CodeDx

Five Application Security Testing Pointers for Software Quality Assurance Professionals

Software quality assurance is a tough job. With frequent costly and extremely embarrassing cyberattacks on company websites and networks, the job of software quality assurance (SQA) professional has gotten even tougher. Traditionally, SQA professionals have been responsible for monitoring the software development process to ensure design quality and to make sure that software adheres to a set of coding standards established by their organization. But in today's time-compressed world of rapid releases and DevOps, the traditional SQA process has changed and broadened in scope.

More SQA professionals are now expected to perform application security testing. While it has been generally recognized that software quality is strongly tied to an application's security, testing with a specific focus on detecting security vulnerabilities requires special knowledge, tools, and procedures. Even if an application passes functional tests and is compliant with certain software standards, the software may still be vulnerable to cyberattacks.

Operating in a world of rapid application development and agile development environments, SQA professionals find themselves faced with more abbreviated testing schedules than ever before. Management wants production quality software fast, leaving less time to properly test and approve software applications for security. Consequently, SQA professionals are pressed for time, have more standards to which their software needs to be compliant, have a set of functional tests that may not reveal the security flaws in their software, and are under pressure from developers and management to declare their software secure.

So what's an SQA professional to do?

Here are five pointers to help SQA professionals build a robust application security testing environment to help reduce software security risks:

1. Study and learn the key aspects involved in application security testing
2. Know what vulnerabilities to look for in the software being developed
3. Understand the mindset of the attacker—know how they think and then test like an attacker
4. Learn how to incorporate security testing into your existing QA testing processes
5. Learn how to leverage application security testing and management tools

Pointer 1. Study and learn—know the key aspects of application security testing

Traditionally, SQA professionals have had just one focus: ensuring the quality and functionality of the software they are testing and approving. Now, however, SQA professionals are also required to wear an additional hat: security test professional.

Organizations often already have a security department, so why ask SQA professionals to pitch in? Because *everyone* needs to be involved in securing the software an organization develops, purchases, or sells. It only makes sense for SQA teams to be relied upon, as their end-user perspectives are invaluable in identifying areas of an application where security risks may be high.

The first step in gaining application security knowledge for any SQA professional is to become familiar with application security best practices. It is important to be trained on the basics of security and secure coding practices, such as the fundamentals of access control, encryption, and identifying and protecting critical data sources. Most importantly, SQA professionals should:

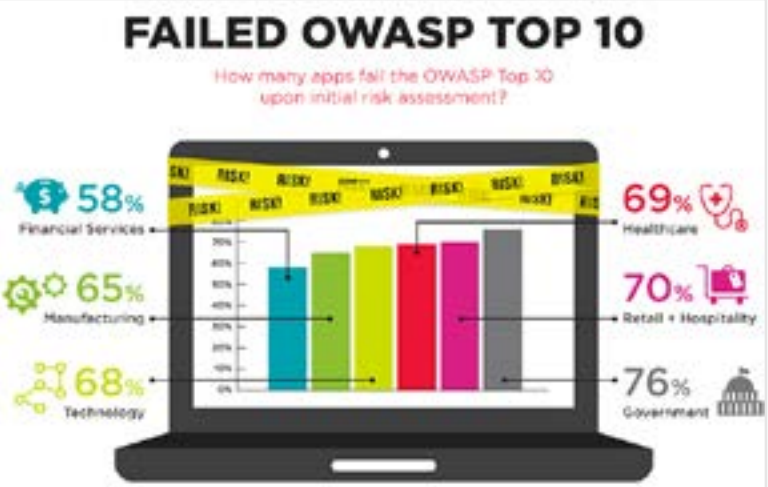
- Understand the types of application security testing, their value, and how to implement them within your security testing environment (Pointer 5 discusses manual, static, and dynamic security testing in greater detail)
- Review the deployment environment for insecure or incorrect configurations, security status of the deployment server, and the protections implemented for valuable data stores
- Understand your typical end-user, and their security aptitude and risk profile
- Stay current on the latest exploits, tools, and testing techniques—both manual and automated
- Stay current on software industry and regulatory standards

So why is security testing rigor so important? Here's a simple example. A fundamental application security consideration is implementation of an application's access control—login and password entry. In November 2011, a SplashData study was done of 6 million username and password combinations using data from companies whose networks had been hacked. The study found 91% of users had used one of the 1,000 most-common passwords. Remarkably, "password" was the leader of them all, in use by 4.7 percent of user accounts. In a followup study five years later, after numerous security lectures, tutorials, and mandates from IT departments and systems administrators, little headway had been made towards users choosing stronger, more hacker-resistant passwords. Users continue to use passwords such as 123456, password, qwerty, welcome, letmein, and abc123. Such dangerous security behaviors by users—and the poorly written software that allows such poor user choices— makes it even easier for hackers to break into an application. Robust password criteria and processes must be created within applications to protect the software from intruders, and from the lack of good judgement of its users.

