

TESTING APPLICATION SECURITY WHITE PAPER,
ACCORDING TO TMAP®

Version information

Version	Date	Particulars	Author
V1.0	06-08-08	White paper ready for launching	PaSS-SC

Mailing list

Name	Version	Reason
Public	V1.0	Launch

CONTENTS

CONTENTS	3
1 INTRODUCTION	4
1.1 Objective	4
1.2 Audience	4
1.3 Scope.....	4
1.4 Definitions.....	5
1.5 Authors and reviewers.....	5
2 APPLICATION SECURITY	6
2.1 History	6
2.2 Learning curve	6
2.3 Types of attackers	7
2.4 Sogeti and information security	8
2.5 Where is the application security positioned?	8
3 SPECIFIC KEY POINTS FOR TESTING APPLICATION SECURITY	10
3.1 The tester	10
3.2 Risks.....	12
3.3 Frequently occurring vulnerabilities.....	13
4 APPROACH FOR SECURITY TESTING	15
4.1 Introduction	15
4.2 Assignment	15
4.3 Product Risk Analysis	16
4.4 Test strategy.....	24
4.5 Test basis	25
4.6 Techniques and test types	26
4.7 Organisational aspects	30
4.8 Infrastructural aspects	31
4.9 The use of tools	32
5 EXPERIENCE AND TOOLS	33
5.1 The toolkit.....	33
6 LITERATURE	34

1 INTRODUCTION

Application security and security leaks are in the news more and more frequently. Companies suffer damage and lose a lot of important data due to applications being open to the outside world. Application security often relates to business-critical information systems because these require a high level of security. By security testing, the tester provides the customer with insight in the quality of the application security.

This paper describes the test approach for successful security testing.

This approach is based on TMap[®] [1], Sogeti's test approach, and the information that OWASP [2] provides.

In addition to this, the knowledge and expertise within the other divisions at Sogeti has also been applied. We would like to express our thanks to all contributors for their input, reviews and the constructive feedback.

The paper has been compiled as followed:

Chapter 2 described the background information, which serves to ensure that the reader fully understands the context in which application security has been placed. This chapter also addresses the origin of the need for security and the security learning curve. In addition to this, the broader scope of Sogeti's integral approach to information security is also outlined.

Along with an approach, a number of important general issues are also discussed. These include the behaviour and mindset of a security tester. Chapter 3 focuses primarily on these issues. This chapter also addresses a number of risks and defines those related to security. The test approach is set out with the background information, applying the mindset and the risks as a basis. The test basis, strategy, and techniques are described in chapter 4.

Chapter 5 described the testers' experience. This can help in estimating how much work and what level of effort will be required to test security. Finally, Sogeti has developed a toolkit, which explains the process for security testing.

1.1 Objective

This paper outlines a framework in which security testing takes place in order to establish a sound strategy and approach for performing an application security test. It gives the test expert a better understanding of implementing a test process for security. The approach is described here step by step. This paper gives the tester the method to set up and implement the test process.

1.2 Audience

This paper is intended for test experts who want to specialise in security testing and security test management as defined by TMap[®]. Designers and developers can also use this paper for building a more secure application. They will gain insight in the approach for security testing. Test experts can support them in the application of security in the design and development process.

1.3 Scope

The scope of this paper is the test approach for testing application security. Process, infrastructure and physical security are not within this scope.

The purpose of security testing is to provide advice on the quality of the test object, specifically aimed at security. An explanation of application security is given in section 2.4.

On the Internet and in the related documentation, various terms are used. In this document, 'security testing' is understood to mean the testing of an information system (application). In practice, the terms 'application security', 'security testing' or 'testing security' are used. Sometimes, however, these terms are also used for other issues that go beyond the scope of testing the security of an application. In this document, only the term 'security testing' is used.

This document provides a framework in which a security test can be performed. In order to fully apply it as a functional tester, the toolkit and course developed by Sogeti are needed to perform a test. More

information is given in section 5.2.

1.4 Definitions

The definitions used in this document are:

- Client side validation: data validation performed at the client side instead of at the server side.
- OWASP: Open Web Application Security Project: a world-wide organisation that is involved in detecting and combating leaks in application security and techniques. This organisation produces a number of aids such as tools and documentation for developing secure applications.
- PCI: Payment Card Industry: defines a set of data security standards that apply to companies that process credit card payments.
- Server-side validation: data validation performed on the server side instead of on the client side.
- Social engineering: attacks on systems by making use of social contacts and non-IT related matters to collect information before cracking the system.

SOx, Sarbanes-Oxley: financial reporting regulations for companies listed on the American Stock Exchange. This is also related to application security. More information about SOx and testing can be found in the white paper: "SOX - WHAT IS IT AND WHAT IS THE INFLUENCE ON TESTING, RCLM AND QA".

1.5 Authors and reviewers

This document is created and written by the members of the PaSS-SC taskforce. The document has developed into its current form with the input of security experts from other PaSS taskforces.

The reviewers of this document are: Leo van der Aalst, Rob Baarda, Dagmar Bakker, Ewald Roodenrijs, Marc Valkier, Johan Vink, Willemien de Vries, to which we express our thanks.

2 APPLICATION SECURITY

2.1 History

Software developed since 1990 was intended to serve as a support for existing business processes. All of these different applications were running on local systems. Many of these systems were not interlinked. Most of these applications could only be accessed with a specific computer and the rights and users could only be set in the relevant application. They were developed as open applications and were non-secure.

This had many implications for the security of the application. Basically, the attacker had to break into the building, break into the room and then hack into the system. Security for this was easily arranged by means of access controls. 80% of the security risks were application related, as opposed to being network related.

The demand for network applications began to grow around the year 2000. Applications had to be interlinked to facilitate data exchange. Networks were required to accomplish this. These networks were set up within one building.

The applications were, however, still developed as open code and were non-secure. After all, this is not necessary if everything is running on one network. Users could move from one application to another. The risks for individual users were already increasing because they could easily access other applications.

The application was much easier to access for external abusers. Simply being in the building was enough to gain access to other applications via one application. 80% of the risks at that time were network related. As a result, firewalls were developed and applied to counteract the network related problems.

Since 2007, 80% of the risks are Internet related. The networks that arose were linked to the Internet. The reason for this is that companies worldwide want to be interlinked through their applications. However, the architecture of the applications that run on the networks has not or hardly changed. If access to other applications can be gained through one application then all applications can be accessed on a network. Whether or not login to an application has been honestly obtained, it is possible to gain access to many applications via the Internet. Firewalls prevent unauthorised access. However, a firewall has difficulty identifying whether the other application that is approaching it by legal means is being illegally used.

The current security requirements are of a completely different nature. The physical security of a building offers little added value when the applications are running outside the building. However, physical security has not become useless. Particularly since 2007 there has been a huge rise in social engineering. "Abusers" use this method to obtain valid login data by dishonest means to penetrate systems.

2.2 Learning curve

As mentioned in the previous section, it is now extremely important that applications are tested on security before they are commissioned. In an open environment, applications are easy to access. Apart from this factor, there are a number of other factors that play an important role in security testing.

Nowadays, there are different kinds of developers. There are many individuals, both amateurs and professionals, who are actively involved with the development of information systems in many different programming languages.

With all the aids that are currently available, such as tools, books, and online tutorials, it is possible for anyone to create for example a web shop.

Many of these developers are, however, unaware of the significance of application security. They build non-secure applications to either sell these or use them themselves to do online business.

Along with the low learning curve in application development, the learning curve in abusing this is also low.

Attacks on bank applications, for instance, are frequently revealed in the media. However, there are many cases that do not make the news or even go unnoticed.

The Internet is full of tips on how to hack an application. There are also plenty of books on the market that give step-by-step instructions on how to crack certain applications.

A low learning curve in application development combined with the low learning curve associated with cracking these applications poses huge risks.

2.3 Types of attackers

Owners of information systems have to deal with various types of attackers. To gain insight into these threat agents, an overview is included below.

The different types of attackers can be divided into two groups, namely, the aware attackers and the unaware attackers. This is illustrated in Figure 1.

