June, 2009

testing experience

The Magazine for Professional Testers

printed in Germany
print version 8,00 €
free digital version
www.testingexperience.com
ISSN 1866-5705

Security Testing

ISTQB® Certified Tester Foundation Level

for only 499,- €
plus VAT

ONLINE TRAINING

English & German

www.testingexperience.learntesting.com

ISTQB
International Software Testing Qualifications Board

Díaz Hilterscheid

te testing experience

Dear readers,

One of my professors at the university said once to all of us: Computer scientists are at some point criminals. What he meant was that we or some of us – computer scientists – at some point like to try things that are not that "legal". The most of us are "clean", but some of us are "free time hackers"!

Nowadays the hackers are almost away from the 17 years old guy, trying to penetrate in some website and so on. They are now adults, with families, cars, pets, holidays and a job. They are professionals earning money for acting as such.

Application Security is not only important and essential for the companies and their businesses, technology and employees. Application Security is a macroeconomic aspect for the countries. There are a lot of secret services or governments agencies working on getting technology or information by advance hacking the server and databases of top companies or governments worldwide. When we hear that some countries could be behind the penetration of the USA electricity network, you can imagine what is going on outside.

Are we testers prepared for that job? I'm not! Last year we had the first tutorial by Manu Cohen about Application Security Testing. It was amazing what you can do in few minutes using the right tools!!! Even as computer scientist your eyes get wide open. We saw after the first tutorial that we need to give the attendees attack skills; they should learn also to attack and to think how a hacker thinks. The second tutorial some weeks ago had two days introduction into practical hacking. It was an even bigger success.

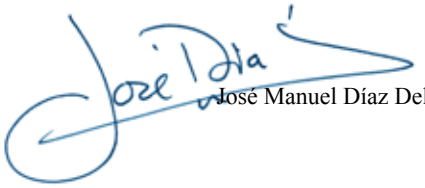
We - as testers - have to be given specific knowledge on security testing to do the job in the right way. As well as this tutorial by Manu Cohen there is an initiative called ISSECO. ISSECO has defined a syllabus for a certification as professional for secure software engineering. This is more than testing; security already starts with the requirements and design of the application. It is a part of the whole process. This is a step in the right direction!

Security is getting essential and that's why we will issue a new magazine on this topic called Security Acts. The first issue is going to be released on October 2009. It appears quarterly too. Please send us your proposals for articles.

The program for the Testing & Finance is ready and I hope to see you there. We have great speakers!

Last but not least I want you to pay attention to our new e-learning portal www.testingexperience.learntesting.com. You can register for ISTQB Certified Tester Foundation Level and very soon for the Advanced Level. Enjoy learning!

Yours sincerely



José Manuel Díaz Delgado

Contents

Editorial 3

Claim-Based Authorization – Next Generation Identity Management 7
by Manu Cohen-Yashar

The Liability of Software Producers and Testers 12
by Julia Hilterscheid

Application Security Fundamentals 14
by Joel Scambray

Security Testing by Methodology: the OSSTMM 20
by Simon Wepfer & Pete Herzog

Wanted: Technical Test Analysts 24
by Erik van Veenendaal

Application Security – Money Still Being Squandered on It 25
by Serge Baumberger

Interview Mike Smith 27

Web Vulnerability Scanners: Tools or Toys? 31
by Dave van Stein

A Risk-Based Approach to Improving Software Security 36
by Rex Black

Case Study: An Automated Software Testing Framework (ASTF) Example 39
by Elfriede Dustin

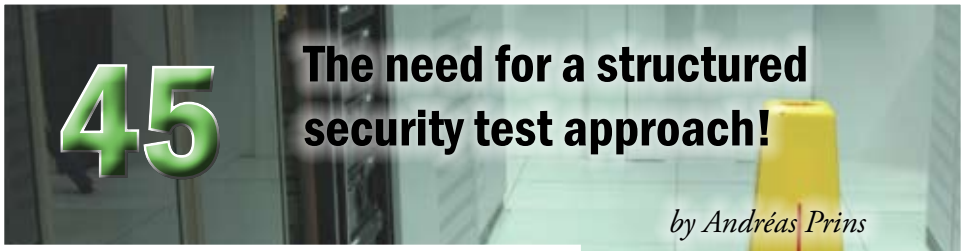
The need for a structured security test approach! 45
by Andréas Prins

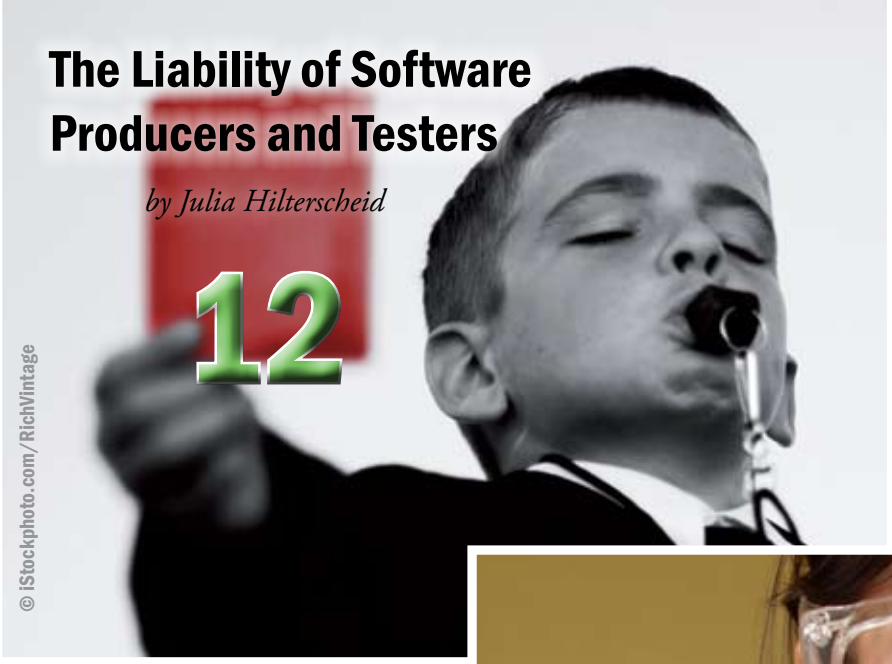
Business Logic Security Testing and Fraud 48
by James Christie

Demystifying Web Application Security Landscape 55
by Mandeep Khera

Customer Success Story - Advertorial 59
by Vladan Konstantinovic

How to conduct basic information security audits? 61
by Nadica Hrgarek





Advanced Software Test Design Techniques, Decision Tables and Cause-Effect Graphs 66
by Rex Black

Load Testing In 10 Steps 71
by Shai Raiten

Testing the Enterprise Security: Anti-Spam and Anti-Virus 74
by Dr. Marian Ventuneac

Software Test Automation: Frame Your Own Requirements 77
by Suri Chitti

Database Auditing 79
by Craig Steven Wright

Project-Based Test Automation 86
by David Harrison

Software Configuration Management-SCM 89
by Mahwish Khan

Align for Good Test Design 93
by Richard van der Pols, Andrew Jong, & Jeanne Hofmans

The new ISTQB® Certified Tester Advanced Level Focus on practical know-how 99
by Professor Mario Winter

Masthead 102

Index Of Advertisers 102

LASSEN SIE DIE ANDEREN STAUNEND AUF DER STRECKE...

Dìaz & Hilterscheid ist
Deutschlands erster Trainingsprovider
mit dem neuen Syllabus für
ISTQB® Certified Tester Advanced Level Test Analyst
(deutsch)

Buchen Sie bis Ende Juni einen CTAL Kurs bei uns
und sie bekommen
15% Rabatt!

www.training.diazhilterscheid.com

ACCREDITED ISTQB Certified Tester Foundation Level TRAINING PROVIDER

ISTQB® International Software Testing Qualifications Board

Díaz Hilterscheid

Claim-Based Authorization – Next Generation Identity Management

by Manu Cohen-Yashar

Identity is one of the most popular challenges applications face today. Almost every application has to know who it is talking to and needs to do something about it. Unfortunately we know that identity is poorly handled, as Identity theft is one of the world's greatest problems today.

What exactly is identity? After decades of working with Identity, we finally understand that identity is nothing more than some information that describes an entity. It turns out that entities have multiple identities, each relevant on a different context of execution. For instance, a person (entity) has one or more professional identities, a personal identity, a social identity etc.

An application that needs to work with an entity's identity needs to know which identity to look for.

The life cycle of identity management from the application's point of view would look something like this:

1. Registration: The entity stores its identity information in some identity management store that will be used by the application later in the life cycle. This store is usually managed by IT professionals.
2. Authentication: The entity sends some form of credentials to enable the application to determine who it is.
3. The application searches the identity stores for all the identity information it needs.
4. The application uses the identity information for authorization, personalization and for its business activities.
5. The application creates an authenticator and caches the identity information. The authenticator is given to the client for immediate interaction with the application, so that he/she will not need to go through authentication every time he/she interacts with the application. The application might use the authenticator as a key to find the identity information in the cache.
6. The user logs out and the authenticator is deleted.

All the above introduces a substantial challenge for the application.

- The identity information is usually sensitive. Credentials are always sensitive. Keeping sensitive information in a persistent store in a secure manner is not easy. A security professional and IT professionals should be consulted before writing the code. To overcome these issues, professional infrastructure for identity management like MS Active Directory or IBM Tivoli are often used. Interacting with these will, however, raise more issues, as we will see later on. Unfortunately today almost every application requires users to register and supply some identity information. It is easy to understand that the more your identity information is spread out, the less secure it is. Users are not happy to register as they know that somewhere there is an application that does not secure their identity. Users might choose not to make business with an application that requires registration.
- When every application has its own identity management, users end up having a growing number of passwords. Unfortunately, a human user cannot handle so many passwords, so passwords are repeated and become weak. The security threat is trivial. Passwords are the weakest form of authentication, but usually this is all we have got. Governments have failed to distribute a strong form of authentication to their citizens, e.g. smart passports, and thus there is no strong authentication for the masses. Some employers and large organizations have managed to do so and they enjoy a much safer authentication.
- After authentication the application needs to look for the identity information it needs. This information might be stored in several stores managed by the IT pros that usually do not know and care about the application. The application needs to know where and how to look.

  In the age of distributed applications, globalization and company mergers and acquisitions, it is quite common that not just the information is stored in several data stores, it is stored using several different technologies. The application must master these technologies if it wants to find the data.

  What happens when the store moves as a result of a company acquisition or simply because of an upgrade?

  It turns out that interacting with professional identity management systems is not that easy.
- When the application finally has the information it needs, it is time to figure out the authorization rules and to apply them. These rules might be complex and subject to frequent changes, so the application has to use some sort of dynamic decision processor to execute authorization. The rules must be protected and the processor must be efficient as everything goes through authorization. This is not an

Subscribe for the printed issue!

Please fax this form to +49 (0)30 74 76 28 99, send an e-mail to info@testingexperience.com or subscribe at www.testingexperience-shop.com:

Billing Adress

Company: _____

VAT ID: _____

First Name: _____

Last Name: _____

Street: _____

Post Code: _____

City, State: _____

Country: _____

Phone/Fax: _____

E-mail: _____

Delivery Address (if differs from the one above)

Company: _____

First Name: _____

Last Name: _____

Street: _____

Post Code: _____

City, State: _____

Country: _____

Phone/Fax: _____

E-mail: _____

Remarks: _____

1 year subscription

☐ **32,- €**
(plus VAT)

2 years subscription

☐ **60,- €**
(plus VAT)

Date

Signature, Company Stamp

easy challenge..

- The user and the application would be very unhappy if every interaction with the application requires a long authentication process, so an authenticator is created for the client. The authenticator is like a ticket that proves to the application the entity has already passed authentication. This authenticator must be heavily protected. It is common to find applications that perform a great deal of authentication, but their authenticator is weak and can be forged easily. The results are disastrous.

So Identity management is not easy and yet everyone needs to do it. Wouldn't it be great if identity management could be outsourced?

The idea of outsourcing had amazing consequences on the world economy, maybe it could help here as well.

What about the application identifying "someone" it trusts and letting him do all the identity management work. The application would get from this "someone" only what it really needs – the identity information. This is exactly what claims-based authorization is all about.

In a claim-based application users present their identity to the application as a set of claims. These claims include all the information about the user like name, e-mail address, and maybe a list of services he/she is allowed to use. The idea is that an external identity system is configured to give the application all the information it needs. The information will be packaged as a token built from a collection of claims each representing a certain piece of information. The token will be cryptographically protected to assure that the identity data come from a trusted source.

With this model, the application will not need to authenticate the user, store the relating data, look for data and integrate with other identity stores.

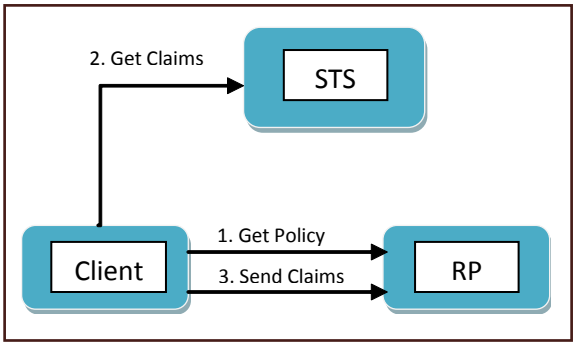
This model might ring a bell, because this is exactly what happens whenever we enter a foreign country using a passport. The approval to enter the designated country begins with the fact that this country trusts our home country. When we try to enter the country, we present a token (i.e. passport) with some information about us (i.e. name, age, etc.) to the immigration officer. He does not authenticate us and he does not need to look for identity information in any identity management store. Everything is written right there. He has all the information he needs. He will check that the token is legitimate and if so, based on the initial trust and the information we presented, will let us in and maybe log the entry.

The model is simple and has many advantages for the application. However, one particular advantage must be mentioned here: the idea of federation. Using a claim- based authorization, federation is much easier to implement as will be shown later in the article.

