Testing for Security

Kenneth Ingham

September 29, 2009

1 Course overview

The threat that security breaches present to your products and ultimately your customer base can be significant. This course is designed to assist testers in updating their testing practices to include testing for security. The goal of this effort is to reduce the number of identified post-release security vulnerabilities.

Many tools exist to assist testers. However, more important is to understand the testing techniques. This class uses the tools to teach the techniques. While the students will learn about some of the available tools, they are not the primary focus of the class.

2 Course objectives

- Understand how the risk analysis drives security testing.
- Learn the concepts of penetration, black box, and white box testing, and understand the strengths and weaknesses of each technique. Know when to apply each.
- Understand the types of attacks (attack patterns) and tools that hackers use against your products in order to develop or utilize testing practices to duplicate the more common forms of these attacks. Thinking like an attacker is important for this understanding, so the class shows the approaches attackers often take.
- Understand what tools are available and appropriate to use in different security testing situations.

3 Student background

If you are attending this class, then we assume that

- You have already had a class on or experience with threat models and risk analysis.

- You understand programming using a major programming language (C, C++, Java, etc).
- You are (or will be) testing products or applications for possible security breaches.

4 Logistics

The class lasts two days. nothing; no class computers are needed The class uses the following software:

- Firefox (on Linux distribution but needed for Windows)
- Firefox Web Developer toolbar
- Java JDK 1.6
- Missing from auth2testing.tex
- Missing from dynamictesting.tex
- Missing from errortest.tex
- Missing from featureinteractions.tex
- Missing from fuzztesting.tex
- Missing from injectiontesting.tex
- Missing from insecurecomm.tex
- Missing from outputtesting.tex
- Missing from resourcetesting.tex
- Missing from staticanalysis.tex
- Nikto (on class web site)
- Paros proxy
- PortSwigger burp suite
- WebGoat v5.2
- WebScarab
- *emacs* (on Linux distribution)
- *firefox*
- *g++* (on the Linux distribution)
- *gcc* (on Linux distribution)
- *gdb* (on Linux distribution)
- *opera*
- *strace* (on the Linux distribution)
- a class web server that can run perl CGI programs
- at least one of *ddd* or the GNU Visual Debugger (on Linux distribution)
- perl 5.8 (On Linux distribution but needed for Windows)
- the Firefox web developer toolbar

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
   (d) What the class covers
2. Security Testing Introduction (Lecture: 25; Lab: 0)
   (a) Introduction
   (b) What is a secure program?
   (c) The Cost of a Serious Security Problem
   (d) Why do security testing?
   (e) Who should perform the testing
   (f) Types of security testing
   (g) Limitations of Testing
   (h) When to test
   (i) Staying current
   (j) System Requirements and Security Testing
   (k) Notes about the class
   (l) Summary
3. Risk-based Testing (Lecture: 35; Lab: 90)
   (a) Introduction
   (b) The threat model
   (c) The assets you are protecting
   (d) Attackers
   (e) Common attack goals
   (f) Example
   (g) Failures of Imagination
      i. Clients, servers, and embedded systems
         A. Embedded system
   (h) Risk analysis
   (i) Using risk analysis to drive testing
      i. Prioritizing Abuse Scenarios
      ii. Focus Security Testing on Targeted Areas
   (j) Summary

   (k) Lab

4. Input Validation Vulnerabilities (Lecture: 30; Lab: 30)
   (a) Introduction
   (b) Beware hidden user input
   (c) Some failures of input validation
      i. Buffer Overflows
         A. A simple buffer overflow example
         B. Finding buffer overflows
      ii. Integer Range Errors
         A. Value truncation
         B. Example
         C. Integer overflow and underflow
         D. Finding integer range errors
      iii. Format string vulnerabilities
      iv. Repeated Input
   (d) Finding input locations
      i. Example
   (e) Tools
      i. Firefox Web Developer toolbar
   (f) Summary
   (g) Lab

