to and logging on

Object Linking and Embedding Database

- OLE DB
  - Set of interfaces that enable applications to access data stored in DBMS
- Designed by Microsoft
  - Faster, more efficient, and more stable than ODBC
- Relies on connection strings
  - Allow the application to access data stored on external device
- Different providers can be used
  - Depends on data source

Object Linking and Embedding Database

OLE DB provider	Description in connection string
Microsoft Active Directory Service	Provider=ADSDSOOBJECT
Advantage	Provider=Advantage OLE DB Provider
AS/400 (from IBM)	Provider=IBMDA400
AS/400 and VSAM (from Microsoft)	Provider=SNAOLEDB
MS Commerce Server	Provider=Commerce.DSO.1
DB2	Provider=DB2OLEDB
Microsoft Jet	Provider=Microsoft.Jet.OLEDB.4.0
Microsoft.ACE	Provider=Microsoft.ACE.OLEDB.12.0
MS Exchange	Provider=EXOLEDB.DataSource
MySQL	Provider=MySQLProv
Oracle (from Microsoft)	Provider=msdaora
Oracle (from Oracle)	Provider=OraOLEDB.Oracle
MS SQL Server	Provider=SQLOLEDB

Table 10-1 OLE DB providers

ActiveX Data Objects

- ActiveX Data Objects (ADO)
  - A programming interface for connecting Web applications to a database
  - Defines a set of technologies that allow desktop applications to interact with Web
- Steps for accessing a database:
  - Create ADO connection
  - Open database connection created
  - Create ADO recordset
  - Open recordset and select data you need
  - Close recordset and database connection

Understanding Web Application Vulnerabilities

- Many platforms and programming languages can be used to design a Web site
  - Application security
    - As important as network security
- Attackers controlling a Web server can:
  - Deface the Web site
  - Destroy the application's database or sell contents
  - Gain control of user accounts
  - Perform secondary attacks
  - Gain root access to other application servers

Application Vulnerabilities and Countermeasures

- Open Web Application Security Project (OWASP)
  - Not-for-profit organization
  - Finds and fights Web application vulnerabilities
  - Publishes Ten Most Critical Web Application Security Risks
    - Built into Payment Card Industry (PCI) Data Security Standard (DSS)

Application Vulnerabilities and Countermeasures

- The OWASP Top Ten list:
  - Injection vulnerabilities
  - Authentication flaws and weaknesses
  - Cross-site scripting (XSS)
  - Insecure direct object reference
  - Security misconfigurations
  - Sensitive data exposure
  - Missing function level access control
  - Cross-site request forgery
  - Using components with known vulnerabilities
  - Unvalidated redirects and requests

Application Vulnerabilities and Countermeasures

- OWASP WebGoat project
  - Helps security testers learn how to conduct vulnerability testing on Web applications
  - Experts from all over the world use WebGoat
  - The following slides contain images of WebGoat

