FAQ

expert answers to frequently asked questions

How Does Security Testing Fit in My QA Process?

by Alan Crouch
alan.crouch@coveros.com

End-users of your software application expect security and privacy. As technology becomes more sophisticated, so do hackers who will look to exploit your application for financial, political, or other gains. Many testers are being asked to certify the quality of the application and to ensure that it is defect-free—which for more and more organizations means free of security vulnerabilities.

The most common question I receive from testers who are new to security testing is, "How does security testing fit into my QA process?" While security testing should be a part of your QA process, security is not typically an item on your checklist that is supposed to take place as a definitive part of your software development lifecycle (SDLC). Your responsibilities for security testing aren't just limited to the testing phase. Instead, security is a pervasive, underlying principle to be applied throughout the entire SDLC.

In practice, not every implementation of a security testing plan works for every organization or project, and no single implementation will guarantee a successful outcome. In order to improve your chances for success in catching security issues, your organization should, at a minimum, include the following tasks.

1. ***Define security requirements along with functional and nonfunctional requirements.*** Security requirements should be concise and descriptive enough to test, like any other well-written requirement. A good tester will define acceptance criteria and tests just like he does for an application's functional tests. For example:
   - *SEC-REQ-01:* The system must encrypt all user data while at rest (on the data or file system) or in transit (using HTTPS/SSL) to prevent unauthorized access or disclosure to sensitive information. Acceptance criteria include the following:
     A. No user data (passwords, financial information, personally identifiable information, etc.) is stored in clear text in the database or on the file system.
     B. All data entry forms are submitted over HTTPS. When you define your use cases, don't forget to define abuse cases. Unlike use cases, abuse cases look at how your system's current requirements might be used to attack your users, application, or data. These abuse cases define ways an attacker might try to use the application in unintended ways even if it comes into conflict with other requirements. Abuse cases also allow us to identify architectural defects early.
2. ***Perform automated vulnerability scanning early and often.*** Don't wait until product release to perform vulnerability scanning. Like with any other defect, the earlier you find the vulnerability, the cheaper it is to fix. Perform weekly static code analysis and security scanning along the way and mitigate any issues. If you've implemented continuous integration, like the inclusion of regression testing, adding security testing as another gate to a release should be a relatively easy task.
3. ***Eliminate false positives with manual testing.*** No tool is perfect. Automated scanning tools will identify false positives and won't necessarily find every vulnerability in your application. You'll want to consider the use of manual tests as a necessary part of the process.
4. ***Perform thorough security testing in a production-like environment and provide yourself enough time to fix any security defects before a release.*** System architecture, OS, integration, and implementation can have huge impacts on your application. Often a vulnerability found elsewhere in the system in conjunction with an application vulnerability can be far more damaging than either alone. Only testing in a production-like environment will help identify issues your customers would encounter in the production environment. Testing in development is not sufficient enough to provide you an accurate picture of what is vulnerable once your application is deployed.

While there's no silver bullet for security, and even the most thorough security testing does not guarantee a vulnerability-free application, adopting some of the best practices listed above and conducting a reasonable amount of security testing can reduce the risk of being exploited. **{end}**