5. Fuzz testing (fuzzing) (Lecture: 35; Lab: 45)
   (a) Introduction
   (b) Types of fuzz testing
   (c) Tools
      i. The Peach Fuzzer
   (d) A quick Python tutorial
      i. Running Python programs
      ii. Variables and data types
      iii. I/O
      iv. Control flow
      v. Data structures and classes
      vi. Errors and exceptions
   (e) Summary
   (f) Lab

6. Injection vulnerabilities (Lecture: 30; Lab: 30)
   (a) Introduction
   (b) SQL injection

   (c) Shell injection
   (d) Finding these vulnerabilities
   (e) Tools
      i. SQL injection
   (f) Summary
   (g) Lab

7. Static code analysis (Lecture: 30; Lab: 30)
   (a) Introduction
   (b) Test types
      i. Type checking
      ii. Style checking
      iii. Property checking and program verification
      iv. Bug finders
   (c) Tools
      i. Commercial
      ii. Open Source
   (d) Summary
   (e) Lab

8. Testing resource management (Lecture: 10; Lab: 30)
   (a) Introduction
   (b) Graceful degradation
   (c) Testing for this problem
   (d) Summary
   (e) Lab

9. Dynamic analysis (Lecture: 30; Lab: 45)
   (a) Introduction
   (b) Tools
   (c) *valgrind*
      i. Example bug finding
      ii. Finding memory leaks
         A. Command-line flags specific to memcheck
      iii. Finding illegal free() calls
      iv. Other uses of memcheck
   (d) Summary
   (e) Lab

10. Complete and correct error handling (Lecture: 15; Lab: 40)
   (a) Introduction
   (b) Examples

   (c) Proper failure state
   (d) Testing for these problems
      i. Fault injection
   (e) Summary
   (f) Lab

11. Output validation (Lecture: 10; Lab: 30)
   (a) Introduction
   (b) Examples
   (c) Testing for this problem
   (d) Summary
   (e) Lab

12. Feature interactions (Lecture: 10; Lab: 30)
   (a) Introduction
   (b) Finding feature interaction vulnerabilities
   (c) Summary
   (d) Lab

13. Data Security Testing (Lecture: 20; Lab: 40)
   (a) Introduction
   (b) Least Privilege
   (c) Separation of privilege
      i. Example: NTP client
      ii. Compartmentalization
   (d) Erasing data
   (e) Erasing Memory
   (f) Testing
   (g) Summary
   (h) Lab

14. Insecure Communication (Lecture: 30; Lab: 30)
   (a) Introduction
   (b) Using cryptography to protect communication
      i. Using public key cryptography to protect communication
   (c) Testing for this problem
   (d) Summary
   (e) Lab

15. Authentication and Authorization Errors (Lecture: 25; Lab: 45)
   (a) Introduction
   (b) Example failures
   (c) Authentication

- (d) Access control
    - i. Access control vocabulary
    - ii. Access Control Matrix
- (e) Testing for these problems
    - i. Bad habits to check for
- (f) Summary
- (g) Lab

Appendices

A. Debugging with gdb (Lecture: 25; Lab: 30)

- (a) Introduction
- (b) Compiling programs to be debugged
- (c) Working in *gdb*
    - i. Starting and exiting *gdb*
    - ii. *gdb* commands
    - iii. Help
    - iv. Info
    - v. Show
    - vi. Running programs
- (d) Breakpoints and watchpoints
- (e) Continuing and single stepping
- (f) Viewing data and the stack
- (g) Demonstration
- (h) GUI front ends for *gdb*
- (i) Lab

B. More debugging with gdb (Lecture: 20; Lab: 50)

- (a) Dealing with core dumps
- (b) Debugging when the program was not compiled for debugging
- (c) Attaching to already-running programs
- (d) **fork** and processes
- (e) Debugging threads
- (f) Signals
- (g) Lab

C. 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