Application Vulnerabilities and Countermeasures

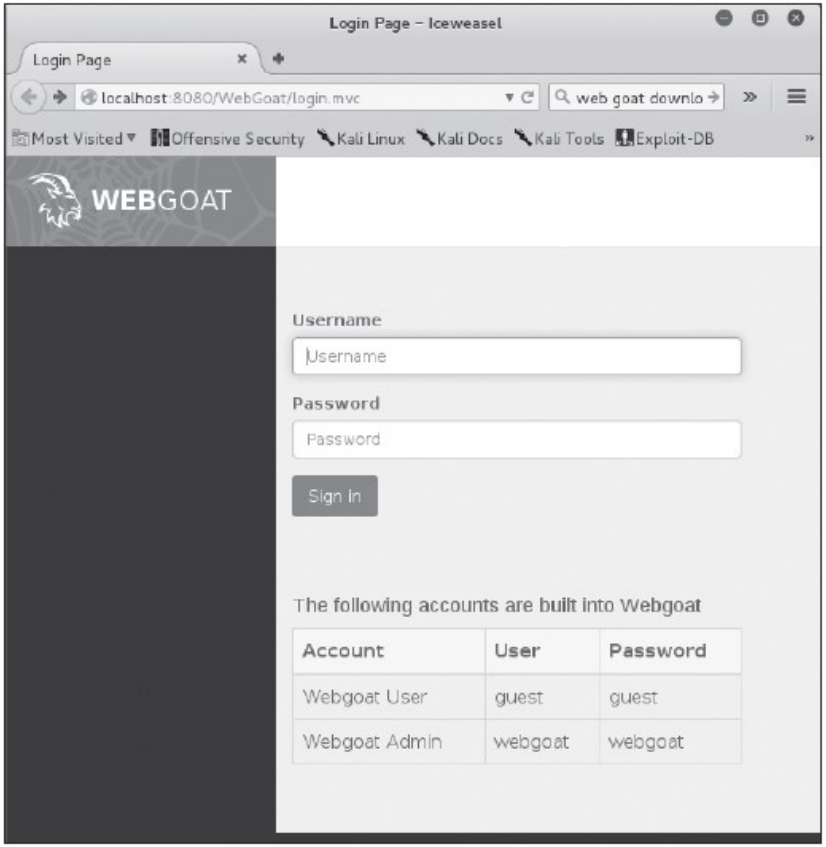

Figure 10-9 The WebGoat start page

Source: GNU GPL

Application Vulnerabilities and Countermeasures

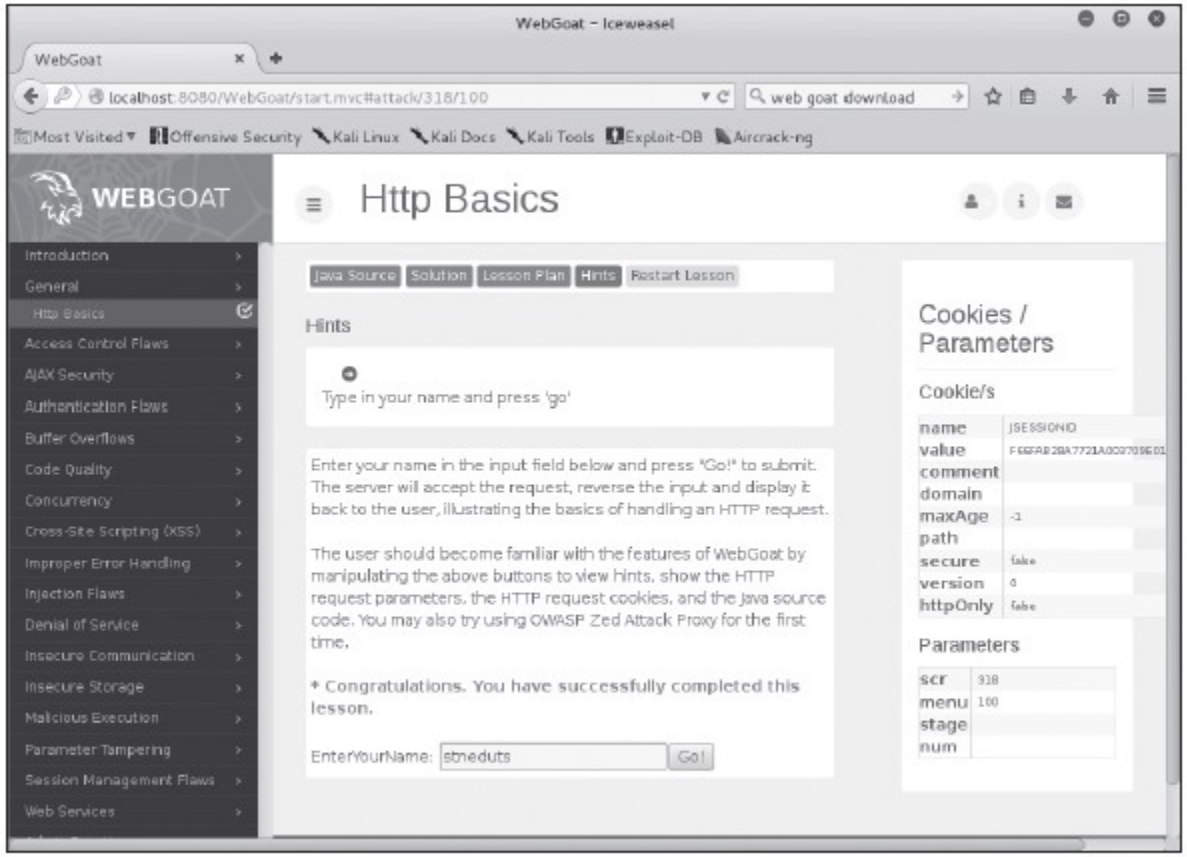

Figure 10-10 The WebGoat Hints menu

Source: GNU GPL

Application Vulnerabilities and Countermeasures

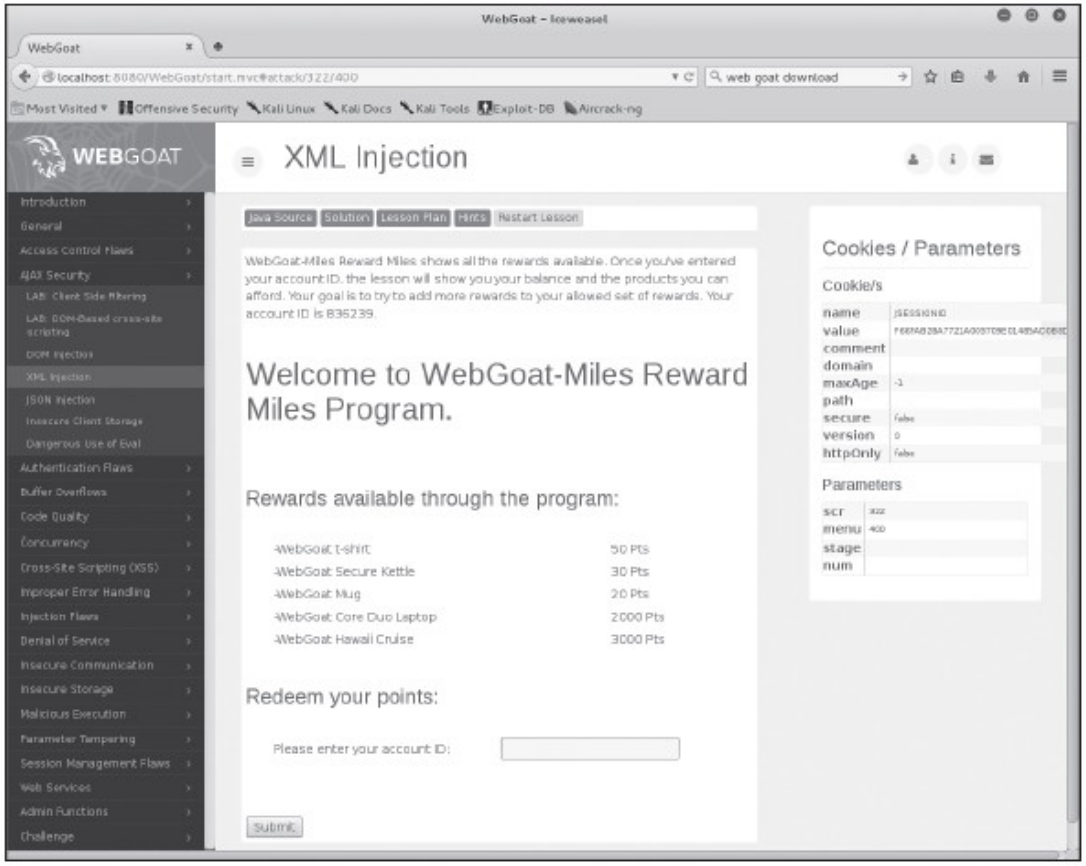

Figure 10-11 The AJAX XML injection exercise

Source: GNU GPL

Application Vulnerabilities and Countermeasures

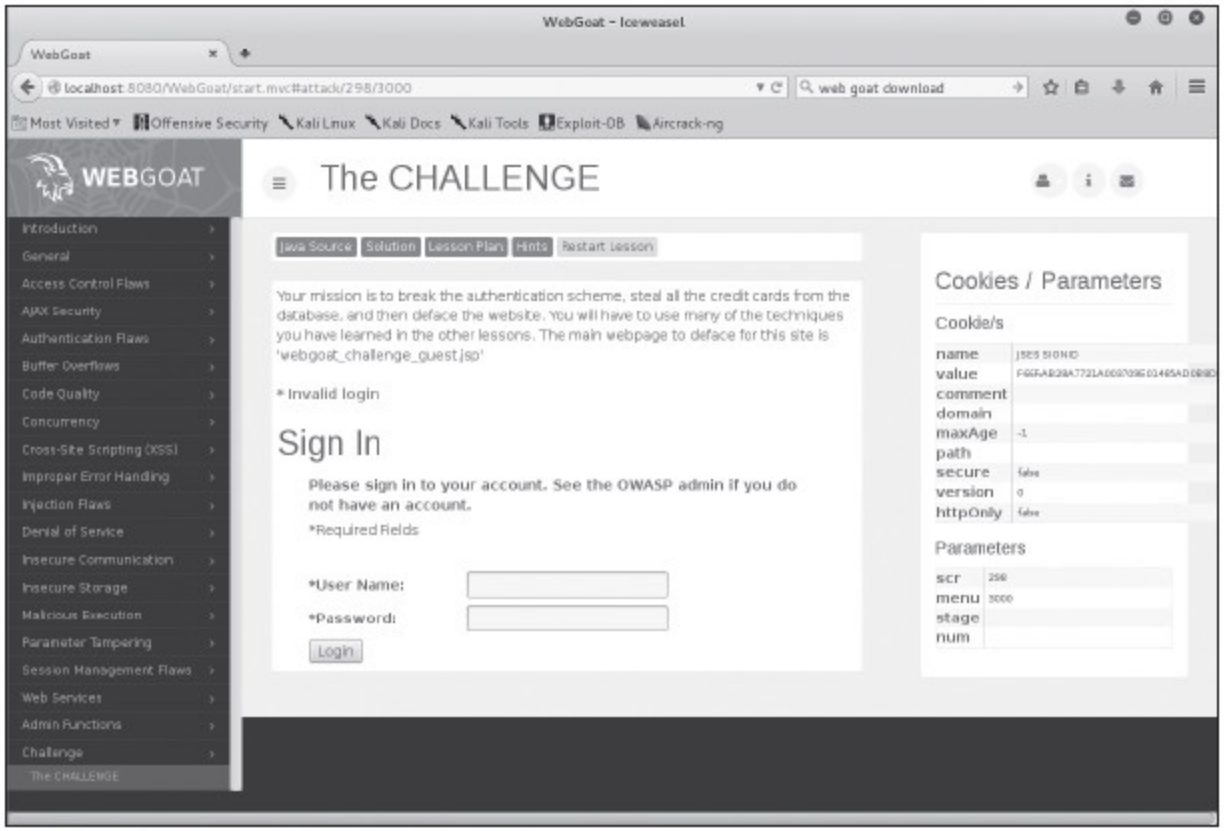

Figure 10-12 WebGoat's Challenge exercise

Source: GNU GPL

Web Application Test Execution

- Two techniques by which an application can be tested:
  - Static Application Security Testing (SAST)
    - Analyzing an application's source code for vulnerabilities
    - A reliable way to enumerate most application vulnerabilities
  - Dynamic Application Security Testing (DAST)
    - Analysis of a running application for vulnerabilities
    - Can be used alongside SAST to prioritize SAST findings

Information Gathering and Architecture Mapping

- Security testers should look for answers to some important questions:
  - Does the application have a database?
  - Does the application require authentication?
  - Does the application have static or dynamic pages?
  - What languages and platform does the application use?
  - Are there devices in-between your Web browser and the application designed to stop attacks from occurring?
  - How does data flow in the application?

Platform Security and Configuration

- Several different platforms and technologies can be used to develop Web applications
  - Attacks differ depending on platform and technology
    - Footprinting is used to discover the OS and DBMS
  - The more you know about a system, the easier it is to gather information about vulnerabilities
- Questions to consider:
  - Do the underlying platforms and components contain known vulnerabilities?
  - Is the Web Server configured to protect confidentiality of users?

Authentication and Session Testing

- Many Web applications require another server (other than the Web server) to authenticate users
  - Examine how information is passed between the two servers
  - Is an encrypted channel used or is data passed in cleartext?
  - Is the server used for authentication properly configured and patched?
  - Are logon and password information stored in a secured location?

Authorization Testing

- Authorization
  - The act of checking a user's privileges to understand if they should or should not have access to a page, field, resource, or action in an application
- Application developers
  - Commonly use hidden fields in tables and obscured URLs to enforce their access control instead of checking users' privileges
- Authorization testing can reveal major areas of concern
  - An important part of any application test

Input Validation

- Input validation
  - The act of filtering, rejecting, or sanitizing a user's untrusted input before the application processes it
- Input validation problems can lead to
  - Data disclosure
  - Alteration
  - Destruction
- Security testers should check for possibility of SQL injection used to attack the system
  - SQL injection: attacker inserts SQL commands in Web application field

Input Validation

- SQL injection example:

```
SELECT * FROM customer
WHERE tblusername = ' OR 1=1 - - AND tblpassword = ‘ ‘
```

