"To paraphrase Lord Kelvin's famous quote, 'You cannot improve what you cannot measure.' Computer security has inhabited this sorry state for years, leaving too much room for snake oil, scare tactics, and plain old bull feathers. Andy's book helps to remedy this problem by sending a strong clear message that metrics are both necessary and possible. Buy this strikingly well-written book today and help put an end to security nonsense."

—**Gary McGraw, Ph.D.,** CTO, Cigital, Author of *Software Security: Building Security In*

SECURITY METRICS

Replacing Fear, Uncertainty, and Doubt

ANDREW JAQUITH

FOREWORD BY DANIEL E. GEER, JR.

Praise for *Security Metrics*

"Throw out the security religion and make informed business decisions now!"

—Mark Curphey
ISBPM, Inc.
"Connecting People, Process and Technology"

"I'm very excited that Jaquith has written a text on metrics, and expect this will be the standard reference for years to come."

—Adam Shostack

"Andrew devotes an innumerable amount of time and effort to helping our profession out at SecurityMetrics.org. His book is wonderful, entertaining, and well thought-out. I found myself nodding my head in agreement more than a few times."

—Alex Hutton
CEO, Risk Management Insight

"Andrew has written a book that most people who work in information protection and those who manage and work with them should read, not because it is particularly informative about information protection, but because it is highly informative about the challenges of measuring protection programs effectively. While lots of books are out there about this or that aspect of security, from a security management standpoint, you cannot manage what you cannot measure, and Andrew puts his stake in the ground with this book about what you should measure and how to do it."

—Dr. Fred Cohen
CEO, Fred Cohen & Associates
http://all.net/

"To paraphrase Lord Kelvin's famous quote, 'You cannot improve what you cannot measure.' Computer security has inhabited this sorry state for years, leaving too much room for snake oil, scare tactics, and plain old bull feathers. Andy's book helps to remedy this problem by sending a strong clear message that metrics are both necessary and possible. Buy this strikingly well-written book today and help put an end to security nonsense."

—Gary McGraw, Ph.D.
CTO, Cigital
Author of *Software Security: Building Security In*

are just the sort of things that an organization subjected to a vicious hacker attack needs to be good at, too. Thus, an important system recovery metric is the amount of elapsed time since the most recent DRP walk-through. It is not necessary to track statistics for all systems—just for nominated critical systems.

CHANGE CONTROL

A critical component of any effective security program is the process used to manage changes to the configuration of the environment. Security standards and configuration change management share a natural affinity. In fact, it is hard to see how you could have one without the other; if it is easy to affect an information system's production environment, it is often easy to change its security state as well.

Companies with poor track records in either discipline tend to conflate one with the other. For example, several years ago I was working with a diversified financial institution that operated an interactive website for its customers, who were nearly all consumers. This company had strong security policies but poor change controls and a complete lack of separation of duties.

One day during the spring of 2000, a developer working to improve the company's website uploaded and installed a new version of a particular part of the company's transaction system. He did this because he had a login account on the production machines, and that was what he'd been accustomed to doing. He wasn't forced to write a test plan or a back-out plan, and there was no "production control" group to implement his fixes in a controlled way. He just simply popped the code onto the box and hoped for the best. At some point later, the new module he'd uploaded crashed. It took the rest of the system down with it, including all of the high-availability replicas. The system took eleven feverish, nerve-racking days to troubleshoot, diagnose, and bring back online. *Eleven days!* That's forever in Internet time—and it knocked three full percentage points off the year's uptime figures.

In telling this story, I mean to make a point other than the most obvious one—namely, that change controls and configuration management are essential. I expect you knew that already. No, the real point of the story was the unintended political consequence that stemmed from the lack of change controls. Because of the lack of visibility to the developer's changes, this company did not know what the cause of the problem was—but it was perfectly prepared to blame security anyway. That's right: management circulated the conventional wisdom that "security problems" had caused the outage—when in reality nothing could have been further from the truth. The security monitoring systems worked fine; they had not detected an intrusion, unauthorized security violation, or privacy breach. When the true root cause emerged, my besieged counterpart

(the company's CSO) received quite a few apologies. Immediately afterwards, he began lobbying for stronger change controls and clearer separations of duties.