Before we dig into the details of the model, lets us review its terminology:

Identity: As stated at the beginning of this article, Identity is a set of information that describes an entity.

Claim: A claim is a piece of identity information such as a name, e-mail address, age, permitted task, etc.



The reason why the word "claim" is used and not the traditional term "attribute" is that it is not the application that goes looking for the data, but it is the user who presents the claims to the application. The application must first examine the claims with a certain measure of doubt. Only after the application is absolutely sure that the claims have originated from a trusted source it can use them. This is exactly what happens when the policeman checks the picture in your passport.

So each claim has an issuer, and it will be trusted as much as the issuer is trusted.

Security Token: A security token is a serialized set of claims digitally signed by the issuing authority.

Issuing authority: An issuing authority is someone who knows how to issue security tokens.

The issuing authority must have enough knowledge about the entity to issue the proper claims for the target application to use. The issuing authority might integrate with some other identity management systems to be able to create the claims.

The issuing authority is the body to which the identity management is outsourced, and as such it must be trusted. When authentication is factored out of the application by relying on claims, the application is actually passing responsibility to the issuing authority to authenticate and manage entities on its behalf.

Security Token Service - STS: The infrastructure that builds, signs and issues security tokens according to interoperable standards and protocols. The STS wraps the issuing authority as a service that supplies tokens. The service has quite a bit of technology to implement in order to create secure and interoperable security tokens.

Relying party – RP: A Relying party is anyone who relies on the issuing authority and uses its claims to give service to some entity. The Relying party is the target application that was described in throughout this article.

The basic scenario:

Here is an example of a claim-based system in action:

The relying party exposes policy that includes a list of claims that the relying party needs, for example a user name, e-mail address, and role memberships. The policy also tells the client the address of the STS where it should retrieve these claims. After retrieving this policy (1), the client makes a request (2) to the STS for the claims that the relying party has asked for. The STS authenticates the user and returns a security token containing all the claims the relying party needs. The client then makes the request to the relying party (3), sending the security token along with the request for the service. The relying party will validate the token and use the claims for servicing the client and create the response.

Interoperability

It is vital that the STS is interoperable, as many different kinds of user and relying parties will use its services. Several WS*-standards are used in the above scenario. Policy is retrieved using HTTP GET, and the policy itself is structured according to the WS-Policy specification. The STS exposes endpoints that implement the WS-Trust specification, which describes how to request and receive security tokens. Most STSs today issue SAML tokens (Security Assertion Markup Language). SAML is an industry-recognized XML term that can be used to represent claims in an interoperable way.

Adherence to standards allows to communicate with an STS on an entirely different platform and to achieve single sign-on across many applications, regardless of platforms. If you want to create an STS, you can just purchase one instead of building it yourself; also it would be interesting to use public STSs like a government's STS that would supply an electronic ID exactly as it does supply passports today.

Federation

When building a claim-based application, it easy to implement federation as the application is decoupled from the identity management and stores. All the application needs is a token from a trusted source. It does not care how the client got the token.

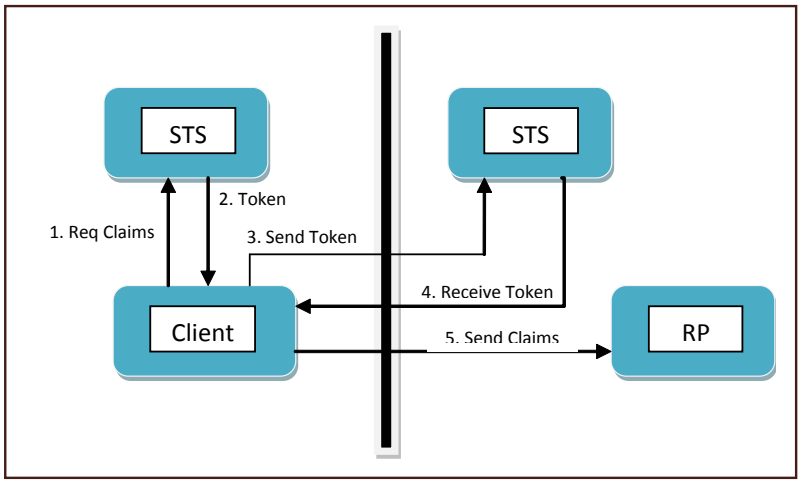
Federation is useful when an application needs to give service to customers outside of its natural domain. For instance, an internal service given to the company's employees is extended

and needs to be given to customers outside of the company. The company does not want to manage those external entities in its identity management systems.

The service policy is very simple: Present a token created by the company's STS and you will be served. For the internal employees it is no problem to get such a token, as they are managed inside the company and the company's STS "knows them". External customers are not managed inside the company, but they are managed somewhere else. If the company's STS trusts an external STS where the external customers are being managed, it will agree to exchange a token given to the external customers by their home STS with another token to be used by the application. Now all the external customers need to do is to present that token to the service and they will be accepted.

Federation enables to broaden the business boundaries traditionally enforced by the identity management infrastructure.

Claim-based implementation.



I hope that by this time you are all convinced that claim-based authorization infrastructure is what you need. The question is where to find it?

Microsoft is about to introduce a family of technologies under the name "Geneva". http://msdn.microsoft.com/en-us/security/aa570351.aspx

1. Geneva Server – A ready-to-use STS.
2. Geneva Framework – A complete object model for building claim-based applications and STS.
3. CardSpace Geneva – An Identity selector to enhance to user usability
   - Helps users manage multiple identities for the Web
   - Helps users select an appropriate identity for a given relying party
   - Protects user privacy
   - Gives consumers non-phishable credentials

IBM has a well-known Identity management infrastructure called Tivoli. It is not a surprise that Tivoly supports claim-based applications: http://www.ibm.com/developerworks/tivoli/library/t-ucitfim2/

http://www-01.ibm.com/software/tivoli/features/soa/soa-mgmt/secure-web-serv.html

Oracle Identity Federation is Oracle's implementation of the claim-based model:

http://download.oracle.com/docs/cd/E10773_01/doc/oim.1014/b25355.pdf

The beauty is that all these technologies interoperate with each other using WS*-standards.

I would like to end this article with an optimistic view to the future and hope that, after years in which the identity issue was forgotten, with claim-based authorization identity will be handled better. Users will feel more secure as applications will handle their identity better and damage from identity theft will decline.

Biography

Manu Cohen-Yashar is an international expert in application security and distributed systems.
Currently consulting to various enterprises worldwide and Germany banks, architecting SOA based secure reliable and scalable solutions.
As an experienced and acknowledged architect Mr Cohen-Yashar was hosted by Ron Jacobs in an ARCast show and spoke about interoperability issues in the WCF age.
Mr Cohen-Yashar is a Microsoft Certified Trainer and
trained thousands of IT professionals worldwide.
Mr Cohen-Yashar is the founder of an interoperability group in cooperation with Microsoft Israel in which distributed system experts meet and discuss interoperability issues. http://www.interopmatters.com/
A wanted speaker in international conventions, Mr Cohen-Yashar lectures on distributed system technologies, with specialization on WCF In which he is considered one of the top experts in Israel.
Manu won the best presentation award in CONQUEST 2007.
Mr Cohen-Yashar is currently spending much of his time bringing application security into the development cycle of major software companies (Amdocs, Comverse, Elbit, IEI, The Israeli defense System...)
Mr Cohen-Yashar is giving consulting services on security technologies and methodologies (SDL etc)
Mr Cohen-Yashar is known as one of the top distributed system architects in Israel. As such he offers lectures and workshops for architects who want to specialize in SOA, and leads the architecture process of many distributed projects.

BE SAFE!
START SECURE SOFTWARE ENGINEERING

Secure software engineering has become an increasingly important part of software quality, particularly due to the development of the Internet. While IT security measures can offer basic protection for the main areas of your IT systems, secure software is also critical for establishing a completely secure business environment.

Become ISSECO Certified Professional for Secure Software Engineering to produce secure software throughout the entire development cycle. The qualification and certification standard includes

- requirements engineering
- trust & threat modelling
- secure design
- secure coding
- secure testing
- secure deployment
- security response
- security metrics
- code and resource protection.

PLEASE CONTACT:
Malte.Ullmann@isqi.org

WWW.ISSECO.ORG

ISSECO®
SECURE SOFTWARE ENGINEERING

iSQI
International Software Quality Institute

The Liability of Software Producers and Testers

by Julia Hilterscheid

A recent decision taken by the German Federal Court of Justice regarding the liability of a freelancer working for a company indirectly effectively reverses the principles relating to the liability of software producers and testers, which had been applicable so far. If certain services or insufficiently tested products cause damage to the customer and require that the customer's employees have to rectify this during their regular working time, the customer can now hold the causer(s) of the damage liable with greater chance of success.

What does this mean for customers, software companies and for testers?

There are various reasons why software products end up being deficient. Firstly, it can and does happen that the requirement definitions are inadequate. When specifying the requirements for a project, the prime concern is the careful planning of the development process and a precise specification of the properties of the product in the contract. Clearly, many customers and contractors are aware of this important criterion. It is, however, all the more amazing that projects are frequently started in a "vacuum", i.e. that contracts or project documentation are only available in a rudimentary form. This means that the requirements for the program were not specified in a way that makes them capable of proof, and it will afterwards be difficult or downright impossible for either party to determine whether the delivered product is in accordance with the objectives, i.e. whether or not it is free from defects.

Another reason why software is often ends up being faulty is that the detailed concept has not been performed properly.

On the basis of a schedule, which is inevitably drafted as a rough concept at the start of the project, and in which the contents and the principal project milestones are specified, the detailed concept is created, in which the various functions are defined. Whilst the rough concept is drafted in coordination between customer and contractor, the technical department, the responsible marketing personnel and the designers, the fine concept serves the technical department and the programmers as a guideline for designing the individual steps. This includes the description of the data architecture and the business logic of the system. Based on the detailed design, the programmers will develop the procedures and functions as well as the database structures required. It is therefore possible for all project stakeholders to look at the detailed design and inform themselves at any time about contents and objectives of the project and to align their activities accordingly.

In order to develop software that is in accordance with the customer's requirements, a well functioning quality management is indispensible. Defects introduced into the software during the definition phase and programming cannot be discovered if the test process is not managed by a professional test manager.

In the development of software, the time pressure also plays an important role as a possible cause for bugs. Each tester knows the panic when time is running out at the end of projects, when the development or release of software is delayed, and if a contractually agreed delivery deadline cannot be met. The customer already threatens with a claim for damages, or – even worse – has even been able to push through a penalty clause that automatically applies when the deadline is exceeded.

As a result, management, technical departments and sales department put pressure on the tester, especially in cases where the delivery deadline cannot be put back e.g. due to legal requirements. Very often, the tester is not even responsible for the delays that have occurred; these could have been caused by bad project planning, late software requirements specified by the customer, or inadequately defined requirements. Nevertheless, the tester is the person that has to somehow cope with the virtually impossible task of delivering a product of at least satisfactory quality. Quality - due to the fact that everybody knows that quality cannot be added by the testers. He will brave the gap and hope that the defect will not occur at the customer as often as it has during testing, and he will have to release the software – albeit reluctantly. For the customer this situation can result in severe implications resulting from the defect.

Up to now, „only" the customers of the software producer had to bear the brunt in the wake of inadequately performed or omitted testing, e.g. if the program did not implement the requirements of the business processes. Development deficiencies of this type not only lead to increased customer costs, but quite frequently also to situations where the customer's employees have to try and rectify the software deficiency during operation. This had the effect that the employees were often less efficient and motivated. In addition to this, an inadequate performance on part of the software producer has an impact both on the customer's and the software producer's external presentation, since both parties can end up with a bad reputation. Moreover, it was the aggrieved customer who was usually the one paying the bill, since he was the one who had to prove judicially or extrajudicially, which work was left undone as a result of the rectification of the damage and the financial loss that has been caused by this.

Up to now, customers have only rarely been in

a position where they could assert a claim for such financially difficult to quantify damages – or if at all then not for the full amount of the damage. The time and effort expended on the rectification of faults in operation, which have already occurred or are expected to occur, was difficult to prove. This was due to the fact that the customer's employees were in the middle of the rectification process before they realized that the work to be performed and the resulting costs to be incurred were threatening to become a bottomless pit. In addition to this, employees usually still do not log the activities they had to perform at what time and for how long, in order to rectify the defects. In court, however, this sort of evidence was a prerequisite for being able to prove the circumstances and therefore for winning the case. Another problem, which in the past was certainly even more decisive in these cases, was that the customer's employees were drawing a salary anyway, which meant that the costs incurred through correcting the defects were not accounted for separately. This means that the damage was not one that could be proven in court.

Due to the uncertain outcome of court actions and because of the time and money involved, customers up to now often refrained from trying to assert their claim, especially since – as we all know – "software is always bound to be faulty". Therefore, software producers have traditionally had an advantage over their customers when the payment of damages was negotiated in court settlements.

This has now fundamentally changed due to the decision taken by the German Federal Court of Justice.

In order to successfully assert claims for damages, it is meanwhile sufficient to present the effort of the employees required for rectification of the damage and the expected effort required due to future malfunctions in operation. According to the Federal Court of Justice, the software producer, as source of the damage, must not gain an advantage from the fact that the customer's employees are in any case employed and paid by the customer.

For software producers, this will mean that in future more accurate planning of the projects will be necessary if the projects are not to become economically unviable through successful claims for damages asserted by customers. The deliverables of the software producer must therefore already be specified as precisely as possible in the contract, e.g. through a clear requirements specification. This is also of advantage for the customer, because the accurate descriptions of the deliverables also makes it possible for the customer to prove – if need be - that the software producer has not fulfilled the contractual obligations.

Furthermore, all necessary software tests must be planned at an early stage and must be performed with due care. A professionally organized quality assurance process tries to achieve the highest possible test coverage at an acceptable risk with the lowest possible number of test cases, in order to deliver the best possible software to the customer within a limited time period.

