The Art of Software Security Testing: Identifying Software Security Flaws

By Chris Wysopal, Lucas Nelson, Dino Dai Zovi, Elfriede Dustin

The Art of Software Security Testing: Identifying Software Security Flaws
By Chris Wysopal, Lucas Nelson, Dino Dai Zovi, Elfriede Dustin

Risk-based security testing, the important subject of this book, is one of seven software security touchpoints introduced in my book, *Software Security: Building Security In*. This book takes the basic idea several steps forward. Written by masters of software exploit, this book describes in very basic terms how security testing differs from standard software testing as practiced by QA groups everywhere. It unifies in one place ideas from Michael Howard, David Litchfield, Greg Hoglund, and me into a concise introductory package. Improve your security testing by reading this book today."

–**Gary McGraw**, Ph.D., CTO, Cigital; Author, *Software Security*, *Exploiting Software*, *Building Secure Software*, and *Software Fault Injection*; www.cigital.com/~gem

"As 2006 closes out, we will see over 5,000 software vulnerabilities announced to the public. Many of these vulnerabilities were, or will be, found in enterprise applications from companies who are staffed with large, professional, QA teams. How then can it be that these flaws consistently continue to escape even well-structured diligent testing? The answer, in part, is that testing still by and large only scratches the surface when validating the presence of security flaws. Books such as this hopefully will start to bring a more thorough level of understanding to the arena of security testing and make us all a little safer over time."

–**Alfred Huger**, Senior Director, Development, Symantec Corporation

"Software security testing may indeed be an art, but this book provides the paint-by-numbers to perform good, solid, and appropriately destructive security testing: proof that an ounce of creative destruction is worth a pound of patching later. If understanding how software can be broken is step one in every programmers' twelve-step program to defensible, secure, robust software, then knowledgeable security testing comprises at least steps two through six."

–**Mary Ann Davidson**, Chief Security Officer, Oracle

"Over the past few years, several excellent books have come out teaching developers how to write more secure software by describing common security

failure patterns. However, none of these books have targeted the tester whose job it is to find the security problems before they make it out of the R&D lab and into customer hands. Into this void comes *The Art of Software Security Testing: Identifying Software Security Flaws*. The authors, all of whom have extensive experience in security testing, explain how to use free tools to find the problems in software, giving plenty of examples of what a software flaw looks like when it shows up in the test tool. The reader learns why security flaws are different from other types of bugs (we want to know not only that 'the program does what it's supposed to,' but also that 'the program doesn't do that which it's not supposed to'), and how to use the tools to find them. Examples are primarily based on C code, but some description of Java, C#, and scripting languages help for those environments. The authors cover both Windows and UNIX-based test tools, with plenty of screenshots to see what to expect. Anyone who's doing QA testing on software should read this book, whether as a refresher for finding security problems, or as a starting point for QA people who have focused on testing functionality."

–**Jeremy Epstein**, WebMethods

State-of-the-Art Software Security Testing: Expert, Up to Date, and Comprehensive

The Art of Software Security Testing delivers in-depth, up-to-date, battle-tested techniques for anticipating and identifying software security problems before the "bad guys" do.

Drawing on decades of experience in application and penetration testing, this book's authors can help you transform your approach from mere "verification" to proactive "attack." The authors begin by systematically reviewing the design and coding vulnerabilities that can arise in software, and offering realistic guidance in avoiding them. Next, they show you ways to customize software debugging tools to test the unique aspects of any program and then analyze the results to identify exploitable vulnerabilities.

Coverage includes

- Tips on how to think the way software attackers think to strengthen your defense strategy
- Cost-effectively integrating security testing into your development lifecycle
- Using threat modeling to prioritize testing based on your top areas of risk
- Building testing labs for performing white-, grey-, and black-box software testing
- Choosing and using the right tools for each testing project
- Executing today's leading attacks, from fault injection to buffer overflows
- Determining which flaws are most likely to be exploited by real-world attackers

This book is indispensable for every technical professional responsible for software security: testers, QA specialists, security professionals, developers, and more. For IT managers and leaders, it offers a proven blueprint for implementing effective security testing or strengthening existing processes.

