PART

One

An Introduction to the Book

CHAPTER 1

Introduction

The following are some sobering statistics and stories that seek to illustrate the growing need to assess the security of Web sites and applications. The 2002 Computer Crime and Security Survey conducted by the Computer Security Institute (in conjunction with the San Francisco Federal Bureau of Investigation) reported the following statistics (available free of charge via www.gocsi.com):

- Ninety percent of respondents (primarily large corporations and government agencies) detected computer security breaches within the last 12 months.
- Seventy-four percent of respondents cited their Internet connection as a frequent point of attack, and 40 percent detected system penetration from the outside.
- Seventy-five percent of respondents estimated that disgruntled employees were the likely source of some of the attacks that they experienced.

The following lists the number of security-related incidents reported to the CERT Coordination Center (www.cert.org) for the previous 4 1/2 years:

- 2002 (Q1 and Q2)—43,136
- 2001—52,658
- 2000—21,756
- 1999—9,859
- 1998—3,734

In February 2002, Reuters (www.reuters.co.uk) reported that "hackers" forced CloudNine Communications—one of Britain's oldest Internet service providers (ISPs)—out of business. CloudNine came to the conclusion that the cost of recovering from the attack was too great for the company to bear, and instead elected to hand over their customers to a rival ISP.

In May 2002, CNN/Money (www.money.cnn.com) reported that the financing division of a large U.S. automobile manufacturer was warning 13,000 people to be aware of identity theft after the automaker discovered "hackers" had posed as their employees in order to gain access to consumer credit reports.

The Goals of This Book

The world of security, especially Web security, is a very complex and extensive knowledge domain to attempt to master—one where the consequences of failure can be extremely high. Practitioners can spend years studying this discipline only to realize that the more they know, the more they realize they need to know. In fact, the challenge may seem to be so daunting that many choose to shy away from the subject altogether and deny any responsibility for the security of the system they are working on. "We're not responsible for security—somebody else looks after that" is a common reason many members of the project team give for not testing a system's security. Of course, when asked who the somebody else is, all too often the reply is "I don't know," which probably means that the security testing is fragmented or, worse still, nonexistent.

A second hindrance to effective security testing is the naive belief held by many owners and senior managers that all they have to do to secure their internal network and its applications is purchase a firewall appliance and plug it into the socket that the organization uses to connect to the Internet. Although a firewall is, without doubt, an indispensable defense for a Web site, it should not be the only defense that an organization deploys to protect its Web assets. The protection afforded by the most sophisticated firewalls can

THE FIREWALL MYTH

The *firewall myth* is alive and well, as the following two *true* conversations illustrate. Anthony is a director at a European software-testing consultancy, and Kevin is the owner of a mass-marketing firm based in Florida.

Anthony: We just paid for someone to come in and install three top-of-the-line firewalls, so we're all safe now.

Security tester: Has anybody tested them to make sure they are configured correctly?

Anthony: No, why should we?

Kevin: We're installing a new wireless network for the entire company.

Security tester: Are you encrypting the data transmissions?

Kevin: I don't know; what difference does it make? No one would want to hack us, and even if they did, our firewall will protect us.

be negated by a poorly designed Web application running on the Web site, an oversight in the firewall's configuration, or a disgruntled employee working from the inside.

This book has two goals. The first goal is to raise the awareness of those managers responsible for the security of a Web site, conveying that a firewall should be *part* of the security solution, but not *the* solution. This information can assist them in identifying and planning the activities needed to test *all* of the possible avenues that an intruder could use to compromise a Web site. The second goal is aimed at the growing number of individuals who are new to the area of security testing, but are still expected to evaluate the security of a Web site. Although no book can be a substitute for years of experience, this book provides descriptions and checklists for hundreds of tests that can be adapted and used as a set of candidate test cases. These tests can be included in a Web site's security test plan(s), making the testing effort more comprehensive than it would have been otherwise. Where applicable, each section also references tools that can be used to automate many of these tasks in order to speed up the testing process.

