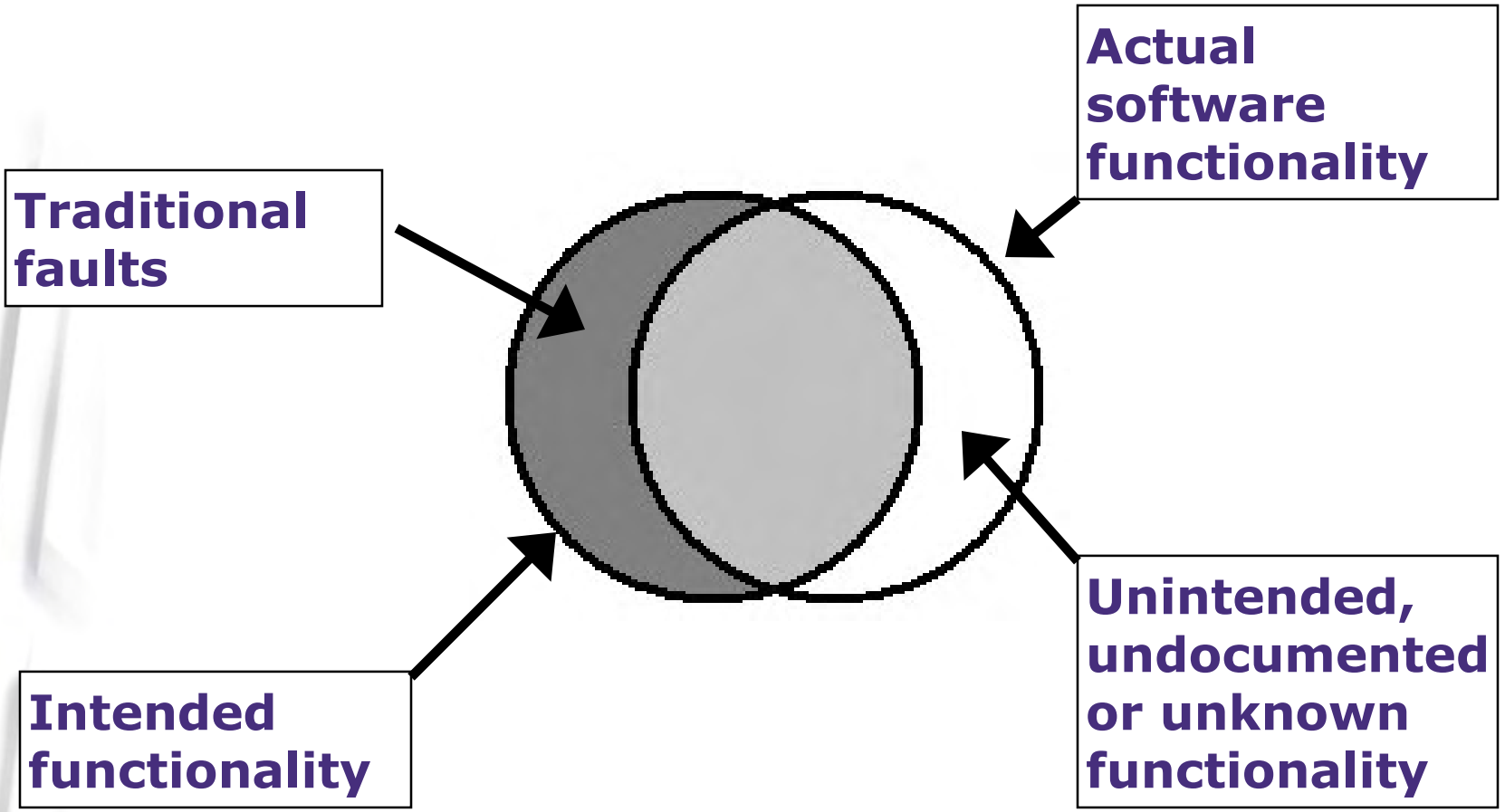

# W05 The QA/Testing perspective on Software Security
# StarEast 2005

Julian Harty
Commercetest Limited

# Contents

- Where does security testing fit in the testing domain?
  - What's different about security testing?
  - The comparison with other areas of testing
- Security testing through the software lifecycle
- Methods of security testing
- Security testing techniques
- Skill sets and experience
- Levels of expertise