The Art of Software Security Testing: Identifying Software Security Flaws

By Chris Wysopal, Lucas Nelson, Dino Dai Zovi, Elfriede Dustin

The Art of Software Security Testing: Identifying Software Security Flaws By Chris Wysopal, Lucas Nelson, Dino Dai Zovi, Elfriede Dustin

Risk-based security testing, the important subject of this book, is one of seven software security touchpoints introduced in my book, *Software Security: Building Security In*. This book takes the basic idea several steps forward. Written by masters of software exploit, this book describes in very basic terms how security testing differs from standard software testing as practiced by QA groups everywhere. It unifies in one place ideas from Michael Howard, David Litchfield, Greg Hoglund, and me into a concise introductory package. Improve your security testing by reading this book today."

–**Gary McGraw**, Ph.D., CTO, Cigital; Author, *Software Security, Exploiting Software, Building Secure Software,* and *Software Fault Injection*; www.cigital.com/~gem

"As 2006 closes out, we will see over 5,000 software vulnerabilities announced to the public. Many of these vulnerabilities were, or will be, found in enterprise applications from companies who are staffed with large, professional, QA teams. How then can it be that these flaws consistently continue to escape even well-structured diligent testing? The answer, in part, is that testing still by and large only scratches the surface when validating the presence of security flaws. Books such as this hopefully will start to bring a more thorough level of understanding to the arena of security testing and make us all a little safer over time."

–**Alfred Huger**, Senior Director, Development, Symantec Corporation

"Software security testing may indeed be an art, but this book provides the paint-by-numbers to perform good, solid, and appropriately destructive security testing: proof that an ounce of creative destruction is worth a pound of patching later. If understanding how software can be broken is step one in every programmers' twelve-step program to defensible, secure, robust software, then knowledgeable security testing comprises at least steps two through six."

–**Mary Ann Davidson**, Chief Security Officer, Oracle

"Over the past few years, several excellent books have come out teaching developers how to write more secure software by describing common security failure patterns. However, none of these books have targeted the tester whose job it is to find the security problems before they make it out of the R&D lab and into customer hands. Into this void comes *The Art of Software Security Testing: Identifying Software Security Flaws*. The authors, all of whom have extensive experience in security testing, explain how to use free tools to find the problems in software, giving plenty of examples of what a software flaw looks like when it shows up in the test tool. The reader learns why security flaws are different from other types of bugs (we want to know not only that 'the program does what it's supposed to,' but also that 'the program doesn't do that which it's not supposed to'), and how to use the tools to find them. Examples are primarily based on C code, but some description of Java, C#, and scripting languages help for those environments. The authors cover both Windows and UNIX-based test tools, with plenty of screenshots to see what to expect. Anyone who's doing QA testing on software should read this book, whether as a refresher for finding security problems, or as a starting point for QA people who have focused on testing functionality."

–**Jeremy Epstein**, WebMethods

State-of-the-Art Software Security Testing: Expert, Up to Date, and Comprehensive

The Art of Software Security Testing delivers in-depth, up-to-date, battle-tested techniques for anticipating and identifying software security problems before the "bad guys" do.

Drawing on decades of experience in application and penetration testing, this book's authors can help you transform your approach from mere "verification" to proactive "attack." The authors begin by systematically reviewing the design and coding vulnerabilities that can arise in software, and offering realistic guidance in avoiding them. Next, they show you ways to customize software debugging tools to test the unique aspects of any program and then analyze the results to identify exploitable vulnerabilities.

Coverage includes

- Tips on how to think the way software attackers think to strengthen your defense strategy
- Cost-effectively integrating security testing into your development lifecycle
- Using threat modeling to prioritize testing based on your top areas of risk
- Building testing labs for performing white-, grey-, and black-box software testing
- Choosing and using the right tools for each testing project
- Executing today's leading attacks, from fault injection to buffer overflows
- Determining which flaws are most likely to be exploited by real-world attackers

This book is indispensable for every technical professional responsible for software security: testers, QA specialists, security professionals, developers, and more. For IT managers and leaders, it offers a proven blueprint for implementing effective security testing or strengthening existing processes.

