Compliments of QUALYS®

Web Application Security

FOR DUMMIES®

Qualys Limited Edition

Scan your website for vulnerabilities

Making Everything Easier!™

FREE eTips at dummies.com®

Mike Shema

Web Application Security

FOR

DUMMIES®

by Mike Shema

A John Wiley and Sons, Ltd, Publication

Web Application Security For Dummies®
Published by
John Wiley & Sons, Ltd
The Atrium
Southern Gate
Chichester
West Sussex
PO19 8SQ
England

For details on how to create a custom *For Dummies* book for your business or organisation, contact CorporateDevelopment@wiley.com. For information about licensing the *For Dummies* brand for products or services, contact BrandedRights&Licenses@Wiley.com.

Visit our Home Page on www.customdummies.com

Copyright © 2011 by John Wiley & Sons Ltd, Chichester, West Sussex, England

All Rights Reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except under the terms of the Copyright, Designs and Patents Act 1988 or under the terms of a licence issued by the Copyright Licensing Agency Ltd, 90 Tottenham Court Road, London, W1T 4LP, UK, without the permission in writing of the Publisher. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, England, or emailed to permreq@wiley.com, or faxed to (44) 1243 770620.

Trademarks: Wiley, the Wiley Publishing logo, For Dummies, the Dummies Man logo, A Reference for the Rest of Us!, The Dummies Way, Dummies Daily, The Fun and Easy Way, Dummies.com and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. Wiley Publishing, Inc., is not associated with any product or vendor mentioned in this book.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER, THE AUTHOR, AND ANYONE ELSE INVOLVED IN PREPARING THIS WORK MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

ISBN: 978-1-119-99487-9

Printed and bound in Great Britain by Page Bros, Norwich

10 9 8 7 6 5 4 3 2 1

Publisher's Acknowledgements

We're proud of this book; please send us your comments through our Dummies online registration form located at www.dummies.com/register/.

Some of the people who helped bring this book to market include the following:

Acquisitions, Editorial and Media Development

Corporate & Custom Publishing: Scott Smith

Project Editor: Rachael Chilvers

Executive Project Editor: Daniel Mersey

Composition Services

Project Coordinator: Kristie Rees

Layout and Graphics: Samantha K. Cherolis

Introduction

Welcome to *Web Application Security For Dummies*! Web applications have become the Achilles heel of IT security. Web application vulnerabilities are now the most prevalent at more than 55 per cent of all server vulnerability disclosures. This figure doesn't include vulnerabilities in custom-developed web applications, so it may be just the tip of the iceberg. This book is all about understanding how to quickly find and fix vulnerabilities in web applications. The goal is to prevent attackers from gaining control over the application and obtaining easy access to the server, database, and other back-end IT resources.

About This Book

This book simply explains all about web application security. After reading this book you'll know how to use a web application security scanner to quickly find vulnerabilities and remediate them for stronger security.

Foolish Assumptions

In writing this book, we assume that you:

- Work in a mid- to large-sized organization and know that you have to secure web applications, but you aren't sure what's required or what you need to do.
- Are familiar with information technology and managing its operations.
- Want to discover the easiest, most effective and direct way to improve web application security.

How to Use This Book

This book is divided into five succinct and easily-digestible parts:

- **Part I: Why Web Security Matters.** Start here for a primer on the importance of web application security.
- **Part II: Establishing a Web Application Security Program.** Here we present a framework of actions you can take to find and fix vulnerabilities in custom web applications.
- **Part III: Using Automated Scanning to Test Web Applications.** This part serves up a guide to choosing and using a scanner to automatically find and prioritize web application vulnerabilities.
- **Part IV: Introducing QualysGuard WAS.** Here you discover the ease and simplicity of using a popular web application scanner from Qualys.
- **Part V: Ten Tips for Securing Web Applications.** Follow this short list of steps to ensure stronger security for your custom web applications.

Icons Used in This Book

We highlight crucial text for you with the following icons:

This icon targets hints and shortcuts to help you comply with policy.

Memorize these pearls of wisdom – and remember how much better it is to read them here than to have your auditor say 'I told you so' later on.

The bomb means 'whoops.' It signals common errors that can happen. Avoid these at all cost.

Prepare for a little bit of brain strain when you see this icon. But don't worry – you don't need to have a doctorate in Web Application Security to successfully make web applications secure.

Where to Go from Here

Check out the section headings in this book and start reading wherever it makes sense. This book is written with a sequential logic, but if you want to jump to a specific topic you can start anywhere to extract good stuff.

Part I

Why Web Security Matters

In This Part

- Noting a hacker's attraction to web applications
- Reviewing the dangers of insecure web applications
- Understanding the potential fallout from a breach
- Resolving to improve the security of web applications

As the world embraces cloud computing, more and more people are transacting business, conducting research, storing information, collaborating with co-workers, publishing personal thoughts, and fostering relationships via web applications.

Web applications use a simple architecture:

- Internet or an intranet for connectivity between user and application.
- Creation of the application with a browser-rendered markup language such as hypertext markup language (HTML).
- Hosting of the application in a browser-controlled environment.
- A browser for user execution of the application on an endpoint device.

Each time you launch a browser and connect to a website, you're using one or more web applications. These enable *thin client* computing, which dramatically reduces resource requirements for the endpoint device. With web applications, the bulk of processing occurs on servers located at remote websites.

As a result, users can run sophisticated web applications from virtually any PC, a low-powered netbook, a tablet computing device, or smartphone. Web applications are generally easy to use, cost little or nothing for the user to operate, are efficient, and pervasive. This is why, as we said in the Introduction, web applications have become the Achilles heel of information technology (IT) security.

Putting Web Applications in Attackers' Crosshairs

The 20th-century American criminal Willie Sutton reportedly said he robbed banks because 'that's where the money is.' As you'll see, web applications hold the same attraction to modern cyber criminals because web application vulnerabilities are now the most prevalent of all server vulnerability disclosures.

Network security professionals are already familiar with many other types of IT vulnerabilities. The process of finding and fixing these is called 'vulnerability management' or VM. For more info about VM, check out a companion book (also written by Qualys), *Vulnerability Management For Dummies*.

Meanwhile, this book, *Web Application Security For Dummies*, is all about understanding how to quickly find vulnerabilities in your organization's web applications and fix them so as to prevent attackers from gaining control over the application and other IT resources.

Understanding the Dangers of Insecure Web Applications

Vulnerabilities in web applications may take dozens of forms. Many attacks use fault injection, which exploits vulnerabilities in a web application's syntax and semantics. In simple terms, an attacker manipulates data in a web page Uniform Resource Indicator (URL) link to force an exploitable malfunction in the

application. The two most common varieties are SQL injection and cross-site scripting. Later we'll dive into details of how these and other vulnerabilities work (and how to get rid of them!). For now, here's the basic idea. Consider a typical URL:

```
http://example/foo.cgi?a=1
```

Executing a SQL injection exploit simply requires modifying the URL. All that's needed might be one odd character to trigger a successful exploit, such as adding an apostrophe to the end of the URL:

```
http://example/foo.cgi?a=1'
```

The 'successful' outcome can give an attacker control over the application and easy access to the server, database, and other back-end IT resources. Needless to say, this access can trigger disastrous results.

Data is the object of desire for attackers – particularly data that converts their efforts into cash. The most lucrative source of this data is a business database containing information that can be sold or used directly by an attacker for profit. Business databases are like pots of gold brimming with bankable opportunities – all in one location. Some of these include:

- Strategic business plans.
- Product plans and other intellectual property.
- Competitive analysis.
- Employee rosters and their personal information.
- Confidential data from business partners.
- Confidential customer data.

Confidential customer data may be the highest value data because it's easy to sell and leverage. It includes personally identifiable data such as names, addresses, birth dates, payment card Primary Account Numbers, email addresses, and so on. Some of the worst data breaches have included the theft of millions of records containing this information. The massive scale of instant damage is unprecedented.

Web application attacks may also target individuals, one by one. Some attacks are executed by infiltrating a trusted website, which then injects malware into computers used by unsuspecting visitors. The malware might redirect links to rogue sites that steal personal information directly from the user's PC. It could trick users into revealing confidential passwords or payment card data. It may even hijack the user's PC and transform it into a spam server or other nefarious mechanism aimed to further the attacker's goals.

Either way, successful attacks on web applications can result in highly negative fallout.

Knowing the Potential Fallout from a Breach

Fallout from a data breach via a web application exploit can range from minor to substantial. Thanks to U.S. federal law and standard practice by the financial industry, the maximum penalty of a consumer whose cardholder data is stolen is just $50; the rest of all related losses are paid by the payment card companies. If the data allows a criminal to access other accounts or steal a consumer's identity however, financial fallout could be severe for that individual. Resolving just one incident of a stolen identity may take years of effort. Personal fallout would be catastrophic if multiple breaches at different merchants occurred during a short period of time.

Sobering statistics for web insecurity

Security researchers are providing gloomy proof of a universal need for stronger web application security. Here are some notable highlights:

Threat is growing. The number of known web application vulnerabilities is growing by about 3,000 to 4,000 disclosures per year, according to the *IBM X-Force 2010 Mid-Year Trend and Risk Report*. It says cross-site scripting and SQL injection vulnerabilities predominate attack techniques. Most platform vulnerabilities (88 per cent) are in browser plug-ins. Client-side vulnerabilities are the second largest category with about a fifth of all disclosures.

See `www-935.ibm.com/services/us/iss/xforce/trendreports/` for the report.

Damage is deadly. Hacking and malware are 'more dominant than normal' based on total records compromised in actual breaches, according to the Verizon Business *2010 Data Breach Investigation Report*, which was conducted with the U.S. Secret Service. For example, SQL injection constituted 25 per cent of breaches caused by hacking – and these contributed to 89 per cent of records breached. Cardholder data is usually a prime target of attack, and for organizations with breaches in 2009, their lowest rates of compliance with the Payment Card Industry Data Security Standard were 21 per cent for Requirement 6: 'Develop and maintain secure systems and applications,' and 25 per cent for Requirement 11: 'Regularly test security systems and processes.' Talk about self-inflicted damage!

Head to `www.verizonbusiness.com/resources/reports/rp_2010-data-breach-report_en_xg.pdf` for more information.

Defense is in disarray. Quite often, there's no clear organizational accountability for web application security, based on the Ponemon Institute's April 2010 study, *State of Web Application Security*. According to respondents: 70 per cent say their organizations devote insufficient resources for web application security; 34 per cent of 'urgent' vulnerabilities remain unfixed; 38 per cent say fixing one vulnerability takes more than 20 hours of developer time; and 55 per cent say developers are 'too busy' to fix security flaws. Visit `www.imperva.com/docs/AR_Ponemon_2010_State_of_Web_Application_Security.pdf` for the full story.