- Because 1 and 1 is always true
  - The query is carried out successfully
  - Double hyphens (--) are used in SQL to indicate a comment

Input Validation

- Basic testing should look for:
  - Whether you can enter text with punctuation marks
  - Whether you can enter a single quotation mark followed by any SQL keywords
  - Whether you can get any sort of database error when attempting to inject SQL statements
  - Sometimes, a Web application will give a tester no indication that a SQL statement was run
    - OWASP calls this “Blind SQL injection” and it has its own set of tests that are required for detection

Error Handling

- A Web application can be configured or written to handle errors in a variety of ways
  - Developers can enable debugging
  - If debugging is left on, it can provide a rich source of information for attackers
- Developers should minimize the amount of information shared with attackers
  - When an application encounters an error

Cryptography Testing

- Many problems in cryptography are due to simple things:
  - Using bad random number generators
  - Using a known weak method of encryption
  - An application doesn’t actually enforce the use of secure channels
  - Using a self-signed certificate instead of a purchased certificate

Business Logic Testing

- Business logic
  - Refers to the flow a user is expected to follow in an application to accomplish a goal
- Example:
  - Before a wire transfer, user must first satisfy the requirement of having at least that amount of money in the transferring account
  - If user doesn’t have funds, transfer is halted
- Business logic testing
  - Involves utilizing creative ways to bypass these checks

Client-side Testing

- Client-side issues
  - Arise from code executing on the user's machine
- Key areas to consider with a client-side test:
  - Does the application store sensitive information on the client's machine in an insecure manner?
  - Does the application allow for client browser redirection if the server is fed a specially crafted request?

Tools for Web Attackers and Security Testers

- After vulnerabilities of a Web application or an OS platform are discovered
  - Security testers or attackers look for tools to test or attack the system
    - All platforms and Web application components have vulnerabilities
  - No matter which platform is used, there is a security hole and a tool capable of breaking into it

Web Tools

- The Kali DVD
  - Is packed with free tools for hacking Web applications
- You can install new tools with a simple
  - `apt-get install packagename` command