Provided that all these aspects have been taken into account, it is likely that any attempts by enterprises to claim damages against the software producers will be destined to fail, even after the new decision of the Federal Court of Justice.

Biography

Ms. Hilterscheid has been solicitor with practising licence since 1997. After stays abroad and studies in law in Berlin, she founded first the solicitor's office Hilterscheid, and one year later, together with her husband, the consultancy Díaz & Hilterscheid Unternehmensberatung GmbH.
In addition, Ms. Hilterscheid has been teaching media law and copyright at the Berlin University of Applied Science.
Ms. Hilterscheid has been supporting enterprises for more than 10 years in the discretionary formulation of contracts, general terms and conditions and license agreements, and accompanies projects legally. She ensures for her clients that legal requirements are adhered to in their correspondence, on-line appearance as well as in their advertising and marketing activities.
Ms. Hilterscheid also offers seminars on the subjects of IT-law, trademark law, copyright and labor law, which can take place in-house if so desired.

Application Security Fundamentals

by Joel Scambray

As the importance of security continues to dawn on the Software Industry 2.0, organizations of all sizes are trying to discover what constitutes software security "due care" for their customers. This brief paper will review key principles surrounding security in the development lifecycle (SDL), covering economic drivers, industry benchmarks (or the lack thereof), a prototypical SDL model, and what we've seen work/not work with real-world SDL implementations.

Few question that software occupies an increasingly central role in our everyday lives. From computer operating systems and applications, to mobile phones, television, the Internet, VoIP, GPS navigation, software-driven medical systems, air traffic control, the electric grid, and so on, human activity (and perhaps even human existence itself) has come to rely heavily on software.

But is that reliance justified? As more and more of our lives becomes digitized, and headlines have begun to trumpet the growth of malicious hacking and other incidents of cyber abuse, deep questions surrounding the confidentiality, integrity, and availability[1] of digital data have been raised. Are the risks underlying this brave new world greater than the rewards?

In parallel, software development organizations of all sizes are trying to discover how to protect their end users from unreasonable risks. Of course, this raises yet another question: what constitutes "reasonable"? Put another way, what is the standard of software security "due care"?

To date, the software industry has adopted the following fundamental approaches to establishing this standard:

1. Ignore
2. React
3. Prevent

Let's examine each of these approaches briefly to set the stage for a deeper discussion of security in the software development lifecycle (SDL).

Ignorance is Bliss

The dirty little non-secret of the technology industry is that few software development-oriented companies are doing anything serious about security. Recent surveys suggest that, despite some uptake of outsourcing and tools, most firms do not allocate significant budget or headcount for application security outside of standard operational IT security processes [1]. Although some in the information security industry would bristle at the implication, the question remains: Is ignoring the problem simply good risk management?

1 Confidentiality, integrity, and availability (CIA) are often cited as the defining properties of information security. Some authorities also include a fourth "A" for "accountability," typically understood to refer to the keeping of tamper-resistant activity logs to provide non-repudiation.

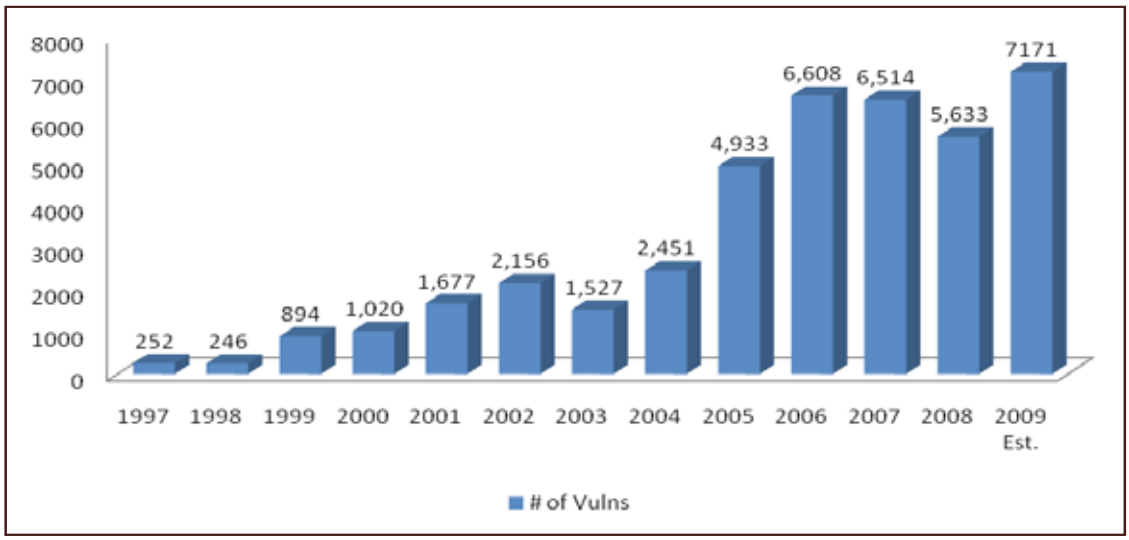


Figure 1: Software vulnerabilities released between 1997 and 2009 (extrapolated), courtesy of National Vulnerability Database

Data on security incidents or breaches has historically not been tracked systematically (with some recent exceptions, albeit focused on operational breaches rather than purely software vulnerability-related [2]). There is good news and bad news in this recent data: while only 6 out of 90 confirmed breaches (derived from over 150 cases) "…resulted from an attack exploiting a patchable vulnerability… the bulk of attacks continues to target applications and services rather than the operating systems or platforms on which they run." (ibid) So, if you're a major operating system vendor, take heart, but if you're writing custom application code, you're increasingly the target of attack.

Another quasi-informative dataset is the National Vulnerability Database (NVD), which tracks advisories on software vendor bulletins [3]. Figure 1 shows the raw count of vulnerabilities tracked in NVD, across all vendors and products, including software flaws (not configuration flaws), across all sources, from 1997 to 2009.

Clearly, the number of visible vulnerabilities is on the rise. Presumptively, based on unrelated studies showing software sales to be dominated by a handful of vendors, most of these vulnerabilities emanate from the same group of vendors. (We haven't done research confirming this.) Given all this activity, does it make sense simply to "vanish in the noise" if you are a small or medium sized software shop, and simply invest minimally in, say, outsourced professional security review during the release cycle? What is the return on investment (ROI) for security in the development lifecycle? We'll return to this question later, but will simply note at this point that there is a robust world-wide research community waiting for you out there.

To close this discussion of the merits of ignoring the software security problem, although it seems counterintuitive to assert positive outcomes from such a mindset, we've found minimal data to quantify the penalties of such an approach for small- to mid-sized software development efforts. Larger-scale development organizations with widely-deployed products are a different story, and anecdotal evidence exists to support investment in security, a topic we'll return to later. Next, we'll examine the other two postures, reactive and preventive.

React vs. Prevent

Many software firms have observed real-world security quality improvements resulting from external security review, and have hired penetration testers to assess their products, typically keeping the results private and selectively fixing some or all of the vulnerabilities. Although these practices can be laudable when performed in conjunction with other measures to be discussed momentarily, simply finding and fixing bugs iteratively between releases is not necessarily the most efficient way to increase code security quality. In fact, it's arguably less efficient than "learning to fish", in other words adapting a culture of prevention and the processes & technology to support it.

This is the heart of the justification for SDL. "Baking security in" rather than "bolting it on" in theory leads to better outcomes for all involved, including the development organization and its customers. Next, we'll describe in more detail the components of SDL and how it drives these outcomes.

SDL History and Philosophy

Of course, the notion of "baking security in" has been around for some time. Some of the "classic" antecedents of security in the development lifecycle include:

- NIST SP 800-64 [4]
- BS7799/ISO17799/27001-2 [5]
- OCTAVE [6]

In our opinion, ISO 17799 Sec 10 and ISO 27002 Sec 12 remain classics from an SDL policy perspective, including such fundamentals as separation of test and production environments, input/output validation, cryptography best practices, transaction integrity/non-repudiation, and so on.

More recent iterations of security in the development lifecycle frameworks include:

- Microsoft SDL [7]
- CLASP [8]
- BSI-MM [9]
- OpenSAMM [10]

Microsoft's SDL is among the most widely recognized currently, although there has been substantial recent attention for the other frameworks in this list (which share some overlaps in pedigree [11]).

SDL Principles & Framework

Obviously, there are a number of approaches to security in the development lifecycle, going back several years. It is therefore more realistic to think of SDL as a framework, or set of principles, that specific organizations can adapt and customize to their own unique purposes. Even Microsoft's SDL documents refer to their "implementation" of SDL. Below we attempt to summarize some of the high-level principles of SDL that are common to many of the above-mentioned frameworks:

1. Distribute the work of assessment and remediation, especially to the development team
2. Independent (of the development team) reviews at key milestones
3. Provide relevant training and re-usable guidance (checklists)
4. Strive for quantitative risk management, and set thresholds
5. Leverage automation

The first point articulates the overall strategy of SDL: accountability for the security quality of software needs to reside primarily with the developers of the software. This creates incentives to make continuous improvements to security quality in the long term. Alternative accountability models, such as where the internal corporate security team takes responsibility for software security, don't scale well in our experience because of conflicting incentives between the business (release feature-rich software to customers) and risk management interests (ensure that security quality is high).

Of course, security assurance cannot be outsourced entirely to the development function, as that creates a "fox guarding the chicken coop" situation (i.e. lack of appropriate segregation of duties). So, point 2 notes that reviews conducted by (or overseen by) parties independent of the development team are necessary at key milestones. For example, the corporate security team could conduct pre-release penetration testing independently of the development team and track the remediation of identified issues.

Point 3 is perhaps self-evident, but nevertheless important: people have a hard time doing the right thing if they aren't told what's the right thing to do. Development team security training (including program managers, testers, and managers) is thus an important component of any SDL implementation. Training programs should provide job-relevant curricula, track comprehension (ideally linked to application on-the-job), and be supported by re-usable guidance, code libraries/routines, and checklists that developers can easily access on the job to enforce good behavior. Application security training could be the topic of an entirely separate discussion, so we'll say little else about it going forward other than to reiterate its importance to the success of the overall SDL effort.

Point 4 acknowledges that information security practices continue to evolve towards more mature, quantitative risk management approaches. These same principles are ideal to apply to software security assurance as well. For example, Microsoft's DREAD [12] risk rating system strives to quantify the severity of software vulnerabilities, and thus define the priority of remediation efforts. (DREAD is somewhat proprietary to Microsoft, but is illustrative of the concept of quantitative assessment; other risk quantification systems include CVSS2 [13], FAIR [14], and FMEA [15].) Beyond just the scoring system itself, it is important to establish thresholds for prioritization, or to put it colloquially, a "bug bar." The bug bar essentially defines for an organization the thresholds at which work will be done to remediate a flaw. It can be immensely helpful to define thoughtful thresholds like this in collaboration with all stakeholders in advance of performing assessments, to avoid disagreements over how to remediate flaws (resulting in delayed release, unacceptably risky flaws in released code, or both).

Point 5 needs little explanation. Automation yields greater efficiency, and SDL is no exception. Some key areas with high potential for improvement through automation include:

- Security code review (although the accu-

racy and relevance of output from common tools remains suspect)

- Fuzz testing (to be defined later)
- SDL process automation, e.g. self-help web portals for workflow management

We've included a brief overview of application security tools at the end of this article.

Pitfalls

We've covered some key SDL principles that can help improve the chance of good outcomes. Are there any practices that should be avoided?

Anecdotally, one of the main reasons for failure of an SDL initiative is lack of focus on benefits to the organization, and by extension, its customers. Many organizations approach SDL assuming that the implied virtues of more secure software will simply make it acceptable to all stakeholders. In addition, there is historical disagreement around whether focusing on return on investment (ROI) for security is worthwhile or achievable (we believe economic justification is imperative, and will return to this concept later). To counteract this tendency, we recommend developing a good "scouting report" on all stakeholders (especially customers!), how they perceive SDL, their key objectives, and expected performance indicators. Often, this basic research can point to simple and easily implemented initial steps that result in easy wins, good momentum, and a strong head start towards a sustainable SDL program.

Culture shock can also torpedo SDL implementations. The culture of software development generally resists structure and discipline, and specific group dynamics can present even further challenges. Often, security is perceived from the start as an outsider, and the various behavioral changes proposed within the SDL initiative are thus viewed with suspicion at the outset. Be prepared to adapt your specific SDL implementation to the general and specific culture of development within your organization to bypass culture shock and outsider perception out of the gate.

Those chartered with initiating SDL are often tempted to set unrealistic expectations for SDL outcomes in order to address the outsider perception issue. Obviously, this is not recommended. Software development cultures are often focused tightly on schedule and resource allocation, and individuals who mismanage those two fundamentals often sacrifice substantial reputational capital that is very difficult to re-acquire for subsequent release cycles.

Lack of alignment with other security initiatives can also introduce "audit fatigue" amongst developers, who typically bristle at being interrupted multiple times for what they perceive is the same issue. One of the typical examples here is web development shops that have to comply with PCI DSS. [16] They are faced with complying with both SDL and PCI-related security initiatives separately if those programs are not well-coordinated.

Finally, and perhaps most importantly, organizational governance is often neglected in the design of SDL programs. The common wisdom is to "get executive buy-in" for initiatives of this nature, and this is of course important. However, development cultures are more often driven by "bottom-up" perceptions, so it's important to consider lobbying of all stakeholders early and often.

SDL Implementation Examples

What does an SDL implementation look like in practice? As we've proposed, it should be well-aligned with the existing development rhythm and culture. Figure 2 shows a mock development lifecycle for a large enterprise.

