Avoiding the CWE/SANS Top 25 Most Dangerous Programming Errors

Kenneth Ingham

September 29, 2009

1 Course overview

The CWE/SANS Top 25 Most Dangerous Programming Errors list are the most dangerous errors that programmers and system designers regularly make. The OWASP Top 10 is a list of the top 10 security-related errors that web application programmers regularly make. Companies producing code that must meet SOX, HIPAA, PCI DSS, and/or other security regulations or laws need programmers trained in avoiding these errors. Companies producing code that they plan to sell will soon be meeting customers demanding that they certify that the code is free from these errors. In order to meet these demands, programmers must understand the errors, how to avoid, and how to test for them. GIAC offers certification for programmers who pass a knowledge test on secure coding concepts; this class can be an important aid in a student being ready to take the test.

The class has examples, specific information, and labs written for C/C++, Java, and C#. Every chapter also has web and/or print references for the student to follow to obtain more information.

2 Course objectives

Students attending this class will learn:

- Something.

3 Student background

If you are attending this class, then we assume that Students attending this class need to have programming experience in C/C++, Java, or C# on GNU/Linux or Microsoft systems. The hands-on labs ask students to write, evaluate, and/or test code.

4 Logistics

The class lasts four days. The student computers need Linux or Windows with a full development environment for the language being used (C/C++/Java/C#). The instructor machine needs to run PostgreSQL with a database already loaded before class begins. A Linux shell script for creating and populating this database is in the class files. The class uses the following software:

- (C++ classes) libpq (client-side C libraries for PostgreSQL on student machines)
- (C++ classes) libpqxx installed on student machines (I used 2.6.9 for development and testing)
- (C++ classes) the GNU Common C++ library
- (Java classes) Eclipse IDE
- (Java classes) Eclipse standard widget toolkit (SWT) for Java (I used version 3.5) from http://www.eclipse.org/swt/
- (Java classes) postgresql-8.3-605.jdbc4.jar (version must match database version) installed on student machines
- Firefox
- Firefox Tamper Data add-on
- Firefox Web Developer toolbar
- Internet access
- Internet access (**not optional for C++ classes**)
- Internet access (not optional)
- Java JDK 1.6 (for WebGoat)
- Missing from Cotheroverflow.tex
- Missing from Cstackoverflow.tex
- Missing from authentication-prog.tex
- Missing from codingerrs.tex
- Missing from interp-overflow.tex
- Missing from leastpriv.tex
- Missing from resourceaccess.tex
- Missing from resourcemgt.tex
- Missing from secsofteng.tex
- PortSwigger burp suite
- PostgreSQL 8.x on the instructor's machine
- WebGoat v5.2
- *telnet* and/or *netcat*
- *wireshark*
- telnet client and server on instructor machine

The student and instructor machines need to be on a network, and the instructor machine will need a web server. The machines should be safe for connection to an internal network.

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
    - (d) What the class covers
2. Secure Software Engineering (Lecture: 45; Lab: 45)
    - (a) Why is Security Important?
    - (b) What is a secure program?
    - (c) Common security myths
        - i. "We have a firewall"
        - ii. "I use anti-virus software"
            - A. Example
    - (d) What you as a software engineer can do to improve security
    - (e) The 2009 CWE/SANS Top 25 Most Dangerous Programming Errors list
    - (f) The OWASP Top 10 web application security errors
    - (g) Threat models
        - i. Example
        - ii. Risk analysis
    - (h) Summary
    - (i) Lab
3. Security and the software development life cycle (Lecture: 35; Lab: 20)
    - (a) Introduction
    - (b) Requirements
        - i. Example
        - ii. Example
        - iii. Use, Abuse, and Misuse cases
    - (c) Design/Architecture
        - i. Design is critical
        - ii. Properly-written specifications
    - (d) Code development

        - i. Implementation is critical
    - (e) Testing
    - (f) Operations/maintenance
    - (g) Agile development
    - (h) Penetrate and patch is the wrong approach
    - (i) Summary
    - (j) Lab

4. Input validation (Lecture: 45; Lab: 60)
    - (a) Introduction
    - (b) Beware hidden user input
    - (c) Example attacks
        - i. Example: 3D3.Com ShopFactory
    - (d) Examples of bad code
        - i. Java example
        - ii. Java example
        - iii. C/C++ example
        - iv. C/C++ example
    - (e) Techniques to avoid this problem
        - i. Input validation frameworks
        - ii. Whitelists
        - iii. Canonicalization
            - A. Example: NTFS
            - B. Example: IIS and Nimda
        - iv. Taint tracking
    - (f) Testing for input validation problems
    - (g) Summary
    - (h) Lab

5. Avoiding SQL injection (Lecture: 30; Lab: 45)
    - (a) Introduction
        - i. Consequences of this weakness
    - (b) Example attacks
    - (c) Example vulnerable code
        - i. Java
        - ii. C#
        - iii. C++
    - (d) Techniques to avoid this problem
        - i. Parameterized queries and pre-stored procedures
            - A. Java
            - B. C++

- (e) Testing for this problem
    - i. Testing tools
- (f) Catching these attacks in production
- (g) Summary
- (h) Lab

6. Avoiding OS command injection (Lecture: 30; Lab: 45)
    - (a) Introduction
        - i. Consequences of this weakness
    - (b) Example vulnerable code
        - i. Java
        - ii. Java example 2
        - iii. C++
    - (c) Techniques to avoid this problem
        - i. Example good code
        - ii. Gotchas
    - (d) Testing for this problem
    - (e) Summary
    - (f) Lab
7. Producing clean output (Lecture: 30; Lab: 45)
    - (a) Introduction
    - (b) Example attacks
        - i. Cross-site scripting attack
        - ii. Log-writing attack
        - iii. Attacking penetration test software
        - iv. Attacking a back-end IRC chat server
    - (c) Techniques to avoid this problem
    - (d) Testing for this problem
    - (e) Summary
    - (f) Lab
8. Cross-site scripting (Lecture: 40; Lab: 55)
    - (a) Introduction
    - (b) Example attacks
        - i. A simple example
        - ii. DoS the user's browser
        - iii. Session hijacking
        - iv. DoS a web server
        - v. Make a web site contents not what the owner expects
        - vi. Port scanning
        - vii. Worms and viruses