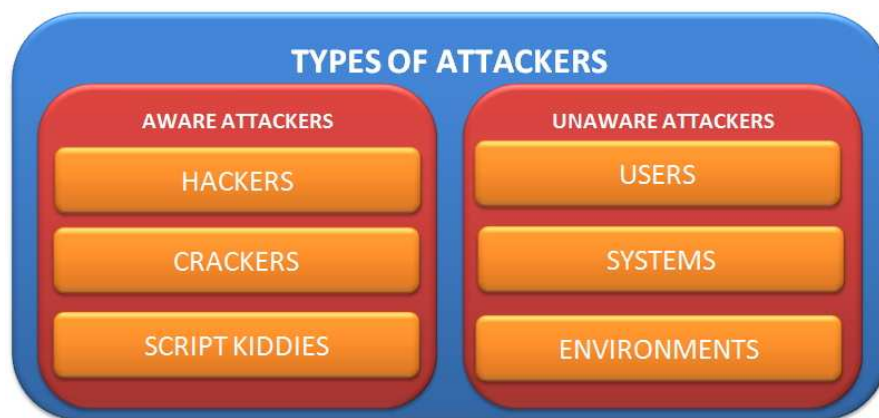


Figure 1 Aware and unaware attackers

Aware threat agents can be divided into three different groups. These groups will appear familiar because they are frequently mentioned in the media. It is important to know the difference between these.

- Hackers: they “crack” or break into systems. The goal of this abuse is not to take down or damage the application. Hackers aim to make the owner or developer of the application aware of the non-secure situation so that the application can be made secure. They are generally well aware of what they are doing and apply various techniques to carry out their attacks.
- Crackers: they are just as knowledgeable on the subject as hackers and also want to crack applications. However, they are in no way ethical and aim to steal information or take down applications. They do this to create havoc for the owner of the application. This is usually for personal gain. Many of these crackers belong to groups that operate from countries like Russia and China.
- Script kiddies: they also set out to cause damage. The main problem, however, is that they have no idea of what they are doing. They implement a number of tricks that have been discovered by others. They often get their malicious script from the Internet or from books. They pose an enormous threat because they do not know what they are doing.

The other group consists of threats that can cause significant damage but have no origin in malicious intent:

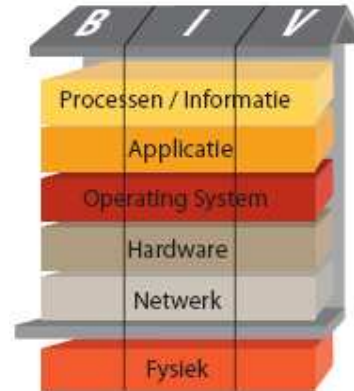
- Users: they can, for example, enter wrong input into fields and inadvertently take down the applications or delete data. This, for example, can be done by entering "--" in a data entry field, which causes an SQL statement to be executed.
- Systems: they can be incorrectly configured and are therefore set up in a non-secure manner. This makes them a threat. Because of the fact that all systems are interlinked, it only takes one incorrectly configured system to cause problems in the entire chain. Errors in one system may then lead to failure of the entire chain.
- Environments: these are often interlinked. If a configuration is incorrect, this can lead to one environment responding to the other environment and possibly influencing it in the wrong way.

2.4 Sogeti and information security

Security testing is only a small part of the overall approach to information security. If security is only taken into consideration during testing, it goes without saying that many security related issues will be discovered during this last phase of software development. To avoid this, developers and designers will also have to take the security aspect into consideration.

A secure application is only a small element within a fully secure environment. Sogeti's approach is geared towards making multiple layers of the application and his environment secure so that a total solution can be safely offered to the client.

Sogeti applies an overall approach that defines several disciplines. This approach defines how information security can be set up using the CIA rating over different layers. CIA stands for the Availability, Integrity, and Confidentiality of data. Within each layer of the CIA model, data must be Available at any random time. The data must be complete (Integrity): is this all of the data there. It is also important that the data is Confidential: can only authorised user access the data?



Figuur 2 The CIA rating

Figure 2 shows that the top layer concerns Processes and Information. This part is secured by the Architecture & Business Solutions (A&BS) department.

The layers that support the application are also shown. These layers are Operating System, Hardware, and Network. The information security activities related to these layers are handled by the Infrastructure Services (IS) department.

As indicated, the Physical layer is not included in the service package provided by Sogeti. However, Sogeti can offer advice in this area (for the physical layer the primary focus should be physical security such as monitoring and defining the areas).

The layer that has not yet been mentioned is the Application layer. The entire V-Model for software development is incorporated into this layer. The emphasis in this layer is therefore the secure design, development and testing of information security.

2.5 Where is the application security positioned?

As illustrated in Figure 3, application security is positioned everywhere. Security testing focuses on one part of this. Not all security can be tested or evaluated. Some elements of security are clearly established in procedures and can be evaluated. Other elements are more oriented on infrastructure.

It is, however, important to be aware of this. Security testing as defined in this paper addresses the areas that are already being tested by testing specialists. The entire information security aspect, such as roles and rights within an organisation, access controls, server area security, design and development, is a comprehensive process and encompasses a lot of information. As defined earlier this paper focuses on a small part of this aspect because otherwise the paper would be too long.

An application is secure if it is clear which user is performing which actions with which role and rights at any given moment of the process. In application security you are therefore paying attention to all of these issues. As illustrated in Figure 3, security is therefore imbedded in each area of the information system.

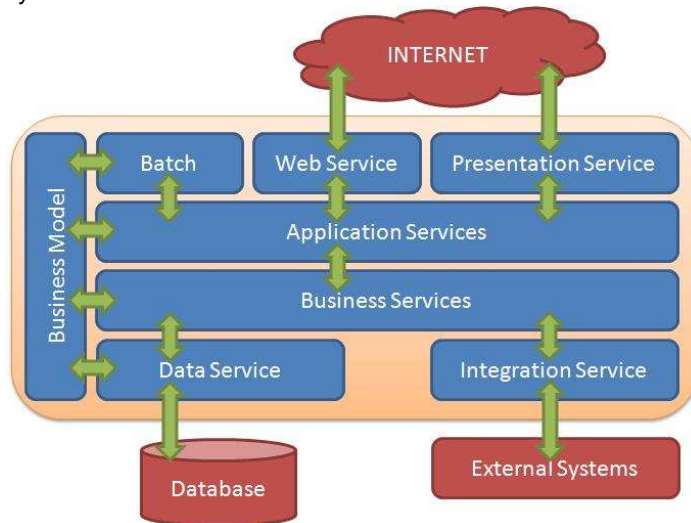


Figure 3 Security is positioned at all areas with arrows.

3 SPECIFIC KEY POINTS FOR TESTING APPLICATION SECURITY

3.1 The tester

This section defines a number of aspects that are characteristic of security testing. The mindset and integrity of the security testers require specific attention.

3.1.1 Mindset of the tester

Experience shows that security testing is different from testing functionality. When testing functionality, the tester focuses on what has been specified. He/she will carry out an additional investigation as an added value and will also detect issues that were possibly not defined in the specification. Security testing also requires the tester to have a degree of creativity, as well as some experience and use of tools.

Possible security issues can be found in two areas (see Figure 4).

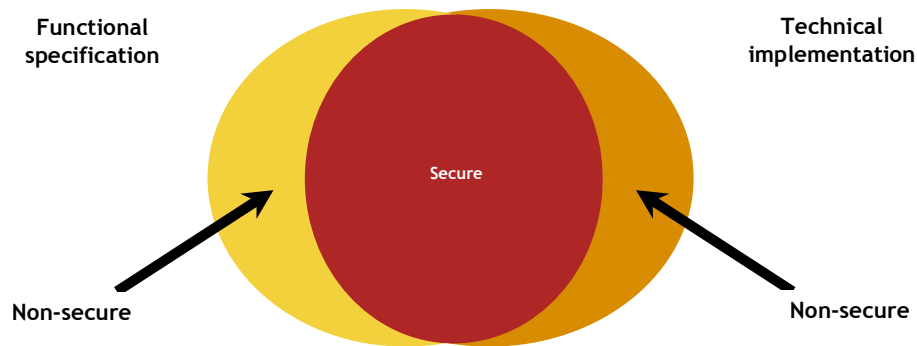


Figure 4 Security issues

The non-secure area on the left has been defined, but not implemented. This is where security leaks are likely to occur.

For example, this could be particular functionality that provides security but has not been implemented.

If a tester detects such issues, he/she will explore further. A missing functionality may have areas of overlap with other functionality and this can possibly cause leaks.

The non-secure area on the right has been implemented, but it has not been defined in the specification. This can, for example, be the use of libraries. A developer could use only part of this, while the library can be more extensive. The user has no knowledge of the undefined functionality.