The Art of Software Security Testing: Identifying Software Security Flaws By Chris Wysopal, Lucas Nelson, Dino Dai Zovi, Elfriede Dustin Bibliography

- Rank: #569426 in Books
- Published on: 2006-11-27
- Released on: 2006-11-17
- Original language: English
- Number of items: 1
- Dimensions: 8.90" h x .70" w x 6.90" l, 1.32 pounds
- Binding: Paperback
- 312 pages

Foreword

Who can argue with testing things before you allow yourself to depend on them? No one *can* argue. No one *will* argue. Therefore, if testing is not done, the reasons have to be something other than a reasoned objection to testing. There seem to be exactly three: I can't afford it, I can get along without it, and I don't know how.

- **Not being able to afford it**—Allowing for economists to disagree over fine points, the cost of anything is the foregone alternative. If you do testing, what didn't you do? If it is to add yet another feature, perhaps you deserve congratulations on choosing a simpler product. Simpler products are in fact easier to test (and for good reason: the chief enemy of security is complexity, and nothing breeds complexity like creeping featuritis). If you didn't do testing, the usual reason given is to "get the product out on time." That reason is insufficient if not petulant. The sort of testing taught in this book is about the future even more than getting the product out on time is about the future. Only CEOs intoxicated on visions of wealth are immune to thinking about the future in ways that preclude testing. Testing is about controlling your future rather than allowing it to control you. Testing accelerates the inevitable future failure of products into the present. When William Gibson famously said, "The future is already here—it's just unevenly distributed," he wasn't thinking of testing as we mean it here. What you explicitly want is to unevenly distribute the future so that your product gets to see its future before your customers (and opponents) do. Since you are reading this paragraph, it's pretty likely you are of a testing frame of mind, so we'll drop the argument and move on.
- **Getting along without it**—Some products probably don't need much testing. They are not subject to innovation; they're nonperishable commodities, or something equally boring. That's not why we are here. We are here to protect security-sensitive products. Which products are those? A product is security-sensitive if, in its operation, it faces sentient opponents. If the only perils it faces are cluelessness ("Hey, watch this!") or random happenstance (alpha particles), the product may well not be security-sensitive. But with software and networks being as they are, nearly everything is security-sensitive because, if nothing else, every sociopath is your next-door neighbor. The burden of perfection is no longer on the criminal to commit the perfect crime but rather is on the defender to commit the perfect defense. Sure, you can get away with not testing, just as you can get away with never wearing protective gear while you band-saw aluminum, mountain-bike in Moab, or scrub down a P3 containment lab. There's always someone who has gotten away with that and more. That doesn't apply here. Why? Because the more successful and widespread your product is, the more those sociopaths, the more those sentient opponents, will adopt you as a special project. Just ask Microsoft. If you want to get widespread adoption, you will be tested. The only question is "Tested by whom?"
- **Not knowing how**—And so we come to the purpose of this book. You are ready, willing, and unable. Or you want to make sure that you're as up to date as your opponents. Or you need raw material for even more extreme sports than what is outlined here. You've come to a right place (there is no "the" right place). This is (let's be clear) a very right place. The authors are proven, and the techniques are current. Although techniques in security have the terrible beauty of never being "done," you won't do much better than these. If you can, there is an audience for your book. In the meantime, absorb what Chris Wysopal, Lucas Nelson, Dino Dai Zovi, and Elfriede Dustin have to teach you, and put it into practice. Skill sets like these do not grow on trees, and they don't stand still any more than the opposition stands still.

As you can see from the table of contents, testing is a way of thinking, not a button to press or a budget item to approve. You very nearly have to adopt this way of thinking—and nothing enforces a way of thinking as much as the regular use of tools and techniques that embody it. This is no joke. The outside attacker is skillful and increasingly professional and has tools and thought patterns. Malware—in particular, malware that turns good citizens into unintentionally bad citizens—has made true the long-standing supposition of security geeks: The real threat is the insider.

Question: What is an external attacker's first measure of success? Answer: Gaining an insider's credentials, access, and authority. If that attacker intends to do so by exploiting software the target insider runs, only your design and your testing stand in the way of the attacker's goals. As shown in the following figure, the idea is not "Does the product do what it is supposed to do?" but "Does the product not do what it supposed to not do?" That question is far harder than the quality assurance question because it is inherently open-ended. It cannot be fully handled by development per se. It has to be tested—preferably by informed testers not tangled up with the build process.