Businesses face their own types of fallout. When a breach occurs, companies face detection, discovery and containment costs for investigating the incident; recovery and remediation expenses; and attorney and legal fees. But this is just for starters. Long-term fallout for all businesses may include:

- Loss of customer confidence.
- Lost sales and revenue.
- Lower use of online stores due to fear of breaches.
- Brand degradation or drop in public stock value.
- Fines and penalties for non-compliance with the Payment Card Industry Data Security Standard (PCI DSS) and other regulations.

- Higher costs for subsequent audits when merchants with a breach must subsequently comply with the penalty of more stringent requirements.
- Termination of the ability to accept payment cards.
- Fraud losses.
- Cost of reissuing new payment cards.
- Dispute resolution costs.
- Cost of legal settlements or judgments.

The potential fallout for large enterprises can be huge, but isn't insurmountable if a company is well capitalized. Smaller companies, however, may have significant trouble weathering a data breach. Think about your company's cash flow and whether it could cover potential damage from a breach. The risk of going out of business should be motivation enough to follow steps to protect your web applications and data. But that's why you're reading this book, right? So let's get to work!

Improving Security in Web Applications

Web application vulnerabilities are often outside the traditional expertise of network managers, even if their main job is network security. The built-in obscurity of web application vulnerabilities helps them evade traditional network defenses – unless an organization takes deliberate countermeasures. Unfortunately, there's no silver bullet for detection! As with network security, the best strategy is a multi-layer approach. Detection and remediation may require source code analysis. Detecting some web application vulnerabilities may require on-site penetration testing.

The good news is that most prevalent web application vulnerabilities can be easily detected with an automated scanner. As you'll see later in this book, scanning web applications acts to supplement and compliment manual testing by performing likely attacks on target applications. Scanning web applications has even become a strategic requirement in some regulations. For example, the Payment Card Industry Data Security Standard (PCI DSS) version 2.0 now requires all merchants accepting payment cards to pass quarterly scans for vulnerabilities in web applications.

Automated scanning can provide many benefits, including:

- Discovering and cataloging all web applications in your enterprise.
- Lowering the total cost of operations by automating repeatable testing processes.
- Identifying vulnerabilities of syntax and semantics in custom web applications.
- Performing authenticated scanning.
- Profiling the target application.
- Ensuring accuracy by effectively reducing false positives and false negatives.

Next, information in Part II will help place scanning in context of overall vulnerability management. Get ready, for you're about to learn how to establish a web application security program. Onward to the good stuff!

Part II

Establishing a Web Application Security Program

In This Part

- Designating someone to lead web application security
- Using the software development lifecycle to address risks
- Adding applications to enterprise vulnerability management
- Using tools to automate the web application security program

IT security managers dream that resolving the challenges of web application security will be an easy check-off on the vulnerability management to-do list. For some aspects, this can be true – especially processes that benefit by the use of automated scanning technology. But achieving web application security entails more than scanning. It's important to step back for a swift look at the bigger picture. You need to understand how scanning and other tasks fit into a *program* that addresses everything it takes to develop, deploy, and maintain secure web applications, and that's exactly what we cover in this chapter.

Deciding Who's In Charge

As successful programs require good management, first you need to address the point of who's in charge.

A recent study by the Ponemon Institute concluded 'there is no clear accountability for Web application security.' That's not to say that security is running leaderless. Twenty-three per cent of respondents to the *State of Web Application Security* survey say the chief information officer (CIO) is mostly in charge, followed by 18 per cent who give the nod to IT operations. Thirteen per cent say the website administrator is 'it.' However, none of the people in these categories even *write* custom web applications. So how will they have the wherewithal to identify the vulnerabilities – or fix them?

You need to resolve the dilemma posed by the lack of clarity of who is, or should be, in charge if your organization is to have a fighting chance at managing web application vulnerabilities. Clearly, many disciplines are linked to the task. Consider:

- Developers write the applications, and are the obvious candidates to correct the code if vulnerabilities are discovered.
- Website administrators deploy and maintain the applications.
- Network managers assist with connectivity and performance management.
- IT managers keep everything working.
- Security administrators handle safety of networks and systems.
- Compliance officers make sure deployment mandates are met, deal with auditors, file paperwork, and resolve findings.
- The CIO coordinates all these efforts.

Cross-discipline cooperation is mandatory for web application security. It's vital when time is of the essence for urgent vulnerability remediation. So decide now who'll take the lead as doing so will help to smooth out operational issues later.

We suggest that overall responsibility for web application security should rest with the security team. These people are responsible for vulnerability management in networks and systems. Adding application security to their watch makes sense because this team already has 'find and fix the vul-

nerabilities' in its DNA and workflow. Integrating the use of automated scanning tools for web applications augments the technical skills of security staffers doing vulnerability management. These tools can guide the security team as it interacts with developers. Remediation details can remain the domain of web application programmers.

Using the Software Development Lifecycle to Address Security of Web Applications

An organization that relies upon custom web applications to implement business processes can have up to thousands of web applications. These may include full-blown applications, or consist of modules such as shopping carts, forms, login pages, and other forms of dynamic content. Those that appear in your network could be developed in house, although some may be legacy sites with no designated ownership or support. Analyzing all of these for vulnerabilities and prioritizing their importance for remediation can be a huge task without organizing efforts and using automation to improve efficiency and accuracy.

The software development lifecycle (SDLC) aims to do this analysis and prioritization. SDLC is rooted in the mature discipline of Software Assurance, which the industry defines as follows: *'Confidence that software, hardware and services are free from intentional and unintentional vulnerabilities and that the software functions as intended.'* (Source: Software Assurance Forum for Excellence in Code.)

The SDLC presents three broad stages:

- Secure development.
- Secure deployment.
- Secure operations.

As mapped over these stages by Securosis in *Building a Web Application Security Program* (see the Qualys website for a copy), web application security consists of seven elements, as shown in Figure 2-1.

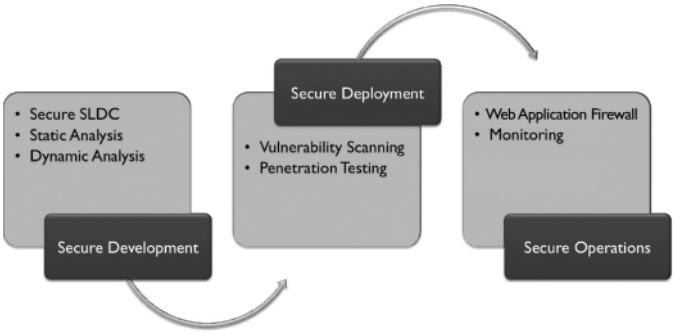


Figure 2-1: Web application security lifecycle (Source: Securosis).

Secure development

The Secure Development phase is all about building security into web applications right from their inception. This is what commercial software companies do because customers expect what they license to be secure. This isn't an automatic process, however, so as your organization strives to be smart with its efforts to secure applications on its websites, it needs to provision for several elements. These include:

- **Secure SDLC.** The software development life cycle is where your organization establishes best practices for secure coding, testing, and proving that software is safe.
- **Static Analysis.** This entails line-by-line analysis of code for errors or improper implementations of open source modules. So-called 'white box' tools (testing code structure) can help automate this process.
- **Dynamic Analysis.** Tools of the 'black box' variety (functionality testing) attack and try to break running applications. They don't analyze source code.

Secure deployment

Web applications are deployed when deemed functional within specifications, and when they're secure. Upon deployment, ensuring their security requires two new elements:

- **Vulnerability Scanning.** Applications must be scanned as a credentialed user and as a non-credentialed user to simulate a full range of tests. Ideally, scanning should occur as part of enterprise vulnerability management. In addition to testing systems and networks, web applications are tested to assess exposure to known vulnerabilities, other threats, and for compliance with the organization's security policy. Applications at risk must be fixed to eliminate vulnerabilities. Remote scanning can be software-based or Software-as-a-Service (SaaS). In cases where SaaS is used, internal scanning often utilizes a trusted Scanner Appliance. This ensures that scanning is thorough from the inside out.
- **Penetration Testing.** Usually done by skilled experts, penetration testing is a deep, targeted effort to break a web application. In exploring vulnerabilities, a 'pen test' helps to measure and prioritize their impact.

Secure operation

The operational phase includes detection and reaction to actual vulnerabilities and potential exploits. Ideally, the security team leads this process with participation by programmers and other application experts as required. New elements for the operational phase include:

- **Web Application Firewall (WAF).** This tool can help provide visibility into web application traffic. It also can block known attacks.
- **Activity Monitoring.** An organization needs perpetual visibility on the operations and security of the web applications, databases powering their operation, and systems that host operations and provide connectivity. Automated tools can provide this visibility and instantly alert the security team when policy is violated.

In addition, you need to continue Vulnerability Scanning and Penetration Testing. Success is only as good as results from the latest security scan.

Adding Applications to Overall Vulnerability Management

As an alert reader, you may have noticed an important lack of distinction in the last section. For the topic of vulnerability management, there's no lower or higher rating of importance ascribed to web application scanning versus other kinds of scanning such as for network or system vulnerabilities. Creating web application security requires your organization to scan for *all* these types of vulnerabilities because they're interrelated. Security in each category can suffer from weaknesses in other categories. This is why comprehensive vulnerability management is essential.

The traditional intention of vulnerability management has focused on vulnerabilities in the network, and attached devices and endpoints. In fact, network vulnerability management was the first business solution addressed by Qualys when it was founded in 1999. As hackers have gained more expertise, vulnerability management has expanded to new risk vectors. Be advised that if you're using a network vulnerability scanner, it usually doesn't address other areas of security such as web applications. And learning how to protect applications is, no doubt, why you're reading this book!

Following are the seven best practices or steps to Vulnerability Management (VM). Although we present these specifically for web application VM, the overlap in processes also applies to network and web application vulnerability management.

Step 1: Track and categorize web applications

With enterprise VM, you need to itemize all systems and categorize them before conducting security audits. Web application VM is no different – you must identify all web applications for testing and decide on their relative priority of importance for assessing remediation. Tools include automated scanners that do most of the work for you. The undeniable fact is that a machine can do some things better than us humans!

Step 2: Scan everything for vulnerabilities

Scanning is an automated process that tests items for security as measured by a dynamic database of known vulnerabilities and likely exploits. A web application scanner includes simulated attacks against each web application. The object is to see if it breaks.