# What's different about security testing?



# Comparing Security with functional testing

- Security testing challenges the functional requirements e.g. to:
  - See if the system can be fooled into accepting fake / false inputs
  - Find out if unauthorised activities are possible
  - Determine authorised activities can be delayed or interrupted

# Comparing Security with Robustness testing


- Robustness testing assesses whether software can cope with unexpected, hostile, or unfavorable inputs and conditions. And includes:
  - Recovery
  - Availability
  - Integrity

- Security testing involves:
  - Confidentiality
  - Integrity
  - Availability

# Comparing Security with Usability testing

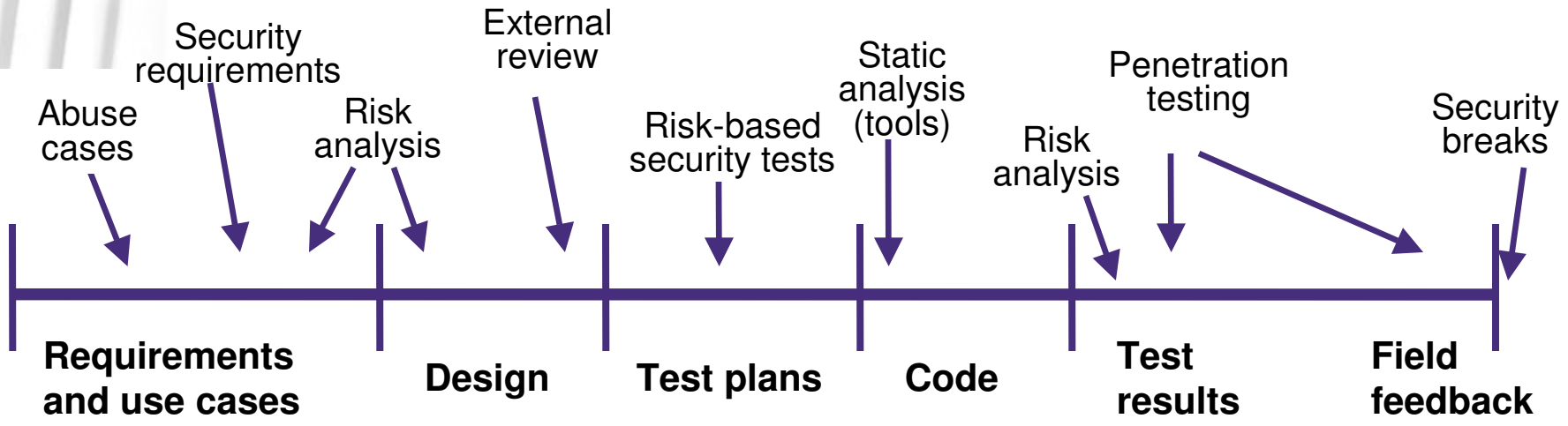

- Usability and Security sometimes conflict e.g.
  - Users resent having to login, change passwords, etc.
- Thoughtful design can minimize the conflict between usability and security
- The testing overlaps when testing the usability of security mechanisms
- *Is an unusable system secure?*

# Comparing Security with Performance testing


- Performance testing includes Load, Volume and Stress testing
- All of these types of testing can be used for denial-of-service testing
- Both Security and Performance testing rely on the testers having technical skills
- Many of the analysis techniques are common e.g. to reverse-engineer Web requests in order to automate the tests
- Performance testing tools can help test for security vulnerabilities e.g. for SQL injection, password guessing, etc.