The tester will have to submit a specific enquiry about this to the developer. Enquiries about these libraries and other undefined issues may be a reason for the tester to examine this in his/her test. However, he/she possibly won't get the complete or usable answer to the enquiry. A tester, therefore, cannot fully rely on this and will have to further examine the issues by, for example, performing an exploratory test.

When security testing, the tester will have to use his/her creativity to not only detect the undefined functionality, but also the undesired functionality. The following example makes this clear.

A tester is testing an insurance application. He/she checks that two particular account types have

the correct authorization in the application.

Type A has few rights and can only see the new applications. Besides the logout functionality, only one other button is displayed in the main screen for this.

Type B has many rights and is also able to change and delete existing insurance policies. In addition to the buttons for type A, the buttons for changing and deleting are also displayed.

The tester tests this as follows. First, a test is performed to see whether the buttons are present for the correct rights; account type A should not have all the buttons of account type B. Type B should have all of the buttons.

The tester also checks that the right screens open when the buttons are clicked.

Apart from running syntactic checks, this is the end of the test. The functionality is present and is working properly.

This is where the security tester steps in. The question is in which ways the access control can be bypassed. How can a user with account type A gain access to the functionality for type B. The tester asks the developers and it soon becomes apparent that they have only made the buttons invisible on the account with fewer rights.

The tester now logs in with account type B and clicks the delete button. He/she then temporarily places the URL that appears in the Address bar onto his/her clipboard.

He/she then logs out and logs in with account type A.

He/she has now set up a session with an account that has few rights. Next, he/she pastes the saved URL into the Address bar. It appears that the application does not prevent this and account A is able to execute the actions of account type B.

This example shows that creativity is required and that the tester has to learn to think differently and that he/she has to have particularly good communication skills in order to be able to uncover matters such as these.

3.1.2 Integrity of the tester

Apart from the fact that a different mindset is required, the tester will also have to be aware of what he/she is doing when the security of an application is being tested.

Many people (clients) place the testing of security on the same par as hacking. To a certain extent, they are right. However, security testing has a different aim than hacking. Testing is performed in advance and is a preventative measure, while hackers try to penetrate an application without rights at later stage.

When a tester detects an issue, he will report this. Each tester will act with integrity with the client. When security testing, vulnerabilities on the part of the customer can come to light, which can directly cause a lot of damage. As such security issues often already exist in the production environment.

The tester will therefore never test the security in a production environment. Not even if the customer requests this. In such a case, the most the tester can do is to point out how a vulnerability can be used.

Prior to security testing, the tester must always obtain permission from his/her test coordinator, project manager, and, for example, a security officer, if present. As a rule, this permission must be given in writing. The commissionee can draw up a standard policy in this respect that can be applied with the various clients.

The tester may never test security outside the test environment. He may, for example, detect an issue in release twelve of an application, while release eleven is operating in production with that same issue. Performing a test to check whether this issue is in fact present in the production environment can have major consequences for the application.

If the tester discovers a defect, he/she must exercise caution when reporting this. This is explained in more detail in section 4.7.2. Setting up and managing the procedure as well as monitoring it is a task for the security officer. It can, however, happen that this person does not fully understand the issue. It may therefore be useful to involve someone like the application manager and ask his/her advice.

Security testing may seem like a somewhat vague area for a tester to venture into; however, this is not the case. After all, if the tester has obtained the client's permission, he/she can just go ahead and perform the test. It is, however, important that the client is the actual owner of the information system. The definition of the scope of, for example, a security scan is therefore also very important. This is particularly the case if any providers and consumers are also involved.

One of the first steps here for the tester is to seek out the applicable policy within the organisation. In the absence of this, an agreement will have to be made with the security officer or the client.

3.2 Risks

In order to test the correct components of the test object, it is important to know what risks the client is running with the application. With security, the motto "No risk, no test" applies just as equally as with all other tests.

This section defines the key points and factors that prompt security risks. The security risk is caused by:

Security risk = Chance of failure * Damage

It is sometimes difficult to express what damage can arise from risks, particularly in the case of security risks. What is the extent of the damage for a company if, for example, data is being tapped over the period of an entire year? The approach described in this document provides some insight into the possible damage that can be suffered as a result of security risks.

The chance of failure due to security risks is not only determined by, for example, inexperienced developers or new development environments, but also by external threats.

The abusers as described in section 2.3 are one of the factors that have to be taken into consideration when determining the security risks. This form of threat, the malicious individual, is usually very competent and well aware of how to exploit vulnerabilities.

In terms of failure, particularly with security, the actual possible crash of an application is not all that should be taken into consideration. The abuse of this is also a failure of the application because it does not succeed in being used as was intended.

To define security risks, a different mindset is required than with other risks. Failure of the application is not so much the primary focus, but rather how an application can be abused. All kinds of different ways are applicable here. An example is a 'defacement'. This maliciously alters the appearance and content of a website. Another method is monitoring the network traffic or taking over the entire application.

To provide more insight into the security risks with the corresponding chance of failure and damage, an example has been set out below. The assignment of an absolute value to the security risks is described in section 4.3.2.

This example is based on the insurance company "Insurance Company". Insurance Company has various applications that have to be replaced by an application that allows customers to manage all of their insurance policies in one application. This application is called 'My Insurance Business Online' and it will be used by more than 10,000 users. In addition to taking out a policy and submitting claims, this application can also be used for payment transactions.

At the beginning of the 'My Insurance Business Online' project, attention was also focused on

the risks. For security risks, special attention was given to damage and the chance of failure. The different types of business and technical damage are, for example:

- Damage to the company's image: 'My Insurance Business Online' can suffer damage to its image, which can be expressed in terms of money lost if, for example, the media makes it known that customer data has been made public because the security was not tight enough.
- The system is offline: if 'My Insurance Business Online' is cracked and, for example, an attacker take over the management of the server, with the result that customers can no longer access it. Each hour that the application is offline costs 'My Insurance Business Online' thousands of Euros. It is, however, also possible that the company takes the application offline on its own initiative because a leak or possible weakness has been detected.

It is therefore not only technical damage expressed in terms of money lost that a company can suffer, business damage can also cause risks.

Apart from the damage, the chance of failure also determines the extent of a risk. Possible factors that influence the chance of failure are:

- No involvement of security experts in the development phase. As no attention was given to security in the early stage, it is possible that the components to be developed are non-secure.
- The type of application plays a large role. Because it is a web application, it is in an open environment and many people can access it. This is unlike, for example, a standalone application on an internal server for internal use.
- Bad coordination in respect of communication between application components. If it is not clear how exactly the communication is affected and which exits and entrances an application has, this can be abused. Non-defined or non-secure 'back doors' can possibly be accessed.
- The high expertise level of abusers: they know from their extensive experience how they can abuse an application and therefore form a significant part of the risks.
- Users' ignorance: this also influences the application. Suppose that a Insurance Company employee adds a new customer to 'My Insurance Business Online' and fills in the date of birth, 30-12-1983, with a double dash (-) between the month and year, this then means that several programming languages will read this as 'everything that follows this is a comment'. The database command will not be executed correctly and all customers in the relevant table will have the same date of birth.

All of these potential forms of failure contribute risk factors at Insurance Company. When building 'My Insurance Business Online' it is therefore essential to consider these factors in advance and list the chance of failure and what the damage would be for each system component.

Because Insurance Company is already thinking about the risks this early in the process, the security can be taken into account within the entire Software Development LifeCycle (SDLC). Certain risks, such as customers being able to see each other's data, must be eliminated. This means that security must be taken into account when designing and building 'My Insurance Business Online'. The tester must cover this risk in the test strategy and when performing the test.

3.3 Frequently occurring vulnerabilities

A common term in security is OWASP or OWASP Top 10. The OWASP is an Open Web Application Security Project. It focuses on detecting and combating security leaks and offers application security solutions.

The OWASP Top 10 (http://www.owasp.org/index.php/OWASP_Top_Ten_Project) is a list of the most frequently occurring, but not all, vulnerabilities in applications. Many organisations use this Top 10 as a guideline. They want to protect themselves against these vulnerabilities. Several of the issues that

appear in the top 10 are related to incorrect input validation. It frequently happens that these validations do not take place, or do take place but then on the client side (usually the user's computer). By placing a simple proxy server between the client and the server, messages can be intercepted and modified.

