

# Integrating Security Testing into Quality Control

## Executive Summary

At a time when 82% of all application vulnerabilities are found in web applications<sup>1</sup>, CIOs are looking for traditional and non-traditional approaches to reduce the number of vulnerabilities present in their application assets. Traditional approaches can range from training developers in secure coding practices to hiring professional security testers.

This paper focuses on a non-traditional approach that leverages existing quality assurance (QA) or quality control (QC) groups to perform high-level functional security testing. The goal of this approach is not necessarily to turn QA/QC groups into ethical hackers, but to add another layer of security testing that is not traditionally employed. Adding this layer can increase the number of vulnerabilities detected and removed in web applications prior to release. This non-traditional approach is best suited for organizations that lack a dedicated software security group (SSG) or security specialists.



### What Is Web Application Security Testing?

Organizations continue to grow the number of application assets they rely on to support or accomplish business goals. These assets are popular targets for attackers, and many large scale application security incidents have been publicized recently. The real threat of being hacked has increased executive managements' interest in reducing the number of vulnerabilities present in their applications.

Web application security testing is the process by which a team of qualified assessors evaluate an application or applications in order to identify opportunities for improvement in security and data validation controls. Ideally this is done as part of a coordinated security program which tracks issues from identification to resolution. Organizations that implement an application security program proactively identify and address security vulnerabilities in their software through the technical implementation of an organization-wide security strategy that aligns with the company's business goals.

Conducting functional security testing can directly measure how well the development team implemented security requirements specified in the project's specification documents. Implementing security requirements early in the design phase of the development life cycle is key to reducing the number of vulnerabilities in a completed application.

Functional security testing is divided into two main areas: feature and risk-based. Feature testing primarily covers positive security requirements. This type of testing is meant to ensure that software behaves according to the customer's expectations. Therefore, it is largely based on software specifications or business requirements. For example, if security requirements state that the length of any user input must be validated, then a feature test suite should be created to exercise the application inputs and verify that this requirement was implemented correctly.<sup>2</sup>

Risk-based security testing primarily covers negative security requirements. Risk-based testing is based on data sensitivity and the chance for the business to be impacted by a loss. Each test is intended to probe for a specific risk or vulnerability. These risks may have been identified previously during the threat modeling or risk assessment process.<sup>3</sup> Common examples of risk-based testing vulnerabilities are Cross-Site Scripting and SQL injection. These vulnerabilities are obviously not functions of the application; therefore, they fall under the negative security requirements umbrella. Risk-based tests often carry out an attack, or try to provide evidence that such an application contains a vulnerability.<sup>4</sup> This whitepaper will focus solely on applying functional security testing to web applications.

### Web Application Threats and Vulnerabilities

Attackers focus on manipulating the application parameters and logic to compromise data or otherwise misuse systems. Due to the nature of HTTP, a clear-text transport-layer protocol, attackers can easily modify URL parameters, HTTP headers, and form fields to alter the application state in an attempt to execute unintended functionality or access unauthorized data. Traditional network-layer security controls cannot inspect application protocol communications nor understand the application context well enough to detect or block maliciously altered web application traffic.

The growing adoption of web-based solutions is driving an increased reliance on web applications for processing both critical and sensitive data. Web-based software components provide a new attack vector that can be used to exploit weaknesses in software quality and business logic in order to:

- Access sensitive or restricted information
- Gain free access to otherwise paid-for content
- Spy on a user's web browsing habits
- Steal a user's identity
- Defame corporations
- Damage Web applications
- Launch Denial of Service attacks

The Open Web Application Security Project (OWASP) has put together an excellent list that defines the top 10 vulnerabilities found in web applications. These flaws may allow for malicious users to alter or implement different functionalities on applications -- sometimes using elevated user privileges. In most cases, attackers target externally facing infrastructure and then utilize trust relationships that the perimeter systems have with other business systems (such as application servers, databases or desktops) to penetrate deeper into the organization; launching attacks on users of the web application as well as the enterprise systems hosting it. The Top 10 OWASP Application Security risks are:<sup>5</sup>

#### A1. Injection

"Injection flaws, such as SQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing unauthorized data."

#### A2. Cross-Site Scripting (XSS)

"XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation and escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites."

#### A3. Broken Authentication and Session Management

"Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, session tokens or exploit other implementation flaws to assume other users' identities."