Next, become familiar with the software application you will be testing, including its security features. SQA professionals should already have a clear understanding of the software application's purpose, use, functionality, and how it should and should not work. However, to perform security testing, SQA professionals must also understand what security features are built into the software, why they were created, how they are intended to work, and how they are *not* intended to work. They should look for unexpected application behaviors and functions that shouldn't be available or active. Finally, they must be familiar with both the public and private interfaces to an application and its key data stores.

Pointer 2. Know what vulnerabilities to look for in the software you develop

Hackers attack software in other ways, not just through compromising user logins. It is critical for SQA professionals to understand all the common vulnerabilities and be cognizant of the security features that should be present in their applications. For those who don't know where to start, the OWASP (Open Web Application Security Project) Top 10 is a document describing the ten most critical web application security flaws faced by organizations. Its purpose is to provide web application security awareness and assist developers, SQA and security teams in securing the software they design, develop, and deploy. The most recent list, published in 2013, includes the following vulnerabilities (in order of most critical to least): Injection, Broken Authentication and Session Management, Cross-Site Scripting (XSS), Insecure Direct Object References, Security Misconfiguration, Sensitive Data Exposure, Missing Function Level Access Control, Cross-Site Request Forgery (CSRF), Using Known Vulnerable Components, and Unvalidated Redirects and Forwards. Early testing for software weaknesses associated with the OWASP Top 10 is an effective first step towards the development of secure code within an organization.



According to Veracode, a large number of applications fail the OWASP Top 10 upon initial assessment. In the government market, 76 percent failed, in retail and hospitality 70 percent failed, in healthcare 69 percent failed, and in financial services 58 percent failed. These are inconceivable statistics when these industries are so heavily regulated with other industry standards, such as HIPAA (Health Insurance Portability and Accountability Act), DISA STIG (Defense Information Systems Agency Security Technical Implementation Guide) and PCI-DSS (Payment Card Industry Data Security Standard).

To assist SQA professionals to better understand these vulnerabilities and test web applications to help build reliable and secure software, OWASP has published the OWASP Testing Guide (4.0), a comprehensive guide and checklist that details successful security testing principles and techniques in relation to various software development life cycle phases. The guide also provides "how to" testing for specific software security vulnerabilities.

Pointer 3. Understand the mindset of the attacker—know how they think, and then test like them

Part of being a security test professional is getting into the mindset of an attacker. Hackers will research their targets and ask questions such as:

1. What information can I gather about this application and its server?
2. What are the application entry points?
3. What open ports are there?
4. How can I bypass user authentication logic?
5. How can I gain root privileges?
6. Where are the "crown jewels" of information stored, and how can I gain access to them?
7. What are the execution paths through the application?

SQA professionals not only need functional knowledge of application security, but also must understand how attackers can attack their software. They cannot dismiss threats that seem bizarre, or assume that "nobody would ever try that." Hackers are extremely creative, and will try anything to gain valuable information or disrupt operations of an organization. Consequently, it is critical to verify what the application will *not* allow a user to do and how the application will behave when given erroneous or unexpected data.

SQA professionals need to know where the most important information is stored and the attack paths that a hacker might take to reach that information and extract it. In our data-rich world, applications usually include databases full of valuable, confidential information that, if leaked, would cause tremendous issues for companies, their employees, their partners, or their customers. A common example is a retail store's point of sale system being hacked and leaking its customers' credit card information. In short, it is important to know what a hacker will be able to access if they find a vulnerability, and how the application will perform if it's compromised.

Most applications implement role-based access control to help administer security in an application. Testing as an administrator, an authenticated user, and as a non-authenticated user is important to cover all the paths a hacker might use to penetrate an application. Administrators have different access to various functions compared to normal and guest users. So, SQA professionals need to test from the perspectives of both less-privileged users who has limited access to functionality as well as administrative users with full access.

Pointer 4. Incorporate security testing into your QA processes

When an application is tested, there is a standard set of processes and tools used to ensure that the software is functionally correct. A comparable set of processes and tools must be implemented and executed when testing for security. The good news is that there is some overlap between quality assurance testing and application security testing practices, and these can be integrated seamlessly into the quality testing process.