If a validation takes place on the client side and the message is then modified, it arrives on the server side in a modified form and possibly manipulate the database. Many vulnerabilities can be prevented by using good server side validation. This means that a proxy server is a simple but effective tool for security testing. The proxy works as follows:

A tool is installed on the client side: the proxy is placed between the client and the server to intercept the network traffic. The tool (for example, webscarab from OWASP) is still on the client side. The outgoing traffic is intercepted by the proxy server and can be modified. Then it continues on to the server. This server does not know that something has been modified. Messages that are returned can also be modified. This is illustrated in Figure 5 How a proxy works

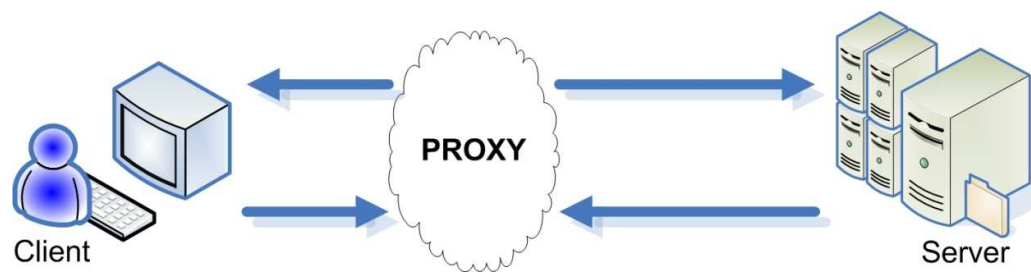


Figure 5 How a proxy works

In addition to what has been mentioned regarding input validation, there are also other matters such as hoaxes, DOS attacks, and buffer overflow. These issues are not addressed in this paper.

4 APPROACH FOR SECURITY TESTING

4.1 Introduction

This chapter describes the approach to security testing. Security testing is a part of a much broader framework, which is described in section 2.4.

When an application is tested on security alone, defects, and therefore, vulnerabilities, appear to emerge. It is necessary that, along with the testing, the designs as well as the development and management are security oriented. This paper does not address this because the entire process involved in the secure development of an application is a process on its own. Sogeti has other taskforces for this purpose who specialise in, for example, designing or programming Java or .Net. More information about this is available at pass@sogeti.nl.

It does not necessarily mean that the organisation is secure when an application has been securely developed. In addition to the application level, there are still other levels that must be secure. For instance, processes, hardware, network, operating systems, and also physical security. This is one of the areas that the Competence Network Security (CNS) task force within Sogeti focuses on. More information about this is available at kerngroepcns@sogeti.nl.

This white paper only addresses the security testing of the application layer. Security is often tested in conjunction with other disciplines, i.e. with designers and developers. Since these experts have a thorough knowledge of the code, they can explain to the testers how certain functions work technically and how functions relate to each other.

This chapter defines a number of factors that should be included in a test plan for security. A standard TMap[®] template can be used for the test plan with modifications to the following components.

4.2 Assignment

When drawing up a test plan for security testing or preparing the security elements of a Master Test Plan (MTP), the assignment should be just as clear as is applicable to a normal test plan.

As is the case with any other test process, the assignment is broken down into a number of elements. These elements are listed below and any variances are explained.

- Client: this is usually the person who initiates the assignment. If testing of security does not constitute part of a test process but has to be separately performed on an application, the client is usually the security officer. In terms of an overall approach, it is advisable that the implementation of security is included within the projects. If the client only gives an assignment for security testing and not for testing secure application designing and building, it will require great diligence to perform the tests. In terms of the policy and processes that the security officer monitors, a new application will often have to undergo a security test.
If, indeed, security testing does constitute part of the overall testing approach, this will be incorporated into the Master Test Plan (MTP). In that case, the client is usually also responsible for all other tests that qualify during the test phase.
- Acceptors: as is the case with normal processes, these are very similar to the client. The client is also an acceptor; however, there are also other acceptors. It is important to inform these entities when security testing. As a security tester, it is necessary to know that there is also an entire group of indirect acceptors. Particularly when application security is concerned, the end user tends to assume that an application is secure. However, this is by no means always the case.
It is important to know that acceptors have no say in drawing up the assignment. This is included here in order to make a distinction between 'client' and 'acceptor'.

- **Contractor:** commits to producing a particular result that supports the client's goal. In most cases, this will be the security tester if he/she is testing the security of an application. If this constitutes part of the total test, the test manager will be the contractor.
The contractor is responsible for drawing up the test plan for security testing. It is therefore not much different from a normal process, however, it could be that, instead of a test coordinator or test manager, the tester is the contractor.
- **The assignment:** focuses on what the contractor does and delivers. This should be brief and to the point. The assignment will not address the security test goals in detail. This will be addressed at later stage when the test goals have been identified together with the client and acceptors.
- **Scope:** this encompasses the scope for both the test object as well as the testing activities.

The scope focuses on the various security elements. For example, the interfaces between applications, the user interface, and, for instance, file exchange.

The testing activities can usually be split into three parts depending on where the tester starts in the project. These are documentation evaluation, code review implementation, and performing a security test. In most test processes, the focus will be on the latter. It is important to be involved at an early stage, particularly for security testers. This is not always possible because experience in the past shows that security issues often get left to last.

- **Preconditions and Assumptions:** these two elements are closely related, however there is a difference which is explained here. Preconditions are requirements that the relevant parties impose on the tests. An example of specific preconditions for security testing would be: 'the security test must be performed in so many days', 'The security test must be performed on an internal network'. For a tester it is important that these requirements are clearly defined. Once the requirements have been set, each requirement is scrutinised. Assumptions are the requirements that the test process imposes on the client. It is essential that the requirements that the test process imposes on the organisation are clearly conveyed. These requirements must be observed in order to prevent damage. One example is 'the test environment must be identical to the production environment'. These requirements must be individually linked to a particular person or function so that it is clear which responsibility rests where.

The contractor receives the test goals from the client. The test goals can be split into various levels. The test goal that the test manager initially receives is often on a global level. In terms of security this is, for example: 'a security test must be performed on the applications'. This is comparable to global test goals such as: 'an Acceptance Test must be performed'.

The test goals are specified in further detail in the Product Risk Analysis (PRA), section 4.3. It is usually the task of the test manager to specify the global test goal 'a security test must be performed' in more detail.

These points are included in the MTP, and then followed by further details of the strategy. This is defined in section 4.4.

4.3 Product Risk Analysis

This section describes the Product Risk Analysis (PRA), which is used as input for determining the strategy. The product risk analysis clearly specifies which subsystems are running which risks. The PRA approach as described in this paper is based on the approach as defined in TMap[®], chapter 9 as well as the risk analysis defined in the OWASP Testing Guide.

In this paper, a number of points have been combined compared to the PRA described in TMap[®]. This has been done because this paper very specifically focuses on the PRA of security. If security

constitutes part of a total PRA, this paper serves as a supplement for the security factor.

4.3.1 Participants and organisation

The first step involved in the PRA is to determine the participants. In this respect, it is important that besides the usual group of participants, there are also participants who are experts in the field of security. This expertise is required in two different areas: practical (knowledge of the operational area), as well as the law and regulations. If possible, these individuals should come from several discipline layers as described in section 2.4.

The parties involved have the following functions:

- client
- project manager
- designer (of processes and application)
- developer(s)
- quality controller (this role is to be filled in by a testing expert)
- administrator
- security expert (specialised in process area)
- security expert (specialised in application area)
- security expert (specialised in infrastructure area)

In practical terms, a PRA works best by organising sessions with these participants. By reflecting on the risks involved in security, the participants complement each other during the sessions. Security covers several disciplines and when identifying the risks, the individuals with various roles complement each other. In this way the day-to-day affairs are combined with security expertise.

Added to this, such a session also generates commitment and a sense of unity, which is of essential importance when building secure applications. It takes more than a good design or good structure to provide security; security has to be applied throughout the entire Software Development Lifecycle.

The preparation for a PRA in respect of security also requires that both the manager and the participants make the necessary preparations.

The test manager has the following tasks:

- To prepare and familiarise him/herself with the organisation and the assignment that he/she has been given, and this with the focus on security. Which business risks already exist from the onset and what business requirements already exist before the PRA begins?
- To draw up a list of security terminology that might be referred to in the PRA and to request clarification before the PRA begins.
- To prepare the PRA kick-off meeting. This should include the goal, the approach, and any security-related terminology. When it comes to security, people usually have varying views and goals in mind. A kick-off meeting is an extremely effective way of creating a communal starting point.
- Sending out invitations and facilitating meetings are activities that belong to the normal course of business affairs. In this respect, however, it is up to the test manager to ensure that all participants understand the matter at hand. Where necessary, he/she can ask the security experts to explain specific subjects.

