



The need for a structured security test approach!

by *Andréas Prins*

Most testers who normally test functionality are unfamiliar with application security testing. In the meantime the business runs enormous risks, because security is not covered by our regular tests. The unfamiliarity with security testing causes us, the testers, to leave out these types of tests in our strategy. Our knowledge of testing, however, is very useful for application security testing. The experience we have with structured testing, the collaboration between disciplines, and the knowledge of risk analyses are instruments we need in this situation. Security testing with a structured approach throughout the entire development lifecycle gives a good understanding of the software quality and protects us from known security risks.

The present situation

At present, most testers don't have any experience with security testing. Those who are a little aware of security, execute a security test (penetration test) at the end of the development lifecycle. This late execution is caused by the fact that it is unknown to the customers as well as to the professional testers that the best results can be achieved by integrating security testing into the test process, and these test activities start already in the design and development phase. One of the reasons for this lack of insight is that security testing seems to be different from functional testing. This is demonstrated, for example, by the different security tooling you need for security testing. The testers are not familiar with this tooling. Another reason for late or non-execution of a security test is that the client does not know what security testing really means.

This approach, where security testing is executed just before the application goes into production, is inefficient and very expensive. The tester should apply his expertise much earlier,

for example to execute a security review at the requirements phase. If a vulnerability is found at the end of the development lifecycle, a lot of rework must be done. The project members have to change the software and the designs, but they have to also change the architecture, this is very expensive. Besides this, you have to execute a couple of retests; just retesting the security issue is not enough. When some basic principles are changed, the performance or functionality could be changed as well. A retest of these is therefore necessary. If the tester had spotted the vulnerability when reviewing the requirements, the changes could have been made even before the coders started their development. This would have saved a big part of the budget, which would otherwise have to be spent on analyzing, fixing, testing and re-testing the fix.

Most professional testers don't give any attention to security testing, even though this is an important part of the quality of the application. If you also realize that just one security vulnerability is enough to make the entire application useless, you cannot deliver a good insight into the quality without testing the application for security. For example, if there is an entry point where SQL-injection is possible in just one field in the application, you can lose all your data. With SQL-injection the data for the database is executed as code to manipulate the data. Some data that is entered in a

field will be seen as code because there is not the right validation, and as a result characters in the data are executed as code. This is explained in figure 1, where you can see that the data in the password field has an effect in the background in the code.

If a security test does not take place or no attention is given to security during development, the application could contain dangerous vulnerabilities, which causes enormous risks for the organization. The assets of the organization are, as an attack happens, in danger.

A structured approach

To avoid all this and gain an insight into the quality of the application, a structured approach is necessary to reduce the costs of damage and risks. This can be achieved by an approach for application security and, more in detail, a test approach.

<p>Banking Application</p> <p>Username: <input type="text" value="6717"/></p> <p>Password: <input type="text" value="'1' or '1'='1'"/></p> <p><i>//entered as data</i></p>		GUI
<p>Select name from users where usnrme='6717' and psswr='1' or '1'='1'</p> <p><i>//executed as code</i></p>		CODE

Figure 1 An SQL-Injection example

Definition of asset:

An asset is a resource of value. It varies by perspective. To your business, an asset might be the availability of information, or the information itself, such as customer data. It might be intangible, such as your company's reputation. To an attacker, an asset could be the ability to misuse your application for unauthorized access to data or privileged operations.

One of the most important parts of this is that the business has to determine the required coverage, and it must do so based on the risks they want to avoid. Only that will guarantee a safe application which meets the expectations. Noticeable in this context is that it is the business that determines the test risk and required coverage and not the test experts. Obviously, the test expert gives advice and does a lot of work, but the assets of the customer need to be protected, and this can only be done by the customer himself, because the customer knows what is of value in his organization. Another important task for the test expert is to inform the business about the possible threats and vulnerabilities. A 100% secure application is never possible. The customer expectations/requirements and the security strategy and coverage need to be well balanced. Only then can a safe enough application be achieved. It is important to use the right mitigations at the right moment in time to avoid loss of data, or damage of reputation.

Risks in the future

It is not possible to defend yourself against new Technologies or combinations of techniques that are not known at this moment. An example is CSRF, for years unknown, but when the world started to notice this, a lot of attacks had taken place because it is easy to misuse. A more recent example is click-jacking which was detected in 2008 but had been possible for many years. You cannot know what the security future brings.

To minimize the costs, a structured test approach delivers the right activities as early as possible. As a professional tester you can execute a review for completeness and testability early in the lifecycle, but you can broaden the scope and execute a security review of the requirements and designs. After the architecture is ready you can design a threat model which you can use later on as input for the test strategy. The static source code analysis is also one of the activities which make the application more secure. This can be done after parts of the code are ready, and it can take place before the application is completely ready. The benefits of this approach are that the quality checks are executed before the software development goes into the next phase resulting in less rework. Although this is a task for a developer, it is a test activity because this gives insight in

the quality.