#### A4. Insecure Direct Object References

"A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data."

#### A5. Cross-Site Request Forgery (CSRF)

"A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim."

#### A6. Security Misconfiguration

"Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server and platform. All these settings should be defined, implemented and maintained as many are not shipped with secure defaults. This includes keeping all software up to date, including all code libraries used by the application."

#### A7. Insecure Cryptographic Storage

"Many web applications do not properly protect sensitive data, such as credit cards, SSNs and authentication credentials, with appropriate encryption or hashing. Attackers may steal or modify such weakly protected data to conduct identity theft, credit card fraud or other crimes."

#### A8. Failure to Restrict URL Access

"Many web applications check URL access rights before rendering protected links and buttons. However, applications need to perform similar access control checks each time these pages are accessed, or attackers will be able to forge URLs to access these hidden pages anyway."

#### A9. Insufficient Transport Layer Protection

"Applications frequently fail to authenticate, encrypt and protect the confidentiality and integrity of sensitive network traffic. When they do, they sometimes support weak algorithms, use expired or invalid certificates, or do not use them correctly."

#### A10. Unvalidated Redirects and Forwards

"Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages."

### Introducing Security into the SDLC

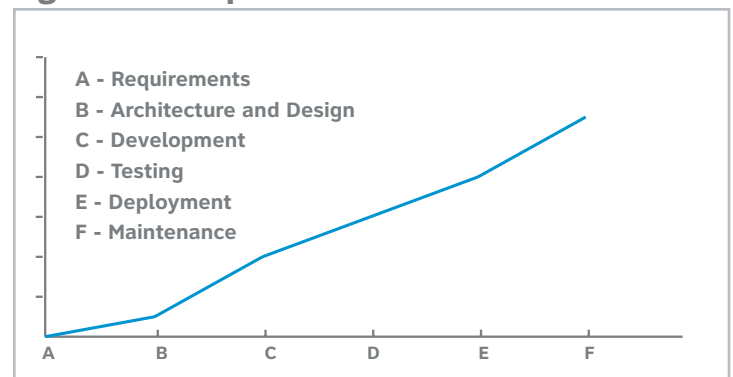
Organizations are constantly looking for ways to reduce the level of risk and lower the probability of breaches coming to fruition. Despite this, Symantec has indicated that there was a 93% increase in web-based attacks in 2010.<sup>6</sup> In order to protect revenue generating assets, organizations are slowly incorporating security into the Software Development Life Cycle (SDLC). The SDLC is a company's overall procedure of developing information systems through a multi-progression process from requirements analysis through design, development, testing, deployment maintenance and ultimately, retirement.<sup>7</sup>

Organizations reap many benefits by incorporating security into the SDLC including:

- Cost reduction
- Increased security awareness of team members
- Seamless security integration
- Improved quality of software

Cost reduction through the identification of vulnerabilities early in the SDLC is arguably the most compelling reason to integrate security into the development process. Gartner states that the cost to fix a security vulnerability found in production is 6.5 times higher than one found in Quality Control (QC).<sup>8</sup> This means that a single security defect that might have cost only \$150 if found in QC could easily cost an organization \$975 if not found until the production phase.<sup>9</sup> This is illustrated in the SDLC cost-justification curve as seen in Figure 1 – Example SDLC Cost-Justification Curve. Organizations that adopt the fundamentals of secure software development processes can reduce such costs significantly.

**Figure 1 – Example SDLC Cost-Justification Curve**



Team members will directly benefit from integrating security into the SDLC. Developers learn principles of secure system design and software coding. Participation in secure code review activities further reinforces secure coding principles and discipline. While secure code review is not a silver bullet, it is one of several layers included in a defense-in-depth approach to application security.<sup>10</sup> Team members can also benefit from writing security requirements that are testable by the QA/QC group. Security should be seamlessly integrated into each phase of the SDLC. The SDLC cost-justification curve clearly indicates the consequences of "bolting on" security in the deployment phase.

Integrating security into each phase of the SDLC will result in overall improved software quality. Serious vulnerabilities detected late in the project life cycle could potentially affect time to market duration for organizations that deem early market penetration as a competitive edge. Organizations that integrate security into the SDLC can experience fewer security-related bugs that delay the release of software to production.