The participants have the task of preparing themselves by reflecting on the possible security risks or test goals that have to be addressed in the PRA.

4.3.2 Compiling and analysing product risks

Test goal specification

The contractor has received a number of test goals from the client. These are at a general level and do not always contain sufficient information for a PRA. Thus more detailed specification is needed.

This specification must be comprehensible to the client. Security testing goals sometimes require additional explanation which can be included as one of the agenda points for the kick-off meeting. An example of such a goal could be that SQL injection should not be possible. The client may have no knowledge of SQL, and thus requires extra information.

According to section 9.6 of TMap[®], the test goals can be split into different types, for example use cases per subsystem or per business process. Security can be an element within all test goal types.

The specification of test goals will, for instance, be done by the test manager or test coordinator and in the end will always be conveyed back to the client.

The test manager will implement the initial specification and refer back to it again during the sessions. This can lead to yet more in-depth discussion during a session.

The OWASP top ten can serve as a good guideline for input for the specification in order to provide a better understanding of security test goals, www.owasp.org/index.php/owasp_top_ten_project.

The following list includes some examples of test goals at the detailed level. These serve as an example and are not all-inclusive. A selection can be made from these, which will then have to be further refined according to individual situations.

- The system can only be accessed by those persons with the appropriate access rights. This applies to both the official route as well as all kinds of back door routes.
- Contact with other applications must be secure. If there is interaction with another system, all incoming messages must be secure.
- If a user has access to an application by means of textual input, this input must be secure at all times and may not give rise to abuse of the application.
- All interaction between a user and the application must be logged.
- It must not be possible to execute script/code anywhere within the application.
- The system may never be rendered inoperative due to external attacks.
- It must not be possible for customers to read or change other people's data.
- It must not be possible to bypass the login function.
- It must not be possible to retrieve any data other than the intended data from the database.
- The data sent must be encrypted by the application and not coded.
- It must not be possible to call any functions other than the desired ones by means of URL browsing.
- Error messages that are displayed must not contain any information unwanted by the end user.

Identifying relevant characteristics per test goal

The characteristic applicable to the security-oriented PRA is the security quality characteristic. When security constitutes part of the total PRA approach, it makes matters more complex and the security quality characteristic can be added for test goals that affect security.

For the rest, this step is not any different from the PRA approach as described in TMap[®].

Identifying object parts

Once the test goals have been identified, it is necessary to identify the object parts. In the PRA, this is done per characteristic. In the case of security that is the security characteristic.

Security is applied within the entire application and not, for example, to the user interface only. At the end of the PRA, it is clear to which elements the security characteristic applies and to which ones it

does not. By identifying the object parts, any shortfalls in the security can come to light because, for example, these may well be technically present, but have not been functionally defined. They can then be incorporated into the design. Often only the functionality that the client needs is defined and not the functionality that facilitates this.

Filling in the risk table

The method described in this section is based on absolute classification. This is not a science, but it does create an order of risk priority. This can be used to classify and make an initial assessment of the risks.

It is, of course, permitted to apply the relative classification and to arrange each risk in order of priority in relation to the others. This is usually done as a correction to the absolute classification.

The risk table that emerges at the end is based on the formula described in section 3.2. The approach specified below specifically defines a method for classifying security risks. It shows that security risks are fully related to the threats caused by malicious activities.

The following steps describe the approach:

1. Compiling the risk table and defining the combinations;
2. Factors for estimating the damage;
3. Factors for estimating the chance of failure;
4. Defining the risk category per combination.

Step 1: Compiling the risk table and defining the combinations

In the first step a table is made in which the object parts are set out in relation to the test goals. This is per characteristic, in this case for security.

All test goals that have been identified in the PRA are placed in the first column. All object parts that have also been identified earlier on are placed in the top row.

Each test goal and object part combination poses a possible risk and will be addressed during the PRA session(s).

Security	Object parts:	Object part 1	Object part 2
Test goals:			
Test goal 1		X	
Test goal 2		X	X

Table 1 Risk table with test goals and object parts

The first step, the compilation of the table, is only done once.

Which combinations are applicable are determined in the session. These are marked by an X. Each risk that has now been identified passes through steps two, three, and four.

Step 2: Factors for estimating the damage

This step is implemented for each applicable combination.

When estimating the damage, it is important to be aware that there are two forms of damage. The first is the “technical damage” to the application, the data that is used for this, and the functions that it facilitates. The example of Verzekeraak given in section 3.2.

In addition to this, there is also the “business damage”. Ultimately, the business damage is perhaps the most important damage and many technical issues can eventually be expressed in terms of business damage.

Both the technical and the business damage have four factors. All of these factors are rated on a scale of 1 to 9.

The technical damage comprises four areas: confidentiality, integrity, availability, and accountability.

Under confidentiality, the issue is how much data can be lost and how sensitive this data is.

- minimum non-sensitive data has “escaped”(2 points)
- minimum sensitive data has “escaped” (6 points)
- maximum non-sensitive data has “escaped” (6 points)
- maximum sensitive data has “escaped” (9 points)

Integrity concerns the quantity of data that can be corrupted. This is the extent to which the data is a reflection of reality and cannot be modified.

- Minimum slightly corrupt data (1 point)
- minimum seriously corrupt data (3 point)
- maximum slightly corrupt data (5 points)
- maximum seriously corrupt data (9 points)

The next step is to identify a number of systems that can possibly be offline due to an attack.

- minimum secondary systems interrupted (1 point)
- minimum primary systems interrupted (5 point)
- extensive secondary systems interrupted (5 points)
- extensive primary systems interrupted (7 points)
- all systems completely interrupted (9 points)

Lastly, the issue is whether the attackers can be defined. Is it possible to trace who has implemented the attack?

- fully traceable (1 point)
- possibly traceable (7 points)
- completely anonymous (9 points)

The business damage stems from the technical damage, but requires a deep understanding of what is important for the company (the business). Certainly, if the client includes the management, this will have to be translated in terms that they can understand. If you are not in a position to do this, then bring in an intermediary layer of people who understand the business and who can therefore translate the technical aspects into business aspects.

The first damage that the organisation can run up against as a result of vulnerability is financial damage.

- the costs of damage are lower than the costs for solving the vulnerability (1 point)
- the damage caused by the vulnerability has little impact on the annual profit (3 points)
- the damage caused by the vulnerability has a significant impact on the annual profit (7 points)
- the damage caused by the vulnerability brings about bankruptcy (9 points)

The next damage is to the reputation or image that the vulnerability can cause for the business.

- minimal damage (1 point)
- loss of major customers (4 points)
- loss of goodwill (5 points)
- brand damage (loss of face) (9 points)

The third question is how much damage does non-compliance with the regulations introduce.

- minor violation (2 points)
- clear violation (5 points)
- major violation (7 points)

The last issue is the loss of personal information. This concerns how much personal data can be lost or stolen.

- one set of individual personal data (3 points)
- hundreds of data sets (5 points)
- thousands of data sets (7 points)
- a million sets of personal data (9 points)

Step 3: Factors for estimating the chance of failure

This step is implemented for each applicable combination.

Once the possible risks have been identified, it is important to estimate the chance of any of the given risks arising. This is at a global level and the estimate is therefore approximate. We do not ultimately need an exact estimate; a subdivision into low, medium and high will suffice.

The scope of the chance of failure is determined by a number of factors. Each factor has different elements. The elements are rated on a scale of 0 to 9. These scores are applied in order to arrive at the ultimate chance estimation.

The first factor is the threat. Determine the applicable number of points for each element. It is important to assign a value to the education/skill level of the threat agent. Pay close attention to who the threat agent of the object part is. The threat agent can sometimes be one of your own colleagues and sometimes someone from outside. The following scores apply:

- no technical skills (1 point)
- some technical skills (3 points)
- advanced computer user (4 points)
- network and programming skills (6 points)
- security penetration skills (9 points)

In addition to this, the threat agent has a certain degree of motivation for cracking the system.

- low or no motive (1 point)
- possible motive (4 points)
- high motive (9 points)

Next, the opportunity to find the vulnerability has to be defined.

- no authorised access (0 points)
- restricted access (4 points)
- full access (9 points)

Lastly, the size of the group of threat agents has to be defined.

