Web Application Security

Kenneth Ingham and Anil Somayaji

September 29, 2009

1 Course overview

Web applications are essential to everything from embedded systems to e-commerce systems. This class looks at the problems unique to the web and shows how attackers target these systems, how easy the vulnerabilities are to exploit, and how to solve these problems. Students will also learn upcoming vulnerabilities in areas such as SOAP and XML use.

Most of the OWASP Top Ten are covered in this course, as well as other security issues. The OWASP Top Ten not covered are covered in the complementary courses (e.g., buffer overflows are in the C/C++ class). This class complements the design and implementation courses, and should not be considered a replacement for either.

This class is language-neutral.

2 Course objectives

- Understand the differences between the web and traditional applications.
- Learn how attackers map a target looking for possible vulnerabilities.
- Recognize that the user controls the client, and how this affects input validation and state.
- Understand cross-site scripting and how to prevent it.
- Learn about directory traversal attacks and how to prevent them.
- Discuss authentication and access controls, what to do, what not to do, and how to do it right.
- Learn more about resource utilization and preventing denial of service attacks.
- Learn about XML, SOAP, Web Services, and possible problems for applications that use them.

3 Student background

If you are attending this class, then we assume that

- You are a programmer or designer of web applications (i.e., you understand programming languages).
- You want to know how to make your application more secure.
- Or, you are a QA or security testing person and want to know where to look for possible problems.

The class does not assume any particular language background, and examples in the class are drawn from many languages, including perl and Java. Not required, but useful:

- an understanding of HTML.
- a basic knowledge of SQL.
- a basic knowledge of working at the Unix or Windows command-line.

4 Logistics

The class lasts three days. The class computers can run either Windows or Linux. Note that Windows requires Cygwin from http://www.cggwin.com/, and this is a lengthy download and install. The class uses the following software:

- Internet access
- A web browser
- Data::Dumper perl module
- Firefox (on Linux distribution but needed for Windows)
- Firefox Web Developer toolbar
- Internet access
- Java JDK 1.6
- Java JDK 1.6 (for WebGoat)
- Nikto (on class web site)
- Paros proxy
- PortSwigger burp suite
- SOAP::Lite perl module
- SOAP::Transport::TCP perl module
- WebGoat v5.2
- WebScarab
- XML::Checker::Parser perl module
- XML::Parser perl module
- *netcat* (get this for Windows as part of Cygwin from cygwin.com; On Linux it is on the distribution media)
- *nmap* (Windows needs Cygwin from Cygwin.com)
- a class web server that can run perl CGI programs

- perl 5.8 (On Linux distribution but needed for Windows)

No class network information specified.

The class needs a web server for the class web site. The instructor's laptop may be this web server; otherwise the machine provided in the classroom for the instructor is a good choice. This machine obviously will need web server software installed.

5 Class outline

1. Introduction (Lecture: 15; Lab: 0)
   (a) Class Introductions
   (b) Class Logistics
      i. Class schedule
      ii. Breaks
      iii. Question policy
      iv. Break room and restroom locations
      v. Assumptions about your background
   (c) Typographic conventions
2. Web application security (Lecture: 30; Lab: 10)
   (a) Introduction
   (b) The Complex Foundation
   (c) Immature Development Model
   (d) Developing on Internet Time
   (e) Ubiquitous Networking
   (f) Hope
   (g) Class software
   (h) Lab
3. How HTTP works (Lecture: 20; Lab: 25)
   (a) Introduction
   (b) Common Gateway Interface (CGI) Parameters
   (c) GET and POST
   (d) Cookies
   (e) Lab
4. Cryptography in Web Applications (Lecture: 60; Lab: 10)
   (a) Introduction
   (b) What is Cryptography?
   (c) Cryptographic Primitives
      i. Cryptographic hash functions
      ii. Symmetric key encryption
      iii. Public key encryption

   (d) Digital signatures
   (e) Certificates
   (f) SSL/TLS Overview
   (g) SSL/TLS authentication
   (h) Public Key Management
      i. The Problem
      ii. Trust Models
   (i) Limitations of SSL/TLS
   (j) WS-Security
   (k) Summary
   (l) Lab

5. Attacking Web Applications (Lecture: 50; Lab: 45)
   (a) Introduction
   (b) PortSwigger's Burp Suite
   (c) WebScarab
   (d) Paros Suite
   (e) Nikto
   (f) Firefox Web Developer toolbar
   (g) WebGoat
   (h) Summary
   (i) Lab

6. The user controls the client: input validation (Lecture: 45; Lab: 45)
   (a) Introduction
      i. Example input validation problems
   (b) Hidden forms are not hidden
      i. Example: Smartwin Technology CyberOffice Shopping Cart
   (c) Never trust other programmers
   (d) Alternate encodings to evade input validation
      i. Example: IIS and Nimda
   (e) Solution: Whitelists
   (f) Solution: Canonicalization
      i. Example
   (g) Solution: Taint tracking
   (h) Summary
   (i) Lab