The Approach of This Book

Testing techniques can be categorized in many different ways; white box versus black box is one of the most common categorizations. Black-box testing (also known as *behavioral testing*) treats the system being tested as a black box into which testers can't see. As a result, all the testing must be conducted via the system's external interfaces (for example, via an application's Web pages), and tests need to be designed based on what the system is expected to do and in accordance with its explicit or implied requirements. White-box testing assumes that the tester has direct access to the source code and can look into the box and see the inner workings of the system. This is why white-box testing is sometimes referred to as *clear-box*, *glass-box, translucent*, or *structural* testing. Having access to the source code helps testers to understand how the system works, enabling them to design tests that will exercise specific program execution paths. Input data can be submitted via external or internal interfaces. Test results do not need to be based solely on external outputs; they can also be deduced from examining internal data stores (such as records in an application's database or entries in an operating system's registry).

In general, neither testing approach should be considered inherently more effective at finding defects than the other, but depending upon the specific context of an individual testing project (for example, the background of the people who will be doing the testing—developer oriented versus end-user oriented), one approach could be easier or more cost-effective to implement than the other. Beizer (1995), Craig et al. (2002), Jorgensen (2002), and Kaner et al. (1999) provide additional information on black-box and white-box testing techniques.

Gray-box testing techniques can be regarded as a hybrid approach. In other words, a tester still tests the system as a black box, but the tests are designed based on the knowledge gained by using white-box-like investigative techniques. Gray-box testers using the knowledge gained from examining the system's internal structure are able to design more accurate/focused tests, which yield higher defect detection rates than those achieved using a purely traditional black-box testing approach. At the same time, 200174_CH01/SplaineX 9/20/02 9:42 AM Page 6

GRAY-BOX TESTING

Gray-box testing incorporates elements of both black-box and white-box testing. It consists of methods and tools derived from having some knowledge of the internal workings of the application and the environment with which it interacts. This extra knowledge can be applied in black-box testing to enhance testing productivity, bug finding, and bug-analyzing efficiency.
Source: **Nguyen (2000).**

however, gray-box testers are also able to execute these tests without having to use resource-consuming white-box testing infrastructures.

Wherever possible, this book attempts to adopt a gray-box approach to security testing. By covering the technologies used to build and deploy the systems that will be tested and then explaining the potential pitfalls (or vulnerabilities) of each technology design or implementation strategy, the reader will be able to create more effective tests that can still be executed in a resource-friendly black-box manner.

This book stops short of describing platform- and threat-specific test execution details, such as how to check that a Web site's Windows 2000/IIS v5.0 servers have been protected from an attack by the Nimda worm (for detailed information on this specific threat, refer to CERT advisory CA-2001-26—www.cert.org). Rather than trying to describe in detail the specifics of the thousands of different security threats that exist today (in the first half of 2002 alone, the CERT Coordination Center recorded 2,148 reported vulnerabilities), this book describes generic tests that can be extrapolated and customized by the reader to accommodate individual and unique needs. In addition, this book does not expand on how a security vulnerability could be exploited (information that is likely to be more useful to a security abuser than a security tester) and endeavors to avoid making specific recommendations on how to fix a security vulnerability, since the most appropriate remedy will vary from organization to organization and such a decision (and subsequent implementation) would generally be considered to be the role of a security designer.

How This Book Is Organized

Although most readers will probably find it easier to read the chapters in sequential order, this book has been organized in a manner that permits readers to read any of the chapters in any order. Depending on the background and objectives of different readers, some may even choose to skip some of the chapters. For example, a test manager who is well versed in writing test plans used to test the functionality of a Web application may decide to skip the chapter on test planning and focus on the chapters that describe some of the new types of tests that could be included in his or her test plans. In the case of an application developer, he or she may not be concerned with the chapter on testing a Web site's physical security because someone else looks after that (just so long as someone actually does) and may be most interested in the chapters on application security.