- developers (2 points)
- system administrators (2 points)
- intranet users (4 points)
- partners (5 points)
- authorised users (6 points)
- anonymous internet users (9 points)

The second set of factors is related to the vulnerability. This set of factors also has to be assigned points for four elements. Here it is important to try to estimate the chance of the vulnerability being discovered and being abused. This should be based on the selected threats mentioned above.

The first is the ease with which the vulnerability can be discovered. How easy is it for the threat agent to discover the vulnerability? To define this requires the necessary knowledge. That is why it is wise to involve a security expert in a PRA session.

- practically impossible (1 point)
- difficult (3 points)
- easy (7 points)
- possible to execute with automated tools (9 points)

The second factor is the ease of exploitation.

- practically impossible (1 point)
- difficult (3 points)
- easy (7 points)
- possible to execute with automated tools (9 points)

The threat agents' awareness of the vulnerability also constitutes part of the chance of a risk arising.

- unknown (1 point)
- hidden (4 points)
- obvious(6 points)
- public knowledge (9 points)

Lastly, intrusion detection is explored. This concerns whether the 'break in' has been logged.

- active detection in the application (1 point)
- usage logged and monitored (3 points)
- usage logged without monitoring (8 points)
- no usage logged (9 points)

Step 4, Determining the risk category per combination.

In this step, the three previous steps are combined and the severity of the risk will be determined by multiplying the chance and the damage. This gives an overall idea of the severity of the issue.

Now, the chance and the damage per risk first have to be subdivided into low, medium and high.

In the previous two steps, the impact and chance were determined per risk. Now an average will be taken for the different elements. This produces the average impact and the average chance. The table below shows that a number of points have been assigned to the low, medium, and high categories.

Chance and damage categories	
0 to <3	LOW
3 to <6	MEDIUM
6 to <9	HIGH

Table 2 Risk categories

The following two tables have to be completed for each risk:

The table below reflects the chance that has been defined. All eight elements have been included here. These eventually result in one overall chance. This result is reached by adding up the eight

elements and dividing the sum by eight. The two green rows have to be filled in.

Threat agent factor				Vulnerability factor			
Skill level	Motive	Opportunity	Size	Ease of discovery	Ease of exploitation	Awareness	Intrusion detection
5	2	7	1	3	6	9	2
Overall chance = 4.375 medium							

Table 3 Chance of failure from the risk

The table below reflects the impact that has been defined. The eight components have been included in this. This eventually produces two different impacts. This result is reached by adding up the four results of the technical damage and dividing the sum by four. The same applies for the business damage.

Technical damage				Business damage			
Loss of Confidentiality	Loss of Integrity	Loss of Availability	Loss of Traceability	Financial	Reputation	Non Compliance	Privacy violation
9	7	5	8	1	2	1	5
Overall technical damage = 7.25 high				Overall business damage = 2.25 low			

Table 4 Chance of failure from the risk

The risk category can now be determined for each risk. This is based on the overall chance of failure and damage. If you can make a good estimate of the business damage, it is useful to use this. This is what has to be conveyed later on. If it is not possible to make a good business estimate, the technical damage can be used instead (generally speaking, the business damage stems from the technical damage).

Risk category per combination				
Damage	High	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Notify	Low	Medium
		Low	Medium	High
Chance of failure				

Table 5 Risk category per combination

Merging object parts and risk categories

Fill in the risk category for each test goal and object part combination. The risk category is the result of Table 5 Risk category per combination.

Some combinations will remain empty because these are not recognised in step 1 (Compiling the risk table and determining the combinations).

Security	Object parts:	Object part 1	Object part 2
Test goals:			
Test goal 1		Critical	
Test goal 2		High	Low

Table 6 Risk table filled in with test goals and object parts

4.3.3 Completeness check

The last part of the PRA is not any different from the PRA described in TMap[®] section 9.7. There are, however, a number of additions that can be implemented as extras for security.

The last step of the PRA is to check whether the result of the sessions is complete. In addition, the test manager has to check that the language used in the PRA is comprehensible. If it contains a lot of security terms, these have to be translated into the language of the business.

A completeness check can be performed in the following manner; this is a check to make sure that all object parts have been covered.

This is performed as follows:

- In table 6, check whether there is any object part without a link to a test goal (in which case the column underneath an object part is empty).
- If an object part is not linked to a test goal, discuss this with the participants and check whether the object part does possibly affect test goals.
 - If there is no link, this can mean that it is not applicable, or that a conscious decision has been made to run the risks.
- In table 6, check whether there are any test goals without a link to object goals (in which case the row next to a test goal is empty).
- If a test goal is not linked to object parts, discuss this with the participants and check whether the test goal does possibly affect object parts.
 - If there is no link, this can mean that it is not applicable, or that a conscious decision has been made to run the risks.

4.4 Test strategy

The assignment and possibly the PRA form the basis for the further development of the strategy.

If an extensive PRA has been conducted, the first two steps will already have been carried out.

If a PRA was not conducted, the first two steps will apply.

It is recommended to conduct a PRA for security. This provides a detailed understanding of the risks of combinations between object parts and test goals related to security.

Determining a strategy consists of a number of steps that ensure that a strategy is established, which is conveyed to the client. Four elements, result, risk, time and costs, play a prominent role.

1. Defining the test goals

The test goals are already known from the PRA and are described in section 4.3.2.

2. Determining the risk category for each characteristic and object part combination.

The risk category has been determined in the PRA and is described in section 4.3.2.

3. Determining light/thorough testing

This assessment will be made in the same way as in the normal process. This is where object parts and risks come together. The greater the risk that is being run, the more extensive the required testing regime will be.

4. Feedback with the client on steps 1, 2, and 3.

The previous steps will be reviewed with the client. When specifying the results, it is important to do this accurately.

In the case of functionality, the user usually has a good understanding of what you mean. Security, and the potential damage in particular, is a very real factor but not a factor that is easily envisaged by the business. That is why the risks should be translated into the language of the business, and, where possible, trying to use as many examples as possible.

5. Allocating test techniques, creating test cases, performing tests

This step will initially be carried out by the test manager. Where necessary it is a good idea to take on an additional security expert. The expert can specify which techniques are available for the indicated risks. It is recommended to do this at all times, because the discipline is subject to continuous change.

In order to proactively help seek solutions, it is wise to inform the client of the fact that a lot of security can already be covered with requirements related to security. The test manager can provide the client with a standard package.

The various techniques are specified in section 4.6. These include estimation techniques and review

techniques as well as specification and implementation techniques. The application of these techniques is also covered in this section.

6. Communication throughout the entire test process

Lastly, the contractor liaises with the client. This is effected in the same manner as for every test process.

The following sections address the selection of the test basis for security and the techniques to be applied.

4.5 Test basis

The selection of the test basis runs almost parallel with the previous phase in the process: defining the strategy. The test basis forms the basis for determining the relevant techniques. It is not, however, the guiding principle, because when security testing, it is often the case that little information is documented.

The test basis can consist of different elements. These can be divided into the following categories.

- documentation regarding legislation and regulations (including PCI and SOx)
- security requirements (and non-security requirements, such requirements may well contain information that is relevant in the context of security)
- security policy
- functional designs (security elements or even missing elements)
- technical designs. A lot of valuable information can be obtained from these for a security test. For example, how an application works with other systems (outward and inward interface).

As a functional tester, you are, for example, examining the functionality and it appears that the application has two connections within the chain. One connection is necessary for supplying data. The second is intended for validating an XML message with W3C. For this purpose, a connection will be made with this side.

It can sometimes happen that this has not been defined in a functional design. Such issues are of vital significance for the security tester.

When selecting the test basis, the tester will have to ask the developers what they use and to what extent this is documented.

4.6 Techniques and test types

4.6.1 Estimating the effort

The following is a guideline for the time division for security testing. This is not standard, but it is used for many security tests. When drawing up the test plans, you must be aware that before using this guideline, the tester has to already have been involved in the test process from an early stage. This guideline is based on the assumption that security is the focus of attention throughout the entire process and that the tester has already been involved in the project right from the stage when the requirements were drawn up.

If the tester became involved in the process at a later stage, the emphasis will be more on performing the penetration test. The following percentage ratios will then not apply.

- 50% of the time is spent on reviewing documentation and assimilating this review. The documentation is based on the test basis.
- 35% of the time is spent on the code review. The code review will be primarily executed by developers. The results of the code review are, however, important to the tester. These are the test situations that will be used as input for the test in the next phase.
- 15% of the time is spent on performing penetration tests. Proportionately, this does not seem like a lot. However, a lot of preliminary study has already been done and it is already obvious that the part to be tested derives from the previous points.