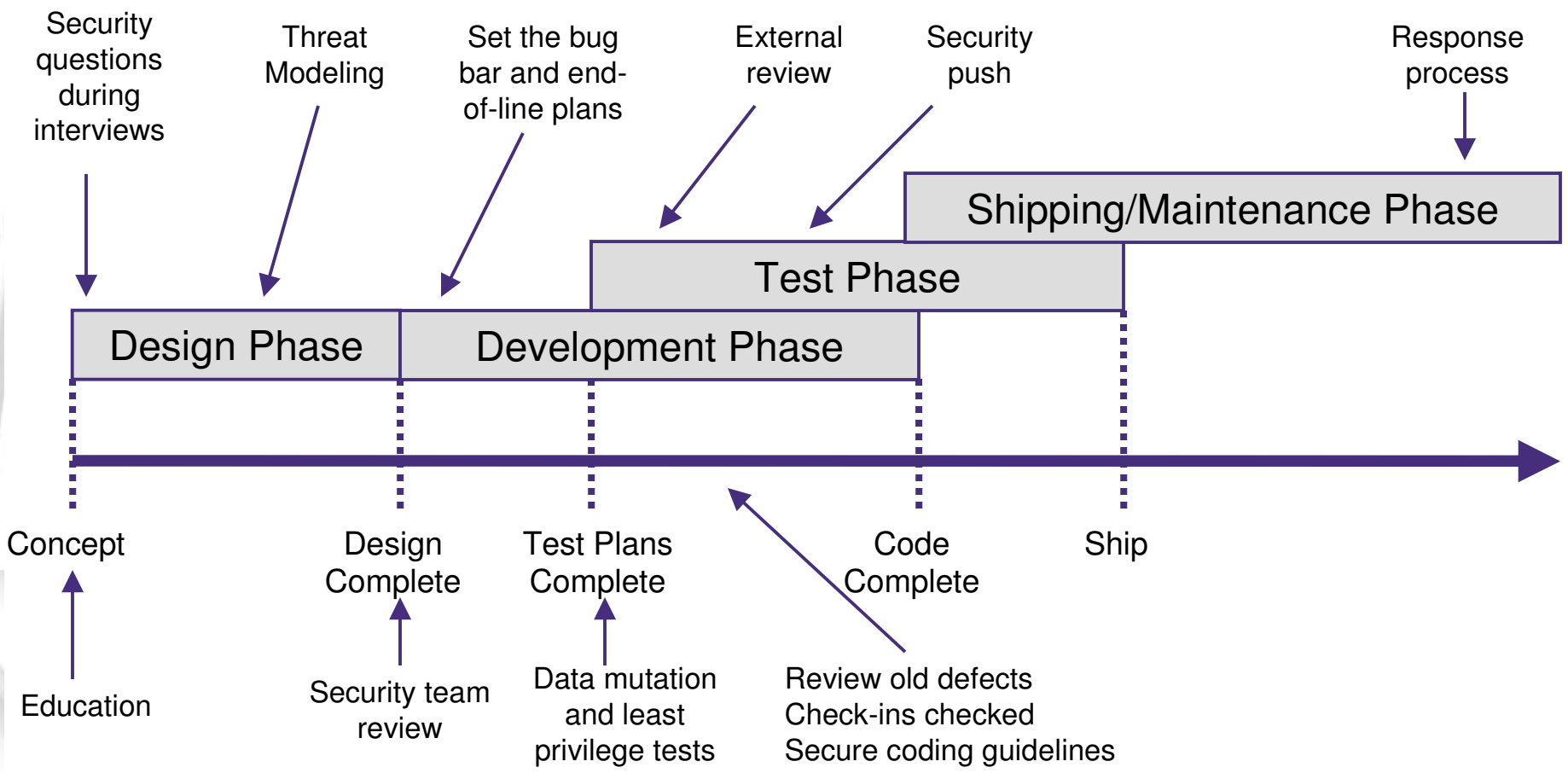
# Security testing as a full-lifecycle activity