Three key metrics can help organizations understand the degree of change control an organization possesses. All of these assume the organization keeps track of changes to production systems:

- The number of production changes
- The number of exemptions
- The number of unauthorized changes (violations)

These metrics are typically tracked on a per-period basis, and for additional insights they can be sliced by business unit or technology area. The latter two, exemptions and violations, are related. An "exemption" represents a change that was granted for exceptional reasons and required implementation outside of normal maintenance hours. Emergency fixes and other out-of-cycle changes require exemptions.

An interesting variation on the exemption metric is one that divides the number of exemptions into the number of changes. This results in a percentage that shows how many changes are made out-of-cycle. When grouped by business unit, this metric provides evidence that helps IT organizations finger the twitchiest and most cowboy-like business units.

Unauthorized changes, also known as violations, measure the number of changes that were applied without approval. This number, obviously, should be as close to zero as possible.

APPLICATION SECURITY

Applications are the electronic engines that drive most businesses. Microsoft Office, web servers, order-management software, supply chain management, and ERP systems are all applications that businesses rely on every day. Applications automate firms' workforce activities, pay their bills, and serve customers. Applications come in many shapes and sizes: in-house developed, packaged, outsourced, and served on demand.

As important as applications are to the fortunes of most organizations, they also represent points of potential weakness. Application threat vectors, although they are less well understood than network-based threats, are just as important. As long ago as 2002, Garter Group stated that 75 percent of attacks tunneled through or used application-related threat vectors.[27]

[27] D. Verton, "Airline Web Sites Seen as Riddled with Security Holes," *Computerworld*, 4 Feb. 2002.

For companies with custom-developed applications, the manner in which the software was developed matters. Software written without sufficient attention to security issues carries much more risk than software that adheres to generally accepted principles for coding secure software—as much as five times more risk, based on my previous research.[28]

Measuring the relative security of application code is hard. The security industry has not arrived at a consensus about exactly what it means to build a "secure application." Although definitions vary, there are at least three potential ways to measure application security (see Table 3-5): by counting remotely and locally exploitable flaws without knowledge of the code (black-box metrics), by counting design and implementation flaws in the code (code security metrics), and by creating qualitative risk indices using a weighted scoring system (qualitative process metrics and indices).

Table 3-5 Application Security Metrics

Metric	Purpose	Sources
Black-Box Defect Metrics		
Defect counting	Shows externally identified defects due to implementation or design flaws	Black-box testing tools
Vulnerabilities per application (number [#]) • By business unit • By criticality • By proximity	Measures the number of vulnerabilities that a potential attacker without prior knowledge might find	Black-box assessments by security consultants
Qualitative Process Metrics and Indices		
Business-adjusted risk	Simple formula for scoring the business impact and criticality of vulnerabilities identified in security assessments	Security assessments Spreadsheets
Application conformance indices	Creates a score for ranking the overall security posture for an application or group of applications	Questionnaires Spreadsheets

[28] A. Jaquith, "The Security of Applications: Not All Are Created Equal," @stake, Inc., 2002.

Metric	Purpose	Sources
Code Security Metrics		
Assessment frequency for developed applications • % with design reviews • % with application assessments • % with code reviews (optional) of sensitive functions • % with go-live penetration tests	Measures how often security quality assurance "gates" are applied to the software development life cycle for custom-developed applications.	Manual tracking Lines of code (LOC)
Thousand lines of code (KLOC)	Shows the aggregate size of a developed application	Code analysis software
Defects per KLOC	Characterizes the incidence rate of security defects in developed code	Code analysis software
Vulnerability density (vulnerabilities per unit of code)	Characterizes the incidence rate of security defects in developed code	Code analysis software
Known vulnerability density (weighted sum of all known vulnerabilities per unit of code)	Characterizes the incidence rate of security defects in developed code, taking into account the seriousness of flaws	Code analysis software
Tool soundness	Estimates the degree of error intrinsic to code analysis tools	Code analysis software Spreadsheets
Cyclomatic complexity	Shows the relative complexity of developed code. Indicates potential maintainability issues and security trouble spots.	Code analysis software