Step 3: Verify vulnerabilities against inventory

This step helps to reduce false positives and false negatives, which can lead to inefficiencies in the VM process. False positives inhibit some scanning by drowning the scan results with vulnerabilities that don't match what's in your inventory of network and IT assets and web applications. Chasing down false positives is a waste of IT staff time and an inefficient way to do VM. Likewise, a false negative may occur when the scanner fails to detect a vulnerability that actually exists in the device or application. This failure to detect actual vulnerabilities may place your applications at serious risk of exploitation by hackers.

Step 4: Classify and rank risks

Fixing everything at once is practically impossible. In fact, in large organizations, the amount of vulnerability data can be overwhelming if it's not properly categorized, segmented, and prioritized in a meaningful fashion. Step 4 defines the most critical issues that could impact the most critical applications – all the way to items of lesser importance. In a nutshell, you need to decide what to fix first.

Step 5: Pre-test patches, fixes, and workarounds

Patches and workarounds are usually meant for insecure networks and systems, but they apply equally to web applications. If an application is vulnerable, it must be fixed. Patches,

fixes, and workarounds are usually administered by security and IT team members. Fixing actual code requires the work of programmers as guided by the in-house development team. Once a fix is devised, the team must thoroughly test it before re-deployment.

Step 6: Apply patches, fixes, and workarounds

Here's where the repairs are applied to vulnerable web applications and other assets.

Step 7: Rescan to verify patching

After conducting steps for remediation, it's useful to re-scan the web application to ensure it was fixed. This step verifies that the fix worked and that it doesn't cause other applications, network devices, services, or other applications to be exposed to additional vulnerabilities.

Using Tools to Implement the WAS Program

There are many kinds of scanners and other tools for IT security, audit, and operations. Our focus here is the web application security (WAS) scanner. Its fundamental purpose is to automatically examine custom web applications that respond to dynamic web page requests with protocols like hypertext transfer protocol (HTTP). Some scanners use the 'point-and-shoot' method of operation without imposing particular parameters. Others need to be 'trained' or configured for a particular environment. However it works, the scanner should be able to find all vulnerabilities in all web applications with a high degree of accuracy. To miss these, or present false positive or false negative identifications is to leave a web application exposed to exploitation.

An auditor's perspective

As someone who's responsible for network and application security, you're frequently urged to 'think like a hacker' in order to create defenses that provide strong protection. If you're responsible for helping your company comply with various regulations for security, it's likewise helpful to 'think like an auditor.' Here's the auditor's perspective on web application security:

> *I am a Certified Public Accountant at a national CPA firm where I perform Sarbanes-Oxley, Service Organization Controls Report (SOC, formerly called SAS70), Payment Card Industry Data Security Standard, and other types of IT audits. Clients come in different shapes, sizes, and states from a policy controls perspective. The easiest audits for me are when it's clear that the client has a continuous and consistent compliance program. I have some really great clients who generate monthly reports to show that they use tools to monitor their web applications, networks, and systems for compliance, and they make those reports available to me. The reports are hard evidence that a client has the required policies and procedures – and that they actually follow them. From an auditor's perspective, it's clear that organizations like these take compliance seriously and that they employ a systematic process for holding themselves and their IT staff accountable. With these clients, I am typically in and out in a matter of days.*
>
> *And then I find some clients who clearly view my audit as a necessary (but perhaps annoying) once-a-year activity. They do a lot of preparation to try to convince me they're compliant by shoveling in a mound of paperwork, but I know better. The results are simple. As I'm less confident in their control environment, I'm forced to test it with a higher sample rate, and that comes at a higher cost to the client. The extra scrutiny inevitably reveals more problems. Sometimes the end result is a qualified report that the client doesn't like, and I don't like having to issue.*
>
> *Simply using IT policy compliance products for continuous control monitoring of web applications, for example, doesn't automatically make audit issues go away. But use the right tools under the right policy framework, stay consistent, and keep your reports, and you can greatly reduce the amount of money you spend every year in audit!*

Specifying a web application security scanner

It goes without saying that specifications for a web application security scanner vary depending on the needs of each organization. Traditional considerations include whether you should go for an Open Source ('free') solution, or purchase a commercial scanner from an established software company. Another consideration is whether to run a software solution on your own infrastructure, or to rent its functionality with Software-as-a-Service. We'll have more to say in Part IV about the advantages of SaaS. Meanwhile, what else should you look for in a scanner? We answer this question in greater detail in Part III, where we also describe the practical ins and outs of using a web application scanner.

Part III

Using Automated Scanning to Test Web Applications

In This Part

- Learning what to look for in an automated web application scanner
- Understanding the types of scanners and how they work
- Explaining how a scanner finds examples of top vulnerabilities
- Presenting tips on how to tune the scanner for efficient scanning
- Evaluating findings of a scan and using them for effective remediation

You can find web application vulnerabilities across the spectrum of sites on the Internet regardless of whether a site has a few dozen pages or a few hundred thousand; or whether the site is for banking, email, social networking, news, or discussion groups. These vulnerabilities are available for anyone to find, including hackers, researchers, developers, and curious visitors looking for problems in a site.

Manually analyzing a site is one way to find vulnerabilities, but it quickly becomes a slow, tedious process – especially considering the sheer number of websites and the complexity and size of modern web applications.

Web application scanners automate the manual techniques that hackers and security researchers alike employ against websites. They range from simple scripts that may simply search HTML content for useful information, to more complex tools that *spider* (that is, discover and crawl through) a website and catalog its content for further manual analysis. The scanner acts like a hacker, albeit one less antagonistic

towards the web application. As with any technology, it's useful to consider what to look for in choosing the right web application security scanner.

Considering Your Requirements for a Web Application Scanner

What should you look for in a web application scanner? One authoritative framework for answering this question is from the U.S. National Institute of Standards and Technology (NIST) in Special Publication 500-269. This guideline, *Software Assurance Tools: Web Application Security Scanner Functional Specification Version 1.0* provides a functional baseline of expectations. (You can read it in full at `http://samate.nist.gov/docs/webapp_scanner_spec_sp500-269.pdf`.)

As a minimum standard, NIST SP 500-269 suggests that a web application security scanner should:

- Identify specified types of vulnerabilities in a web application.
- Generate a text report indicating an attack for each identified vulnerability.
- Identify false positive results at an acceptably low rate.

The standard suggests the following as optional elements in a scanner tool:

- Produce a report compatible with other tools.
- Allow particular types of weaknesses to be suppressed by the user.
- Use standard names for weakness classes.

The top priority for a scanner is helping you keep on top of the constantly changing universe of web application vulnerabilities. In particular, your scanner must be capable of identifying the top vulnerabilities such as cross-site scripting and SQL injection. Your scanner will provide more value if it incorporates standard industry resources for this information, such as:

- CWE/SANS Top 25 Most Dangerous Software Errors: `www.sans.org/top25-software-errors/`
- Open Web Application Security Project Top Ten Project: `www.owasp.org/index.php/Category:OWASP_Top_Ten_Project`
- Web Application Security Consortium Threat Classification: `http://projects.webappsec.org/w/page/13246978/Threat-Classification`

Aiming Higher: Desirable Features in a Scanner

Elements within the inventory of desirable features in a web application scanner may be weighted differently depending on what your organization requires, and on the available feature set of a particular scanner. Many of the following features are suggested in the *Web Application Security Scanner Evaluation Criteria Version 1.0*, published by the Web Application Security Consortium. (See the document for full details at `http://projects.webappsec.org/Web-Application-Security-Scanner-Evaluation-Criteria`.)

- **Minimal training / ease of use.** This will be on the mind of each IT staffer whose job includes web application security! Look for a solution that doesn't require a doctorate in Web Application Security to get results. Reducing time to learn and use the scanner is more efficient – and lowers operating costs for the organization.
- **Scalability.** Consider if your organization needs bells and whistles, or can make do with a lesser-featured scanner. If your company is small (say, under a hundred people), a stripped-down scanner may be perfect. But if your company is bigger and poised to grow, consider scalability of the scanner. The optimum solution will cost-effectively provide powerful, easy-to-use features for any-sized organization. In Part IV, we explore various options to effectively address this requirement.
- **Accurate discovery.** A scanner must be able to automatically discover and catalog all web applications in your enterprise. A large enterprise can have tens of thousands of web applications, so establishing an accurate

foundation for scanning is essential. Without this capability, your web application security team could easily spend most of its time in the identification phase instead of testing and remediation!

- **Protocol Support.** Time to turn to technical features. Your scanner must support communication protocols used by web applications and devices on or within the network edge. These include network transport protocols such as HTTP and SSL/TLS, plus proxy support, which enables multiple machines to use a shared Internet connection.
- **Authentication.** You want a scanner that supports various authentication schemes such as Basic, Digest, HTTP Negotiate, HTML Form-based, Single Sign On, and Client SSL Certificates. Seek support for custom implementations if required.
- **Session management.** A scanner needs the ability to maintain live, valid sessions for each web application. This is true for web crawling during the discovery phase, and for testing the application by sending it HTTP requests. The scanner needs session management capabilities through the use of tokens for easier authentication of users; this includes support for session management token type, token detection configuration, and token refresh.
- **Crawling.** This spidery term is the process of a scanner browsing through page by page of a website. The scanner needs to support granular configuration so it can efficiently address precise criteria for finding all vulnerabilities on every page. Crawling functionality should include identification of hostnames, support automated form submission, detect error pages and custom 404 responses, support redirects, plus identify and accept cookies and AJAX applications.
- **Parsing.** This process entails mapping the structure of a web application and its functionality. The scanner should support the usual web content types (such as HTML, JavaScript, XML, ActiveX Objects, Java Applets, Flash, and CSS), support character encoding, tolerate partial or disorganized content and customizations, and extract dynamic content with code that executes on individual clients.

- **Testing.** Testing is the meat and potatoes functionality of a scanner. It should enable granular configuration

for precise control over what the scanner tests, such as IPs, URL patterns, file extensions, and so on. Testing capabilities should include authentication, authorization, client-side attacks, command execution, information disclosure, testing customization, and policy management. We present more on this later in the next section, in 'Considering Strategies for Web Application Scanning.'

- **Command and control.** The scanner interface governs the usability and efficiency of this tool. It should allow scheduling automatic scans, pause-and-resume, examination of real-time scanning progress, use of configurable templates, do multiple scans simultaneously, support multiple users and roles, and support remote/distributed scanning. The scanner needs to integrate with your organization's existing vulnerability management systems for fast, smooth remediation. An Application Programming Interface (API) is mandatory for organizations that require customization.