- Penetration testing, while popular, is only practical once the software has been written and installed
- Security testing should start from the beginning of the software development lifecycle (SDLC)
- Should start before the software and end after a system has been decommissioned




*From 'Building Security In' by Brian Chess and Gary McGraw*
*Used with permission from Gary McGraw @ Cigital.com*

# Microsoft's Security Development Process






From 'Writing Secure Code' 2nd Edition, Microsoft Press

# Methods of security testing

- Manual testing
- Semi-automated e.g.
  - Reverse engineering
  - Source code analysis
  - Scripting tools such as for performance testing
  - Copy-and-paste technology e.g. for:
    - SQL injection
    - Command injection
    - Buffer overruns
- Fully-automated
  - Anomaly injection tools
  - Pre-coded, often for known vulnerabilities

# Automation support

- Commercial software such as
  - WebInspect
  - Holodeck
  - Codenomicon

  can automatically test your software to find problems
- Open-source and free tools
  - E.g. WebScarab from www.owasp.org
- Home-brew automation also has a useful part to play
  - E.g. Perl scripts to attack networked applications & web sites

# Security testing techniques


- Threat modeling
- Code reviews
- Data mutation
- ‘Functional’ testing techniques
- Security Innovation’s 19 attacks
- Automated vulnerability tests

# Threat Modeling


- Threat modeling is important, and virtually essential
- A number of useful techniques exist, e.g.:
  - STRIDE
  - Anti-Goals
  - Misuse cases
  - Abuse cases

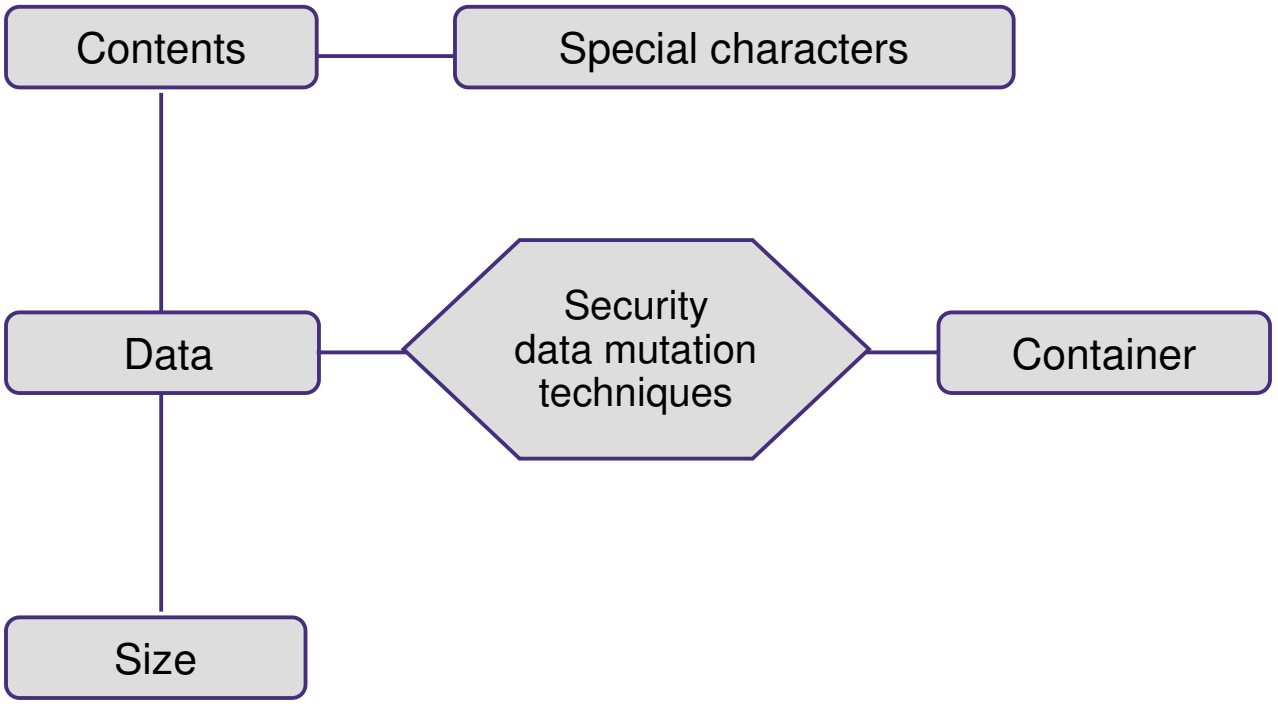
  They overlap, yet each offers a unique perspective
- Consider combining several of the techniques to catch more issues, earlier