That, again, is where this book comes in. It tells you how to exert the kind of expert pressure that does accelerated failure time testing. You learn how to do so efficiently enough to be willing to do the testing and not think that you can get away without it. In other words, this is hunting in the bush. You can learn to do it by yourself, but following an expert tracker is a faster education than learning everything the hard way. Absorb everything that is here, and you'll either be a formidable hunter or you'll be in a position to be a tracker yourself. Remember, all skill is the result of practice. These authors are well practiced; it is your turn, and they have given you a leg up.

Daniel E. Geer, Jr., Sc.D.
24 July 2006

Endnote

1. Herbert H. Thompson and James A. Whittaker. Testing for Software Security. *Dr. Dobb's Journal*, 27(11): 24–32, November 2002.

Preface

With the growth of threats targeting computer systems, various organizations and institutions are looking for solutions that protect brand equity and customer confidence while minimizing maintenance costs.

The challenge is growing because of the widening scope of threats. Attackers started with UNIX-based Internet services and then moved to Windows-based PCs. Now they are beginning to target Apple MacOS X systems, networked video game consoles, wireless handheld devices, and even cell phones. After a software vulnerability is exposed, it often takes less than a week for hackers to come up with an exploit for it. And although the window of vulnerability has shrunk, it still takes about six weeks before software vendors issue a security patch. [1]

Symantec has reported that attackers are moving away from large, multiple-purpose attacks against traditional security devices such as firewalls and routers. At Symantec, 69% of the vulnerabilities reported in the last half of 2005 were in Web applications.

Instead, attackers are focusing their efforts on regional targets, desktops, and Web applications that potentially allow an attacker to steal corporate, personal, financial, or confidential information. A new Symantec Internet Security Threat Report cites the growing trend of attackers using bot [2] networks, targeted attacks on Web applications and Web browsers, and modular malicious code.

Security issues often translate into the loss of revenue and reputation for an organization. They also can result in the loss of market share, which may be vital to the organization's future. The security firms Counterpane Internet Security and MessageLabs estimate that a piece of malware with a modest infection

rate could cost a small company $83,000 a year—and may cost a large company $1 million or more. They add that these are direct losses and do not include indirect losses such as losses of reputation.[3]

Dave Cullinane, chief information security officer of the financial firm Washington Mutual in Seattle says, "If you have an application exposed to the Internet that will allow people to make money, it will be probed." Cullinane believes the consequences of being breached are not only financial but also damaging to the company's reputation. "The reputation risk can literally put you out of business. Twenty percent to 45% of customers will leave you if you report a security breach."[4] CardSystems, which processes credit card transactions, nearly went out of business because of software that let criminals steal the private financial data of millions of customers. The com-pany that purchased CardSystems was ordered by the FTC to undergo independent audits for the next 20 years.

This book discusses how this type of insecure handling of sensitive data could have been prevented. Specifically, testing for these types of scenarios is covered in Chapters 6 and 7.

Additionally, this CardSystems security disaster could possibly have been prevented if security testing had been part of the company's process and if it had learned how to detect this type of situation. Then, when it was detected, the company could have taken the removal steps discussed. Keep in mind that it is against the VISA PCI regulations to store most secrets at all. This is a credit card security standard and security policy requirement that should have been implemented, as further discussed in Chapter 3.

The threat is enormous, according to Gartner, which says that 70% of business security vulnerabilities are at the application layer. This is compounded by 64% of in-house business software developers admitting that they lack the confidence to write secure applications, according to research done by Microsoft.

Implementing security testing early on can help block a wide spectrum of current and future threats. Protecting the security and integrity of customer data—including, for example, online transactions containing personal information—is critical to maintaining customers' trust.

This book addresses the challenges facing today's software security engineers, software project managers, and other software professionals responsible for their applications' security.

These professionals work to develop and deploy systems. They are under pressure to complete secure development efforts, incorporate upgrades, and maintain secure systems ahead of the competition.

This book's objective is to teach the engineers who build and test software that to protect their customers, they must perform security testing throughout the development process. Unless security is addressed throughout the development lifecycle, software will inevitably have security issues that can lead to the theft of financial information, the loss of data and performance, and the compromise of personal data. In the past, software vendors have been able to get away with not focusing much on security testing. They simply wait for an external security researcher or customer to find a security problem and report it to them. Then they fix the problem and issue a patch. Enterprise customers are becoming overwhelmed by the hundreds of patches they must test and install, sometimes on thousands of machines.