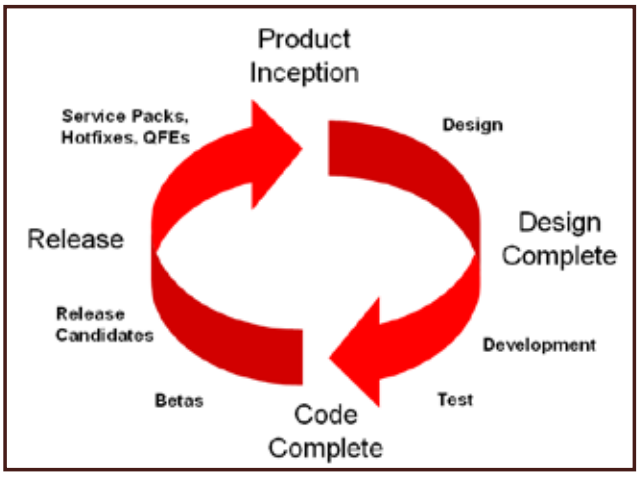


Figure 2: A mock large enterprise development lifecycle

Beginning with the lifecycle in Figure 2 as a baseline, in Figure 3 we overlay some common SDL milestones. This is one possible implementation for a large-scale organization with substantial resources.

It's important to note how Figure 3 aligns with the SDL principles we articulated earlier:

Figure 3: A sample SDL implementation overlaid on top of the previously introduced mock development cycle.

Principle	Implementation in Figure 3
Distribute the work of assessment and remediation, especially to the development team	A Security Liaison is assigned at project inception, who will be accountable for managing security workflow throughout. [2]
Independent review at key milestones	The red lines indicate milestones where independent review can occur. Note that these are closely aligned to existing development process gates.
Provide relevant training and re-usable guidance (checklists)	Training is an SDL gate that occurs early in the cycle. [3] Also, the "Build Standards" gate at the "Test" milestone illustrates an opportunity to provide re-usable checklists.
Strive for quantitative risk management, and set thresholds	The iterative nature of the overlay cycle provides multiple opportunities to check metrics (such as DREAD score mitigation) during the current and in future releases.
Leverage automation	A number of gates could require automated checks, such as the "Security Testing" and code review milestones.

2 Note that the security liaison manages workflow, not security outcomes, such as code security quality and other metrics. Ultimately, the development team leadership/executives are accountable for outcomes.

3 In practice, the number of developers who receive required training fluctuates between and within cycles, but the idea here is to enforce training early in a given cycle to ensure people have the training they require to do their jobs.

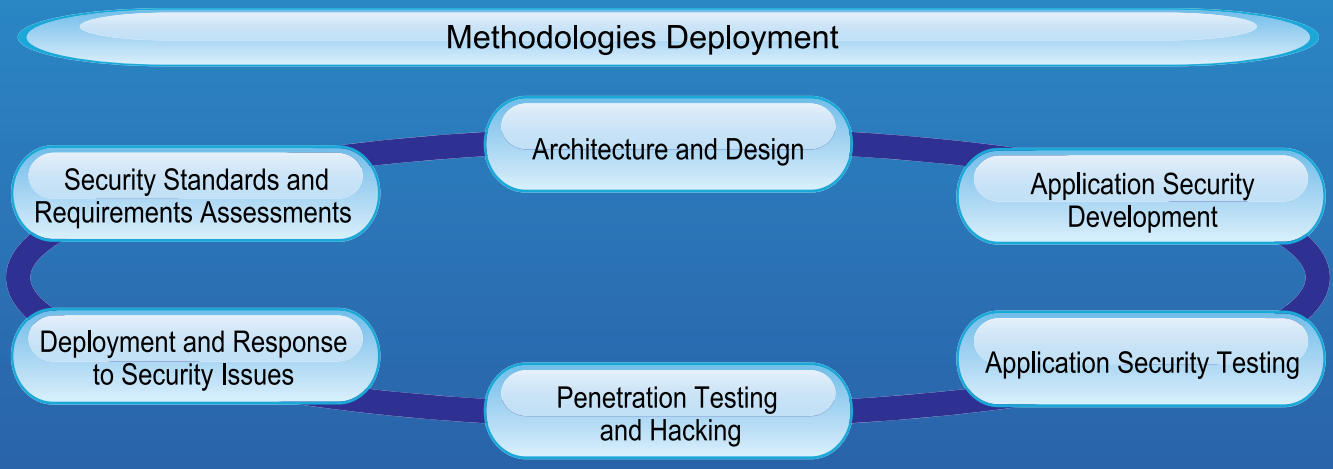
We See The Whole Picture...

Addressing Application Security from a Holistic Perspective

Applications today can no longer rely on the infrastructure to secure their assets. The applications must be built with integrated security design and that is a great challenge.

We can help you to secure your software!

SELA, a founding organization of ISSECO®, offers you comprehensive services helping you to integrate application security into the product life cycle. The services are provided for each and every step of the development life cycle:



Available Courses:

- ISSECO® Certified Professional for Secure Software Engineering
- Application Security Testing
- Application Security Design
- Introduction to Information Security
- Application Security using C++
- Web Services Security
- Windows 7 Security
- Application Security using the Microsoft .NET Framework
- Vista Security

Next ***ISSECO Certified Professional for Secure Software Engineering*** Courses

Country	*City*	*Date*
Israel	Tel Aviv	August 4-6
Canada	Toronto	September 23-25
India	Pune	October 28-30
Singapore	Singapore	November 2-4
Israel	Tel Aviv	November 18-20
Canada	Toronto	November 25-27

solutions@sela.co.il
www.sela.co.il/en

ISSECO SECURE SOFTWARE ENGINEERING

SELA TECHNOLOGY CENTER

Microsoft GOLD CERTIFIED Partner
Learning Solutions
Networking Infrastructure Solutions
Information Worker Solutions
Custom Development Solutions

Although Figures 2 and 3 are illustrative of how SDL principles might be implemented on a large scale, we've stressed the importance of starting small with SDL and iteratively growing the program to a scale that is sustainable for a given organization. Figure 4 provides an example of a smaller-scale SDL implementation based on what we assert are the minimum components for success.

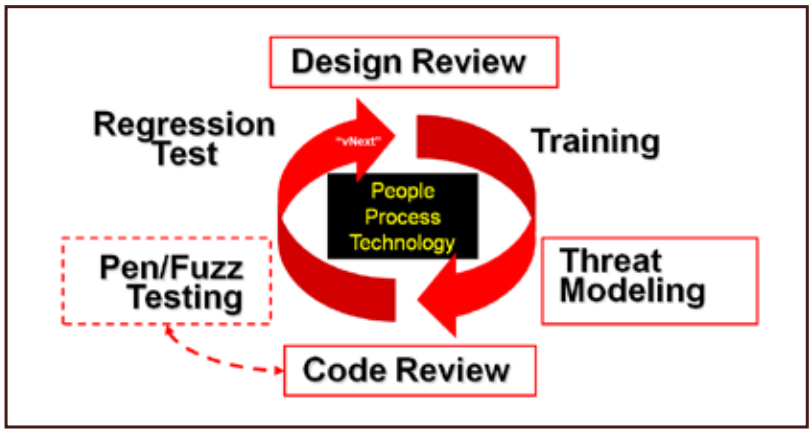


Figure 4: A light-weight SDL implementation example.

Note that many of the enterprise-scale SDL checkpoints (shown in Figure 3) have been eliminated in the example shown in Figure 4. The boxes highlighted in red in Figure 4 comprise an even more minimal SDL implementation made up of Design Review, Threat Modeling, and Code Review (with optional Penetration/Fuzz Testing). The terms shown in Figure 4 are defined in the Glossary at the end of this article.

Development Infrastructure

It's worthwhile to pause for a moment to highlight the importance of basic fundamental software development hygiene. Of course, many opinions exist (especially within development communities) about the exact meaning of "basic software development fundamentals," and we're not interested in starting such a debate here. Our primary point is that much of the theory and practice of SDL depend upon certain fundamentals being in place, and most SDL initiatives will not be successful without at least some structure to which it can be anchored. Some of the common key enablers of successful SDL implementations are listed in Table 1.

What About Alternative Programming Models?

The topic of software development fundamentals raises a popular question: Can SDL be applied successfully to iterative/unstructured Agile programming methods like Scrum and Extreme Programming? Absolutely, [17] but there is a point at which a lack of structure or too much "adaptive-ness" can hamper SDL. SDL is based on the premise that a minimal level of structure exists to which systematic evaluations of security quality can be anchored. If there is no structure to which anything can be anchored, then SDL will likely be challenging to implement. This again highlights the challenge of cultural integration for those without substantial development experience (e.g. security professionals attempting to implement SDL): it can be hard to differentiate natural resistance to change from active attempts to cover up an overall poorly managed existing development effort.

SDL ROI

To this point, we've discussed the "what" and some of the "how" of SDL. Before concluding, we'll take a step back and briefly survey the "why." Our basic premise is that, at least in business scenarios, the key driver of SDL should be economics, with emphasis on the "should be". Finding and interpreting data to support this contention is challenging.

Generally, there is a lack of systemic empirical evidence supporting measurable economic outcomes following implementation of SDL. Most studies to date have focused on hard operation costs yielding intangible benefits. For example:

- Savings-to-cost ratios ranging from break-even to 8 or 9 times [18]
- Average loss of 0.76% market value when a vulnerability is disclosed [19]
- Return on investment equated to 21% [20]

Other studies have shown the value of good design, user education, and automated response technology over finding and fixing bugs. [21, 21]

Microsoft has published data showing a sharp reduction in security bulletins published from Windows 2000 Server to Windows Server 2003. [23] Combining this data with separate claims by Microsoft that a security bulletin costs the company approximately $100,000 (in 2002 dollars) [24], one can impute that Microsoft saved approximately $3.7M due to their "Secure Windows Initiative" push that was one of the primary progenitors of their brand of SDL. Of course, this does not quantify tangible investment in SDL beforehand (let alone even as a percentage of overall spend); it just illustrates benefits in terms of hypothetically realized savings from un-issued bulletins.

Admittedly, this brief review of economic data in support of SDL has not done justice to the topic. Our sense based on anecdotal experience is that, like most things, the ideal risk/reward balance is not one-size-fits-all, but is rather best gleaned from experimentation and keen focus on tying SDL metrics to economic outcomes early and often. To end with one final piece of guidance on quantitative data supporting SDL, we paraphrase the Pareto Principle: Invest more in finding the "vital few" issues that cause the vast bulk of security vulnerabilities. We've provided one last table to help illustrate this point:

	Budget	Quantity Found	Quality Impact
React • Find & fix bugs • Bolt-on	20%	80%	20%
Prevent • Improve dev practices • Baked-in	80%	20%	80%

Table 2: Planning for SDL should focus on finding the "vital few" issues that cause the preponderance of security vulnerabilities.

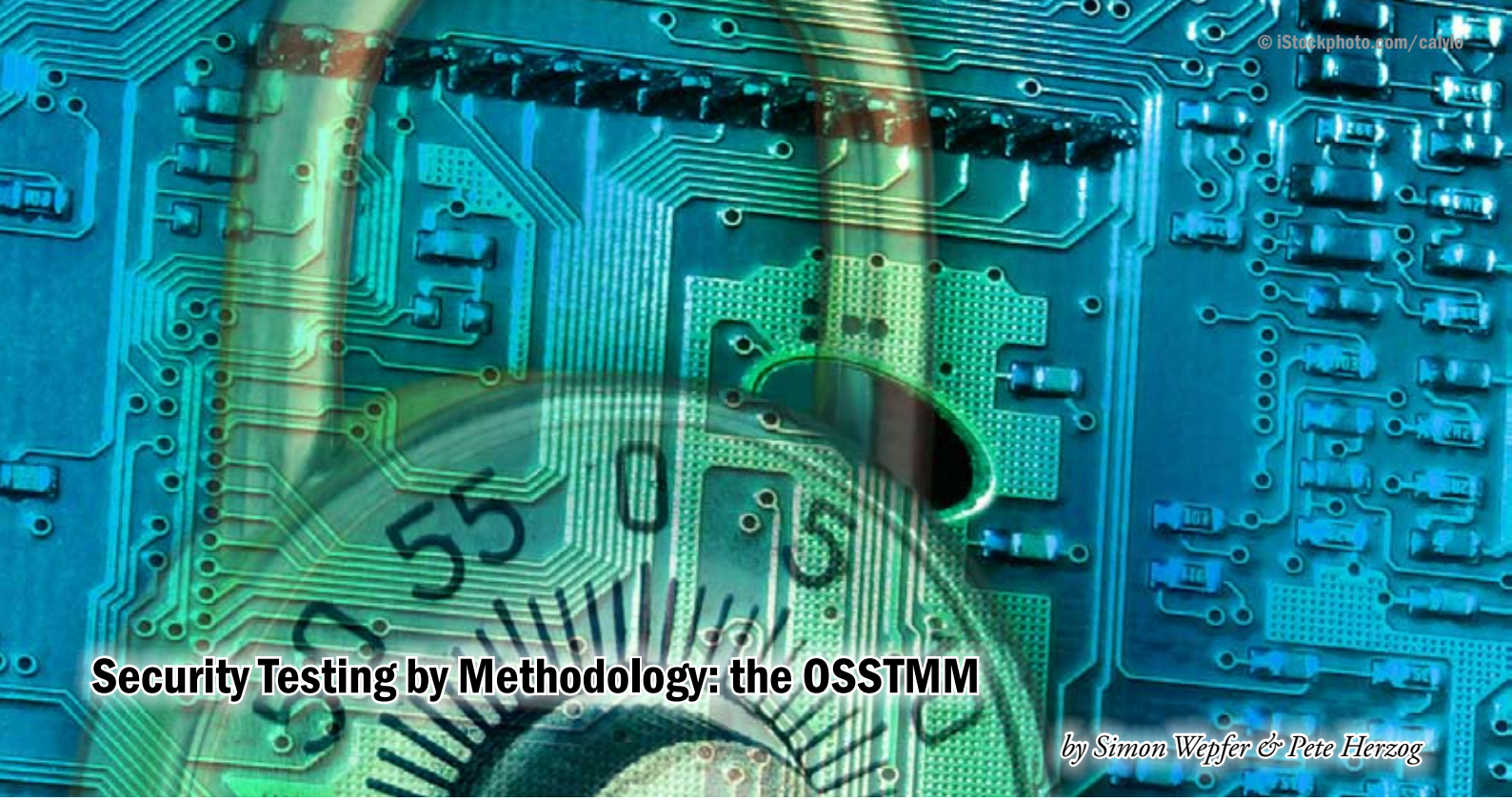
Fundamental Dev Practice	Assists SDL By
Consistent test, build environments	Separating test and production data, ensuring expected run-time parameters
Concurrent versions system (CVS)	Enforcing known-state versioning and providing reversion capability
Defect management system	Provides a central repository for managing and measuring defect reduction
Reporting	Provides consistent communication of data to appropriate decision-makers

Table 1: Development practice fundamentals that enable SDL.