- **Reporting.** Reports are the product of scanning, so they need to be as detailed and customized as your team requires for effective reporting and remediation of web application vulnerabilities. Consider:
    - An **executive summary** is mandatory, because you can use this snapshot to convince senior business managers why what you're doing is useful to the organization.
    - **Technical detail reports** guide programmers and other staffers involved with remediation.
    - **Trend reports** help assess progress over time.
    - **Compliance reports** provide documentation for auditors. As an example, the PCI Data Security Standard mandates in Requirement 6.6 that organizations verify that public-facing web applications are reviewed regularly using either manual or automated vulnerability assessment tools or methods.
    - The **scanner's reports** should automatically provide documentation of compliance.
    - **Advisory reports** help place each vulnerability in context of industry findings.

  Reports also need to support standard human and machine readable formats such as PDF, HTML, and XML.

Configuring and using the scanner, along with extracting results through its reporting functionality are accessed with the scanner's user interface (UI). As an example of a UI, Figure 3-1 shows the main portal for the QualysGuard WAS scanner.

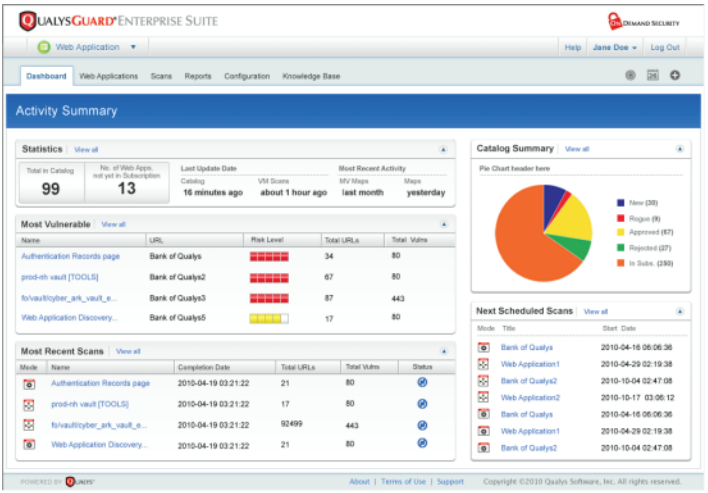


Figure 3-1: Main portal for QualysGuard WAS scanner.

Considering Strategies for Web Application Scanning

Presuming you select a scanner that meets requirements described in the previous section, it's good to know how to use the scanner for maximum efficiency and effectiveness. One of the primary functions of a scanner is emulating what a skilled hacker would do to compromise your web applications. So the rest of this part describes how to use strategies for scanning that would reveal typical attacks used by a hacker.

For example, a basic approach to hacking a web application essentially involves injecting random values into the parameters and fields found in links and forms. The attacker intends to find some string of characters that produces a useful error message or abnormal response.

To illustrate, consider that a site may always expect an email address to contain a single '@' symbol or to never contain double quotes. Or, the site may assume that usernames are always letters and numbers, rather than a mix of symbols that include HTML formatting. Perhaps the site always uses positive numbers to refer to product IDs and behaves strangely in the face of negative values. Any error message or abnormal response provides the clues an attacker needs to refine the randomly injected characters into a working exploit against the site.

Of course, more sophisticated or more focused techniques can skip such generic injection and turn instead to common payloads that target a particular type of vulnerability. For example, one attacker might focus on finding cross-site scripting problems in order to spread malware while another may look for SQL injection problems in order to extract credit card numbers. These steps mirror the basic approach quite closely, but use more carefully selected payloads as opposed to blasting the site with garbage data.

People, while being very good at pattern recognition and understanding the purpose of a website, don't always have to rely on injecting garbage data or hand-crafted payloads into a web application. Attackers are often able to identify vulnerabilities by breaking the application's assumptions of a visitor's behavior or leveraging developers' mistakes in enforcing authorization and other controls.

Rather than requiring someone to click through the web application and inject bits of garbage data here and there, an automated scanner enables the web application's developers to consistently and exhaustively reproduce many kinds of attacks that hackers use. In Figure 3-2, you can see the process a hacker would follow, from initial probe to a successful exploit.

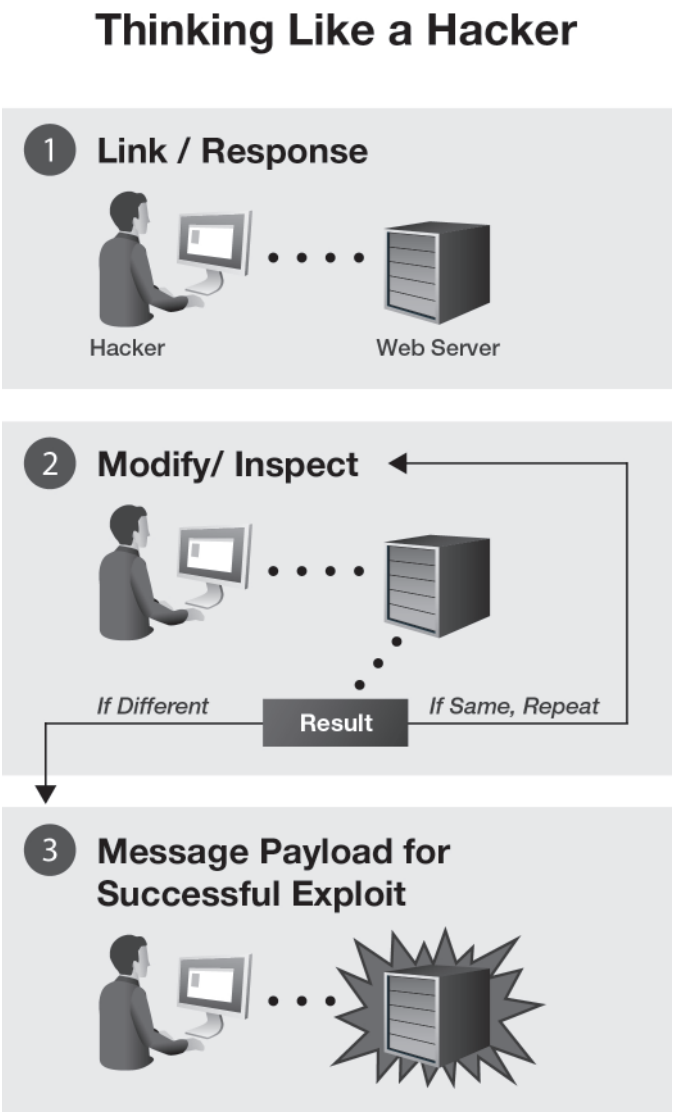


Figure 3-2: Thinking like a hacker for effective scanning.

Manual testing by an expert will always be the most comprehensive and accurate way to determine a site's security, but even the most accomplished security testers incorporate tools into their work.

Why some manual testing is always necessary

Automation provides benefits for many situations. For web scanning, it enables better scalability and repeatability for addressing web application security. However, websites implement enormous technological complexity in order to be simple for users. At their core, websites are a mixture of HTML, JavaScript, and server-side programs. This potpourri of technology creates a huge array of design patterns and implementations.

People, rather than machines, excel at understanding and dealing with the complexity of these sites. Someone who understands web security will be able to identify risks, determine possible threats, and analyze how a site might be impacted by such things.

More importantly, people understand the logic behind a website and the expectations that underpin how you use its features. For example, a visitor to an e-commerce site quickly figures out how to find a widget, put it into their shopping cart, head for the checkout page, and purchase the item. Someone testing the site's security will enumerate the different ways that process might break down – perhaps one of the checkout steps can be skipped, or someone else's order can be redirected to a different address, or a discount coupon can be applied multiple times.

These types of vulnerabilities, often described as 'business logic' flaws, pose a particularly difficult challenge for automated scanners. Whereas scanners do well in gathering links and randomly (or intelligently, depending on the tool) tweaking values just to see what happens, they fare poorly against identifying multi-step workflows and understanding the logic, or true functionality, behind a web page.

Scanners are vital but remember this basic security principle: no single tool is a magic panacea for protecting a website.

The Three Colors of Scanning

The three major approaches towards testing the security of a web application are often referred to as *black box*, *white box* and *gray box* scanning. Figure 3-3 compares black and white box scanning.

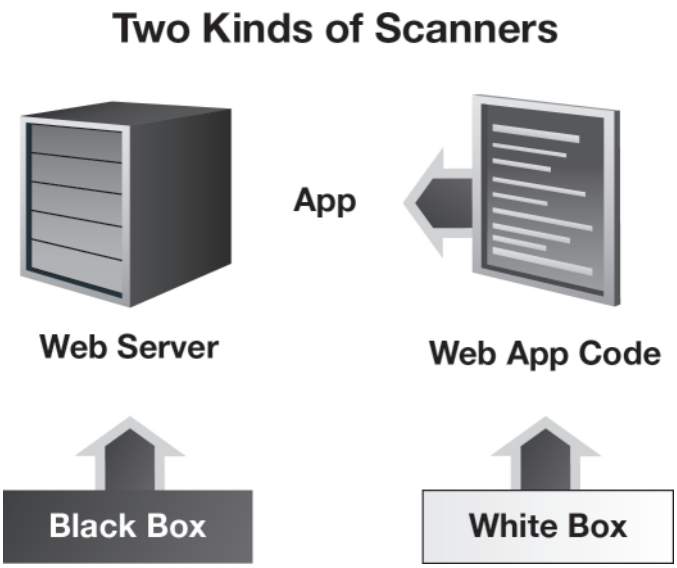


Figure 3-3: White box versus black box scanning.

- **Black box scanning** looks at the application in its deployed state, just as visitors of the site would see it. These types of tools interact with the site over HTTP or HTTPS and have a view of the final, rendered source code that shows up in a visitor's browser. The advantage of black box scanning is that the tool sees the site in a complete, finished state. All of the complex components – from the database, to application code, to HTML and JavaScript – are working together, which means testing should give an accurate view of the completed site's security during normal operations. This is the main type of scanner we focus on in this part.
- **White box scanning** tools review the source code that runs a website. Consequently, these tools have better visibility into areas where input validation is missing or cross-site scripting may occur. White box scanning can pinpoint vulnerabilities to their exact line of code.
- **Gray box scanning**, as you might guess, represents any hybridization of the previous approaches. The scanning tool, or tools, combine information gleaned from

the finished website as well as its source code. Gray box scanning can be more time-consuming and require more coordination of tools and results. But the results provide advantages of both black box and white box testing.