Firefox and Chrome Built-In Developer Tools

- Both come with similar set of developer tools

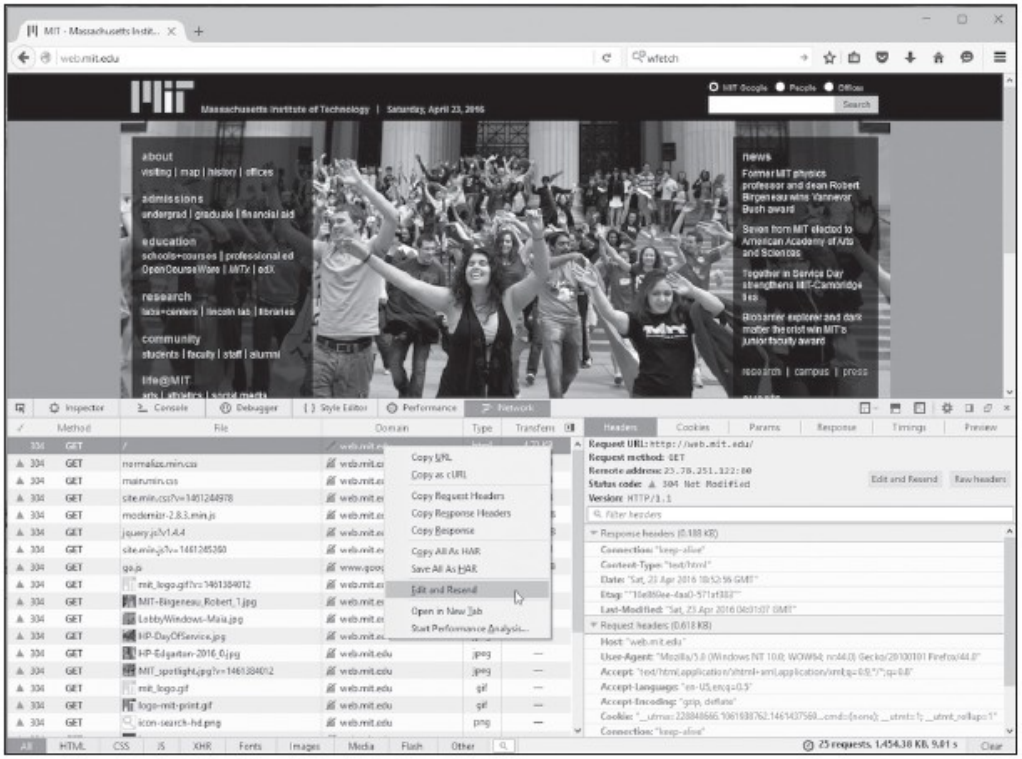

Figure 10-13 Firefox Developer Tools

Source: Mozilla Public License ("MPL") 2016

Burp Suite and Zed Attack Proxy

- Burp Suite
  - Included in Kali Linux
  - Offers the tester a number of features for testing Web applications and Web services
  - Allows you to intercept traffic between the Web browser and the server to inspect and manipulate requests before sending it to the server
- Zed Attack Proxy
  - Can be used interchangeably with Burp Suite

Burp Suite and Zed Attack Proxy

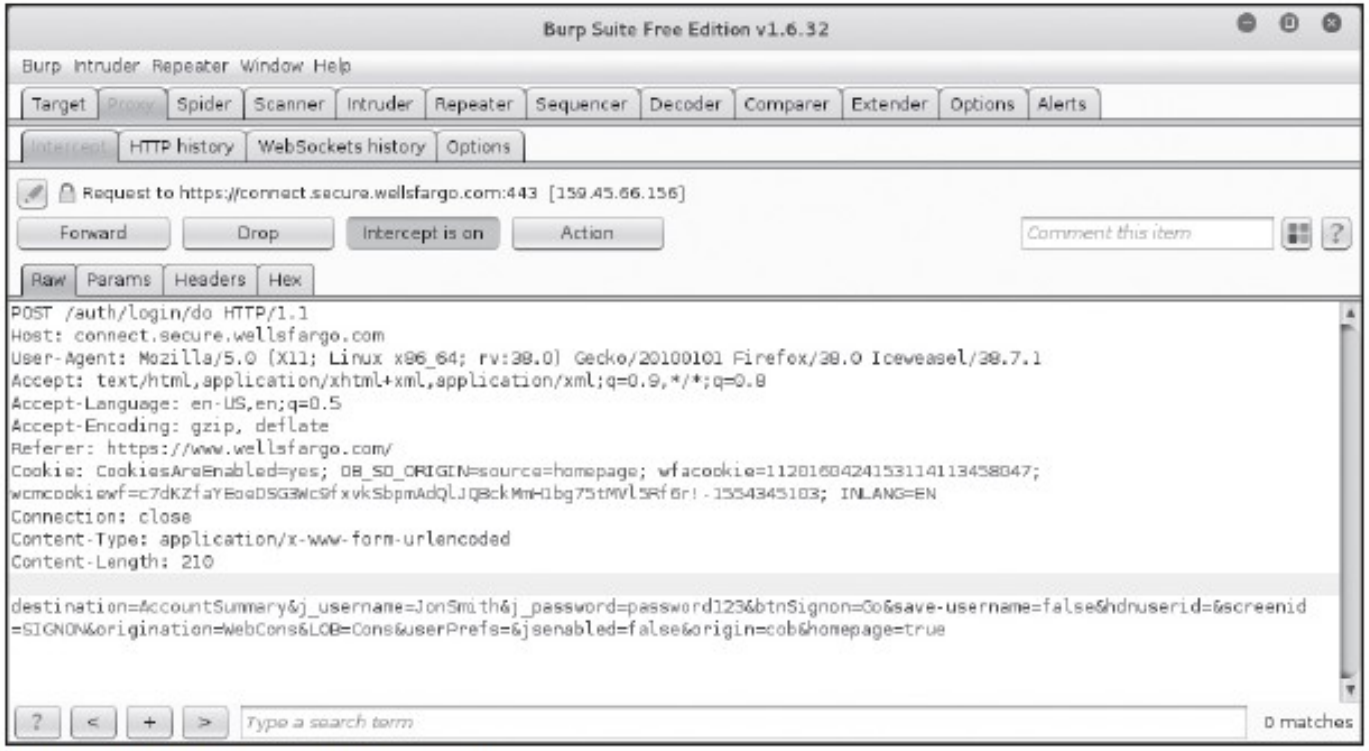

Figure 10-14 Burp Suite intercepting proxy

Wapiti

- Wapiti: Web application vulnerability scanner
  - Uses a black box approach
    - Doesn’t inspect code
  - Inspects by searching from outside
    - Ways to take advantage of XSS, SQL, PHP, JSP, and file-handling vulnerabilities
  - Uses “fuzzing”
    - Trying to inject data into whatever will accept it