7. State and the web (Lecture: 25; Lab: 65)
   (a) Overview
   (b) Ways of tracking state
      i. Hidden fields in forms

        ii. Cookies
        iii. CGI parameters
        iv. HTTP Referer field
    (c) Session hijacking
    (d) Solutions
    (e) Example vulnerable code
    (f) Testing for this problem
    (g) Summary
    (h) Lab

8. Cross-site scripting (XSS) (Lecture: 40; Lab: 30)
    (a) Overview
    (b) A simple example
    (c) Example XSS attacks
        i. DoS the user's browser
            A. Session hijacking
        ii. DoS a web server
        iii. Make a web site contents not what the owner expects
        iv. Port scanning
        v. Worms and viruses
    (d) Locations to place script references
    (e) Ways attackers try to obscure XSS
    (f) Types of XSS attacks
    (g) XSS is not just for HTML
    (h) XSS solutions
    (i) Summary
    (j) Lab

9. Fail securely (Lecture: 25; Lab: 15)
    (a) Introduction
    (b) Failure-related code is often poorly written and tested
        i. Examples
    (c) Proper failure state
    (d) Complete error/exception handling
        i. Example: Linux ELF binary loader
    (e) Resource issues
    (f) Failures take unexpected paths in the code
    (g) Backwards Compatibility
        i. Example: Windows file sharing
    (h) Reporting Errors Securely
    (i) Failing Functionally yet Securely
    (j) Summary

(k) Lab

10. XML Security (Lecture: 25; Lab: 45)
    (a) Introduction
        i. Example
    (b) XML syntax
        i. XML prolog
        ii. XML tags
    (c) Schemas and DTDs
        i. Name spaces
        ii. Schemas
        iii. Schema security issues
    (d) Well-formed and valid XML
    (e) XML security problems
        i. Improper Type Validation
        ii. Improper Sematic Validation
    (f) XML Injection
    (g) XPath Injection
    (h) XML Firewalls/Security Gateways
    (i) Summary
    (j) Lab

11. AJAX Security (Lecture: 20; Lab: 30)
    (a) Introduction
    (b) AJAX History
    (c) The Asynchronous Problem
    (d) Client Side Code
    (e) Input Validation
    (f) JSON
    (g) DOM-based XSS attacks
    (h) AJAX Frameworks
    (i) Securing AJAX Applications
    (j) Lab

12. Cross-site request forgery (CSRF) (Lecture: 40; Lab: 45)
    (a) Introduction
        i. Example: CSRF in Gmail
    (b) Session State and CSRF
    (c) AJAX and CSRF
        i. CSRF solutions
    (d) Testing for CSRF
    (e) Summary

   (f) Lab

13. Mashups (Lecture: 40; Lab: 45)
    (a) Introduction
    (b) Understanding Same Origin Policy
    (c) Circumventing Same Origin Policy
    (d) Mashup attacks
    (e) Securing Mashups
    (f) User Scripts
    (g) Summary
    (h) Lab
14. Other Injection attacks (Lecture: 15; Lab: 35)
    (a) Introduction
    (b) SQL injection
        i. Overview
        ii. Solutions
    (c) Shell code injection
        i. Overview
        ii. Solutions
    (d) Summary
    (e) Lab
15. Web services security overview (Lecture: 20; Lab: 15)
    (a) Introduction
    (b) Web services issues
        i. Example
    (c) Resources vs. Activities
    (d) Representation
    (e) Representational State Transfer (REST)
    (f) Other APIs
    (g) Summary
    (h) Lab
16. SOAP Security Issues (Lecture: 25; Lab: 40)
    (a) Introduction
        i. SOAP history
        ii. General SOAP security issues
    (b) SOAP message structure
        i. SOAP header
        ii. SOAP body
    (c) SOAP attacks and common vulnerabilities

    (d) Testing SOAP with soapUI
    (e) Summary
    (f) Lab

17. Web Services (Lecture: 40; Lab: 45)
    (a) Introduction
    (b) Web Services security
    (c) Security support in Web Services
        i. "We use WS-*"
    (d) How Web Services work
    (e) Web Services Description Language (WSDL)
        i. Example
    (f) Universal Description, Discovery and Integration (UDDI)
    (g) Attacks
        i. Attack (and testing) tools
    (h) Summary
    (i) Lab

Appendices

A. Mapping the target web site and server (Lecture: 40; Lab: 45)
    (a) Introduction
    (b) Server error messages
    (c) Guessable secrets
    (d) Tools
        i. *nmap*
        ii. Portswigger burp spider
        iii. WebScarab
    (e) Vulnerability databases
    (f) Summary
    (g) Lab