BLACK-BOX DEFECT METRICS

Perhaps the most dramatic and headline-grabbing type of application security metric is of the black-box variety—that is, how many holes we can drill in one application compared to another. Black-box testing involves assessing an application, typically remotely via the web. The method of assessment varies. For high-volume testing, automated black-box testing tools from SPI Dynamics, Cenzic, and Watchfire allow companies (or consultants) to quickly scan a large number of deployed applications for potential vulnerabilities.

Automated tools are best suited to testing web applications. A typical black-box web security testing tool "spiders" an application by starting at a known URL (http://www.foo.com/myapp) and following every related hyperlink until it has discovered all the website's pages. After the spider enumerates the application's pages, an automated "fuzzer" or "fault injector" examines the web forms on each page, looking for weaknesses. For example, the fuzzer might see an account registration form that contains a field into which a new user is meant to type her first name. The goal of the fault injector is to see what happens when it sets the field value to something the server-side logic won't expect—like 10,000 letter A's, SQL statements, or shell code.

In the nonautomated camp are security consultants who conduct tests as part of a formally scoped engagement. Consultants tend to be much more expensive than an automated black-box tool but can find issues that the tools cannot. They can also exercise their creativity to dig deeper and find root causes. On the other hand, the level of analytical rigor and degree of methodological consistency vary from consultant to consultant.

Regardless of the method used, the objective of testing is the same: to find vulnerabilities (defects) that can be exploited to compromise the application's integrity, confidentiality, or availability. The categories of flaws that black-box tools and consultants tend to find include:

- **SQL injection:** Manipulating submitted web form fields to trick databases into disgorging sensitive information
- **Command injection:** Executing native operating system commands on the web server
- **Parameter tampering:** Changing submitted web form fields to change the application state
- **Cross-site scripting:** Submitting malformed input that will cause subsequent users to execute malicious JavaScript commands that hijack their sessions or capture data
- **Buffer overflows:** Overfilling a server-side buffer in an effort to make the server crash, or to take it over remotely

At the end of the assessment, the tool or consultant adds up and summarizes any defects found for prioritization. Results of black-box tests are typically simple counts of what defects were found, the category to which they belong, and where. Security consultants generally, as part of the engagement, prioritize the vulnerabilities they find and assign them a criticality rating (high, medium, low).

Enterprises that rely on black-box testing techniques to provide application security metrics, in my experience, do not care too much about defects that aren't marked as

critical. They *do* fix severe issues that could lead to a remote compromise or disclose sensitive data. Thus, in Table 3-5 we recommend that organizations group vulnerabilities by criticality. Other cross sections that companies find useful include by business unit and by proximity. Was the defect remotely exploitable, or could the exploit succeed only when the attacker was logged in locally to the server?

QUALITATIVE PROCESS METRICS AND INDICES

Qualitative assessments earlier on in the application life cycle uncover issues before they become bona fide vulnerabilities in the field. Assessments go by many names. During my tenure at @stake, we performed all manner of application assessments at different stages in the application development life cycle (see Table 3-6):

- **Design reviews** at the midpoint of the design stage
- **Architecture assessments** at the midpoint of development
- **Code reviews** (optional) at the end of development for sensitive functions
- **Penetration tests** prior to deployment

Table 3-6 Qualitative Assessments by Phase of Software Development

	Design Review	Architecture Assessment	Code Review	Penetration Test
Test Type				
Goals	Validation of security engineering principles	Verification of implemented security standards	Focused examination of sensitive functions	Identification of deployment flaws
	Identifies gaps compared to security standards	Finds potential architectural weaknesses	Finds development flaws	Finds "real-world" vulnerabilities
Recommended Testing				
External public-facing	Yes	Yes	Yes	Yes
External partner-facing	Yes	Yes	Yes	Yes
Internal enterprise	Yes	Yes	Optional	Optional
Internal departmental	Optional	Yes	Optional	Optional