Showing a Black Box Scanner at Work: Three Top Application Vulnerabilities

In their simplest form, black box web application scanners crawl the links and forms on a website, then resubmit contents of those URL and form requests with modified values. The scanner changes the values in links, form fields, and even HTTP headers in order to elicit an error from the website or cause the site to react in a way its developers didn't intend. The result may bypass a defense or otherwise break the site's security.

The scanner must collect links and forms in order to find potential attack vectors. Providing a thorough analysis is essential because all web applications have to deal with an untrustworthy end-point: the browser. A site's developers have full control over the code and behavior of the web server. But they're forced to trust the browser to display content and submit requests in the way the site is designed to work. There's nothing inherent to HTTP or HTML (the building blocks of the web) that prevents the user from changing a browser's behavior or requests.

The web application scanner analyzes a multitude of potential attack vectors with a variety of tests. Its purpose is determining how well the website validates the data it receives, whether it effectively manages a user's access to data, and whether a malicious user can leverage the site to attack others.

To see how this occurs, the following sections explain three of the top vulnerabilities for web applications – cross-site scripting, SQL injection, and information disclosure – and describe exactly how a scanner finds points of potential trouble.

Cross-site scripting

A web application has two fundamental elements: the software that runs on your site's servers, and the HTML that displays in a visitor's browser. The browser is required for visitors to see and interact with your site. The browser does this by executing the mixture of HTML and JavaScript it receives from a website in much the same way a binary program is launched on the desktop.

Attackers take advantage of the browser's execution environment with exploits like cross-site scripting (XSS). Web applications often customize their content for each visitor. For example, many sites greet a user who's just logged in with a brief message somewhere in the page:

```
<div id="first_name"><b>Welcome, Mike!<b></
div>
```

Notice that the name, Mike, is mixed in directly with the page's HTML. The site has put this content together on the fly by inserting the user's first name. If this mixture of HTML and data isn't handled carefully, an attacker can directly influence the browser's actions. For example, what if the first name happened to contain the <script> tags used to execute JavaScript?

```
<div id="first_name"><b>Welcome,
Mike<script>evil_function()</script>!<b></
div>
```

In the preceding example, the insides of evil_function( ) are left to your imagination, but real-world attacks have included everything from adding followers on social networking sites, to launching phishing attacks, to downloading malware.

Cross-site scripting attacks don't have to cross sites or necessarily use JavaScript to be effective, but the term 'cross-site scripting' has been commonly used in a generic sense for more than a decade. A scanner looks for this type of problem by injecting HTML and JavaScript payloads into links, forms, and even cookies, and then watches the web application for signs of subversion to its regular HTML content.

SQL injection

Cross-site scripting occurs in the browser through security failures in a website's server-side code and HTML content. A different set of vulnerabilities called *SQL injection* result from problems in the site's server-side code.

Databases are the central players here, for they store the information that sites need to display customized content, and store the comments, orders, and profiles of its visitors. An attack using SQL injection piggybacks malicious database statements onto values otherwise used by the site to make legitimate queries. For example, a website may use an 'id' parameter to track products in its catalog:

```
http://web.site/products.page?id=19
```

Just like an attacker, a scanner looks for SQL injection problems by adding special characters to the 'id' value. One of the usual characters is a single quote because it serves as a delimiter for values, usually strings or sequences of characters, within SQL statements. Your site is in serious trouble if an attacker can change the syntax of a SQL statement, perhaps by changing a record lookup to grab data from another table. Other avenues of attack leverage SQL injection to execute commands on the database's server.

The first step towards determining if a web application is vulnerable to SQL injection is watching for specific error messages when link parameters and form fields have special characters or garbage data thrown into them. In the worst case, a website will serve the same verbose error message only seen by a developer. Ideally, the web application should return no detailed information when reacting to attempts at SQL injection. Remember, information is power, so providing less of it to an attacker helps keep your site more secure.

Information disclosure

Not all vulnerabilities lead immediately to compromise, but you can't be complacent about the danger of a breach. Some vulnerabilities, such as information disclosure problems, can

be stepping stones to bigger attacks or to leak sensitive information. As an example, consider a link with a single parameter that displays a news article:

```
http://web.site/news.page?id=19
```

Cross-site scripting and SQL injection can target the parameters of the link, but that's not the only possible attack vector. A scanner may also modify the link itself, such as changing '`news.page`' to '`news.page.bak`' or '`news.page.old`' to see if a developer accidentally left a backup of the file on the server. Next, the scanner could try modifying cookie values or some other headers to see how the website responds. Scanners very often take a shotgun approach towards finding flaws in a website: spray it with many different tests and see which ones break a security assumption.

The shotgun approach requires balancing the intrusiveness of a scan. Some scanners strive to automate the process of exploiting a vulnerability after it's found. Other scanners focus more on identification rather than exploitation. Be aware that scanners may address different needs and also carry different consequences. Some exploits could adversely impact a production website, while some vulnerabilities may require active exploitation in order to adequately determine their risk. Addressing these optional requirements entails adjusting coverage of the scanner before you launch a scan.

Ensuring Adequate Test Coverage With a Scanner

Testing the security of all of a website's functionality is the main goal of a web security program. Web application scanners are integral to this goal so it's important to understand the capabilities and scope of a web scanner. Some scanners may be single-purpose tools for identifying weak passwords or common files and directories. Other scanners may provide more in-depth analysis of a website. In either case, you'll want to know what areas of the site are tested to enable follow-up work – either with a different tool or by a manual process.

Websites range in size from dozens to tens of thousands of links. A web application scanner would be hard-pressed to complete an exhaustive scan of a website that has thousands of links if every combination of link, parameter, and form field were to be tested with every one of the scanner's payloads. While no one wants to miss a vulnerability, no one can afford to wait days to complete a scan either.

Test coverage is hard to measure, but keep in mind that you still attain good coverage without requiring the scanner to visit every link on a site. In many cases, perhaps even the majority of cases, links with similar pages and parameters are exercising the same program flows on the server. It may be sufficient to request only a handful of similar links in order to reach coverage of the site's functionality even if many links aren't touched. Here's where the benefit of risk-based sampling can provide you with acceleration of the scanning process.

No one would have the time or patience to test every link on a site with 10,000 pages. Nevertheless, you could still produce an accurate report of the site's problems. This is because people, as we mentioned earlier, are good at finding patterns. You can make educated guesses whether one link is hitting the same code path as another. A scanner can also take this approach, though perhaps more clumsily than a person would. The scanner can't finish an exhaustive scan within a reasonable time, so it spreads out its checks in order to reach a balance of accurately measuring the site's risk with the time required to obtain that measurement.

As you become more familiar with a web application scanning tool, strive to get a sense of the types of vulnerabilities it finds as well as the ones it misses. As with all security tools, use a scanner to complement your broader vulnerability management effort. Don't rely on a scan that reports zero vulnerabilities to provide 100 per cent assurance that no vulnerabilities exist!

Web application scanners can't find every type of vulnerability. Scanners are an important component of a security program, but not the only component. Knowing where the scanner works well and where it doesn't helps narrow the focus of more time-intensive testing.

Understanding Why Developers Must Help with Vulnerability Remediation

Simply finding vulnerabilities is but the first step towards securing web applications. The critical challenge is translating scan results into effective remediation of the vulnerabilities. Achieving this result isn't always a simple process. Although web applications share common foundations of server technologies, programming languages, and databases, they often bear little resemblance to each other due to developers' styles, design patterns, engineering decisions, and more. There's no easy fix because, unlike host-based security problems, it's rare when you can download and apply a patch to repair a programming flaw in a custom web application.

Organizations that create their own custom web applications have nowhere to turn for patches other than their own developers. Organizations that incorporate Open Source applications or contract third-party developers to create websites face the same problem with developing patches, although the Open Source web applications may have developers working independently on fixes. But because they're often coding for free, your expectation of timely, effective patches is, at best, misplaced and likely to result in ongoing risk of a breach.

Regardless of where a web application originates, many organizations face the significant problem of operating applications on sites that become *orphans*. This happens when the original developers leave, move on to different applications, or any other way that the site loses someone to be in charge of its maintenance. These applications will likely need custom programming to fix a vulnerability. Unfortunately, documentation may be poor to non-existent. As a result, hands-on knowledge and understanding of its source code may be lost forever.

The problem amplifies when a vulnerable application runs on multiple websites. For example, a cross-site scripting vulnerability can hit several different websites with a single payload. The fundamental techniques for blocking XSS are

well understood, but implementing the fixes changes from site to site depending on how it's coded. This is where it's necessary to engage your developers in the web security process. The scanner provides an external view of the website by processing its public links and parameters. Your developers are required to see how each parameter's value moves through the source code.

Seeing Past the Bug

Resist the temptation to focus only on fixing a vulnerability identified by the scanner. The critical requirement is to remediate the *systemic cause* of the vulnerability. For example, a simple approach to an SQL injection vulnerability might be to blacklist a single quote character or the word SELECT because it was used as part of the scanner's test payload. While this might improve security from the scanner's perspective (the scanner's payloads no longer work), it unfortunately papers over the real problem. A better approach is to analyze *how* the parameter was misused, apply recommended countermeasures (such as using prepared statements in the code), and then search the source code for similar patterns. A single vulnerability found by a scanner may point to a more endemic problem just waiting to be discovered by a malicious attacker.

Remember that web vulnerabilities are essentially programming errors – bugs with potentially nasty consequences for the site's data or users. Bugs will forever be a part of the programming process, which is precisely why savvy organizations build secure development processes for tracking, fixing, and verifying them. Security bugs should be no different (though perhaps have a higher priority).

Security vulnerabilities, like other bugs, can be time-consuming to fix. Nevertheless, you can create temporary workarounds either in code or with other tools such as a web application firewall. Once the vulnerability is fixed and verified, we recommend you add regression tests to the development process to ensure the vulnerability doesn't resurface later. As always, using automated tools whenever you can will help to speed the application vulnerability management process.

Part IV

Introducing QualysGuard WAS

In This Part

- Understanding what QualysGuard WAS can do for you
- Automating the website crawl for a thorough, efficient scan
- Testing web applications for an accurate ID of vulnerabilities
- Scaling the solution scan websites of any complexity

The QualysGuard vulnerability management platform pioneered cloud-based identification and management of vulnerabilities in the network and network-attached hosts. The Web Application Scanning (WAS) Service extends the QualysGuard platform by also securing your web applications.