Biography

Joel Scambray, CISSP, is co-founder and CEO of Consciere, provider of strategic security advisory services. He has assisted organizations ranging from small startups to Microsoft Corp. address information security challenges and opportunities for over a dozen years, in diverse roles including consultant, corporate leader, entrepreneur, and co-author of the Hacking Exposed book series.

The author wishes to acknowledge important contributions to this article from Andre Gironda, Birgit Lahti, and Kevin Rich.



Security Testing by Methodology: the OSSTMM

by Simon Wepfer & Pete Herzog

Security tests are an important part of the risk management process and executives realize the benefits of an independent security test: It introduces a neutral view on the target and can improve security when the proposed sensible measures are successfully applied. But there are often also questions to answer after such an audit.

How secure is the target, and are there aspects that have not been tested? How much has our security improved since the last test? How does our security compare to other companies in our industry? This article is a brief introduction into the Open Source Security Testing Methodology Manual (OSSTMM), which can answer these and other follow-up questions.

OSSTMM is a freely available manual that provides a methodology for a thorough security test of physical, human (processes) and communication systems. A core aspect are the security metrics – the Risk Assessment Values (RAV) – which express the final security level of the tested system as a numerical value. The current release candidate of OSSTMM 3.0 is an approximately 150-page document and is a complete re-write from the 2.X version series incorporating the results of the last 6 years of research.

The main purpose of the OSSTMM is to provide a scientific methodology for the accurate characterization of operational security and is adaptable for penetration tests, ethical hacking, security assessments and so forth. In the EU-sponsored project, Open TC, it became the standard for testing and measuring trusted computing systems. Most of all, an OSSTMM compliant test defines the target clearly, and results are reproducible, something unusual in the current methods of ethical hacking and penetration testing.

Preparation and Testing

Before the test can actually start, the assets that have to be secured must be defined. The protection mechanism for the assets are the targets to test. The engagement zone is the area around the assets, the test scope is everything needed to keep the assets operational, for instance, processes or network protocols. The test vector defines the interaction points of the scope. For instance, a DMZ may be tested from the internet or from the LAN as well – with obviously different results. Then, the testing channels have to be defined. Our example DMZ may be tested not only on the communication layer but on the process layer as well (e.g. patching process).

The test type defines the knowledge about the target and the test. Common known testing types are black box and white box; the OSSTMM, however, distinguishes six types, each detailing different results. The rules of engagement are protecting the customer and the tester on legal, ethical and procedural aspects.

When all the above has been defined well, it is clear which tests in the OSSTMM have to be performed on the scope. OSSTMM tests define only what is to be done, but do not dictate any tools. One test for data networks for example is requesting that server uptime has to be verified to latest vulnerabilities and patch releases. Another example is that responses to UDP packets with bad checksums to a collection of ports have to be verified. The tools to use and how to use them is left up to the tester.

Classification	Description
Vulnerability	is the flaw or error that: (a) denies access to assets for authorized people or processes, (b) allows for privileged access to assets to unauthorized people or processes, or (c) allows unauthorized people or processes to hide assets or themselves within the scope
Weakness	is the flaw or error that disrupts, reduces, abuses, or nullifies specifically the effects of the five interactivity controls: authentication, indemnification, resistance, subjugation, and continuity.
Concern	is the flaw or error that disrupts, reduces, abuses, or nullifies the effects of the flow or execution of the five process controls: non-repudiation, confidentiality, privacy, integrity, and alarm.
Exposure	is an unjustifiable action, flaw, or error that provides direct or indirect visibility of targets or assets within the chosen scope channel.
Anomaly	is any unidentifiable or unknown element which has not been controlled and cannot be accounted for in normal operations.

An OSSTMM compliant test is much more than running an automated vulnerability scanner and printing the report. It relies on the tester's in-depth knowledge and experience, and on human intelligence for interpreting the results. This does not mean that automated tools will not be used at all, but they will be used as what they are: just a tool without real intelligence.

Risk Assessment Value

Once a risk is detected and verified, it has to be categorized. OSSTMM is naming these limitations; the inability of protection mechanisms to work correctly, see table

OSSTMM knows five „risk" classifications

The limitations are one of the three factors for calculating the final RAV. The operational security is a second one, derived from visibility (a means of calculating opportunity for an attack), access (counting the interactive access points) and trust (fall-back to unauthenticated access to trusted systems). The third factor for calculating the RAV are the controls implemented for each point identified in the operational security section. Controls are grouped in class A (authentication, indemnification, subjugation, continuity and resilience) and class B (non-repudiation, confidentiality, privacy, integrity and alarm).

Certification

ISECOM (Institute for Security and Open Methodologies) offers several OSSTMM-specific certification and training schemes. The ISECOM Licensed Auditor (ILA) program provides quality assurance and support for obtaining OSSTMM certified audits from a properly accredited auditing company. OPST (OSSTMM Professional Security Tester) and OPSA (... Analyst) is a certification for persons. Additional information may be found on http://www.isecom.org/.

Biography

Simon Wepfer is COO at OneConsult.

Pete Herzog is founder of the OSSTMM and Director of ISECOM.

Agile TESTING DAYS

Berlin, Germany

The Agile Testing Days are the European conference for the worldwide professionals involved in the agile world. We are very proud to count on the support and the work of Lisa Crispin, Elisabeth Hendrickson, Tom Gilb, Stuart Reid, Isabel Evans, Tom and Mary Poppendieck and last but not least Alessandro Collino.

We decided to choose Berlin as one of the best connected capitals in Europe and also for its very good price/service ratio for hotels and flights. Berlin offers also a lot of fabulous places to visit in your spare time.

Please have a look at the program and enjoy the conference!

October 12
Tutorials

"Using the Agile Testing Quadrants to Cover Your Testing Needs"
by Lisa Crispin

"Managing Testing in Agile Projects"
by Isabel Evans and Stuart Reid

"State of the art and future of Agile Testing"
by Alessandro Collino

"Agile Inspection Leader - How to perform an efficient agile inspection"
by Tom Gilb

"Acceptance Test Driven Development (ATDD) in Practice"
by Elisabeth Hendrickson

"Designing a Lean Software Development Process"
by Mary and Tom Poppendieck

October 13
Conference

Key Note - "Are Agile Testers Different?"
Lisa Crispin

"Agile Testing: A Report from the Frontline"
Joke Hettema, Ralph van Roosmalen

"Open Source Agile Testing"
Péter Vajda

"The Agility GPS" Ulrich Freyer-Hirtz Enabling Agile Testing through Continuous Integration" Sean Stolberg Introduction to Robot Framework"
Pekka Klärck

"How to develop a common sense of 'DONE'?"
Alexander Schwartz

"Key Note"
Elisabeth Hendrickson (TBD)

"Identifying the value of Performance testing in an agile world!"
Mieke Gievers

"HtmlUnit: An Efficient Approach to Testing Web Applications"
Marc Guillemot

"Quality and Short Release Cycle"
Lior Friedman

"Agile Quality Management –Axiom or Oxymoron?"
David Evans

"'Testify' – One-button Test-Driven Development tooling & setup"
Mike Scott

"Test Driven TDD - TDD quality improvement based on test design technique and other testing theory"
Wonil Kwon

"Key Note" *Tom Gilb (TBD)*

October 14
Conference

"Key Note"
Mary and Tom Poppendieck (TBD)

"Improved Agile Testing using TPI"
Cecile Davis

"Configure a build server in 60minutes"
Emanuele DelBono, Alessandro Melchiori

"Learning is key to Agile success: Building a learning culture on your Agile team"
Declan Whelan

"The Missing Link for ATDD & Example-driven Development"
Gojko Adzic

"Promoting the use of a quality standard"
Eric Jimmink

"Test Driven Development of Embedded Software at MiPlaza"
Marcin Czenko, Wim van de Goor, Martijn Gelens

"Key Note" *Stuart Reid (TBD)*

"Bridging the gap between agile and traditional testing"
Anko Tijman

"Automated Integration Testingin Agile Environments"
Slobodanka Sersik, Dr. Gerald Schröder

"Agile Practices in a resistant-to-change environment"
Markus Gärtner

Clossing Session José Díaz & Alessandro Collino - Podium Diskussion

Please fax this form to +49 (0)30 74 76 28 99 or send an e-mail to register@agiletestingdays.com.

Agile TESTING DAYS

Participant

Company: _____

First Name: _____

Last Name: _____

Street: _____

Post Code: _____

City, State: _____

Country: _____

Phone/Fax: _____

E-mail: _____

Early Bird ends by June 30th, 2009

Billing Address (if differs from the one above)

Company: _____

First Name: _____

Last Name: _____

Street: _____

Post Code: _____

City, State: _____

Country: _____

Phone/Fax: _____

E-mail: _____

Remarks/Code: _____

Tutorial

- ☐ Using the Agile Testing Quadrants to Cover Your Testing Needs, by Lisa Crispin
- ☐ Managing Testing in Agile Projects, by Isabel Evans and Stuart Reid
- ☐ State of the art and future of Agile Testing, by Alessandro Collino
- ☐ Agile Inspections Leader - How to perform an efficient agile inspection, by Tom Gilb
- ☐ Acceptance Test Driven Development (ATDD) in Practice, by Elisabeth Hendrickson
- ☐ Designing a Lean Software Development Process, by Mary and Tom Poppendieck

Early Bird **600,- €** (plus VAT)

700,- € (plus VAT)

Conference

- ☐ 1 day (first day), 450* / 550 EUR
- ☐ 1 day (second day), 450* / 550 EUR
- ☐ 2 days 700* / 850 EUR

Early Bird **700,- €** (plus VAT)

850,- € (plus VAT)

Included in the package: The participation on the exhibition, at the social event and the catering in course of the event.

Notice of Cancellation

No fee is charged for cancellation up to 60 days prior to the beginning of the event. Up to 30 days prior to the event a payment of 50% of the course fee becomes due and up to 15 days a payment of 100% of the course fee becomes due. An alternative participant can be designated at any time and at no extra cost.

Settlement Date

Payment becomes due no later than the beginning of the event.

Liability

Except in the event of premeditation or gross negligence, the course holders and Díaz & Hilterscheid GmbH reject any liability either for themselves or for those they employ. This also particularly includes any damage which may occur as a result of computer viruses.

Applicable Law and Place of Jurisdiction

Berlin is considered to be the place of jurisdiction for exercising German law in all disputes arising from enrolling for or participating in events by Díaz & Hilterscheid GmbH.

Date

Signature, Company Stamp

Wanted: Technical Test Analysts

by Erik van Veenendaal

Security Testing

Security testing is a relatively new topic to me, but one that is becoming more and more important in today's internetbased society. Having studied the techniques and tools available for security testing during the last month, one thing became very clear to me: It requires extensive technical know-how and skills. It is not that difficult for someone with networking and/or programming background, but a domainbased tester will have difficulty in performing security testing and using the tools. This brings us back to a very fundamental discussion: "What is required to become a professional tester?". "Does a programming background help in becoming a better tester?". In the past the gurus have shown to be in disagreement on this. Having been a programmer myself, I have always found it to work to my benefit. It is easier to understand where defects can be found, it is easier to see technical risks and, of course, it is easier to communicate with a developer. In fact, most developers will take you more seriously if you are able to speak their language.

Agile development

With agile development becoming more and more popular these days, many are doing their stand-up meetings, 4-week sprints, developing user-stories etc. This also changes the role of the traditional test analyst. Within agile development, the role of a test analyst is often characterized by the following tasks:

1. Offer the business analyst support in making test cases
2. Offer assistance to preparing unit tests
3. Participate in design, code and test case reviews
4. Provide pro-active feedback
5. Develop and execute system test cases
6. Build automatic regression test suites
7. Make notes for retrospective meetings

Note that no less than three of these tasks (2, 3 and 6) require extensive technical knowledge. A tester that wants to survive in an agile team must understand unit testing, coding and have some knowledge of test automation. In agile projects, I see many so-called domain-based testers, who are not performing adequately in such an environment.

Test consultancy

Where a few years ago the focus for test process improvement was almost always on improving the system test and/or acceptance test, we nowadays encounter more and more clients who understand that if you want to improve testing, a thorough unit and integration test are also needed. To only improve the higher test levels is much less efficient. Also, different types of defects will be found by unit and integration testing, making the test process more effective. With this in mind, a test consultant needs to also be able to consult developers on improving their testing. Extensive knowledge of structure-based techniques and tooling such as dynamic analysis, static analysis and, of course, unit test frameworks are requirements for the 2009 test consultant.

The Technical Test Analyst

As a result of a number of trends, I see a need for testers with substantial technical background. Security testing has already been mentioned, but also testing other non-functionals such as reliability and performance requires technical skills. Agile development and the different focus of many test improvement actions are also important drivers for requiring more technical test professionals. The ISTQB Advanced Level scheme offers a complete 5-day module on technical testing which may seem irrelevant to many testers today, but I'm convinced will become more and more important to the testing professional in the near future or even already today.

I'm off to my daily scrum meeting, and will be reviewing a set unit test cases thereafter …….

