Finding your Inner Evildoer

LEVERAGE YOUR ALTER-EGO TO MORE THOROUGLY SECURITY TEST YOUR WEB APPLICATIONS

Joe Basirico

Director of Security Services, Security Innovation

SecurityInnovation
THE SOFTWARE SECURITY COMPANY

187 Ballardvale Street, Wilmington, MA 01887 +1.978.694.1008 **www.securityinnovation.com**

Overview

In my experience as both a software security engineer and trainer, it has become clear to me that some students catch on to the concepts more quickly than others. Often times, those who tend to excel don't immediately show signs of brilliance. Frequently, a seasoned tester that can hunt down functional bugs in the weirdest of places can't make the transition to security testing very easily. In the last few years, having trained development teams from some of the world's largest software vendors, I've examined their behavior and attempted to distill what makes a great security tester. I've come up with three pillars that every security testing foundation must have:

Imagination - Many times we don't have all the information we'd like to have as security testers. When exploiting a SQL injection vulnerability, for instance, the security tester has to make certain leaps of faith about the underlying system and make educated guesses about what is really going on to create a really effective test.

Complete knowledge of the system – A great security tester must know about each component of the system he or she is testing. For Web applications this often means in depth knowledge of JavaScript, XML, server side code (asp, jsp, ruby, php, etc.), databases, Web services and more. The tester must be able to recognize when things are out of place, and when components may be used incorrectly. This complete knowledge comes with time and expertise, but can be aided by intense research of each subject with a security focus in mind.

Evil streak – The previous two pillars of expertise will only take a security tester so far in his/her quest for security testing nirvana; the pillar that is a game-changer is the ability to think like the attacker. Being able to anticipate the way the attacker will visualize the system is an integral part of testing the system. Similar to mapping out the many ways a burglar might be able to break into your house, the similar thought process is needed for security testing so that you cover all the creative ways an attacker could exploit your application.

Below I'll discuss each of these pillars in more depth and provide a few tips on how to become an expert in each field.

A great security tester has a great imagination

A great imagination extends beyond the ability to imagine a system as it could be, but extends to envision the truly interesting bugs and vulnerabilities in a system. Most security assessments are performed "blackbox" - without source code, documentation or access to internal systems. When a security tester approaches a security assessment with little information he/she must make certain assumptions and inferences about the system he/she is testing. Sometimes these can be verified later through focused testing, but often times, they cannot.

Take for example an AJAX-enabled Web application. Things that are immediately obvious may be the use of JavaScript, the server language, or the use of SSL. The tester must then imagine what the server topology looks like and ask him/herself a few questions:

- Are they using Microsoft SQL server or MySQL, or are they storing everything in an XML datastore?
    - If they're using SQL, are they using Stored Procedures or Parameterized Queries?
- Where/What is the input checking being done?

- On the client in JavaScript?
- On the server?
- Did they write their own input validation routines or are they relying on controls provided by the language?

There are many other questions that a seasoned security tester will ask to better understand what is going on behind the scenes in this software system. Once the architecture and topology has been discovered it is then time to start discovering how secure the system really is.

SQL injection is an exceptional example of a vulnerability that requires a creative imagination to be discovered. For these vulnerabilities, a tester must be able to envision how certain features in the Web application would be executed on the database. Take for example a login control that requests a username and a password from the user and uses that information to query the database to locate and login the user. What would the SQL look like that would perform this function? After constructing a hypothesis of the query the security tester can verify the guess against the live system by trying certain inputs in the username and password fields. If the test passes and a security vulnerability is found, the tester may choose to do some further testing to leverage this vulnerability against the system to discover vulnerabilities deep within the system.

In this case the security tester is presented with an SQL statement that queries the database and returns some information. This is an exciting statement because it means that a well crafted SQL injection string may allow the tester to arbitrarily read from the database. An improperly secured database may relinquish all control to the user at this point by allowing multiple SELECT, JOIN, INSERT, DROP or EXEC statements to be executed in succession.

Without the ability to imagine the SQL statement or the backend architecture of the system, this attack may not be possible. Even if a less imaginative tester were to stumble across this vulnerability he/she may miss other vulnerabilities with higher risk.