Being in the cloud makes QualysGuard WAS a Software-as-a-Service (SaaS). This means instead of purchasing software, WAS – like the rest of the QualysGuard platform – is purchased like a utility, used whenever you need it with a standard web browser. Through this secure interface, you obtain centralized access to scanning and web application vulnerability management.

QualysGuard WAS automates most of the web scanning process. You don't need any prior knowledge about a website in order to use QualysGuard WAS. For example, the scanner automatically discovers and catalogs all web applications in your enterprise. It automatically knows how to define common behaviors related to authentication or custom error pages in your site's applications. In addition to saving you the burden of

manually executing vulnerability tests, automation ensures that scanning provides a consistent level of accuracy – regardless of your web security experience.

This part describes the functionality of QualysGuard WAS and explains how it keeps your web applications secure.

Crawling Your Custom Web Applications

Accurate, comprehensive crawling is a fundamental requirement for a web application scanner. The closer a crawler emulates how visitors use a browser to interact with the site, the better the scanner's ability to find all the vulnerabilities that could result in a breach.

Obtaining good link coverage

QualysGuard WAS uses a crawling strategy that emphasizes coverage of the site's functionality as opposed to testing every redundant link. WAS takes this into account by balancing the number of links it requests based on similarities among each link's page, parameter names, and parameter values. In this manner, WAS saves time by not requesting excessive links within a narrow section of the target application. It's a risk-based strategy that experts consider essential for real-world implementation of effective vulnerability management.

QualysGuard WAS tests for vulnerabilities across all kinds of operating systems and programs. When a vulnerability check is added to the platform, it receives an assigned number called a Qualys ID (or QID). Manually adding QIDs to a scan turns on the relevant checks for related vulnerabilities. Excluding a QID prevents the scanner from checking its related vulnerabilities. For example, QID 150009 lists the links crawled during a scan. QualysGuard WAS might discover thousands and thousands of links on the site, but only the ones it actually requested make it onto this list. Selectively using QIDs can significantly improve scanning performance.

Good reasons for throttling the crawler's scan rate

There are many reasons to adjust the bandwidth used during a web application scan. One factor is the web application's processing capacity: can it handle dozens of concurrent requests without crashing? And equally important: will the scan slow a site's responsiveness for visitors?

It's rare for a scan to overwhelm modern web applications. But if there's any question about site response during a scan, the QualysGuard WAS crawler provides you with five performance settings that affect the number of concurrent requests it will make, and whether any delays will be added between requests.

QualysGuard WAS automatically adjusts its request speed if the server's average response time passes a certain threshold or experiences a sharp spike during a scan.

The QualysGuard WAS scanner monitors connection timeouts and server errors, such as HTTP 500 messages. If a scan produces too many timeouts, the scanner automatically cancels the crawl. This capability prevents a scan from unintentionally causing a denial of service to the website. It also helps to ensure accurate scans. For example, during a scan, a site may go into an error state that causes pages to respond differently, or not at all, which can hide vulnerabilities that would otherwise be found when site processing isn't in a state of overload.

Restricting the crawler to specified designations

QualysGuard WAS crawls your site very much like a normal visitor would with a browser. Crawling includes areas of the site that rely on JavaScript to create dynamic pages. While the crawler emulates a visitor's browser as much as possible, it does have restrictions:

- By default, it won't request links that point to a host name or IP address other than the target website. Exceptions must be explicitly defined by the user.
- Form submissions can be disabled.
- You can instruct WAS to conform to permissions defined in a robots.txt file.

In some cases, a scan may require using more granular controls. With QualysGuard WAS, you may restrict areas the crawler explores with exclusion lists which define specific links that may or may not be requested during the scan:

- **Blacklists** define links or patterns that prohibit requests by the scanner.
- **Whitelists** define exceptions to blacklist entries. This makes it convenient to create blacklists where an entire directory is blocked, such as `/admin/.*`, but permit particular links to bypass the blacklist, such as `/admin/login.cgi`.

Exclusion lists with QualysGuard WAS can contain explicit links or regular expressions. When a list is defined with links, each entry should be the absolute uniform resource identifier (URI) such as:

```
http://site/dir/send_email.cgi
http://site/feedback.cgi?p=723
```

Regular expressions offer a more powerful approach to defining exclusion lists because a single entry can match groups of links based on complex patterns. Here are examples showing two types of patterns:

```
send_email\.cgi
feedback\.cgi\?p=.*
```

WAS uses the Perl-compatible regular expression (PCRE) syntax for these entries. Take care to ensure they're well-formed and their syntax conforms to the types of links you expect to match them.

Authenticated scanning

Some web applications require authenticated access to the majority of their functionality. With QualysGuard WAS, authenticated scanning can be configured for two different situations: server-based authentication (HTTP Basic, Digest, NTLM, or SSL client certificates) and HTML forms (think of login pages for email or banking sites). Server- and form-based authentication may be combined.

Configuring scans to use HTML form-based authentication is simple – merely supply a valid username and password. Throughout the crawl, QualysGuard WAS watches for login forms. When encountered, it applies some logic to complete the appropriate fields required for processing form. All this is done automatically behind the scenes, including detailed profiling to ensure the scanner stays logged in once it finds the login page.

Scanning your web applications may trigger exceptions requiring more advanced configuration. Some web applications use complex login forms that prevent WAS from completing a proper profile. In these situations, you may need to create a customized authentication record. The customized authentication record uses the name/value pairs of each input field within a form.

Make sure the credentials supplied to QualysGuard WAS are valid! The reason a scan fails to authenticate to a website may be as simple as a typo in the username or password.

Passive analysis of the site

Website vulnerabilities, or precursors to vulnerabilities, don't always require active testing by injection or customized payloads. QualysGuard WAS uses active tests to identify well-known problems like SQL injection and cross-site scripting. It also includes steps to passively examine the content served by a website during normal use.

QualysGuard WAS watches the traffic sent back and forth to the website during the crawl phase in order to spot issues that might affect the privacy or security of the site's visitors. Consequently, it's one of the least intrusive methods of identifying potential problems.

Certain passive analysis tests address how well the site follows best practices for handling cookies and forms securely, such as the use of SSL or cookie attributes.

WAS may also be configured to scrape the site's HTML for unmasked credit card or Social Security numbers. The scanner takes care to identify only values that match expected patterns so as to avoid producing false positives. This capability can be useful for audits, such as for the Payment Card Industry Data Security Standard (PCI DSS) or other regulations addressing security of personally identifiable information. (For more information on compliance, read our sister publications, *PCI Compliance For Dummies* and *IT Policy Compliance For Dummies*.)

In addition to search patterns that are predefined in QualysGuard WAS, you can instruct the scanner to search for a pattern defined by you. These custom pattern formats use the same Perl-compatible regular expression (PCRE) engine used by the exclusion lists mentioned earlier in this part. The customized pattern feature lets you search for specific words or phrases important to your organization or environment. For example, the scanner could highlight pages that contain developer names or phrases like 'Proprietary and Confidential.' This data-loss detection capability can also provide useful documentation for policy compliance auditors.

Application Security Testing for Vulnerabilities

QualysGuard WAS takes the information collected during the crawl phase to create an appropriate set of tests for execution against your website. These tests identify common

vulnerabilities in the site. As the scanner runs these tests, it reviews content of site's responses in order to determine the presence of a particular vulnerability.

QualysGuard WAS also uses the crawler's profiling data to tune the vulnerability tests. A significant number of false positives are reduced by the scanner's ability to determine the differences between default landing pages for a directory, custom 404 pages, and custom error pages used by the website. In this way, the scanner doesn't have to rely solely on signatures to correctly report security problems.

Minimizing the intrusiveness of web application testing

A common concern among users of web application scanners is whether the scanner is safe or non-intrusive. This is an important consideration for scanning a website. The last thing you want is for a scan to cause downtime for a production system – and a direct hit to revenue.

It's impossible for any scanner to provide 100 per cent assurance that a crawl or other tests conducted against the site won't cause an adverse result. However, special technology in QualysGuard WAS minimizes its effect on a site and reduces the intrusiveness of its tests to the maximum extent possible.

The security tests used by QualysGuard WAS use the minimum effort necessary for identifying vulnerabilities. Some scanners attempt to test everything, and execute techniques aimed to fully exploit vulnerabilities; but this is often a wasted effort. For example, the SQL injection tests attempt to generate errors in the web application, but they don't go so far as to create temporary tables, extract information from a database, or otherwise modify the database.

With QualysGuard WAS, we suggest you use exclusion lists to prevent the scanner from requesting links that produce unwanted side-effects. For example, some forms may generate emails each time the form is submitted. The WAS security

tests may require hundreds or thousands of visits to the form, with each visit producing an email. Trust us on this: you don't want to receive thousands of emails resulting from a web application security scan!

Evaluating the results achieved with scanning

The QualysGuard WAS scanner tests for vulnerabilities across several potential attack vectors:

- Files, directories, and parameters referenced in a link.
- Form input fields.
- Cookies.
- Common HTTP headers.

The payloads for each test are designed to find errors, misconfigurations, and coding mistakes in the web application that could reveal sensitive information or lead to a compromise of site security.

Two of the most common vulnerabilities in web applications are SQL injection and cross-site scripting. QualysGuard WAS includes technology that makes tests for these, and other vulnerabilities both comprehensive and accurate.

Not every problem discovered by QualysGuard WAS is immediately exploitable. The scanner also reports findings to help the site's developers restrict the amount of information about the site's architecture, protect users' privacy, and maintain recommended security practices. Some examples include:

- Information disclosure such as web pages that display references to source code line numbers, error messages that reveal software or version numbers, or other items an attacker may use to exploit your website.
- SSL, or lack thereof, as applied to login forms, cookies, and actions that may affect the security of the site or privacy of its visitors.

The test phase typically uses 90 per cent or more of the time required to complete a full scan. As mentioned above, you can reduce the scan time by including fewer Qualys IDs. For example, you could run one scan that just looks for XSS-related QIDs. Running fewer tests significantly improves scan times. However, network latency and poor server responsiveness also contribute to longer scans. Unfortunately, there's nothing WAS can do to reduce the effect of those on a scan.

Reducing false positives

A web scanner's accuracy directly leads to its usefulness. If you can't trust the results of a scan or spend too much time confirming results that seem questionable, then the benefits of automation are greatly diminished.

False positives are vulnerabilities that don't exist in the web application, but were reported (incorrectly) by the scanner. False positives happen for a variety of reasons, most typically because the scanner's detection technique was insufficient.

QualysGuard WAS collects custom profiling data about the target website. This data helps reduce many types of false positives. QualysGuard WAS also reduces false positives by combining detections based on patterns and error messages with more behavior-oriented techniques that compare similarities across a group of responses.