Erik van Veenendaal is a leading international consultant and trainer, and recognized expert is the area of software testing and quality management. He is the director of Improve Quality Services BV. At EuroStar 1999, 2002 and 2005, he was awarded the best tutorial presentation. In 2007 he received the European Testing Excellence Award for his contribution to the testing profession over the years. He has been working as a test manager and consultant in software quality for almost 20 years.
He has written numerous papers and a number of books, including "The Testing Practitioner", "ISTQB Foundations of Software Testing" and "Testing according to TMap". Erik is also a former part-time senior lecturer at the Eindhoven University of Technology, the vice-president of the International Software Testing Qualifications Board and the vice chair of the TMMi Foundation.

Application Security – Money Still Being Squandered on It

by Serge Baumberger

Security is especially important in online applications, yet far too little attention is paid to it. Perform a quick risk assessment just prior to implementation, jerry-rig a plug for the biggest holes, and you'll have the software up and running in no time. Then a few days later, the first bytes of customer data are **stolen**. Does it have to happen this way?

In an incredibly short time, online security risks have become the primary risk for businesses. One reason is that nowadays **business-critical applications** have to be accessible online at any time to customers, partners, and internal users. Continuous availability opens the floodgates for friend and foe. Interactive contents exacerbate the whole thing. Statistics show that hackers don't have to be asked twice to do what they do. Currently, one of the favorite modes of attack is cross-site scripting (**XSS**). For example, a hacker tries to manipulate a web application in such a manner that damaging script code is embedded in the displayed page (e.g., in a guest book or an auction site page). The browser processes this actually trustable website including the harmful code and thereby sends the current login information back to the hacker. Even though companies take security absolutely seriously, reports regarding successful attacks are a daily occurrence. What's happening here?

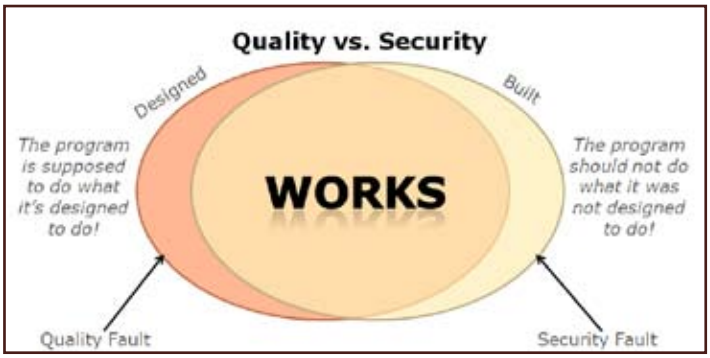


Fig. 1: Quality vs. security

Finally Rethinking The Situation

The lion's share (over 90%) of an IT security budget is invested in **network security**, e.g., for firewalls and intrusion detection systems, even though according to **Gartner**, 75 percent of the hacker attacks take place directly via the application and not the networks. To spend money specifically on where the greatest danger lurks requires rethinking the situation. Ideally, pains are taken in **software development** to ensure that no **security loopholes** even exist, or to remedy these immediately. **Those who are late with their testing are wasting money.** However, the problem isn't that easy to fix, because the software is geared to preferably provide what is described in the functional specifications. As a result, checks are performed on what the applications are supposed to do, but not on what they are not supposed to be able to do. Developers with expertise in the field of secure development are a rare breed. One of the rare examples of this can be seen in the development of new operating systems. **Testing** also has to be tailored to the new requirements. Unfortunately, one rarely sees a **penetration test** as part of a **test concept**. Likewise, the skills profile of a tester, who is supposed to verify the functionality, is different to that of a security tester. While one is looking for things that don't work, the other should be looking for the undesired functionalities that allow too much to take place – in other words, we're talking about real detective work.

WHAT IS SECURITY TESTING?

There are various definitions and terminology involved in this relatively new area of software quality assurance. Many choose to cut corners by including security constraints as functional requirements and test these in the same way as other functional requirements. Even more prefer to avoid the topic altogether and simply shift the responsibility to an external provider. This may prove to be a cost saver in the short run, but the actual costs of reduced security consciousness is reflected by the latest published figures. In order to implement and maintain a secure software application, dedicated security testing is essential.

Some terminology to consider in this regard:

Security test: The process to determine that an IS (Information System) protects data and maintains functionality as intended.

Penetration test: A penetration test is a method of evaluating the security of a computer system or network by simulating an attack from a malicious source, known as a hacker, or cracker.

XSS: Cross-site scripting (XSS) is a type of computer security vulnerability typically found in web applications which allow code injection by malicious web users into the web pages viewed by other users.

Solution Approaches

Making online applications secure requires specific measures:

1. Implementing security rules, guidelines, and regulations
2. Creating security requirements and attack scenarios
3. Performing specific security architecture reviews
4. Developing and complying with secure coding guidelines
5. Performing:

   White box penetration tests

   Grey box penetration tests

   Black box penetration tests
6. Monitoring operating systems continuously
7. Making assessments and feedback loops part of the first step

Tool Support

Once the activities are defined, they can be accelerated. Large-scale software manufacturers such as **HP** or **IBM** have recently shored up their capabilities in regard to security testing of web applications through targeted acquisitions. Certain fields of use may also have corresponding **open source solutions**. IBM, like HP, offers security-related tools that support an application's **entire lifecycle** from its creation to its replacement:

For developers:

- Code is checked for security while being entered
- Solution recommendations and links appear if requested

For testers:

- Automated security tests check web applications and services for flaws
- Any discovered flaws are saved with the corresponding priority depending on the security risk and are assigned to the responsible developer

For security experts:

- A 24/7 solution combs through complete web applications looking for security loopholes
- Automatic verification pertaining to legal, company-internal, and regulatory provisions

The tools described are becoming better and better. Experts scan the Internet daily for new risks. As for anti-virus solutions, the tools are kept current with new signatures and detection patterns by means of updates.

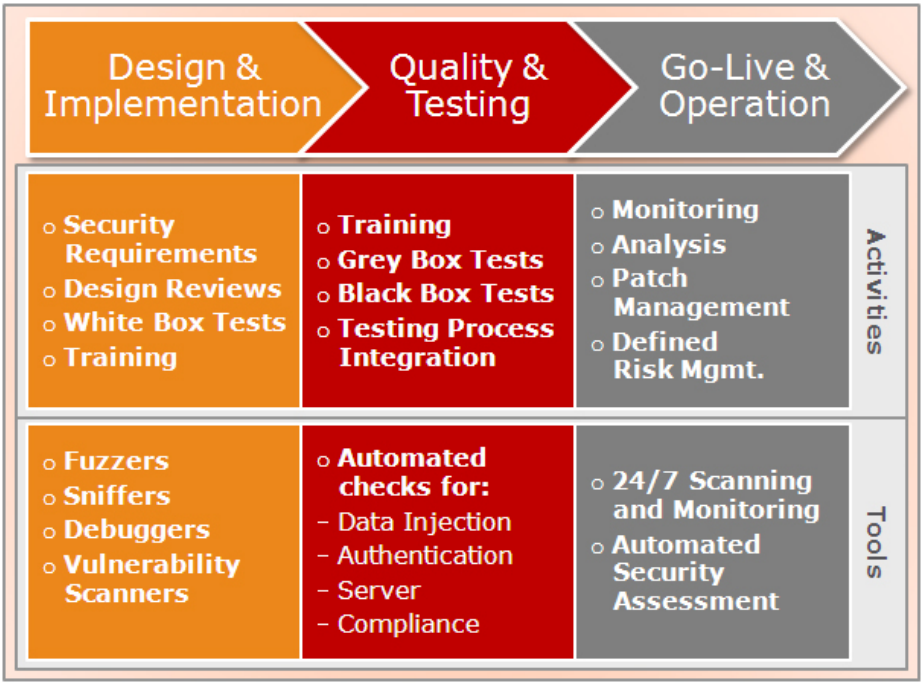


Fig. 2: Toolbox for actual implementation measures

Acute Need For Action

Secure programming and performance of penetration tests are still in their infancy. One of the reasons is that this topic is not properly addressed at universities. There most certainly is a need to make applications more secure and to include the entire **software development cycle** when creating secure applications. By implementing the solution approaches above, one can gradually increase the application security's maturity. However, it takes a considerable amount of time and money in the short term, because systematic training of the test and development units is necessary. In any event, setting up one's own security testing squad does make sense.

Summary

Targeted investments are necessary. Regardless of the maturity level an organization has reached, there is always room for improvement. The investment is certainly worthwhile because besides financial losses, the company's image is also at stake.

Biography

Serge Baumberger is a leading international consultant and trainer. He is a recognized expert in the area of software testing and quality management. Serge is head of Application Quality Management at beteo, which is a specialist IT-consulting company in Application Lifecycle Management.

Through his previous activities as project manager, performance engineer and test automation expert, Serge has many years of experience, especially in the area of Application Quality Management. Most recently he was test manager at Credit Suisse for the front products and had a leading role in building the Test Factory for the entire bank.

Serge is currently writing his master thesis about security-/penetration-testing in large companies.

© Jana Noack

Interview Mike Smith

- Testing Experience and the new 'Learntesting' online training service

An interview with Mike Smith, CEO of Testing Solutions Group (TSG)

In April, Testing Experience introduced a new concept in online learning by partnering with TSG to bring 'Learntesting' to the global market.

Jose Diaz interviews Mike Smith, CEO of TSG to find out how Learntesting is breaking new ground in the field of virtual learning for the testing community:

1. What is the new 'Learntesting' all about?

The first thing to say is that Learntesting itself is not new; it has been around for over 5 years providing online ISTQB Certified Tester Foundation Level training, fully accredited by ISEB! However, we have now used the experience of two previous implementations to transform the technology and business model into a completely new global service. As well as providing a range of online content to help testers achieve certification, the new Learntesting service provides ongoing support via its approach to Virtual Learning.

2. What do you mean by Virtual Learning?

There are many different terms used in this field, including distance learning, CBT (Computer Based Training), Virtual Education, Virtual Learning Environments (VLE) and Learning Management Systems (LMS). With Learntesting, we have built our own standards-based Virtual Learning Environment utilising an 'industrial-strength' LMS to support the current and ongoing needs of anyone interested in software testing. This environment provides:

- High-quality course content presented in a variety of formats to cater for different learning styles, including:
  - Tutor videos;
  - Automated slide presentations;
  - Professional voice-overs;
  - Course text pdf transcripts.
- Interactive quizzes and exercises with supporting voice-overs;
- Exam-style questions and answers with supporting explanations;
- Formal 'mock' exam assessments;
- Live Virtual Classes led by tutors to support higher-level learning objectives, course exercises and exam revision sessions;
- A communication centre, including:
  - email;
  - forums;
  - noticeboard.
- A private library of testing and testing-related ebooks;
- Administration and accredited tutor support from a growing 24x7 global network of accredited training providers;
- Self-registration system for students;
- Additional resources, including papers, presentations, templates for testers and sample exam questions;
- Provision of ISTQB Foundation Level exam vouchers redeemable at test centres throughout the world.

3. From your experience, what is important for successful online certification training?

There are many different ways that students can prepare for certificated exams. Study options range from simply reading the syllabus to sitting accredited courses, whether in the classroom or online. There are many books, sample questions and other support materials widely available at a range of costs.

From our experience, we use a term called 'horses for courses'; that is, a student's best option depends upon their previous experience, the level of certificate they are studying and their preferred learning style. Some experienced testers decide not to do any formal study, preferring to rely on their experience. Whist this works for some, others actually have real problems because their experience may be too narrow, or they have used a proprietary approach to testing not in line with the ISTQB.

Even with a prime objective of just passing the exam, not taking any study can be a false economy with expensive exam re-sits. Additionally, most large organisations that buy into the concept of certification are also buying into the concept of learning and development – not just proving that someone can pass a test on a given day.

With the above in mind, when providing online learning it is important to cater for a range of backgrounds. The most inexperienced students will need a lot of material available and ways of reinforcing the knowledge that they are acquiring. The more experienced students need ways of assessing their knowledge levels and easy access to the areas of learning in which they are weakest.

For all learners, it is important that the course provides an interesting learning experience. This is the most difficult challenge to anyone trying to build online training solutions. Although there is an increasing range of 'rapid authoring' tools available, it requires expert subject matter knowledge, online content design skills, a lot of time and a large budget to build successful online training materials.

Interactive quizzes and exam assessments are critical and in many respects, the most important feature for the success of online training. This involves a huge amount of development work, since setting good exam questions is hard enough – and providing the supporting information as to why each answer option is correct or not is an even bigger challenge!

With Learntesting, we have chosen to get our courses leading to certification accredited. This means we cannot take short cuts and need to cover all aspects of the syllabus as well as ensuring the online learning experience is good. Students of Learntesting are buying a guarantee that our courses properly prepare them for the exam. Individuals and organisations will soon be unhappy if the training fails to deliver this, and our pass rate at Foundation level of over 95% in the past 5 years is testament to the quality of the learning experience.

4. What sets Learntesting apart from other self-study training options?

With the growing demand for self-study options, it is becoming increasingly important to provide an environment that not only leads to the best chance of success in exams for students but also supports them in their career development and future certification needs. By investing in both high-quality online training and a virtual learning environment, Learntesting can deliver on both of these key objectives **and** provide a rewarding and enjoyable learning experience along the way.

In addition, the vast majority of self-study options aimed at the ISTQB Certified Tester scheme are not accredited by a recognised body. Learntesting has chosen to invest in providing accredited certification courses, so individuals and organisations can be sure that the learning experience is high-quality as well as providing in-depth coverage of the complete syllabus.

By creating its own virtual learning environment, Learntesting does not just offer online courses which are available to run on a one-off basis within a simple hosted environment. It goes much further by providing a number of value-adds that are appreciated by business and individual alike and which serve as key differentiators. All students registering with Learntesting courses are provided with their own unique 'seat' in the LMS and access to a range of resources. Learntesting course students are provided with 1-year renewable access to a unique service that includes:

1. A private library of ebooks, some of which are written specifically to support the ISTQB Certified Tester scheme;
2. A professional tutor-video from its accredited ISTQB Certified Tester Foundation Level (CTFL) course recording, with one of the most experienced testing consultants in the world – ideal to act as a 'refresher' of knowledge and as preparation for those going on to the Advanced courses;
3. A 24x7 support forum from a global network of accredited training providers;
4. A growing private library of presentations, papers and other testing-related content;
5. An annual subscription to Testing Experience Magazine.