The benefits of an approach that starts as early as possible are visible in the figures the Systems Sciences Institute of IBM published in 2005. If you start with security during the design phase, it costs you one time the security costs. If you start security with static analysis during the development phase, you have to do some rework in the design phase and the costs are six times higher. If you execute the first security tasks in the testing phase during system or integration testing, the costs for security are already 15 times higher. And if the first security activities start in the field, the rework must be done through the total lifecycle, the costs are 100 times higher. There could be more costs, because data can be lost or other kinds of damage may happen. These figures point out that if you need a secure application, you have to start early and continue throughout the entire development process.

The specific tasks for security testers are described in detail in the different Secure Development LifeCycles (SDLC). One of them is CLASP of owasp.org, which focuses on all the roles and activities. Another one is the SDLC of Microsoft. If the professional tester has knowledge of a particular structured test approach, these SDLCs can be very useful. In combination with a test methodology, for example TMap®, it is possible to embed these SDLC tasks within the normal test activities.

The other way to do security testing is in a non-structured approach. However, there are a lot of disadvantages:

- There is no insight in the total quality of the application, because you don't know what happens where in the SDLC.
- A gap in activities is possible; this increases the security risks because not every mitigation is used.
- An overlap in activities is possible; this results in higher costs than necessary.
- The coverage and quality of the mitigations depend on the quality of the expert, while in a structured approach this is based on the methods and best practices.

A structured collaboration

A structured approach between the different disciplines in the development lifecycle is necessary to make the entire application secure. To obtain this approach, the use of the described SDLCs is required. Especially CLASP provides a clear view of the different roles and their activities. It clearly demonstrates the moment when these activities have to take place. If one of these activities is not executed, for instance because there is not enough budget allocated, this will result in a lack of security countermeasures.

During the development of a secure application, interaction between the disciplines is needed. This is necessary within a "normal" development project, but even more important within a project aiming to achieve a secure

application. Some examples of the required interaction are: A threat model must be made by the business, an architect, and a test manager. The choice for the right security solution is made by the designer and the architect and implemented by the developer. A code review is executed by a developer and a tester. This combination uses the "technical" knowledge of the developer and the test knowledge of the tester. The outcomes of the code review can be used by the test team as test scenarios when the application is ready for testing. There is at least interaction between the developer and the application manager to configure the application in the right manner for production. If you have tested the application in a secure configuration, and the application is in production without this secure configuration, the quality advice you gave is not trustworthy because the circumstances are totally different.

Throughout all these activities, a central management and monitoring role is necessary to ensure that the right coverage is achieved through all individual activities. This could be the task of the project manager because he has the overview of the project and the different roles. But it could also be the role of a security specialist who has experience throughout the total SDLC. However, if this key position is not filled, there is a good chance that people work separately from each other and gaps occur in the security coverage.

It is best to start in small projects to set up this security approach. Therefore you can start with these quality steps:

- Architecture phase: Verify the security in the architecture.
- Design phase: execute a review of the requirements and use a security checklist, the results must be specific security requirements.
- Design phase: execute a review of the design to see whether the security requirements have been translated correctly into security solutions..
- Development phase: use best practices and coding standards which the programming language or framework offers.
- Development phase: execute a manual review of your colleague's code focussing on security which is described in the best practices and coding standards (if there is enough budget, use an automated tool for static source code analysis)
- Testing phase: define a strategy with needed test coverage, execute a test and use the right tools for security testing; a lot of freeware or open source tools are available.
- Deployment phase: verify the security configuration.

As the security budget or the need for a secure application grow, you can use more professional tooling and experts to build the SDLC in your organization.

Because security testing is still unknown to many of our customers, it is our task as professional test experts to convince them of the need of a structured approach. The professional tester should realize that he has a key position in this. The results of a security assessment can create awareness with the customers, which in turn could be the start of a secure application development!

- i. More information about assets <http://msdn.microsoft.com/en-us/library/ms978516.aspx>
- ii. More information about threat modeling <http://msdn.microsoft.com/en-us/library/aa561499.aspx>
- iii. http://www.owasp.org/index.php/Category:OWASP_CLASP_Project
- iv. <http://msdn.microsoft.com/en-us/security/cc420639.aspx>
- v. http://eng.tmap.net/Images/Whitepaper_application_security_testing_English_tcm9-52302.pdf



Biography

Andréas Prins is a senior test coordinator, specialized in security testing and is a member of the business development team within Sogeti Netherlands B.V. Andréas has developed different security courses and trains both clients and his own colleagues.

As a member of the development team, he works on different test innovations like cloud testing and the testing of non-functional requirements..

As a test expert, Andréas sets up test innovations for the customers of Sogeti. Also, Andréas actively helps the customers of Sogeti to innovate their test teams and methods. Besides that, he is a speaker on many national and international forums, addressing many different test topics.

Advertisement

© iStockphoto.com/ Palto



Díaz Hilterscheid

IREB

**Certified Professional for
Requirements Engineering
- Foundation Level**

<http://training.diazhilterscheid.com/>
training@diazhilterscheid.com

15.07.09-17.07.09	Berlin
16.09.09-18.09.09	Berlin
18.11.09-20.11.09	Berlin