A good example of behavior-based analysis is the Blind SQL Injection module. With Classic SQL Injection, a scanner finds vulnerabilities by looking for error messages generated by payloads that corrupt a database query. The Blind SQL Injection module generates a group of tests intended to elicit different responses from the site. The scanner looks for consistent responses to each request to determine if the database can be manipulated, but these responses don't need to contain specific error messages.

If the scanner misses something – after all, web applications differ immensely in design patterns and complexity – then it's often possible to update the scanner's detection via a

QualysGuard WAS *Vulnsigs* update. The Vulnsigs releases contain payloads and signatures that WAS uses to create security tests. These updates have quick release cycles, often within a day or two of vulnerability's initial identification.

We mentioned previously that since QualysGuard WAS is a Software-as-a-Service, new versions of WAS are released quickly – and from a customer's viewpoint, automatically and without any requirement for user intervention. In other words, you don't have to do anything! As an added benefit, the changes that go into fixing a false positive for one customer are immediately available to all QualysGuard WAS customers, regardless of whether or not their scans have experienced the same false positive identification.

Addressing false negatives

False negatives represent actual vulnerabilities that a scanner failed to detect. They occur for a variety of reasons. In some cases, the scanner didn't find the link to be tested, didn't implement the test necessary to find the vulnerability, or the scanner's detection technique wasn't robust.

Missed vulnerabilities are always a concern. The QualysGuard WAS development team continually monitors feedback from customers and creates new test cases to ensure that WAS security tests work as intended. Qualys addresses missed vulnerabilities just as it fixes false positives. The underlying cause of the missed vulnerability determines if a Vulnsigs update, patch release, or new version is necessary to address the issue.

Managing the Web Application Scanning Process

QualysGuard WAS provides workflows for managing vulnerabilities within a single website, as well as the larger view of multiple sites across a network.

At the individual level, web application scans can be launched on an ad-hoc basis or run automatically on a predetermined schedule. As shown in Figure 4-1, the results of each scan are presented by vulnerability and link to provide you with different views into the status of your site's security.

At a broader level, you can use QualysGuard WAS to automatically scan network IP ranges that designate target web servers. Discovering web servers automatically makes it easier to catalog the websites on a network. Plus, it makes finding rogue websites much easier. The QualysGuard Vulnerability Management service examines the servers for known vulnerabilities or missing patches. The associated QualysGuard WAS service enables users to go one step further into determining what vulnerabilities the site itself may have. Figure 4-2 shows a QualysGuard WAS discovery report, which provides identification and details of every web application found on a site scan.

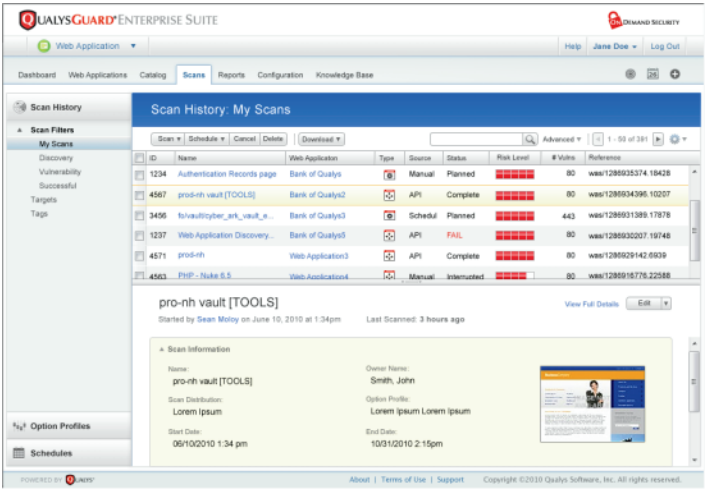


Figure 4-1: Results of a QualysGuard WAS vulnerability scan.

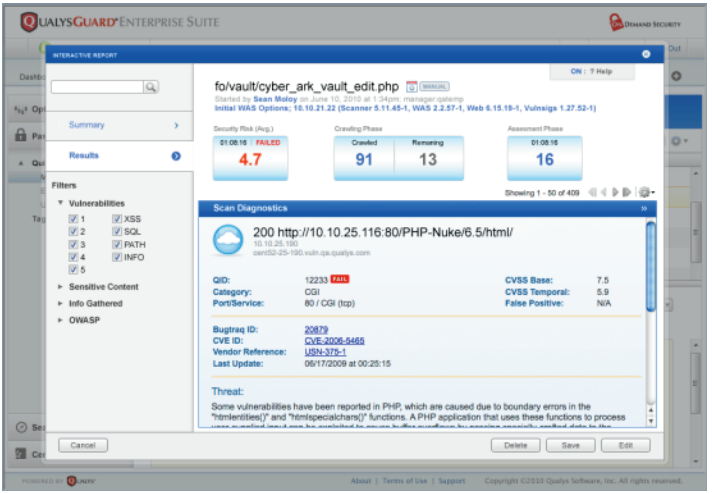


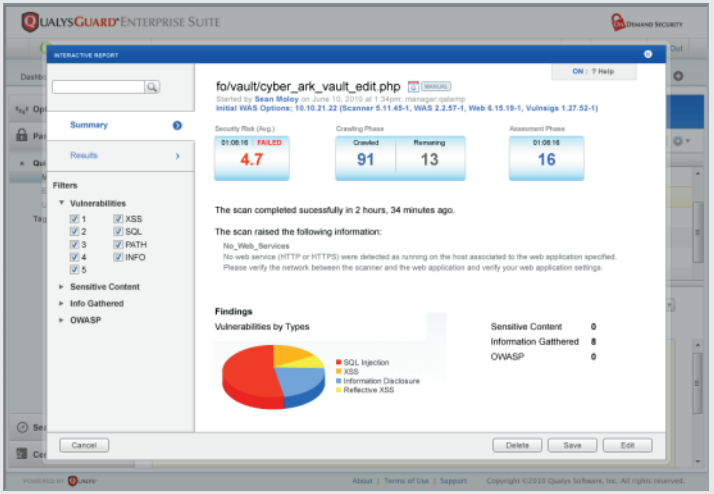
Figure 4-2: Results of a QualysGuard WAS discovery report.

Navigating application security with the QualysGuard WAS user interface

QualysGuard WAS presents web application vulnerability information with an easy-to-use user interface (UI) for standard browsers. This centralized command center makes it just as easy to deal with multiple web targets as it does for a single website. The UI provides easy navigation of WAS features, plus it integrates and clearly displays analytical and testing results from massive amounts of vulnerability scan data. A new tagging system streamlines web application vulnerability management processes executed by your security team. Template-based reporting provides the additional benefits of allowing you to instantly view results in standard, popular formats – or customize views as required by your company's needs. The UI also allows you to export any report in a wide range of outputs including DOC, XLS, PDF, XML, and PPT. To encourage flexibility, you may also access features in the user interface via the QualysGuard WAS Application Programming Interface.

The QualysGuard WAS user interface makes it easy to know in an instant where your website stands

on security. As you can see in the screenshot below, the UI displays a concise summary of threats by vulnerability type. For purposes of PCI DSS compliance, it also identifies vulnerabilities based on the industry standard Common Vulnerability Scoring System. CVSS, as it's called, ranks everything by High, Medium, and Low severity. With QualysGuard WAS, your application security team will always have a clear path for remediating the most important vulnerabilities first on your website.



Part V

Ten Tips for Securing Web Applications

In This Part

- Understanding web application scanning
- Protecting your website
- Improving scanning and remediation with automation

With the importance of web applications in running a business, protecting your website is an essential fact of life. The tips in this part can serve as a handy checklist to help you improve the security of your site's web applications. These ten tips can help you determine what you need to do for scanning and fixing vulnerabilities, so that you can proceed with confidence!

Read This Book

If you're reading this part first, consider that (in our humble opinion) there's no quicker way for an organization to get the gist of protecting web applications than to read the whole of *Web Application Security For Dummies*. This book describes what it is, how to prepare, and how to tap the benefits of automation in the ongoing process of protecting web applications. Start between these covers – you won't be sorry!

Acknowledge the Critical Need for Web Application Security

In Part I we explain why there's a universal need for stronger web application security. Like everyone else, your organization is probably using more web applications than it thinks – many of which are likely to provide crucial support to primary business processes or access to sensitive information. Those applications are prime targets for exploitation – particularly if they contain vulnerabilities.

Research shows the threat of exploitation is growing because vulnerabilities are everywhere. But more to the point: damage is deadly, as breaches caused by hacking and malware represent the biggest share of total records compromised. Unfortunately, many organizations devote few resources to finding and fixing web application vulnerabilities. Your organization's first step in fixing this problem is acknowledging the threat and committing to place a higher priority on eradicating website vulnerabilities. Get senior managers involved because you may need their blessing to shift priorities for web application security.

Measure Potential Fallout of a Breach to Justify Your Program

Naturally, it's useful for justifying your web security program to assess exactly how much is at risk if there's a breach through your organization's web applications. Evaluate potential fallout and assign monetary values to determine overall risk. Consider what you have to lose if:

- Customers take their business elsewhere.
- Sales decline.
- Fewer people use your online store for fear of a breach.
- Your company is fined for non-compliance.

- Your company is successfully sued for legal settlements or judgments.
- Your company loses the ability to accept payment cards in return for goods or services.

A smaller company could be wiped out.

Establish a Web Application Security Program

Achieving web application security entails more than scanning. A successful venture requires a *program* that addresses everything it takes to develop, deploy, and maintain secure web applications. Start by obtaining the official appointment of a company employee to be in charge of web application security. The industry's Software Development Lifecycle is a good program model with three phases: secure development, secure deployment, and secure operations. With a comprehensive security program, you can position web application security efforts with other processes required for enterprise vulnerability management.

Add Web Application Scanning to Vulnerability Management

Vulnerability management has long been a mainstream term among IT professionals for controlling risks – particularly to an organization's network and attached hosts. The seven best practices or steps to web vulnerability management (described in Part II) are a perfect structure for finding and fixing vulnerabilities in web applications. These steps begin with automated discovery and cataloging of all web applications in your enterprise, and end with re-scanning to verify remediation of vulnerabilities. Supplementing this process with specialized tools, such as a web application vulnerability scanner can automate your efforts to speed identification and fix trouble points on your website.