Also, by introducing Live Virtual Classroom Tutor-led sessions to support Advanced courses and exam revision sessions, plus exam vouchers redeemable across the world, Learntesting is providing an 'end-to-end' service to its students and a truly global solution for a global industry.

5. What courses and options are available in Learntesting?

Foundation Level

The new Learntesting service has been launched with its fully accredited ISTQB Certified Tester Foundation Level (CTFL) course available in several 'packages':

- Course only;
- Course plus exam voucher;
- Course plus revision plus exam voucher.

In addition, Learntesting offers a virtual classroom revision session for CTFL which is available as a separate package with the exam voucher.

The CTFL course has now been released in English and German and is under development in Spanish (using the Spanish Testing Board as its accreditation body).

Learntesting also has a Foundation 'tutor-video' only option, which is ideal for those who already hold the CTFL Certificate as a refresher and preparation for moving on to Advanced Level.

Advanced Level

At Advanced Level, a course leading to the Test Manager Certificate is being finalised and will be submitted for accreditation with an expectation of this being complete by August. This is a major exercise since we believe that typical self-study options for such a long and demanding syllabus will not lead to success for most students.

As such, the Advanced Level online courses being developed by Learntesting will have the following structure:

- Full Course video;
- Two 3-hour Live Virtual Classroom sessions each accompanied by two experienced accredited tutors;
- Virtual Classroom exam revision session.

The courses will be supported by:

- Exercise course workbook;
- Interactive exercises and quizzes;
- Formal 'mock' exam assessments;
- 24x7 accredited tutor support;
- Advanced Level ebook access within the Learntesting ebook library.

Other Content

There is a range of other content currently available with much more under development (see below).

- Full 'mock' ISTQB examination assessments;
- Introductory course on Non-Functional Testing;
- Transition course (for those who studied the pre-2005 ISTQB Syllabus);
- Templates for testers;
- Various papers and presentations;
- Sample questions and answers.

6. You mention an exam voucher. What are my options for exams if I study with Learntesting?

Learntesting is working in conjunction with a number of exam bodies to provide flexible options for exams. These include Pearson VUE, who have exam centres across the world and iSQI, who operate in Germany, Austria, Switzerland, Spain, France, Russia, Ukraine and more than 30 other countries all over the world.

In addition, many ISTQB Boards run a public exam schedule and private/closed exams can be arranged for groups of students who study accredited courses.

7. What are the development plans for Learntesting?

Learntesting will be developing further language options for its ISTQB CTFL course over and above English & German (already available) and Spanish (under development). We would be pleased to hear from anyone requesting a language other than those (contact info@learntesting.com).

As well as the Advanced Level Test Manager course currently being completed, an Advanced Level Test Analyst course is also under construction for release later this year. A course is also under development for the Intermediate Certificate in the ISEB scheme, which we believe to be complementary to the ISTQB Certified Tester scheme.We are also looking to provide support for other related schemes such as IREB (Requirements Engineering) and QAMP (Quality Assurance Management Professional).

In addition, some smaller modules of training that support specific areas, such as Agile Development, Non-functional Testing and TSG's 'Practical Test Academies' that are run in the classroom are in the pipeline for development.

8. How do I get access to Learntesting and what do I get if I register as a client for free?

You can access Learntesting via the Testing Experience Portal www.testingexperience.learntesting.com

You will see the content available in the Catalogue, a regular 'Opinion Piece' written by a range of authors, sample course material, questions & answers, and some free downloadable content.

If you register as a client in Learntesting, you will also be given free access for 3 months to a range of other content before signing up for any courses, including:

- Exam hints and tips;
- Papers and Presentations;
- Templates for Testers;
- Sample exam questions and answers, updated on a regular basis;
- Archive of weekly opinion pieces.

As a special bonus, Testing Experience is offering a limited number of pre-pay places on the Advanced Level Test Manager course (prior to formal accreditation) at a very special price of €450 (plus VAT). You will get immediate access to the CTFL Tutor-video and a private library of ebooks which will help you prepare for the Advanced Level course starting in August.

These places are only available by registering as a client on Learntesting and are strictly limited to the number of places in the initial virtual classroom sessions.

© Jana Noack

learntesting

For a totally new experience in online learning,
register for the Testing Experience 'Learntesting' Service

Fully accredited ISTQB Certified Tester (CTFL) online course

(English and German versions)

Course plus exam special offer of €599 (plus VAT)

2 months access to full accredited material,
1-year VLE (Virtual Learning Environment) support package

- A private library of ebooks on testing and testing-related subjects;
- The Foundation tutor-video course;
- A 24x7 support forum from a global network of accredited training providers;
- Testing Experience Magazine subscription;
- 20% discount on any additional classroom or online courses booked with Testing Experience or Learntesting.

limited special offer

Advanced Test Manager course, available in August

Testing Experience can offer a limited number of pre-pay courses at **half-price**

€ 450 (plus VAT)*

- Full online course materials;
- Two 3-hour Live Virtual Classroom sessions each accompanied by two experienced accredited tutors;
- Live Virtual Classroom exam revision session.

Plus, this price includes **immediate 1-year** access to
1-year VLE (Virtual Learning Environment) support package:

- A private library of ebooks on testing and testing-related subjects;
- The Foundation tutor-video course as refresher and preparation for the Advanced courses;
- A 24x7 support forum from a global network of accredited training providers;
- Testing Experience Magazine subscription;
- 20% discount on any additional classroom or online courses booked with Testing Experience or Learntesting

Register at www.testingexperience.learntesting.com to take advantage of this special offer

* Note this offer is valid until 31st July subject to places remaining available. The exam price is not included and is payable separately on booking an exam date.

Web Vulnerability Scanners: Tools or Toys?

by Dave van Stein

Executing a web application vulnerability scan can be a difficult and exhaustive process when performed manually. Automating this process is very welcome from a tester's point of view, hence the availability of many commercially and open-source tools nowadays.

Open-source tools are often specifically created to aid in manual testing and perform one task very well, or combine several tasks with a GUI and reporting functions (e.g. W3AF), whereas commercial web vulnerability scanners, like e.g. IBM Rational Appscan, HP Webinspect, Cenzic Hailstorm, and Acunetix Web Vulnerability Scanner, are all-in-one test automation tools designed for saving time and improving coverage.

Web Application Vulnerability scanners basically combine a spidering engine for mapping a web application, a communication protocol scanner, a scanner for user input fields, a database with attack vectors and a heuristic response scanner combined with a professional GUI and advanced reporting functions. Commercial vendors also provide frequent updates to the scanning engines and attack vector database.

Evaluating web vulnerability scanners

Over the past years many vulnerability scanner comparisons have been performed [1, 2, 3] and the most common conclusion is that the results are not consistent.

This great diversity in results can partially be explained by the lack in common testing criteria. Vulnerability scanners will typically be used by testers with various backgrounds like functional testers, network security testers, pen-testers, and developers, each of them having a different view on how to review these tools, causing different result interpretations. Sometimes this leads to comparing web vulnerability scanners with other security-related products, which is like comparing apples and oranges[4].

Another explanation is the diversity in a test basis. The vast amount of technologies and ways to implement these in web applications make it difficult to define a common test strategy. Each application requires a different approach, which is not always easy to achieve, making it hard to compare results.

Finally, like testers, vendors also have different views on how to achieve their goals. Although vulnerability scanners might look the same on the outside, the different underlying technologies can make interpretation and comparison of results more difficult than they appear to be.

The Web Application Security Consortium (WASC) started a project in 2007 to construct "*a set of guidelines to evaluate web application security scanners on their identification of web application vulnerabilities and its completeness.*"[5]. Unfortunately this project has not reached its final stage yet, although a draft version has been recently presented[6].

This article focuses on the difficulties reviewing and using vulnerability scanners. It does not provide the best vulnerability scanner available, but discusses some of the strengths and weaknesses of these tools and gives insight in how to use them in a vulnerability analysis.

Using a web vulnerability scanner

Vulnerability scanners are like drills. Although the first are designed to find holes and the latter for creating them, their usage is similar.

Using drills out-of-the-box will possibly yield some results, but most likely not the desired ones. Without doing some research into the several configurations of the machine, the possible drill-heads, and the material you are drilling into, you are more than likely to fail in drilling a good hole and may come across some surprises. Likewise, running vulnerability scanners out-of-the-box will probably show some results, but without reviewing the many configuration options and structure of the test object, the results will not be optimal. Also the 'optimal configuration' differs in each situation. Before using a vulnerability scanner efficiently, it is necessary to understand how scanners operate and what can be tested.

In essence, scanners work in the 3 following steps:

1. identify a possible vulnerability
2. try to exploit the vulnerability
3. search for evidence of a successful exploit

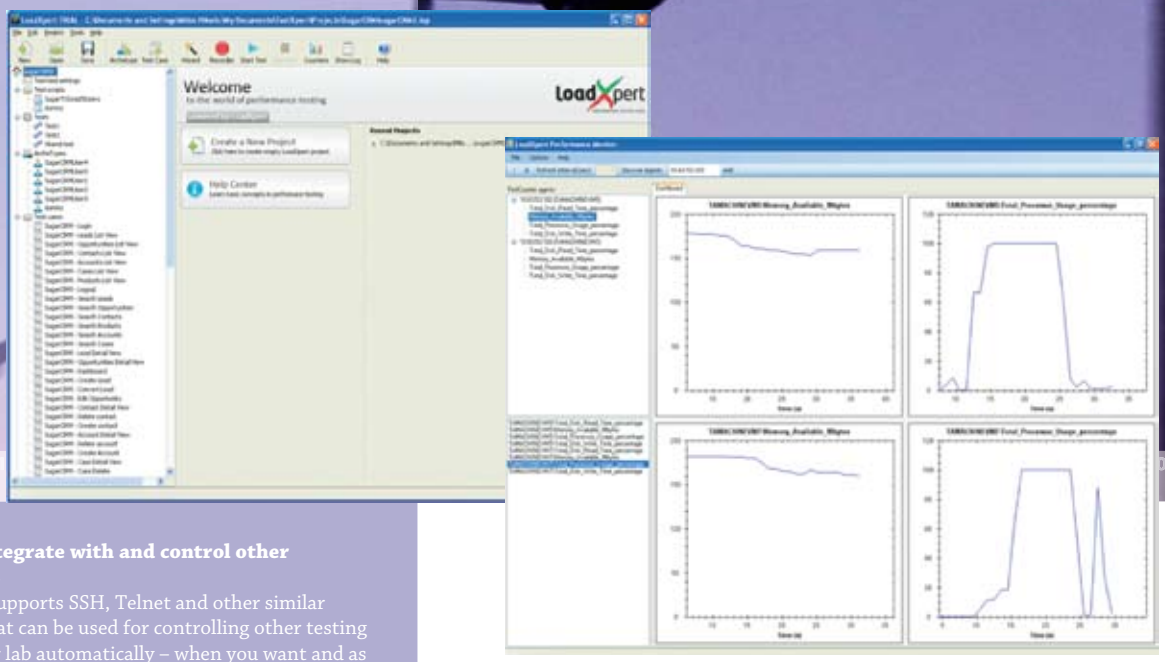
For each of these steps to be executed in an efficient way, the scanner needs to be configured for the specific situation. Failing in configuring one of these steps properly will cause the scanner to report incomplete and untrustworthy results, regardless of the success rate of the other two steps.

1 http://www.networkcomputing.com/rollingreviews/Web-Applications-Scanners/
2 http://ha.ckers.org/blog/20071014/web-application-scanning-depth-statistics/
3 http://anantasec.blogspot.com/2009/01/web-vulnerability-scanners-comparison.html
4 http://en.hakin9.org/attachments/consumers_test.pdf
5 http://www.webappsec.org/projects/wassec/
6 http://sites.google.com/site/wassec/final-draft

Low hardware requirements, scalable system with variable number of load generator machines.
System is a client-server application, where clients are controlled by the server. Server dispatches the load generation tasks onto as many clients as you wish, one client can be installed on one machine. You can use whatever PC machines you have in your LAB.

Easy to use script recording process
Preparing your test scripts has never been easier with our one-click-recording process. You can start the recorder and then continue using your application as you would normally do. LoadXpert will capture all the traffic that goes between your browser and your application. Later that traffic will be parameterized and multiplied onto as many instances as you want.

Test your applications Just as your clients do. And do it affordably.



Ability to integrate with and control other testing tools
LoadXpert supports SSH, Telnet and other similar protocols that can be used for controlling other testing tools in your lab automatically – when you want and as much as you want. Let LoadXpert be your main tool for your complex performance tests.

Flexible and easy to use load design wizard
After you have recorded your test scripts, you can easily design the load that you want to throw at your application, by using LoadXpert load design wizard. You can choose how you want the load to reach its maximum and then how long you want your load test to last, by following on-screen instructions in our wizard.

Server performance monitoring system, which monitors windows/linux performance counters in real-time.
With LoadXpert Monitor you can keep track on your server performance in real-time and record all performance measurements into a database for later processing. LoadXpert Monitor supports both Windows and Linux based servers and applications. It can be used prior, during and after load tests.

Fast growing happy customers list!
Proven tool for performance testing of Web Applications (Cisco and Hypo-Alpe-Adria bank as customers)

- Integrated Development Environment (IDE) for test script editing and traffic recording
- Load Script Wizard, for quick and easy way to design the load scenario during the test
- HTTP/HTTPS Recorder, simple utility within IDE used for capturing HTTP and HTTPS traffic between users and application
- Scalable Client-Server architecture, for emulating concurrent user workload, where clients are controlled by the server. Server dispatches the load generation tasks onto as many clients as you wish, with one client software per machine.
- Scheduled Test Execution, you can schedule your tests, to start at any given moment in time (overnight testing etc)
- Real-time Server Performance Monitoring System, for monitoring and recording server performance by collecting performance counters from all underlying hardware and software servers (both Windows and Linux servers are supported). Performance counters are recorded to a database, for later detailed analysis. Web viewer version available also.
- Automatic Report Generation, test execution report with graphs and data analysis results is just a few clicks away.