Software developers can prevent the bad publicity and customers' loss of trust by adding security testing to their development processes. Every software publisher has a quality process in which QA professionals create and execute tests to verify the software's functionality. Security testing can extend this process to verify that the software does not contain common classes of security vulnerabilities. In the past, poor-quality software providers have been chided for using their customers as testers. Today, companies that do not perform security testing are putting their customers at risk while they wait for an external person to come forward and tell them about their security problems. Unfortunately, not everyone is honest. Some people

who discover security vulnerabilities in software will not tell the software vendor and may instead use the vulnerability to steal information.

In December 2005 criminals in Russia used a vulnerability in Internet Explorer called WMF to compromise unsuspecting Web surfers' machines and install bot software. One of the main reasons that this attack affected thousands of computers was because the vulnerability was discovered as it was being exploited in the wild before Microsoft could produce a fix. The window of vulnerability—the time between a vulnerability's being discovered and the vendor's having a fix available—is the riskiest time for customers who have a piece of vulnerable software. Security testing aims to eliminate the vulnerability. The window of vulnerability can be reduced by responsible disclosure. We can't prevent the window of vulnerability because it was found by "bad guys," but we can prevent the vulnerability itself.

Attacks on the Internet have changed over the last few years to become more criminal and insidious in nature. Three or four years ago the people taking advantage of software vulnerabilities were more interested in making a name for themselves with their peers, by writing and releasing viruses and worms, than in financial gain. The trend today is to use software vulnerabilities to help criminals gather financial, corporate, and government identity information that is then used to commit crimes. When home machines are compromised today, it is likely that they are used for phishing attacks or to relay spam. The software we use has not significantly improved with regards to security quality, yet today's attackers are more motivated than ever.

As a customer, you need to demand that your software suppliers integrate security into the development process. Request to see their process and tools. If the software is guarding financial data or other sensitive information, demand to see third-party validation of the supplier's security quality. Unless the industry voluntarily accepts security quality standards or the government mandates it, it is up to the customer to require proof of a certain level of security diligence. People can't occupy a building until the building inspector confirms that it is up to code. Automobiles and other potentially dangerous products also have requirements that must be independently tested. Today it is up to the software customer to perform these tests—or, more likely, to have a trusted third party perform the testing. Most corporations, however, do very limited third-party application security testing. Previously, this was done in federal organizations, but now it is spreading to larger financial institutions and enterprises. But even there, they don't have enough people adequately trained in black box security testing to be completely effective. Customers are required to do their own security testing (or hire third parties) because the vendors are not doing enough of it themselves.

Computer users accustomed to treating adware and spyware as just low-level annoyances were in for a surprise recently. According to Reuters,[5] a California man was indicted on federal charges of creating a robot-like network of hijacked computers that helped him and two others bring in $100,000 for installing unwanted adware.

The people behind adware always make money from low-level annoyances. However, keylogging spyware is the more insidious threat. Adware is just an annoyance, but hackers can use keylogging spyware to retrieve your financial site login/password and then wire-transfer money, which is a much bigger issue.

The Symantec Internet Security Threat Report, published every six months, analyzes and discusses Internet security activity. It covers Internet attacks, vulnerabilities, malicious code, and future trends. The latest edition of the Threat Report, covering the first six months of 2005, marks a shift in the threat landscape.[6] Attackers are moving away from large, multipurpose attacks on network perimeters and toward smaller, more focused attacks on client-side targets, such as Web browsers, media players, and instant-messaging programs. The new threat landscape will likely be dominated by emerging threats such as bot networks, customizable modular malicious code, and targeted attacks on Web sites. Unlike traditional attack activity,

many current threats are motivated by profit. Attackers often attempt to perpetrate criminal acts, such as identity theft, extortion, and fraud.

Numerous security analysts are working hard to keep the public informed of any vulnerability that has the potential to turn into an exploit. For example, Symantec's Deepsight Threat Management System [7] provides this type of intelligence covering the complete threat lifecycle, from initial discovery and disclosure of vulnerability to active attack.

BugTraq [8] is a high-volume, full-disclosure mailing list for the detailed discussion and announcement of computer security vulnerabilities. BugTraq is the cornerstone of the Internet-wide security community.

The SecurityFocus Vulnerability Database provides security professionals with the most up-to-date information on vulnerabilities for all platforms and services.