To make it easier for readers to hone in on the chapters that are of most interest to them, this book has been divided into four parts. Part 1 is comprised of this chapter and provides an introduction and explanation of the framework used to construct this book.

Chapter 2, "Test Planning," provides the material for Part 2, "Planning the Testing Effort," and looks at the issues surrounding the planning of the testing effort.

Part 3, "Test Design," is the focus of this book and therefore forms the bulk of its content by itemizing the various candidate tests that the testing team should consider when evaluating what they are actually going to test as part of the security-testing effort of a Web site and its associated Web application(s). Because the testing is likely to require a variety of different skill sets, it's quite probable that different people will execute different groups of tests. With this consideration in mind, the tests have been grouped together based on the typical skill sets and backgrounds of the people who might be expected to execute them. This part includes the following chapters:

Chapter 3: Network Security

Chapter 4: System Software Security

Chapter 5: Client-Side Application Security

Chapter 6: Server-Side Application Security

Chapter 7: Sneak Attacks: Guarding against the Less-Thought-of Security Threats

Chapter 8: Intruder Confusion, Detection, and Response

Having discussed what needs to be tested, Part 4, "Test Implementation," addresses the issue of how to best execute these tests in terms of who should actually do the work, what tools should be used, and what order the tests should be performed in (ranking test priority). This part includes the following chapters:

Chapter 9: Assessment and Penetration Options

Chapter 10: Risk Analysis

As a means of support for these 10 chapters, the appendix provides some additional background information, specifically: a brief introduction to the basics of computer networks as utilized by many Web sites (in case some of the readers of this book are unfamiliar with the components used to build Web sites), a summarized list of the top-20 critical Internet security vulnerabilities (as determined by the SANS Institute), and some sample test deliverable templates (which a security-testing team could use as a starting point for developing their own customized documentation).

Finally, the resources section not only serves as a bibliography of all the books and Web sites referenced in this book, but it also lists other reference books that readers interested in testing Web security may find useful in their quest for knowledge.

Terminology Used in This Book

The following two sections describe some of the terms used in this book to describe the individuals who might seek to exploit a security vulnerability on a Web site—and hence the people that a security tester is trying to inhibit—and the names given to some of the more common deliverables that a security tester is likely to produce.

Hackers, Crackers, Script Kiddies, and Disgruntled Insiders

The term *computer hacker* was originally used to describe someone who really knew how the internals of a computer (hardware and/or software) worked and could be relied on to come up with ingenious workarounds (*hacks*) to either fix a problem with the system or extend its original capabilities. Somewhere along the line, the popular press relabeled this term to describe someone who tries to acquire unauthorized access to a computer or network of computers.

The terminology has become further blurred by the effort of some practitioners to differentiate the skill levels of those seeking unauthorized access. The term *cracker* is typically used to label an attacker who is knowledgeable enough to create his or her own hacks, whereas the term *script kiddie* is used to describe a person who primarily relies on the hacks of others (often passed around as a script or executable). The situation becomes even less clear if you try to pigeonhole disgruntled employees who don't need to gain unauthorized access in order to accomplish their malicious goals because they are already authorized to access the system.

Not all attackers are viewed equally. Aside from their varying technical expertise, they also may be differentiated by their ethics. Crudely speaking, based on their actions and intentions, attackers are often be categorized into one of the following color-coded groups:

White-hat hackers. These are individuals who are authorized by the owner of a Web site or Web-accessible product to ascertain whether or not the site or product is adequately protected from known security loopholes and common generic exploits. They are also known as *ethical hackers*, or are part of a group known as a *tiger team* or *red team*.