Weigh Criteria for Choosing the Right Website Scanner

Web application scanners automate the manual techniques that hackers and criminals employ against websites. Scanners search HTML content, and spider through a website, cataloging its content for specific vulnerability tests and for further manual analysis. Part III notes eleven desirable features in a web application scanner. Some will be more or less desirable to your organization depending on its unique requirements. We suggest you play close attention to these in choosing a scanner. Be sure to note the differences between white box, black box, and gray box scanners as their capabilities directly affect how you're able to crawl a website and test its links and forms for security problems.

Let Strategy Guide Scanning for Maximum Efficiency

It makes sense to adjust a scanner's settings in a manner that closely mimics how an attacker might attempt to breach web applications on your company's site. Controlling how a scanner injects random values and parameters into links and forms will provide vital clues, via resulting error messages or abnormal responses. A hacker can use this information to breach your site. You can use this same information to fix those vulnerabilities and protect the site. Strategic scanning – particularly how you tune a scanner's operation – also helps improve operational efficiency and minimizes degradation to normal performance of the website.

Some manual testing is always required to uncover flaws in business logic underpinning your site's web applications.

Be Aware of Top Vulnerabilities and Scanner Capabilities

Web application vulnerabilities are constantly changing, so you need to keep abreast of the top challenges to website security. In particular, your scanner must be capable of identifying the top vulnerabilities such as cross-site scripting and SQL injection. Consult industry resources for this information, such as:

- CWE/SANS Top 25 Most Dangerous Software Errors: `www.sans.org/top25-software-errors/`
- Open Web Application Security Project Top Ten Project: `www.owasp.org/index.php/Category:OWASP_Top_Ten_Project`
- Web Application Security Consortium Threat Classification: `http://projects.webappsec.org/w/page/13246978/Threat-Classification`

If you use a cloud-based scanner such as QualysGuard WAS, incorporation of tests for identifying the top vulnerabilities is done for you automatically.

Use Automation to Analyze Results and for Remediation Workflow

The critical challenge for securing web applications is translating scan results into effective remediation of the vulnerabilities. Your scanner should automate the bulk of vulnerability discovery and analysis. Many people tasked with responsibility for website security are more versed in network security than coding and debugging web applications. For this reason, automation can help ease the burden and allow non-specialists to isolate the bulk of issues before turning web application vulnerability remediation over to the programmers.

Try QualysGuard WAS as Your Web Scanning Solution

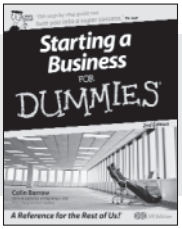
Last but by no means least, we'd be remiss without inviting you to a free trial of QualysGuard WAS. As a cloud-based SaaS solution, QualysGuard WAS can be deployed immediately and provides rapid assistance in finding and fixing vulnerabilities in your web applications. QualysGuard WAS is fully integrated with other services in the QualysGuard vulnerability management platform. If you already use QualysGuard for network security VM or other compliance initiatives, adding WAS is a simple evolution in security management. And if you've never tried Qualys before, now's a prime opportunity to get control of securing applications on your website!

FOR DUMMIES®

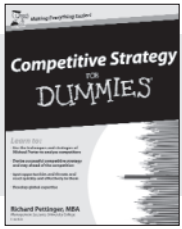
Making Everything Easier!™

UK editions

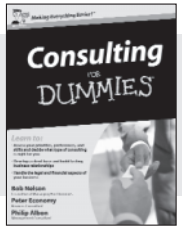
BUSINESS



978-0-470-51806-9



978-0-470-77930-9



978-0-470-71382-2

Body Language For Dummies
978-0-470-51291-3

British Sign Language For Dummies
978-0-470-69477-0

Business NLP For Dummies
978-0-470-69757-3

Cricket For Dummies
978-0-470-03454-5

Digital Marketing For Dummies
978-0-470-05793-3

Divorce For Dummies, 2nd Edition
978-0-470-74128-3

eBay.co.uk Business All-in-One For Dummies
978-0-470-72125-4

English Grammar For Dummies
978-0-470-05752-0

Fertility & Infertility For Dummies
978-0-470-05750-6

Flirting For Dummies
978-0-470-74259-4

Golf For Dummies
978-0-470-01811-8

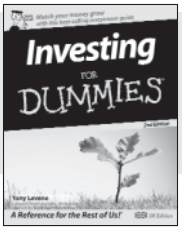
Green Living For Dummies
978-0-470-06038-4

Hypnotherapy For Dummies
978-0-470-01930-6

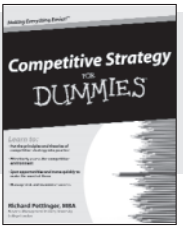
Inventing For Dummies
978-0-470-51996-7

Lean Six Sigma For Dummies
978-0-470-75626-3

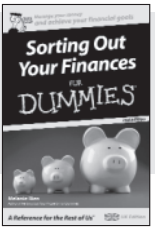
FINANCE



978-0-470-99280-7

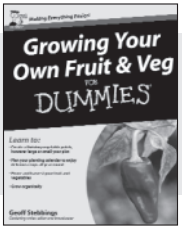


978-0-470-77930-9

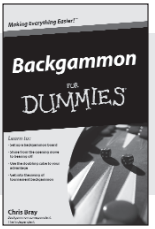


978-0-470-69515-9

HOBBIES



978-0-470-69960-7



978-0-470-77085-6

978-0-470-75857-1

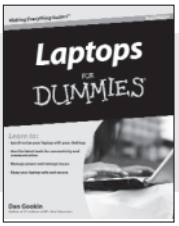
Available wherever books are sold. For more information or to order direct go to www.wiley.com or call +44 (0) 1243 843291

08079_p1

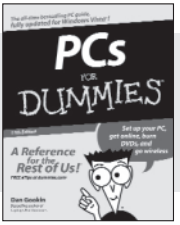
FOR DUMMIES®

Helping you expand your horizons and achieve your potential

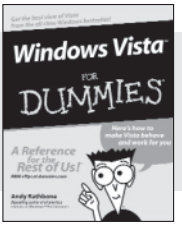
COMPUTER BASICS



978-0-470-27759-1

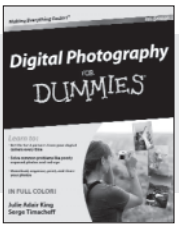


978-0-470-13728-4

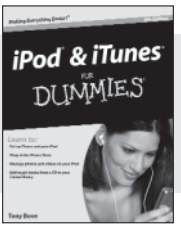


978-0-471-75421-3

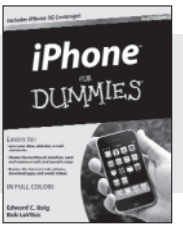
DIGITAL LIFESTYLE



978-0-470-25074-7

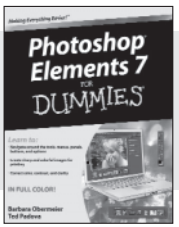


978-0-470-39062-7

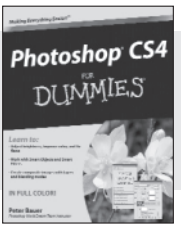


978-0-470-42342-4

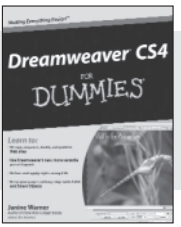
WEB & DESIGN



978-0-470-39700-8



978-0-470-32725-8



978-0-470-34502-3

Access 2007 For Dummies
978-0-470-04612-8

Adobe Creative Suite 3 Design Premium All-in-One Desk Reference For Dummies
978-0-470-11724-8

AutoCAD 2009 For Dummies
978-0-470-22977-4

C++ For Dummies, 5th Edition
978-0-7645-6852-7

Computers For Seniors For Dummies
978-0-470-24055-7

Excel 2007 All-In-One Desk Reference For Dummies
978-0-470-03738-6

Flash CS3 For Dummies
978-0-470-12100-9

Green IT For Dummies
978-0-470-38688-0

Mac OS X Leopard For Dummies
978-0-470-05433-8

Macs For Dummies, 10th Edition
978-0-470-27817-8

Networking All-in-One Desk Reference For Dummies, 3rd Edition
978-0-470-17915-4

Office 2007 All-in-One Desk Reference For Dummies
978-0-471-78279-7

Search Engine Optimization For Dummies, 3rd Edition
978-0-470-26270-2

The Internet For Dummies, 11th Edition
978-0-470-12174-0

Visual Studio 2008 All-In-One Desk Reference For Dummies
978-0-470-19108-8

Web Analytics For Dummies
978-0-470-09824-0

Windows XP For Dummies, 2nd Edition
978-0-7645-7326-2

Available wherever books are sold. For more information or to order direct go to www.wiley.com or call +44 (0) 1243 843291

06439_p4

Qualys: The Leader of On Demand Security and Compliance Management

Qualys is the leading provider of on demand IT security risk and compliance management solutions — delivered as a service. Qualys solutions perform more than 500 million IP audits per year, and are the widest-deployed security on demand solutions in the world. Among these is QualysGuard WAS, which provides automated crawling and testing for custom web applications to identify application vulnerabilities including cross-site scripting and SQL injection. The automated nature of the service enables regular testing that produces consistent results, reduces false positives, and easily scales for large numbers of websites. QualysGuard WAS is integrated with other services in the QualysGuard platform, including network vulnerability management, policy compliance, PCI compliance, malware detection, and a seal of security assurance.

QualysGuard Awards

QualysGuard is overwhelmingly recognized as the leader in its space. QualysGuard has won awards ranging from Best Vulnerability Management Solution, Best Security Product, Best Security Company, Best Network Protection Service, and much more!

Web application scanning needn't be scary!

Successfully learn how to automatically scan your web site for vulnerabilities — on demand!

Web application security may seem like a complex, daunting task. This book is a quick guide to understanding how to make your website secure. It surveys the best steps for establishing a regular program to quickly find vulnerabilities in your site with a web application scanner. This book also tells you about the leading solution for automating website vulnerability management — QualysGuard Web Application Scanning.

Discover:

- *Why web security matters*
- *How to establish a web app security program*
- *The benefits of automated scanning*
- *How automation can ease finding and fixing web app vulnerabilities*

THE DUMMIES WAY®

- *Explanations in plain English*
- *"Get in, get out" information*
- *Icons and other navigational aids*
- *Top ten lists*
- *A dash of humor and fun*

Get smart! @ www.dummies.com

- *Find listings of all our books*
- *Choose from many different subject categories*
- *Sign up for eTips at etips.dummies.com*
- *An electronic version of this book is available at www.qualys.com/wasfordummies*

ISBN: 978-1-119-99487-9
Not for resale.

For Dummies®
A Branded Imprint of
WILEY