# Code reviews

- Recommended by a number of leading experts
- Relies on the reviewer / tester having the ability to find vulnerabilities
  - Often performed by software engineers
- Automated tools exist and can help to find some of the problems
  - however they may produce lots of false positives
- See ‘Static Analysis for Security’ article in IEEE Security & Privacy for a good overview

# Data mutation

- Simplified version of Microsoft's diagram
- 25 techniques identified for attacking software




From 'Writing Secure Code' 2nd Edition, Microsoft Press

# ‘Functional’ testing techniques

- Functional testing techniques are useful for security testing e.g.
  - Syntax testing: to determine whether the system correctly rejects undesirable input such as SQL injection code
  - Boundary Value Analysis: to see whether invalid boundary values expose problems
  - Equivalence partitioning: to reduce the set of infinite test cases to a viable subset e.g. only try a few SQL attacks for each field
- *Are there any functional test design techniques that **don’t** apply?*

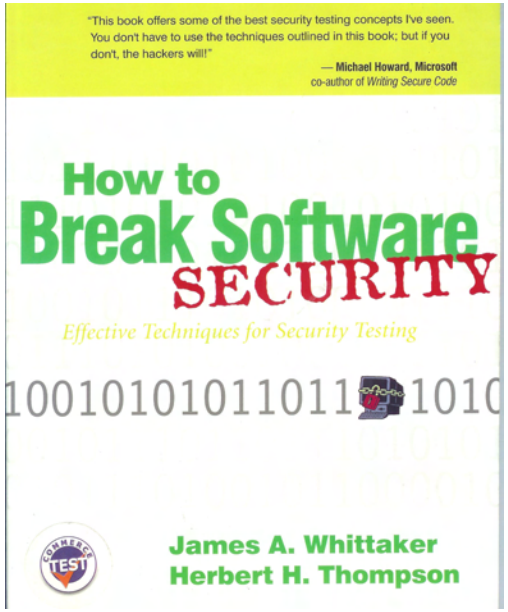
# Security Innovation's 19 attacks

Practical black-box test techniques that attack 4 areas

- Software dependencies (on external objects)
- Breaking security through the user interface
- Attacking the design
- Attacking the implementation

Described in the 'how to break software security' book

Includes a great tool to help expose problems



# Automated vulnerability tests

A number of companies offer automated software tools that test for vulnerabilities e.g.

- Codenomicon provide test tools that ensure software complies with relevant protocols such as BGP for routers, HTTP for web servers, etc
- Penetration testing tools (see the vendor booths)
- Fuzz testing, a frighteningly simple way to crash many software utilities
  - Submits random input to programs and detects whether they crash…

    http://www.cs.wisc.edu/~bart/fuzz/fuzz.html

# Skill sets and expertise


- Security testing requires a different mindset to find security bugs
  - A willingness to think like an attacker
  - Persistence, to keep going until you find a bug
- Techniques such as threat modeling don't require programming skills
- Code analysis does…
- Numerous training courses and certification schemes exist to help you to build up your skills