These are just a few of the numerous resources dedicated to tracking anonymous traffic. Monitoring the vast amount of information (malcodes, security risks, domain alerts) can be somewhat overwhelming.

Ideally, Web sites must be resistant to malicious attacks from Internet users, safeguarding both the site and users' confidential data. The Web application's end users must have confidence that they can use the system without unauthorized users accessing transactions or personal information stored by the Web site.

Material Coverage and Book Organization

This book is a one-stop shop for analyzing and testing applications and networks for security. It provides much-needed technical depth.

Some other books rely on using proprietary software costing upwards of $1,000, but all the tools described in this book are freely available. This book provides tool overviews for QA professionals and presents valuable strategies for penetration testers.

This book shows you how to test what is most meaningful and relevant and/or highest-risk so that you don't waste your expensive testing resources on less likely attack scenarios.

The general consensus still seems to be that specialists are required to conduct security testing. "The task of security testing may have to be performed by specially trained staff because normal quality assurance testers don't have the training to do the job," concluded a panel of corporate security executives, academics, and professional software developers at the RSA Conference in February 2006. [9]

There is a lot of confusion as to whose responsibility security testing is, yet it is the responsibility of many.

This book outlines the concept of the Secure Software Development Lifecycle (SSDL). It consists of six phases and parallels the software development lifecycle. Each chapter details the various phases of the SSDL (which is outlined in Chapter 3) and discusses the importance of incorporating and addressing security issues early in the lifecycle. This book touches on the security tasks, roles, and responsibilities of different team members during each phase to help clarify the tasks and whose responsibility security testing really is.

Security guidelines and rules and regulations have to be understood and/or defined early on. They are considered the umbrella guidelines for the security implementations. Whether your software development security standards are dictated by a standard, such as the VISA standard, or HIPPA or SOX, or if your security standard is homegrown, a security policy should exist and serve as the baseline for the SSDL.

During the business analysis and requirements phase, the security requirements are often overlooked but should be defined. During the prototype/ design/architecture phase, the security group supports this effort by conducting architectural reviews and threat modeling (discussed in Chapter 4). Doing so points out any potential security holes and determines the highest-risk parts of the application.

Secure coding guidelines help prevent defects from occurring in your code (Chapter 2) and need to be adhered to during software development. White/ gray/black box security testing techniques (discussed in Chapter 5) are used during the integration and testing phase.

Part II of the book focuses on discovering and attacking applications over a network. Then different methods of attack are discussed, including attacks against Web sites' authorization functionality. First you learn how to execute SQL injection attacks. Then you take a look at other common attacks against Web servers (Chapters 6, 7, and 8). Finally, you learn how to use different proxies to test applications for a number of different vulnerabilities (Chapters 9 and 10). Each chapter in Part II summarizes test attack patterns, which will serve the QA professional or test engineer as a starting point for test outlines or testing checklists.

Determining exploitability (discussed in Part III) is done throughout the entire lifecycle, yet results are finalized during the system development production and maintenance phase. A patch management program has to be in place. Security program review and assessment activities need to be conducted throughout the testing lifecycle to allow for continuous improvement activities. Following security test execution, a final review and assessment using the metrics collected throughout should be conducted to allow for adequate and informed decision-making.

This book is broken into three major parts.

Part I

Part I discusses the fact that security testing requires a paradigm shift from traditional testing methods and teaches the security tester to think like an attacker. It discusses how vulnerabilities get into all software and how threat modeling and white, gray, and black box testing can help uncover those vulnerabilities.

Chapter 3 in Part I lays out how the parts that make up the Secure Software Development Lifecycle (SSDL) are described throughout the book.