Gray-hat hackers. Also sometimes known as *wackers,* gray-hat hackers attack a new product or technology on their own initiative to determine if the product has any new security loopholes, to further their own education, or to satisfy their own curiosity. Although their often-stated aim is to improve the quality of the new technology or their own knowledge without directly causing harm to anyone, their methods can at times be disruptive. For example, some of these attackers will not inform the product's owner of a newly discovered security hole until they have had time to build and publicize a tool that enables the hole to be easily exploited by others.

HACKER

***Webster's II New Riverside Dictionary* offers three alternative definitions for the word *hacker*, the first two of which are relevant for our purposes:**

1a. Computer buff

1b. One who illegally gains access to another's electronic system

COLOR-CODING ATTACKERS

The reference to colored hats comes from Hollywood's use of hats in old black-and-white cowboy movies to help an audience differentiate between the good guys (white hats) and the bad guys (black hats).

Black-hat hackers. Also known as *crackers*, these are attackers who typically seek to exploit known (and occasionally unknown) security holes for their own personal gain. *Script kiddies* are often considered to be the subset of black-hatters, whose limited knowledge forces them to be dependent almost exclusively upon the tools developed by more experienced attackers. Honeynet Project (2001) provides additional insight into the motives of black-hat hackers.

Of course, assigning a particular person a single designation can be somewhat arbitrary and these terms are by no means used consistently across the industry; many people have slightly different definitions for each category. The confusion is compounded further when considering individuals who do not always follow the actions of just one definition. For instance, if an attacker secretly practices the black art at night, but also publicly fights the good fight during the day, what kind of hatter does that make him?

Rather than use terms that potentially carry different meanings to different readers (such as hacker), this book will use the terms *attacker*, *intruder*, or *assailant* to describe someone who is up to no good on a Web site.

Testing Vocabulary

Many people who are new to the discipline of software testing are sometimes confused over exactly what is meant by some of the common terminology used to describe various software-testing artifacts. For example, they might ask the question, "What's the difference between a test case and a test run?" This confusion is in part due to various practitioners, organizations, book authors, and professional societies using slightly different vocabularies and often subtly different definitions for the terms defined within their own respective vocabularies. These terms and definitions vary for many reasons. Some definitions are embryonic (defined early in this discipline's history), whereas others reflect the desire by some practitioners to push the envelope of software testing to new areas.

The following simple definitions are for the testing artifacts more frequently referenced in this book. They are not intended to compete with or replace the more verbose and exacting definitions already defined in industry standards and other published materials, such as those defined by the Institute of Electrical and Electronics Engineers (www.ieee.org), the Project Management Institute (www.pmi.org), or Rational's Unified Process (www.rational.com). Rather, they are intended to provide the reader with a convenient reference of how these terms are used in this book. Figure 1.1 graphically summarizes the relationship between each of the documents.

Test plan. A *test plan* is a document that describes the *what, why, who, when*, and *how* of a testing project. Some testing teams may choose to describe their entire testing effort within a single test plan, whereas others find it easier to organize

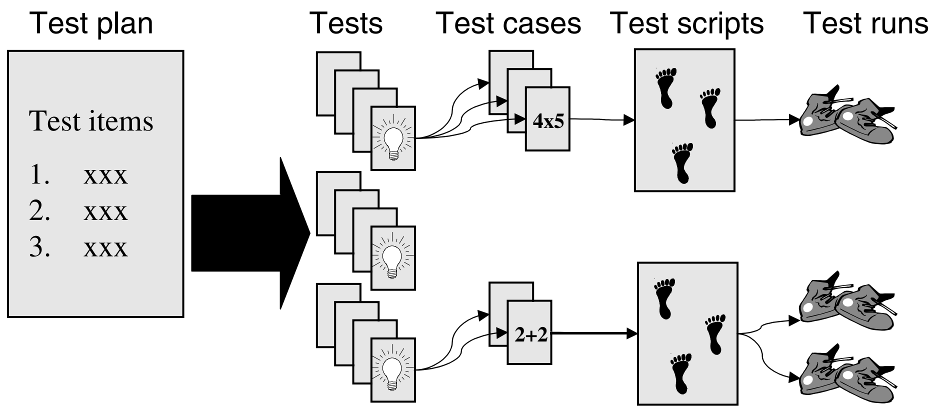

Figure 1.1 Testing documents.

groups of tests into two or more test plans, with each test plan focusing on a different aspect of the testing effort.