### Quality Control vs. Quality Assurance

ISO 9000 defines quality control and quality assurance as:

- Quality Control
  - The operational techniques and activities that are used to fulfill requirements for quality.<sup>11</sup>
- Quality Assurance
  - All those planned and systematic activities implemented to provide adequate confidence that an entity will fulfill requirements for quality.<sup>12</sup>

While similar, there is an easy way to understand the differences between the two disciplines. QC tests defects in a product and is reactive. An example of QC would be testing a production web application for security defects. QA, on the other hand, is proactive in nature, and is focused on prevention of defects. QA assesses the process that was used during the testing of the web application. A lot

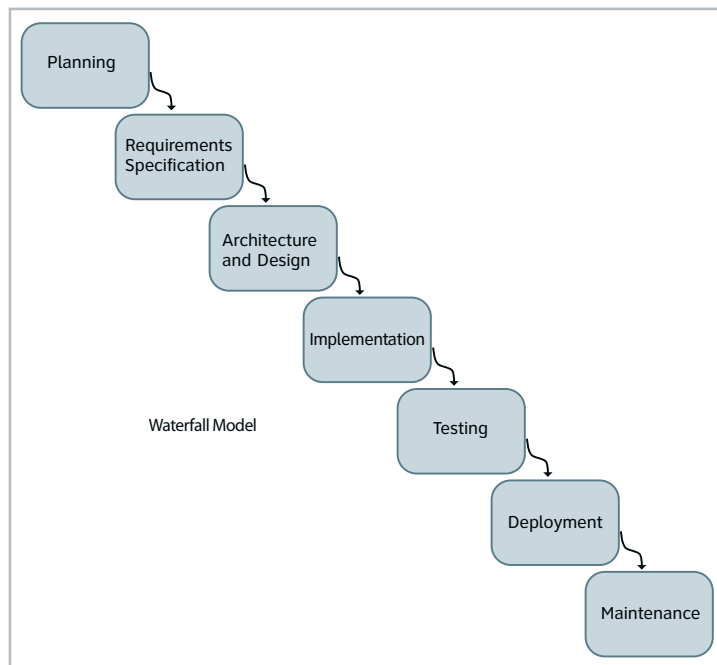
of organizations fail to understand the difference and just label their testing process QA by default. For the purposes of this paper, we will be talking about QC and not QA.

### How Does Security Testing Fit In With QC And The SDLC?

Choosing a SDLC methodology will ultimately determine how security testing fits within QC. There are multiple development methodologies in use today. We will briefly discuss one popular methodology: the Waterfall Model.

The Waterfall Model has been used since the 1970's and is one of the most common and accepted methodologies for building software. The Waterfall Model assumes that a number of phases are completed in a strictly ordered sequence: planning, requirements specification, architecture and design, implementation, testing, deployment and finally maintenance.<sup>13</sup> Thus, creating a waterfall effect as seen in Figure 2 - Waterfall Model. Proper security checkpoints should be incorporated into each phase of the Waterfall Model with the majority of the time being spent in the testing phase.

Figure 2 - Waterfall Model



### Performing QC Security Testing For Web Applications

During the testing phase of the Waterfall Model, the QC team should have a full set of functional requirements to test. These requirements are to be gathered in the requirements specification phase. Assuming that security was integrated into the previous phases of the SDLC, the QC team will also have a set of security requirements to validate. Security requirements go through the same creative process as normal functional requirements. Security requirements should have test cases/test plans created during the requirements specification stage and should be provided to the QC group. In order to increase the level of security assurance of the application through validation of the security test cases, the QC team must be provided with a level of training in application security concepts and vulnerabilities.

Earlier we learned that there are a standard set of web application vulnerabilities on which the QC team should focus. Industry standard lists like the OWASP Top Ten can be a great baseline for security testing inside the QC organization. A common approach to functional security testing is to start with automated vulnerability scanning followed by manual validation and manual testing. These tasks should all be performed in the test phase of the SDLC.

### Automated Vulnerability Scanning

Automated scanning is a great first step for a QC organization. Keep in mind that automated scanning tools will not find all the vulnerabilities in an application. Automated tools do well in identifying many simple, commonly occurring issues. These tools provide value by identifying a variety of application flaws quickly. Potential Cross-Site Scripting, information leakage and web server configuration defects are examples of items typically identified with automated tools. However, automated tools are not foolproof. Automated tools usually do not find business logic vulnerabilities, architectural defects, authentication or authorization issues, or technical vulnerabilities that involve complex interactions with the application like Stored Cross-Site Scripting or complex SQL Injection vulnerabilities.