In addition to the guideline for subdividing the activities within the test process, it is also important to make an estimation of the time required.

In the case of security, this is extremely complex. It can sometimes take a long time before a security leak is detected. This is, for example, due to the quality of the application, the existence and quality of the documentation, and the possibility to inspect the source code.

This is why it is important to work with time boxes (proportional effort estimation) in respect of security. This entails a number of steps that converge in a table at the end.

1. Create a table with 7 columns.
2. Fill in all object parts in the first column and the corresponding risk category in the second column.
3. In the third column, fill in the size of an object part. This may be a relative size in relation to the other object parts.
4. In the fourth column, fill in the factors that correspond with the combination and risk category of the object part; add these up.
5. Subtract 30% from the total time to be spent; the remaining time can be used for the start of the investigation, exploration, and, at the end, for any exploits of possible security leaks.
6. Determine the scale factor by dividing the result of step 5 by the result of step 4.
7. Calculate the number of hours for each object part by multiplying the result of step 6 by the size x factor of an object part.

Risk category	Factor
High	4
Medium	2
Low	1

Table 7 Risk category and corresponding factor

Example:

In the example, the total time spent is 120 hours. 84 hours is then to be spent on the planned test, the remaining 36 hours is for exploration and closer investigation.

Object part	Risk category	Size	Factor	Size x factor	Scale factor	Number of hours
A	High	10	4	40		52.4
B	High	3	4	12		15.7
C	Low	5	2	10		13.2
D	Medium	2	1	2		2.7
Total				64	1.31	84

When performing tests, it is now important to adhere strictly to the planning. 30% of the time is to be spent on time overruns and closer investigation.

The time required for performing a test for security varies a lot. Depending on the organisation and the moment of involvement in the project, more or less test effort is required.

If an organisation has an effective security policy, due consideration will already have been given to security in the requirements, designs, and development. This test is therefore based on sound documentation and the possibility to inspect source code. Security testing is now integrated into the entire process and costs relatively little time.

The other extreme is if nothing has been done about security during the entire development process and a security scan is only done in retrospect. The extent of the test planning, specification, and performance will now be extensive because it is unclear when what should be done.

4.6.2 Review techniques

Early involvement in the project means that the security tester also has an evaluative role. By being involved in the project early on, errors can be detected at an early stage: as early as the unit level of software development.

The code review has been included in TMap[®], section 7.2.7, as a measure to increase the quality of the developed products. Security is an important element of quality and will therefore be evaluated in that way.

The code review process consists of a number of steps, which in this paper have been tailored to the testing of security. The evaluation is a statistical activity and has already been performed early on in the development process. The review can start when the pieces of code (units) are ready. This is often at the moment that the unit test is performed.

The following steps are part of the process:

1. Gathering of documentation. Which standards and best practices for security are available to the developers? These are often imposed by the organisation and describe a method for developing software securely. Which project documentation is available and which documentation specifies the security? If such documentation is not available, you, as the security tester, can ask the organisation why this is not available and point out the need for it. The lack of this documentation does not mean that following steps cannot be carried out.
2. Does the product (product component) satisfy the requirements of the assignment? Have the security requirements set out in the project documentation been implemented correctly, completely, and demonstrably? In the case of security, this is essential. Even if only one of the requirements has not been implemented, this can have a huge impact on the security chain. If it turns out that certain requirements have not been implemented, there should be another solution available for this.
3. Does the product satisfy the following criteria: is it internally consistent, does it comply with the standards and best practices and the best possible security solution? In this step, the

standards and best practices mentioned in step 1 are to be evaluated. Are these being used and are they correctly applied? The best possible security solution usually involves a stress field. Security user-friendliness is not necessarily an advantage. Choices have to be made depending on the type of application and the possible risks. An internal application that does not contain business-critical information can have a different approach to security than an online insurance company that has access to all personal data.

4. Does the product support the project and architectural goals? Consistency with other products and sub-products is just as important here as the associations with other products. Security should also be an area of focus in this respect. A tooling process can be applied to define the architecture. This runs through the software and identifies the paths and dependencies.
5. Is the product suitable for use in the next phase of development? Do the applied standards and guidelines for security have to be modified? As the technology evolves, so too can new risks. Just like a change in developments, it can happen that the standards selected in step 1 are no longer satisfactory. In that case, it is standard practice to modify this. In addition to this, the evaluation of the previous steps is performed in this step and a decision may be made not to incorporate the software, to redevelop it, or indeed to use it again.

In this step, the tooling process together with the interaction of the human brain is a good combination for a review. The output from the tooling process can be applied as input for the manual review; the output from the tooling process is also used in the evaluation report.

4.6.3 Specification and implementation techniques

Abuse case testing

A use case is a description of the complete interaction between a system and one or several actors. This often gives a thorough and accurate account of what a user is permitted to do and with which rights.

An abuse case is a description of the complete interaction between a system and one or several actors whereby the result of the interaction is damaging for the system, an actor, or one or several stakeholders of the system.

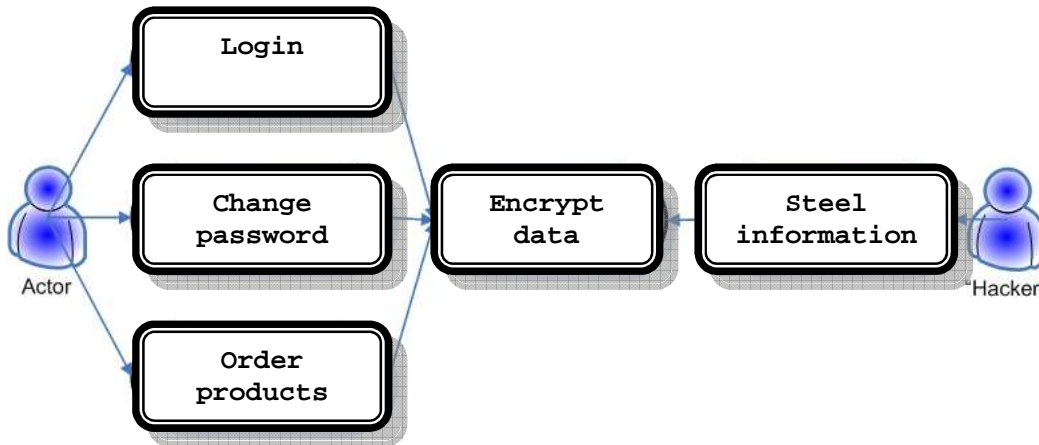


Figure 6 Abuse case testing

In a development process, usually the focus is only on to the use case and not the abuse case. If the tester's task is to test security and use cases are available, the tester will have to create the abuse cases him/herself. The best scenario, however, is when the FOs already include abuse cases along with the use cases.

An abuse case is illustrated schematically in Figure 6 Abuse case testing. We can see from the illustration that not only an actor is present but also a hacker (malicious). A possible solution has also been incorporated into the figure.

Objective

The objective of abuse cases is to learn to think like a hacker and to identify abuse. Abuse cases make it easier for users who have no security expertise to understand the possible security issues and show them what the application looks like from a hacker's perspective. An abuse case gives an idea of the range of different possible security issues. In addition, the solution for possible threats can be identified early on and can therefore also be incorporated into the development process.

Area of application

The abuse cases can be compiled on the basis of security requirements. If, however, these are not available, they will have to be devised.

Abuse case testing is extremely adaptable if use case testing has already been applied. An abuse case could be incorporated into a figure with the use case. For this purpose, security testing is no longer a detached issue, but rather it can be incorporated into the normal test that has to be performed as a standard.

How to apply an abuse case

- Use a use case as a basis.
- Add the hacker.
- Add possible abuse to the (already existing) functionality.
- Add possible solutions.
- Identify various test situations from the abuse (use the OWASP Top 10 for this).
- Apply the test situations as you would for any normal test case.

Data analysis

The analysis of the incoming and outgoing data generated from an application must be analysed to identify if this is correct.

Objective

The objective of data analysis can include:

- To check that encryption has been properly effectuated.
- To check that the username and password have not been sent unencoded across the transmission line.
- To perform an analysis in order to gain more knowledge of how the application works.
- To obtain a better understanding of how the application works.

Areas of application

The area of application of data analysis is the connection on which the server data is transmitted to the client or vice versa, or the storage of files on the server side and the client side.