Understand the security goals established for the project or application. To achieve software quality goals, development teams make decisions about coding standards they will use to develop and verify software, such as use of GNU and MISRA for C, PEAR for PHP, and CERT Secure Coding Standards for several other programming languages. Similarly, a set of prioritized security goals should be identified and agreed upon. For example, development, SQA, and security teams can agree that an application should be in compliance with the OWASP Top 10 or comply with security rules stemming from an industry-specific standard, such as HIPAA or PCI DSS.

Learn the organization's security policy. Know the protection mechanisms an application should have, and the actions that should be implemented within an organization in the event of an attack. For example, if you discover that a website breach has occurred and data compromised, what actions should be taken? Will the website be shut down until the breach has been remedied and data restored, or will the site remain active and data remain available to users despite the compromise? Does the organization consider the availability of the website and its data to users a higher priority than the website's data confidentiality or integrity?

Create misuse test cases. SQA professionals often create use cases to develop functional testing scenarios. Similarly, they can create *misuse* test cases that will help drive the development of software security requirements. Misuse cases capture the possible attacks on an application as well as the mitigation steps used to counter them. These test cases should focus on breaking the application and what can be done to it rather than what the application *should* do. When defining misuse cases, SQA professionals should consider all application users and their roles, including hostile users. They should also reflect user omissions and errors, and careless or accidental inputs and actions. Some examples of misuse cases may include: manipulating user registration logic, providing erroneous data or SQL into form fields and URLs, reviewing in-transit data for lack of encryption, compromising application authentication, and manipulating or exploiting server configurations.

Generate verifiable security requirements. With security goals and security policy defined and understood, security requirements can now be established. SQA professionals can use the same methods for developing functional requirements (defining what an application should do) to develop security requirements (defining what an application should not, do or should prevent from happening). Just like good functional requirements, good security requirements should be explicit and *verifiable*. Security requirements can be categorized according to security objective, for example, and prioritized based on the risk that failure of the requirement might pose to the application. Examples of verifiable security requirements are:

The Confidentiality, Integrity, and Availability triad (CIA) is a model that helps security teams better understand the security policy decisions made by an organization. Essentially, these three security objectives are interdependent, and if any one of them is compromised it will impact the other two areas. The first objective, confidentiality, involves protecting and concealing information from unauthorized users and ensures that authorized users have access to the appropriate data. Integrity ensures the trustworthiness and accuracy of the data and that its representation remains uncompromised. Availability involves ensuring that information is promptly accessible to authorized users at all times. Understanding the importance and priority of each objective with respect to an organization's applications will help teams establish the application security requirements.

Confidentiality requirements:

1. The system shall encrypt all user personally identifiable information, including username, password, name, address, contact information, account numbers, transactions, and balances.

2. The system shall authorize access only to those who have been identified and authenticated.
3. The system shall implement and update an audit log that includes information on all failed login attempts, including username and password used, timestamp, number of attempts, and lockout time.

Data Integrity requirements:

1. A complete backup image of the server shall be generated every four hours.
2. The operating system shall be configured to implement RAID 5.
3. GNU Privacy Guard (GPG) shall be used to encrypt all customer files.

System Availability requirements:

1. Security upgrades and patches shall be deployed to the server within eight hours.
2. The application's user data shall be available to users 99 percent of the time.

Pointer 5. Leverage the right combination of application security testing (AST) tools

Application security testing has been rapidly gaining momentum in recent years. In fact, 40% of business-critical applications are continuously tested for security, while only 23% were in 2012. Although application security testing (AST) is on the rise, in order to truly secure the nation's software supply chain 100% of business-critical applications need to be tested. Many organizations are not familiar with application security testing methodologies, don't know how to use AST tools, and can't afford the cost or time to configure and run them. Yet there are easy and affordable ways for developers and SQA professionals to quickly learn and conduct application security testing.

There are a few different AST methodologies. The first is "white box," or static testing. Just as with functional white box testing, additional testing that looks at actual code from a security perspective must be done. It is also good to establish "black box," or dynamic testing. This type of testing, which examines an application's security functionality without actually looking at the code, can identify critical functions that are penetrable and open to compromise due to software design flaws. In addition to static and dynamic application security testing (SAST & DAST), there is also hybrid testing, which correlates results of static and dynamic tests to help determine which weaknesses are exploitable by an attacker.

Once the security goals, policies, misuse test cases, and requirements are established, there are a variety of tools designed to help uncover the exploitable security vulnerabilities in software. Manual code reviews are still conducted and often find vulnerabilities that automated tools don't, but they can be extremely time consuming and hard to scale. Therefore, most organizations will need to rely on automated mechanisms to review code for security vulnerabilities.