Once automated scanning is complete, the next step is interpreting the results and validating the reported vulnerabilities. Not doing proper validation may result in developers investigating false-positive findings, wasting time and resources. More importantly, over time, it often leads to the development organization questioning the validity and value of the QC process. It is important to validate the scan results before vulnerabilities are assigned to developers in the bug tracking system. This is where manual testing comes into play.

### Manual Testing

During the manual testing phase, QC has two main goals:

1. Eliminating false positives from the automated scanning results
2. Manually testing for vulnerabilities that automated tools cannot find

Automated tool results generally include false positives. The tasks of eliminating false positives can be a rather large endeavor, but over time, patterns of false positives for a particular tool/application combination become more apparent. As the QC team gains more experience with security testing, false positive identification will speed up. It is important that each vulnerability reported in the bug tracking system is validated as a positive finding and that steps to reproduce the related test case or exploit are captured. The key take away from the first step is that only valid, verified findings should be reported to the development team.

After automated scanning output has been validated, the task of manually searching for vulnerabilities starts. This step usually takes the most effort and highest skill level. Vulnerabilities discussed by the OWASP Top Ten or SANS/CWE Top 25 would be further explored and exhausted during this phase.

Beyond technical testing for software flaws, QC also needs to focus on internal security policy requirements and regulatory requirements that intersect with security. QC should be aware of these requirements and be able to test against them. The first step is which internal security policies are applicable to the software that is being tested. Internal security policies usually fall into two categories: regulatory

policy requirements and technical policy requirements. Regulatory security policy requirements may include policies such as: the Health Insurance Portability and Accountability Act (HIPPA), Payment Card Industry Data Security Standards (PCI DSS) and the Sarbanes-Oxley Act (SOX). Examples of technical security policy requirements may include: password controls, account provisioning, account and session management, encryption and [authentication/authorization](#).

The quality control organization should be able to take these requirements and create a test matrix to validate the adherence of an application to the policies. A test matrix is often a spreadsheet used to show what types of tests are needed for a particular type of testing to ensure complete application coverage. The test matrix will visually document a desired level of test coverage based on the test criteria. Once the matrix of regulatory and technical security requirements is created, the QC organization will then be able to create a checklist that will assist in the development and mapping of test case creation.

### **Maintaining Security Integrity After Release**

During the maintenance phase, it is important to ensure that security remains tightly integrated into the SDLC methodology. Therefore, a plan needs to be developed for potential post-release issues, such as new vulnerabilities. Having documented, repeatable security processes in the QC organization is a key step to maintaining security integrity. When a new vulnerability is found post-release, the QC organization should validate the vulnerability before the new finding gets assigned to the development team. Once the development team fixes the new vulnerability, QC should follow the security testing processes for a new version release, including security regression testing. A formal change control process should also be followed when new vulnerabilities are found and versioning takes place.

The development team will also need to be engaged. Like QC, it is important that the development team follows their security best practices and processes such as secure coding, secure code walkthroughs and proper unit testing. The development team should also determine the root cause for the vulnerability. Identifying why and where the vulnerability occurred will allow the development team to take proper action to ensure that the code is not only fixed, but that similar vulnerabilities are not present elsewhere in the application or introduced in the future.

### **AT&T Security Consulting Can Train Or Augment QC Staff To Increase Accuracy And Speed Up The Automated Scanning Process**

[AT&T Security Consulting](#) has extensive experience providing application security scanning services to organizations. AT&T Security Consulting can augment an existing QC group to offload work during peak demand or to help organizations meet specific deadlines. AT&T Security Consulting also provides application security training to QC groups. This training will equip team members with the necessary skills to execute on security testing tasks quickly and accurately. AT&T Security Consulting can also help in the following areas:

- Selection of automated scanning tools
- Use of automated scanning tools
- Interpreting tool output and results

### **AT&T Security Consulting Can Train Or Augment QC Staff To Increase Coverage Or Speed Up Manual Testing.**

AT&T Security Consulting also has extensive experience manually assessing software of all types including thick-clients, web services, web applications or other complex environments. Just as in the case of automated scanning, we can help offload work during peak demand, help meet specific deadlines and provide training in application security to identify vulnerabilities commonly discovered using manual testing techniques. AT&T Security Consulting can also help in the following areas:

- Selection of manual testing tools
- Use of manual testing tools
- Interpreting tool output
- False positive identification and reduction
- Positive findings validation
- Manual testing techniques

### **AT&T Security Consulting Can Educate Software Developers And QC Personnel To Identify Issues**

AT&T can provide your QC team with the skill set needed to integrate security into the QC organization. AT&T offers security training that will enable your QC team to get up to speed rapidly in order to take advantage of the benefits that seamless security integration offers.

Without training in secure coding standards and practices, developers can unknowingly inject serious vulnerabilities into an application. Ongoing training to use security best practices and secure coding standards help ensure software flaws are minimized and allows weaknesses to be identified in the QC process instead of in production.

AT&T can help your team understand issues to test for tools that work well, and ways to effectively manage identified vulnerabilities. With training, your team will be better educated in security best practices.

### **Summary**

Integrating security into the QC organization adds another layer of security and defense. Training the QC organization in security fundamentals increases the overall quality of the software being released to production and, in the end, saves the organization money. At a time when high-profile systems containing card holder data are breached on a regular basis, costing organizations millions of dollars, the need for integrating security in the SDLC becomes extremely important. The cost of early flaw detection and correction vs. the costs associated with a compromise of customer data has the potential of taking a successful, even booming organization, straight into the red – “no passing go, and no collecting \$200.” With these risks in mind, addressing security in QC makes perfect financial and business sense.<sup>14</sup> By contrast, taking no action can be detrimental to the entire organization. In summary, AT&T can provide the training in QC and in every phase of the SDLC, to help you in your efforts to reduce the overall risk to the organization.

**Notes**

1. Source: Cenzic Web Application Security Trends Report – Q3-Q4, 2009, Cenzic Inc. ([www.cenzic.com/downloads/Cenzic\\_AppsecTrends\\_Q3-Q4-2009.pdf](http://www.cenzic.com/downloads/Cenzic_AppsecTrends_Q3-Q4-2009.pdf))
2. Source: C. C. Michael, Will Radosevich , Ken van Wyk, Cigital, Inc. , “Risk-Based and Functional Security Testing”, July 23, 2009, (<https://buildsecurityin.us-cert.gov/bsi/articles/best-practices/testing/255-BSI.html>)
3. Source: C. C. Michael, Will Radosevich , Ken van Wyk, Cigital, Inc.
4. Source: C. C. Michael, Will Radosevich , Ken van Wyk, Cigital, Inc.
5. Source: OWASP, The Open Web Application Security Project, “OWASP Top Ten – 2010, The Ten Most Critical Web Application Security Risks”, April 19, 2010, (<http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202010.pdf>)
6. Source: SecurityWeek News, “Symantec Report: 286 Million Threats Surfaced in 2010, Sophistication Increases”, April 6, 2011, (<http://www.securityweek.com/symantec-report-286-million-threats-surfaced-2010-sophistication-increases>)
7. Source: on point, “Incorporation Security into the System Development Life Cycle (SDLC)”, ([http://www.onpointcorp.com/documents/Security\\_in\\_the\\_SDLC.pdf](http://www.onpointcorp.com/documents/Security_in_the_SDLC.pdf))
8. Source: Ryan English, “Justifying Security Testing in QA “, December 11, 2006, (<http://www.stickyminds.com/sitewide.asp?ObjectId=11752&Function=edetail&ObjectType=COL>)
9. Source: Ryan English
10. Source: OWASP Code Review, “Security Code Review in the SDLC”, ([http://www.owasp.org/index.php/Security\\_Code\\_Review\\_in\\_the\\_SDLC](http://www.owasp.org/index.php/Security_Code_Review_in_the_SDLC))
11. Source: International Organization for Standardization (ISO), (<http://www.iso.org/iso/home.htm>)
12. Source: International Organization for Standardization (ISO)
13. Source: Per Kroll, “Transitioning from waterfall to iterative development”, April 16, 2004, (<http://www.ibm.com/developerworks/rational/library/4243.html>)
14. Source: Ryan English

**For more information contact an AT&T Representative or visit [www.att.com/business](http://www.att.com/business).**