How to apply data analysis

One of the tools for applying data analysis is the proxy server. This tool can be used to intercept the data traffic. An example of such a proxy server would be WebScarab (http://www.owasp.org/index.php/OWASP_WebScarab_Project). Various tools, such as decoders, can be applied for monitoring messages.

- To identify situations to be tested, like for example:
 - coded information
 - information such as IDs, prices, and other sensitive information
 - SOAP messages
 - cookies
- Adjust the software on the client side so that all traffic runs through a proxy server.
- To start the web application and to work with this application.

- To intercept all incoming and outgoing messages at that particular moment.
- To examine these messages and check them against the previously defined test situations.

Exploratory testing

This test technique is described in detail in the TMap[®] Next book. Exploratory testing can also be applied to security testing. The technique is based on the fact that the tester knows what he/she is doing and knows how to apply various techniques. This also applies to the techniques for security testing.

Example:

The tester wants to examine the application for injection flaws (the entering of a piece of code, which is then executed). For this purpose, he/she is looking for the various "entry points". In doing so, the tester therefore knows what injection flaws are and likewise where he/she can find the "entry points".

While performing the test, the user establishes which "entry points" exist, which injection he/she has released and what the defects are.

This is how, based on suitable expertise, several techniques can be applied to the inspection of the application. It is recommended that this is accurately logged and an overview of the tested parts is made for the client.

Objective

The objective of this technique is to identify the most significant and most frequently occurring security problems while a limited test basis is available or the time allocated for testing is restricted.

Consequently, the exploratory testing of security is a very efficient and focused method for security testing. No matter what type of test is performed, exploratory testing is always one of the components. If this technique is applied alone, it makes it more difficult to establish what, for example, the coverage ratio was. It is very effective, but it is difficult to manage. In combination with, for example, abuse case testing, it is better to adhere to the formulated strategy.

Area of application

The area of application is extremely wide and encompasses the entire application. It can be applied to a technical test. It can be applied, for instance, to database connections, but also to a functional test such as testing a web application.

Experience shows that nowhere near all requirements in the area of security have been defined. This technique and expertise can be applied to investigate whether the security has been adequately implemented.

This technique can also be easily applied in addition to the existing techniques that are being used to test the application.

How to apply exploratory testing

- Decide in advance what at the least should be tested.
- Ask for permission to examine the security.
- Make a thorough, accurate, and straightforward log of the tests that have been performed.
- Explore the application; if, for example, a field that is susceptible to data injections is detected, check to see whether similar fields in the application contain the same error.
- Make use of the various types of tools that have been mentioned earlier on in this document.

4.7 Organisational aspects

4.7.1 Different roles within the organisation

For testing security, the roles are no different from those in a normal test process. There is, however, one exception. In small-scale test processes for security, the tester almost always has a dual role.

He/she is both the test engineer as well as the test coordinator. However, in large-scale processes, several roles will be involved.

Retrospective communication of the approach and evaluation are also particularly important when security testing.

Added to this, the tester will also come into contact with the client's security officer. An individual/role with which the tester would not have any or hardly any dealings with in a "normal" process.

The security officer in the organisation usually has the following tasks:

- To translate the managerial view and mission into a security approach for the organisation.
- To set up and monitor processes regarding data security.
- To collect and manage security metrics.

The bottom line is that the security officer is usually the contact person in the organisation with regard to data security.

The defect procedure

When security testing a different type of issue presents itself than when testing manageability or effectivity for example. When the application to be tested is for instance the 11th release and the 10th release is in production, it is important to apply the defect procedure. It may be the case that the security issue already exists in the production stage. If this is a web application the consequences can be huge. In the case of "functional" defects, it is possible that a certain functionality will not work. However, with security defects, a lot of damage may already have been suffered as a result of abuse. If a security issue has been implemented in a detection management tool, it will be visible to a lot of people. It is therefore recommended that an agreement is made with the client to set up a separate process for this.

We recommend that this is incorporated into the test plan and that the following procedure is applied:

The tester discovers a defect -> reports it to the test coordinator -> reports it to the client or security officer (preferably the latter) -> assigns it to a developer -> assigns it to the tester for retesting.

It is noticeable here that only a few individuals are involved with the issue and that it is not visible for the other testers, but it is also not visible for other developers.

Various defect tools that are available on the market already offer the possibility to work with several levels where defects can be made invisible to others.

4.8 Infrastructural aspects

Permission and authorisation

The client must give explicit approval for security testing.

When testing the security, the tester is privy to a lot of information about the system and how it runs under particular circumstances. The tester may come across security leaks. If the client has not given express permission for this, things could escalate, even possibly resulting in a court case.

The testing of security is usually well intended, however, if it is performed without permission, then it is hacking, which is a legal violation (for more information about the integrity of the tester, please refer section 3.1.2.).

Remote environment

Expressly incorporate into the test plan that the test environment for security testing is completely remote from the production environment and preferably from all other environments as well (Development and Acceptance).

The test coordinator will also have to verify this with the person in the management organisation who

is ultimately accountable.

Identical to production

Especially when security testing, certain peripheral matters are extremely important. For instance, what has been previously written about the hardware, the network, and the OS? When setting up a test environment, these issues should be identical to those of the future production environment as far as possible. If this is not the case, an extensive regression test will have to be performed in the production environment in order to detect possible weaknesses. This is far from desirable, as it can have huge consequences for the other applications.

Using a test environment as identical as possible to the product environment can increase testing efficiency. After all, in that case, the environmental variables on the application are equal. This reduces the risks when the application is put into production.

4.9 The use of tools

Tools are one of the elements that are essential when security testing. If a security test has to be performed, tools are especially necessary, for example, when intercepting the message traffic.

The tools for security testing can be split into a number of different categories. It is important to recognise the difference between these so that they can be applied in the correct manner. Each tool is listed with a description of what it does and what it can be used for. There are many more types and variants of tools available, however this white paper does not address these here because that is a separate subject in itself.

- Proxy server: this is a server (in a tool) that is positioned between the user's (client) computer and the computer where the information is coming from (server). When security testing, the web proxy (a type of proxy server) can be deployed to intercept the data and to modify it before it actually goes to the server. A lot of freeware or open source web proxies can be found on the Internet. WebScarab is an extremely suitable tool for this purpose. This can be found and downloaded at: http://www.owasp.org/index.php/Category:OWASP_WebScarab_Project
- Static code analysis tool: this type of tool can be used to analyse the source code without having to run the corresponding program. The tool searches the code for possible vulnerabilities. These vulnerabilities can then be manually tested (if the program component has been built in) and then applied as input for the penetration test.
- Dynamic program analysis: this type of tool can be used to analyse a program that has already been built. This tool can be applied at a later stage than the static tool. This type of tool also detects possible vulnerabilities in the application. Further manual investigation is necessary to validate whether the weaknesses are actual vulnerabilities in the security.
- Defect management tool: for logging and processing defects. It is important to note that this should be a tool that offers good functionality to block security incidents from individuals who have no business accessing them.
- Other: there are many small tools available for supporting the testing of security, for example, for the automatic detection of an SQL-injection point, for scanning the network traffic, and for Base64 decoding.

It is important to be constantly aware that tools have a supportive function and that a structural approach and effective use of the human brain are essential. Tools may only have a supportive function, but they certainly shouldn't be overlooked.

5 EXPERIENCE AND TOOLS

5.1 The toolkit

In order to incorporate the testing of security into mainstream processes, the PaSS-SC task force has developed a toolkit. This toolkit is a dynamic tool that makes it possible to test the security characteristic based on the best practices.

The toolkit has been developed for various degrees of expertise and application levels. It has four levels, from beginner to guru.

Prior to using the toolkit, it is necessary to follow a course. Information about this course is available at pass-sc@sogeti.nl. The first course addresses the theory from the white paper; supplemented by the beginner level of the toolkit.

The toolkit has been dynamically built and contains a number of threats for each technique. An explanation is given for each technique. An explanation is also given for each threat and the corresponding guidelines on how the threat should be tested and investigated.

At this moment the toolkit is still available in Dutch for employees of Sogeti Netherlands.

6 LITERATURE

1. [TMap Koomen, Aalst, Broekman, Vroon 2006]
TMap[®] Next, for result-driven testing (2006)
2. [OWASP]
Open Web Application Web Application Security Project www.owasp.org
3. [The Security Development Lifecycle. Howard, Libner 2006]
The Security Development Lifecycle. Howard, Libner (28 June 2006)
4. [Software security McGraw 2006]