Wfetch

- Wfetch: GUI tool that queries status of Web server
  - Attempts authentication using:
    - Multiple HTTP methods
    - Configuration of hostname and TCP port
    - HTTP 1.0 and HTTP 1.1 support
    - Anonymous, Basic, NTLM, Kerberos, Digest, and Negotiate authentication types
    - Multiple connection types
    - Proxy support and client-certificate support
    - Capability to enter requests manually or read from file
    - Onscreen and file-based logging

Wfetch

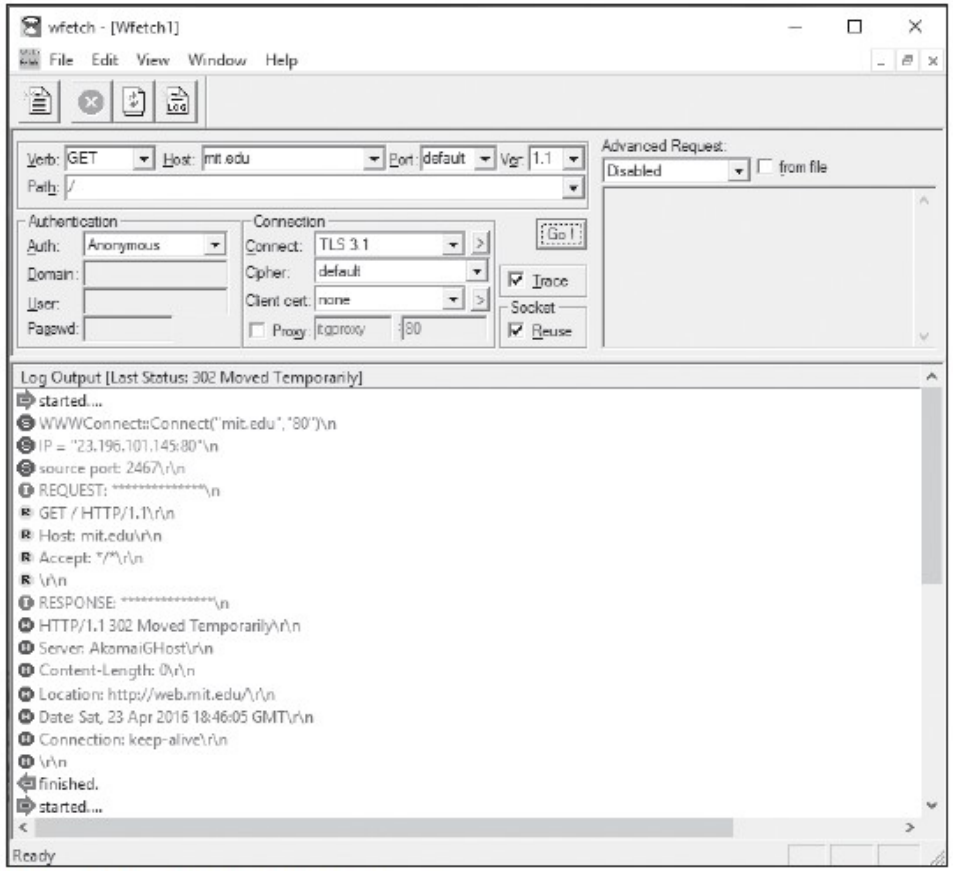

Figure 10-15 Using the Wfetch program

Source: © 2016 Microsoft

Summary

- Web applications
  - Can be developed on many platforms
    - HTML pages can contain forms, ASP, CGI, and scripting languages
- Static Web pages
  - Have been replaced by dynamic Web pages
    - Dynamic Web pages are created using CGI, ASP, etc.
- Web forms
  - Allow developers to create Web pages with which visitors can interact

Summary

- Web applications
  - Use a variety of technologies to connect to databases (e.g., ODBC, OLE DB, and ADO)
- You can install IIS
  - Test Web pages in Windows
- Web application vulnerabilities
  - Can have damaging consequences
- When conducting security tests on Web applications
  - Various considerations

Summary

- Web applications that interact with databases
  - Might be vulnerable to SQL injection exploits
- Many tools for testing Web application vulnerabilities are available
  - Burp Suite
  - Wapiti
  - OWASP open-source software