To foster better communication across projects, many organizations have defined test plan templates. These templates are then used as a starting point for each new test plan, and the testing team refines and customizes each plan(s) to fit the unique needs of their project.

Test item. A *test item* is a hardware device or software program that is the subject of the testing effort. The term *system under test* is often used to refer to the collection of all test items.

Test. A *test* is an evaluation with a clearly defined objective of one or more test items. A sample objective could look like the following: "Check that no unneeded services are running on any of the system's servers."

Test case. A *test case* is a detailed description of a *test*. Some tests may necessitate utilizing several test cases in order to satisfy the stated objective of a single test. The description of the test case could be as simple as the following: "Check that NetBIOS has been disabled on the Web server." It could also provide additional details on how the test should be executed, such as the following: "Using the tool nmap, an external port scan will be performed against the Web server to determine if ports 137–139 have been closed."

Depending on the number and complexity of the test cases, a testing team may choose to specify their test cases in multiple test case documents, consolidate them into a single document, or possibly even embed them into the test plan itself.

Test script. A *test script* is a series of steps that need to be performed in order to execute a test case. Depending on whether the test has been automated, this series of steps may be expressed as a sequence of tasks that need to be performed manually or as the source code used by an automated testing tool to run the test. Note that some practitioners reserve the term *test script* for automated scripts and use the term *test procedure* for the manual components.

Test run. A *test run* is the actual execution of a test script. Each time a test case is executed, it creates a new instance of a test run.

Who Should Read This Book?

This book is aimed at three groups of people. The first group consists of the owners, CIOs, managers, and security officers of a Web site who are ultimately responsible for the security of their site. Because these people might not have a strong technical background and, consequently, not be aware of all the types of threats that their site faces, this book seeks to make these critical decision makers aware of what security testing entails and thereby enable them to delegate (and fund) a security-testing effort in a knowledgeable fashion.

The second group of individuals who should find this book useful are the architects and implementers of a Web site and application (local area network [LAN] administrators, developers, database administrators [DBAs], and so on) who may be aware of some (or all) of the security factors that should be considered when designing and building a Web site, but would appreciate having a checklist of security issues that they could use as they construct the site. These checklists can be used in much the same way that an experienced airplane pilot goes through a mandated preflight checklist before taking off. These are helpful because the consequences of overlooking a single item can be catastrophic.

The final group consists of the people who may be asked to complete an independent security assessment of the Web site (in-house testers, Q/A analysts, end users, or outside consultants), but may not be as familiar with the technology (and its associated vulnerabilities) as the implementation group. For the benefit of these people, this book attempts to describe the technologies commonly used by implementers to build Web sites to a level of detail that will enable them to test the technology effectively but without getting as detailed as a book on how to build a Web site.

Summary

With the heightened awareness for the need to securely protect an organization's electronic assets, the supply of available career security veterans is quickly becoming tapped out, which has resulted in an influx of new people into the field of security testing. This book seeks to provide an introduction to Web security testing for those people with relatively little experience in the world of information security (*infosec*), allowing them to hit the ground running. It also serves as an easy-to-use reference book that is full of checklists to assist career veterans such as the growing number of certified information systems security professionals (CISSPs) in making sure their security assessments are as comprehensive as they can be. Bragg (2002), Endorf (2001), Harris (2001), Krutz et al. (2001 and 2002), Peltier (2002), the CISSP Web portal (www.cissp.com), and the International Information Systems Security Certifications Consortium (www.isc2.org) provide additional information on CISSP certification.