   (c) Locations to place script references
   (d) Ways attackers try to obscure XSS
   (e) Types of XSS attacks
   (f) XSS is not just for HTML
   (g) Techniques to avoid this problem
   (h) Testing for this problem
      i. XSS testing tools
   (i) Summary
   (j) Lab

9. 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

10. Logging and error messages (Lecture: 20; Lab: 35)
   (a) Introduction
   (b) Examples
   (c) Example bad code
   (d) Techniques to avoid this problem
   (e) Testing for this problem
   (f) Summary
   (g) Lab

11. Cryptography Fundamentals (Lecture: 30; Lab: 25)
   (a) Introduction
      i. Cryptographic Applications
      ii. Open design
   (b) Limits of Cryptography
   (c) Cryptographic Primitives
      i. Cryptographic Hash Functions
      ii. Symmetric key encryption
      iii. Public key encryption
   (d) Digital signatures
   (e) Random Numbers
   (f) Parameter sizes
   (g) Insecure Cryptography

    (h) Do Not Innovate in Cryptography!
    (i) Summary
    (j) Lab

12. Using cryptography to enhance security (Lecture: 40; Lab: 60)
    (a) Introduction
        i. SSL versus TLS
    (b) Consequences of cryptographic problems
    (c) Example problems
        i. Developing custom cryptography
        ii. Key management problems
        iii. Poor random number quality
        iv. Leaking sensitive information
        v. Improper use of public key cryptography
        vi. Weak cryptographic algorithms
        vii. Implementation issues
    (d) Long example: Web Services
        i. "We use WS-*"
    (e) Example problems and cryptographic solutions
    (f) Other hints
    (g) Testing
    (h) Summary
    (i) Lab

13. Authentication (Lecture: 40; Lab: 60)
    (a) Introduction
        i. Biometrics
    (b) Bad habits
    (c) Example failures
    (d) Example vulnerable code
        i. Java client
        ii. Java server
        iii. C++ client
        iv. C++ server
    (e) Techniques to avoid this problem
        i. Rate limiting
        ii. Best Practices from OWASP
        iii. Storing encryption keys
    (f) Testing for these problems
    (g) Summary
    (h) Lab

14. Least privilege (Lecture: 45; Lab: 20)
    (a) Introduction
    (b) Example failures
    (c) Techniques to avoid this problem
        i. Dropping privileges after obtaining a resource
        ii. C/C++ Example on GNU/Linux
        iii. Java example
    (d) Separation of privilege
        i. Implementing separation of privilege
        ii. Example: OpenSSH
            A. Applying separation of privilege to OpenSSH
        iii. Example: NTP client
    (e) Testing for this problem
    (f) Summary
    (g) Lab
15. Authorization and Access Control (Lecture: 45; Lab: 75)
    (a) Introduction
        i. Access control vocabulary
    (b) Access control models
        i. Discretionary access control (DAC)
        ii. Access control lists (ACLs)
        iii. Mandatory access control (MAC)
        iv. Role-based access control (RBAC)
    (c) Example failures
    (d) Techniques to avoid this problem
    (e) Testing for this problem
    (f) Summary
    (g) Lab
16. 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

17. Stack overflows for C/C++ (Lecture: 60; Lab: 90)
   (a) Introduction
      i. Memory layout
      ii. Consequences of this weakness
   (b) Example vulnerable code and attacks
      i. Stack
   (c) Techniques to avoid this problem
   (d) Testing for this problem
   (e) Summary
   (f) Lab

18. Other buffer overflows for C/C++ (Lecture: 60; Lab: 90)
   (a) Introduction
   (b) Example vulnerable code and attacks
      i. Heap
      ii. Other data segments
   (c) Techniques to avoid this problem
   (d) Testing for this problem
   (e) Summary
   (f) Lab

19. Buffer overflows and interpreted languages (Lecture: 20; Lab: 20)
   (a) Introduction
   (b) Example attacks
   (c) Example vulnerable code
   (d) Techniques to avoid this problem
   (e) Testing for this problem
   (f) Summary
   (g) Lab

20. Race conditions (Lecture: 25; Lab: 10)
   (a) Introduction
   (b) TOCTTOU race conditions
      i. passwd command races
      ii. Temporary Files
      iii. Avoiding TOCTTOU problems
   (c) Memory corruption Race Conditions
      i. Multithreaded Processes
      ii. Signal race conditions
      iii. OS Kernel race conditions

    (d) Summary
    (e) Race Conditions Lab

21. Resource access (Lecture: 30; Lab: 45)

    (a) Introduction
    (b) Example attacks
    (c) Example vulnerable code
    (d) Techniques to avoid this problem
    (e) Testing for this problem
    (f) Summary
    (g) Lab

22. Resource management (Lecture: 30; Lab: 45)

    (a) Introduction
    (b) Example attacks
    (c) Example vulnerable code
    (d) Techniques to avoid this problem
    (e) Testing for this problem
    (f) Summary
    (g) Lab

23. Coding errors (Lecture: 30; Lab: 45)

    (a) Introduction
    (b) Example attacks
    (c) Example vulnerable code
    (d) Techniques to avoid this problem
    (e) Testing for this problem Summary
    (f) Lab