# Levels of expertise

- We start, unaware of security testing techniques
- As we learn about the issues we try some techniques but aren't yet competent, this includes:
  - Being a 'script kiddie' – able to use automated attack tools
  - Able to copy and paste attack code and observe the results
- As our experience increases we start being able to work with other 'craftsmen' e.g. developers to find vulnerabilities
- Able to analyse software for possible areas of concern, even if we can't pinpoint the problems

| | | |
|---|---|---|
| Conscious | Apprentice | Journeyman |
| Unconscious | Ignorant | Master |
| | Incompetence | Competence |

# Levels of expertise


- Journeymen are able to use and apply many of the techniques mentioned here to find relevant security vulnerabilities
- Masters, appear to make security testing and bug finding effortless ☺

| | | |
|---|---|---|
| Conscious | Apprentice | Journeyman |
| Unconscious | Ignorant | Master |
| | Incompetence | Competence |

# Questions & Answers


Please get in touch if you'd like more information

julian.harty@commercetest.com

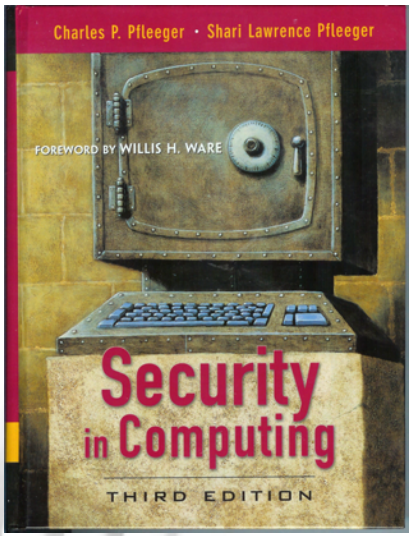
***Acknowledgements***

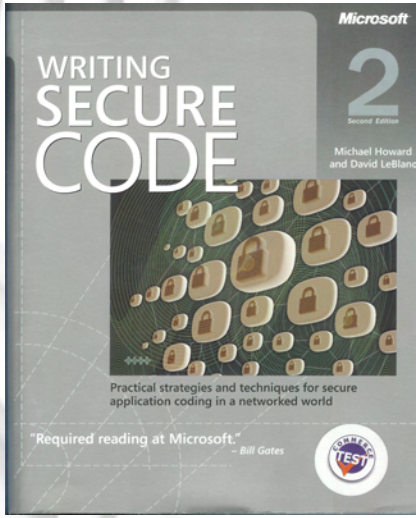
***Thanks to Hugh Thompson, James Whittaker, Michael Howard, and others who gave permission to incorporate some of their material.***

# Security References


- www.commoncriteriaportal.org - Common Criteria material
- www.owasp.org - various tools and whitepapers (Java and .NET)
  - WebScarab: helps testers to find vulnerabilities & attack web sites
  - WebGoat: sample web application with inbuilt tutorials on common attacks
- www.microsoft.com - Threats_Countermeasures.pdf
- //easyweb.easynet.co.uk/~iany – Material on misuse cases, etc.
- www.testingstandards.co.uk - software standards and examples
- IEEE Security & Privacy Journals

# Great Security books




- Security in Computing 3rd edition

  Charles P. Pfleeger, Shari Lawrence Pfleeger

  ISBN 0-13-035548-8, Prentice Hall © 2003

- How to break software security

  James A. Whittaker, Herbert H. Thompson

  ISBN 0-321-19433-0, Addison-Wesley © 2003

- Writing Secure Code 2nd edition

  Michael Howard and David LeBlanc

  ISBN 0-7356-1722-8, Microsoft Press © 2003

- See also:

  – An information security handbook

    John M. D. Hunter

    ISBN 1-85233-180-1, Springer © 2001

  – Testing Web Security

    Steve Splaine

    ISBN 0-471-23281-5, Wiley © 2002