Great security testers have complete knowledge of a system

The most common Web application vulnerability by far is Cross Site Scripting. At Security Innovation, our engineers maintain a knowledgebase of all security vulnerabilities we have found over the years of security testing. More than 85% of vulnerabilities found in web applications are due to poor implementation of Cross Site Scripting. Often times they are so ubiquitous that after finding dozens of them we actually stop looking and instead provide guidance to our customer's development team so they can fix them and so we don't waste our testing efforts.

For this reason Cross Site Scripting is a great example for this subject. Ideally the system is protected by defense-in-depth. Initially any user input should be checked in the Web browser, and then should be validated on the server using a whitelist regular expression. Finally, when that data is displayed back to the user it should be whitelist encoded to ensure no errant characters slip by and are executed on the client's browser.

Let's examine each of the stages of defense-in-depth individually and discuss what should be done at each stage. The first step is to provide the user immediate feedback on the input provided. This is done by checking input in

JavaScript in the browser. A good security tester must be able to recognize where error messages originated and when they can be easily bypassed. JavaScript provides no mechanisms for security since it resides in the browser and can be easily disabled. It does however provide a great mechanism to give the user specific feedback as to what quality of input is expected on a form. This makes users happier, accelerates the form completion process by not requiring a complete post back to the server, and relieves server workload by performing checks on the data before being passed along to the server.

As I mentioned earlier, client side input checking is easily circumvented and cannot be used for a security defense mechanism. JavaScript provides no security benefit. For this reason every input must be checked as thoroughly or more thoroughly on the server.

In this stage the security tester must know how to circumvent the client side input checking mechanisms to exercise each server side check individually. The security tester must also be able to recognize what layer this error message is coming from. Is this error being returned because of an input validation routine, or is it trickling all the way into the system and being stopped by some internal component as it is executed? If the security tester is able to bypass the input validation routines he or she must finally understand what type of attack to apply in this case. Often Cross Site Scripting vulnerabilities are the result of improperly encoded tags or attributes, so closing a string or tag may be necessary before beginning the exploit JavaScript. Alternately this input string may be loaded into a JavaScript block directly, so opening another JavaScript tag would break the system.

Finding your inner evildoer

The final and, in my opinion, the most important pillar of expertise for a great security tester is being able to understand how the system can fail and to think maliciously once you've got your foot in the door. The moment a potential vulnerability is discovered it must be assessed for risk. The most common risk rating system is DREAD which stands for Discoverability, Reproducibility, Exploitability, Affected users and Damage potential. A tester with a healthy understanding of the latest exploits and a bit of an evil streak may be able to convince developers and managers to escalate the vulnerability to a higher risk rating and increase the likelihood of getting it fixed quickly.

Thinking like an evildoer also means keeping abreast of the current security issues in the area that may affect your software system. There are dozens of helpful mailing lists, Web sites and conferences that are great resources for this information. Security testers will find a community to learn and discuss the latest security vulnerabilities and exploits. This gives testers advanced knowledge of the kinds of vulnerabilities that are now common and surfacing in similar types of applications. Additionally, if the software system relies on other systems it is a good idea to search through the mailing lists for those systems for vulnerabilities that may be inherited from less secure components.

Conferences are also good place to gain expertise in the field and learn the latest technique for security testing.

Summary

Security testers have one of the most exciting and creative jobs in the industry. They are tasked with finding elusive security bugs in complex software systems and convincing the rest of the team of its importance. They must prioritize their time and efforts to make sure the best (or worst) security vulnerabilities are found and fixed.

To do this the security tester should have the best resources available: access to external classes, conferences, magazines and books. The security tester has to find as many critical security bugs with limited resources before the major ship deadline; the attacker only has to find one – and has all the time in the world after ship to do so. Level playing field? No. That is what makes the tester's job so exciting, critical and challenging.

Every great security tester has these three qualities: a great imagination, complete knowledge of the system they are testing and an evil streak so they can think like an attacker, and beat them at their own game. By mastering these three pillars of expertise a tester will be well on his/her way to becoming an exceptional security tester.