- Chapter 1, "Case Your Own Joint: A Paradigm Shift from Traditional Software Testing," talks about securing your software by trying to break it. This requires testers to work as detectives and necessitates a paradigm shift from traditional testing. A list of high-level security testing strategies is provided. Security testing requires a different way of thinking about testing. The tester needs to think of herself as not only a verifier but also as an attacker who has to do some detective work to determine the application's weakest areas and to design the appropriate attack. Because not all attacks are equally destructive, the tester must evaluate and prioritize the security test scenarios and remediate any uncovered potential vulnerabilities.
- Chapter 2, "How Vulnerabilities Get into All Software," discusses the various vulnerabilities and how to prevent them from sneaking into your programs (such as design versus implementation vulnerabilities). A design vulnerability is a design mistake that precludes the program from operating securely, no matter how perfectly it is implemented by the coders. Implementation vulnerabilities, on the other hand, are caused by security bugs in the actual coding of the software. This chapter discusses how to avoid the various types of security bugs that can slip into your project.
- Chapter 3, "The Secure Software Development Lifecycle," discusses the importance of incorporating and addressing security issues early in the lifecycle. In the traditional lifecycle, security testing is often an afterthought. However, it is bad practice to delay security testing efforts until after the software has been

developed. This chapter covers early implementation of security testing efforts and discusses how security needs can be addressed in the software development lifecycle, starting with the earliest phases.
- Chapter 4, "Risk-Based Security Testing: Prioritizing Security Testing with Threat Modeling," discusses the uses of threat modeling for security testing efforts. It discusses the various steps in threat modeling, such as information gathering (including meeting with the architects or conducting runtime inspection), carrying out the modeling process, ranking priorities, and using mitigation strategies.
- Chapter 5, "Shades of Analysis: White, Gray, and Black Box Testing," discusses the different types of security testing and their advantages and disadvantages. It includes information about how to set up a lab environment for security testing and the software tools required.

Part II

This part of the book starts with techniques that allow testers to think like an attacker. It discusses how to perform the attacks in detail. Attacks such as generic network fault injection, application attacks, proxies, and local fault injection are discussed.

- Chapter 6, "Generic Network Fault Injection," focuses on discovering and attacking applications over a network. First, you learn how to determine the application's network footprint. Then different methods of attack are discussed. The chapter concludes with a discussion of using a man-in-the-middle random fuzzer.
- Chapter 7, "Web Applications: Session Attacks," focuses on attacks against Web site session management functionality. You learn about brute-forcing passwords, cookie analysis, and cross-site scripting.
- Chapter 8, "Web Applications: Common Issues," shows you how to execute SQL injection attacks. Then the chapter looks at other common attacks against Web servers.
- Chapter 9, "Web Proxies: Using WebScarab," shows you how to use a freeware Web proxy, WebScarab, and its features.
- Chapter 10, "Implementing a Custom Fuzz Utility," shows you how to use rapid prototyping languages to implement a customized fuzzing utility.
- Chapter 11, "Local Fault Injection," details specific techniques for testing local application attack surfaces, including ActiveX interfaces, file formats, command-line arguments, and shared memory segments.

Part III

This last part of the book provides an approach to the analysis and shows you how to determine exploitability.

- Chapter 12, "Determining Exploitability," describes how to identify a vulnerability's scope and severity based on its potential impact on the application. You see how vulnerabilities are triggered at the processor instruction level so that you can identify whether an application crash may result in code execution.

Target Audience

This book addresses the pragmatic concerns of and provides information needed by software security and test engineers who need to perform security testing more thoroughly and quickly. These concerns also may apply to the software developer who is responsible for security testing (in addition to unit and integration testing). This book also supports the needs of the software project manager, who is responsible for security testing on a project. This book supplies the project manager with information such as the security test's goals and objectives, how to decide where and whether to implement a specific security test, how to introduce security testing on the project, and what needs to be reflected in test planning, development, and execution of security testing efforts.

This book addresses the pragmatic concerns and information needed by the following software professionals:

- **Security engineer**—This book contains a great deal of information related to the SSDL, including different methods of attack that the security engineer can apply. Many security engineers have been consumed in the past with network and host-level security. This book allows the security engineer to shift the focus to the application layer and to work with software developers and testers to deploy more secure software while also being involved in software patch management.
- **Software test engineer**—This book is useful for QA/test engineers seeking a more comprehensive technical understanding of potential security issues. It provides a useful guide for understanding the applications of security testing, as well as the methods with which to introduce, manage, and perform security testing on a project. In the past, many software test engineers have implemented security testing as an afterthought or not at all. The result is often a band-aid to security issues uncovered in production—issues that should have been addressed much earlier. This book outlines how to avoid these missteps and how to fully leverage the investment in the SSDL. In-depth discussions are accompanied by step-by-step testing instructions for test engineers. Each chapter summarizes test attack patterns, which can serve as testing outlines or checklists.
- **Software developer**—This book is a guide to security testing for the software developer who is responsible for development. It offers developers insight into and knowledge of how to perform security testing and helps you keep security in the forefront during software development.
- **Software development manager**—This book provides greater technical insight into the key security testing success factors. It is an informative guide to the application of various kinds of security testing strategies and security testing tools. It shows you how to make informed decisions about software security and how to prioritize security test implementation.
- **Project/test manager**—For project and test managers, this book can be used as a pragmatic guide to support security test planning, design, and execution. It also shows you how to prioritize security tests, how to introduce security testing on the project, and what needs to be reflected in test planning.
- **Quality assurance (QA) engineer**—This book helps the QA engineer perform quality reviews on the security policy. The SSDL addresses security issues early in the software development lifecycle, from program inception. This includes security standards and policies, security requirements, security design, security test procedures, results of testing, and patch management policies.
- **Configuration management (CM) specialist**—This book helps the CM specialist understand security requirements and the benefits of baselining security test scripts.
- **Requirements management (RM) specialist**—This book helps the RM specialist understand the relationship between defined security requirements and test design/scripts.
- **Chief information officer/senior manager**—This book serves as an informative guide on the availability of various kinds of security test strategies as well as security tools and their application to corporate information system development. This book offers senior managers, and accountable officers within government and industry, guidelines for considering and implementing security testing within their development lifecycle process. It also provides insight to help managers understand the results of software security quality measurements for decision-making purposes.
- **Classroom instructor**—This book may also be used to provide classroom instruction on software security testing using the latest strategies and test tool capabilities. Students are introduced to application security and the importance of software security testing. They also receive instruction on the different kinds of security tests that may be performed.

Endnotes

1. Symantec Corporation's Internet Security Threat Report, March 2006.
2. Bot software is a malicious program that takes control of a home or business computer. The computer is

then used for phishing, as a spam relay, or simply to record the user's keystrokes as she logs into financial sites or her work network.
3. 2005 Attack Trends & Analysis report, released by Counterpane Internet Security and MessageLabs, http://www.counterpane.com/cgi-bin/attack-trends4.cgi.
4. "Businesses Should Pay More Attention to Software Security," article by Tim Greene, 2006 *Network World*. Summarizes the findings of a panel of corporate security executives, academics, and professional software developers speaking at the RSA Conference in February 2006.
5. "Recent Developments in Adware and Spyware," Symantec, February 21, 2006, article ID: 6476.
6. Symantec Internet Security Threat Report, Volume VIII, September 2005.
7. http://www.tms.symantec.com.
8. http://www.securityfocus.com/about.
9. "Businesses Should Pay More Attention to Software Security," article by Tim Greene, 2006 *Network World*.

Read The Art of Software Security Testing: Identifying Software Security Flaws By Chris Wysopal, Lucas Nelson, Dino Dai Zovi, Elfriede Dustin for online ebook

The Art of Software Security Testing: Identifying Software Security Flaws By Chris Wysopal, Lucas Nelson, Dino Dai Zovi, Elfriede Dustin Free PDF d0wnl0ad, audio books, books to read, good books to read, cheap books, good books, online books, books online, book reviews epub, read books online, books to read online, online library, greatbooks to read, PDF best books to read, top books to read The Art of Software Security Testing: Identifying Software Security Flaws By Chris Wysopal, Lucas Nelson, Dino Dai Zovi, Elfriede Dustin books to read online.

Online The Art of Software Security Testing: Identifying Software Security Flaws By Chris Wysopal, Lucas Nelson, Dino Dai Zovi, Elfriede Dustin ebook PDF download

The Art of Software Security Testing: Identifying Software Security Flaws By Chris Wysopal, Lucas Nelson, Dino Dai Zovi, Elfriede Dustin Doc

The Art of Software Security Testing: Identifying Software Security Flaws By Chris Wysopal, Lucas Nelson, Dino Dai Zovi, Elfriede Dustin Mobipocket

The Art of Software Security Testing: Identifying Software Security Flaws By Chris Wysopal, Lucas Nelson, Dino Dai Zovi, Elfriede Dustin EPub

0L1WKHD3Q2N: The Art of Software Security Testing: Identifying Software Security Flaws By Chris Wysopal, Lucas Nelson, Dino Dai Zovi, Elfriede Dustin