Although SQA professionals already use many software quality assurance testing tools, it is recommended they add AST tools to their testing tool chest as well. Some existing quality assurance tools offer some limited static analysis capability, however they may lack the vulnerability correlation, de-duplication and reporting capability that is important in helping to reduce the time consumed by application security testing. Since there are many different AST tools on the market, it's important to understand the characteristics of this group of tools and the features and functions that are essential in developing a robust AST environment.

The AST tool market is comprised of a healthy mix of commercial, free and open source tools. Commercial tools are typically expensive, and while open source tools are free, they may also be difficult to use, outdated, or not regularly maintained. What's important to understand is that no single tool will find every security vulnerability. The average SAST tool identifies an average 14 percent of the security vulnerabilities within a codebase. (Source: National Security Agency's (NSA) Center for Assured Software [CAS]). Each tool has its strengths and weaknesses. Some tools, for example, may be excellent at identifying SQL injection vulnerabilities, whereas others may be better at identifying Cross-Site Scripting vulnerabilities. Consequently, multiple tools should be used to achieve identification of the most comprehensive set of security vulnerabilities.

There are several key features to look for when evaluating AST tools. Keep this list in mind when selecting the right tools for your organization:

Number of supported programming languages Some tools support a single language while others support several. Consider the languages used now and in the future, but remember those in legacy systems.

Configurability of security rules The ability to configure and create custom rules helps stem the tide of false positives and better manage the volume and types of vulnerabilities that are important to your organization.

Vulnerability filtering A robust set of filters will help expedite the isolation of critical vulnerabilities, or those associated with specific software and industry standards (such as OWASP Top 10, HIPAA, and PCI DSS), codebase locations, or remediation statuses.

Tool scalability If you have large codebases to analyze, this is an important consideration. How does the size of the codebase impact performance? How long to get results?

Software library analysis Analysis of third-party software libraries used in the code base is also an essential part of performing a comprehensive codebase security analysis.

Tool reporting Customizable reporting that fits the needs of development, quality, security, and management teams is a key feature to consider.

Tool collaboration Good communication between development, quality, and security teams will help expedite remediation of security vulnerabilities.

Eliminate the complexity of using multiple tools. Once the right SAST and DAST tools have been identified and run, the results of each tool need to be combined, correlated, de-duplicated, prioritized, and communicated to developers and management clearly and effectively. Each tool's interface is different, and each assesses vulnerabilities differently—naming conventions, severity scales, and reporting mechanisms are all different. Processing such a large set of results is a very time-consuming effort if performed manually.

It is here where software vulnerability management systems have their greatest value. They provide features that can reduce time, effort, and cost. Here are some key features to keep in mind:

Compatibility Can the vulnerability management system accept outputs of multiple commercial and open source AST tools, both SAST and DAST?

Vulnerability consolidation and de-duplication Can the system take manual review results and multiple AST tool results, eliminate overlaps, and combine similar results?

Vulnerability status management Can it filter and prioritize correlated vulnerabilities and effectively track progress of their remediation?

Reporting capabilities Can it generate unified and customizable reports of the identified vulnerabilities to accommodate both development and management teams?

Collaborative capabilities Does the system help developers, QA professionals, and security teams easily communicate results and remediation recommendations effectively?

SDLC integration Can it be integrated into Software Development Lifecycle (SDLC) environments? Does it seamlessly interface with continuous integration systems, integrated development environments, bug tracking and version control systems?

Considering the security risks in releasing applications into production, it is disconcerting how little focus application security receives. Data breaches are not only enormously expensive to remediate, but also produce a loss of consumer confidentiality, confidence and trust. That is why application security processes, *especially* testing, should be an integral part of **every application's software development lifecycle**.

Just as importantly, SQA professionals need to play an integral role in verifying that applications (and their associated data) are protected and secure. Education of SQA teams is key in developing or enhancing an organization's AST program. Understanding the security goals and policy of an organization, defining clear security requirements and test cases for applications and equipping SQA teams with the right tools will enable an organization to balance the responsibility for application security between development, SQA and security teams. Such active involvement of SQA professionals will help to streamline and strengthen the AST and management process, while achieving reduction of an organization's application security risk.

About Code Dx

Code Dx, Inc. provides easy software vulnerability management systems that help developers, testers, and security analysts find and manage vulnerabilities in software at any stage of development. The Code Dx solutions combine multiple static, dynamic, and interactive Application Security Testing tools and manual reviews into an aggregated, unified interface for simple triage and remediation. The core technologies developed by Code Dx, Inc. were partially funded by DHS Science & Technology to help secure the nation's software supply chain.