LoadXpert
PERFORMANCE TESTING TOOL

- Performance testing tool offering features of contemporary products
- Low hardware requirements tool for SMB and enterprise sector
- Easy and intuitive to use environment with quick test preparations
- True simulation of real-life load using archetype methodology
- Quick test reports generation with more time for results quality judgment
- Variability and scalability of test process
- Predictability and productivity for performance testing professionals
- Return on your investment based on your quantitative results

Contact us for more information, trial version or try-out option at
http://www.loadxpert.net

Knowing what to test

Before a vulnerability scanner can start looking for potential problems, it first has to know what to test. Mapping a website is essential to be able to efficiently scan for vulnerabilities. A scanner has to be able to log into the application, stay authenticated, discover technologies in use, and find all the pages, scripts, and other elements required for the functionality or security of the application.

Most vulnerability scanners provide several options for logging in and website spidering that work for standard web applications, but when a combination of (custom) technologies are used, additional parameterization is needed.

Another parameter is the ability to choose or modify the user agent the spider uses. When a web application provides different functionality for different browsers or contains a mobile version, the spider has to be able to detect this. A scanner should also be able to detect when a website requires a certain browser for the functionality to function properly.

After a successful login, a scanner has to be able to stay authenticated and be able to keep track of the state of a website. While this is no problem when standard mechanisms (e.g. cookies) are used, custom mechanisms in the URI can easily cause problems. Although most scanners are able to identify the (possible) existence of these problem-causing techniques, an automatic solution is rarely provided. When a tester does not know or understand the application, used techniques and possible existence of problems, the coverage of the test can be severely limited.

The Good

Vulnerability scanners are able to identify a wide range of vulnerabilities, each requiring a different approach. These vulnerabilities can roughly be divided into four aspects:

- information disclosure
- user input independent
- user input dependent
- logic errors

Information disclosure handles all errors that provide sensitive information about the system under test or the owner and user(s) of the application. Error pages that reveal too much information can lead to identification of used technologies and insecure configuration settings. Standard install pages can help an attacker successfully attacking a web application whereas information like e-mail addresses, user names, and phone numbers can help a social engineering attack. Some commercial vendors also check for entries in the Google Hacking Database[7]. Its use, however, is limited in the development or acceptance testing stage.

User-independent vulnerabilities cover insecure communications (e.g. sending passwords in clear text), storing passwords in an unencrypted or weakly encrypted cookie, predictable session identifiers, hidden fields, and having enabled debugging options in the web server.

Checking for both information disclosure problems and user-independent vulnerabilities manually can be very time-consuming and strenuous. Vulnerability scanners identify these types of errors efficiently almost by default.

The Bad

Bigger problems arise when testing for user-dependent vulnerabilities. These problems occur due to insecure processing of user input. The most known vulnerabilities of this kind are Cross-site scripting (XSS)[8, 9], the closely related Cross-site request forgery (CSRF)[10, 11], and SQL injection[12, 13].

The challenge for automated scanning tools, when testing for these vulnerabilities, lie in detecting a potential vulnerability, exploiting the vulnerability and detecting the results of a successful exploit.

SQL injection

SQL injections are probably the best known vulnerabilities at the moment. This attack already caused many website defacements and hacked databases. Although the most simple attack vectors are no longer a problem for most web applications, the more sophisticated variants can still pose a threat. Even when an application does not reveal any error messages or feedback on the attack, it can still be vulnerable to so-called blind SQL injections. Although some blind injections can be detected by vulnerability scanners, they cannot be used for complete coverage, mainly due to performance reasons. Blind SQL injections typically take a long time to complete, especially when every field in an application is tested for these vulnerabilities. Most vendors acknowledge this limitation and provide a separate blind SQL injection tool to test a specific location in an application.

XSS and CSRF attacks

Cross-site scripting (and cross-site request forgery) attacks are probably the most underestimated vulnerabilities at this moment. The consequences of these errors might look relatively harmless or localized, but more sophisticated uses are discovered each day like hijacking VPN connections, firewall bypassing, and gaining complete control over a victim's machine.

The main problem with XSS is that possible attack vectors run into millions (if not more). For example; an XSS thread on sla.cker.org[14] has been running since September 2007, contains close to 22,000 posts so far and new vectors are posted almost daily.

There are several causes that contribute to the vast amount of possible attack vectors:

- It is possible to exploit almost anything a browser can interpret, so not only via the traditional SCRIPT and HTML tags, but also CSS templates, iframes, embedded objects, etc.
- It is possible to use tags recursively (e.g. <SCR<SCRIPT>IPT>) for applications that are known to filter out statements.
- It is possible to use all sorts of encoding (e.g. unicode[15]) in attack vectors
- It is possible to combine two or more of these vectors, creating a new vector that is possibly not properly handled by an application or filtering mechanism.

With AJAX the possibilities increase exponentially. Obviously it is impossible to test all these combinations in one lifetime, not in the least for the performance drop this would cause. Vulnerability scanners therefore provide a subset of the most common attack vectors, sometimes combined with fuzzing technologies. This list is, however, insufficient by nature, so additional attack vectors should be added or manually tested.

Another problem is the diversity of XSS attacks; the two most known types are reflective and stored attacks. With reflective attacks the result of the attacks is transferred immediately back to the client, making analysis relatively simple. Stored or persistent attacks on the other hand are stored at some place and not immediately visible. The result might even not be visible to the logged-on user and may require logging in as another user and understanding the application logic to detect them.

Stored or persistent user input vulnerabilities can basically be checked in two ways:

- Exploit all user input fields in a web application and scan the application completely again afterwards for indications of successful exploits
- Check what is stored on the server after filtering and sanitizing

Most scanners opt for an implementation of the first method, but detecting all successful exploits is a difficult task. Especially when an application has different user roles or an extensive data-flow, successful exploits can be hard to detect without understanding and taking into account the application logic.

Acunetix uses an implementation of the second method in a technology called AcuSen-

7 http://johnny.ihackstuff.com/ghdb/

8 http://en.wikipedia.org/wiki/Cross-site_scripting

9 http://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

10 http://en.wikipedia.org/wiki/Csrf

11 http://www.owasp.org/index.php/Cross-Site_Request_Forgery

12 http://en.wikipedia.org/wiki/SQL_injection

13 http://www.owasp.org/index.php/SQL_injection

14 http://sla.ckers.org/forum/read.php?2,15812

15 http://en.wikipedia.org/wiki/Unicode_and_HTML

sor[16]. Although this technology shows good results in detecting for example stored XSS and blind SQL injection attacks, the biggest drawback is that it has to be installed on the web-server in order to use it. This might not be problematic in a development or even acceptance environment, but in a production environment this is often not an option or even allowed.

The Ugly

The most difficult errors to find in a web-application are application and business logic errors. Although these errors are usually a combination of other vulnerabilities, they also contain a functional element contributing to the problem. Examples of logic errors are password resets without proper authentication or the possibility to order items in a webshop and bypassing the payment page. Since logic errors are a combination of security problems and flaws in the functional design of an application, practically all vulnerability scanners have problems detecting them. Commercial vendors like IBM and Cenzic do have a module for defining application logic attacks, but these are very basic modules and require extensive parameterization.

When testing for logic errors, manual testing still is necessary, although vulnerability scanners can be used for the repetitive or strenuous parts of the test. Practically all commercial vendors, but also e.g. Burp Suite Pro, have an option to use the vulnerability scanner as a browser. Hereby the tester chooses the route to test the application, while the scanner can perform automatic checks in the background.

Conclusions

Vulnerability scanners can be very useful tools in improving the security and quality of web applications. However, like any other testing tool, being aware of the limitations is essential for a proper use. With their efficient scanning of communication problems and bad practices, they can save time and improve the quality and security early in the development of web applications. When used for testing user input filtering and sanitizing, they can save time by rapidly injecting various attacks. However, manual reviewing of the results is essential and, due to the limited amount of attack vectors, additional manual testing remains necessary.

Fully automated testing of business and application logic is not possible with vulnerability scanners. Here vulnerability scanners have the same limitations as other test automation tools. However, when used by experienced security testers, they can save time and improve the test coverage when used to automate the most strenuous parts of the security testing process.

Web application vulnerability scanners can be powerful tools in the hands of experienced testers, but running them out-of-the-box with default settings demote them to nothing more than expensive toys.

16 http://www.acunetix.com/websitesecurity/rightwvs.htm

Biography

Dave van Stein is a senior test consultant at ps_testware. He has close to 8 years of experience in software and acceptance testing and started specializing in Web Application Security at the beginning of 2008. Over the years, Dave has gained experience with many open-source and commercial testing tools and has found a special interest in the more technical testing areas and virtualization techniques. Dave is active in the Dutch OWASP chapter, and he is both ISEB/ISTQB-certified and EC-Council 'Certified Ethical Hacker'

Become IREB® Certified Professional for Requirements Engineering.

The task of requirements engineering (RE) as first step within system development is to determine the requirements on a system, to document them adequately, to test and to manage them throughout the entire lifecycle. A number of disciplines rely directly on the results of RE - if they are incorrect, the associated projects often exceed their budgets and deadlines or fail altogether.

The IREB® Certified Professional for Requirements Engineering certificate and training is about:

- **Factors Influencing Requirements Engineering**
- **Requirement Types and Description**
- **Requirements Documentation**
- **Acquiring Requirements**
- **Requirements Management**
- **Tools**

For further information please contact: jan.schubert@isqi.org

iSQI®
International Software Quality Institute

A Risk-Based Approach to Improving Software Security

by Rex Black

If you are a software tester, software developer, development or test manager, or another other software professional concerned with quality and security, you probably know that developing secure software is no longer simply desirable—it's completely essential.

Some developers might assume that most security problems arise from the operating system or networking layers, well below the application code they are working on. However, recent figures for Web-based applications from the Open Web Application Security Project (www.owasp.org) show that over three-quarters of security exploits arose from applications (see Table 1).

So, you know you need secure code, but how to get there? What are your security risks? What security failures and bugs do you have? What do these security risks, failures, and bugs mean? How can you reduce security risk in a way that doesn't create new problems? How do you monitor your progress over time? This article will outline seven steps that will allow you to answer these and other questions as you improve your software's security.

Exploited Vulnerability	Percent Occurrence
Server Applications	41%
Non-Server Applications	36%
Operating System Issues	15%
Hardware Issues	4%
Communication Protocol Issues	2%
Others	2%
Network and Protocol Stack Issues	1%
Encryption Issues	0%

Table 1: Occurrence of Security Exploits by Vulnerability

Assess the Risks

Applications tend to have characteristic security risks. These risks often arise from the implementation technology. For example, C and C++ are notorious for their lack of inherent array range checking, and consequent buffer-overflow bugs, which allow hackers to insert malicious code into very long input strings. People writing applications with databases have to worry about SQL injection, where hackers put queries into otherwise-benign fields and gain access to sensitive data.

Security risks can also arise from the business application domain. For example, since they deal in money, banking applications are attractive targets for criminals and a major source of worry for bank IT departments. Applications that store personal information, such as medical history, are subject to regulations like HIPAA that require strict privacy controls.

Risk awareness is the first step in risk reduction. Companies have been reluctant to let outsiders know about the security failures they've had, but some of their failures make the news, and users report others. For example, the Open Web Application Security Project, www.owasp.org, provides good information for those developing Web applications, as does the World Wide Web Consortium's security page, www.w3.org/Security. Carnegie-Mellon's Software Engineering Institute's CERT Coordination Center, www.cert.org, provides a broader look at computer security issues. Last but not least, check out the searchable Risk Digest archives, catless.ncl.ac.uk/Risks, for great anecdotes and commentary on software risks, including security-related risks.

In addition to being aware of the failures, you need to be aware of the underlying bugs themselves. Depending on the kind of applications you're writing, you'll want to read appropriate books and Web sites for hints on common insecure coding constructs, how developers can avoid them, and how testers can find them. For example, entering "secure programming" in the Amazon.com search engine yields dozens of books, some general, some quite specific.

Once you are aware of the kinds of security risks that could affect your software, do a security risk analysis. Identify the specific risk items that you should be aware of. Meet with stakeholders to determine the level of risk in terms of likelihood and impact. Likelihood relates to the chances of any given risk becoming an actual security bug in your software. Impact relates to the effect on customers, users, and your software should the bug be exploited. Your analysis of the risks and their associated levels of risk will allow you to create a prioritized list of potential security failures. [1]

Test to Know Where You Stand

If you're like most software development organizations, you don't have the luxury of starting over with new code on every project. How secure is that collection of existing code? If you're like many organizations, you haven't really had a chance to check. So, check the security of your existing software through a security test.

This type of test is often called a penetration test. Its purpose, as the name suggests, is to discover ways in which hackers and other unauthorized users can penetrate your system. Such a test is useful to check for security fail-

1 I describe the process of risk analysis in my books Managing the Testing Process and Pragmatic Software Testing.

ures that your application already presents to the real world.

Remember that the best lock in the world does no good if it's installed in a door made of rotten wood. Similarly, applications with great security features that are installed in insecurely-configured environments can be hacked.

Do your installation procedures, user documentation, provisioning processes, and notification mechanisms support or impede security? I recently signed up for an account on an e-commerce site that seemed to have good security at first. I was asked to create a user name and password. The application enabled SSL encryption during this process. The input field masked the password when I entered it. I was then told that the application would e-mail me an activation notice after it verified my information. When I received the activation notice, the user name and password were in the e-mail, unencrypted and available to anyone who saw or intercepted that e-mail! Private and identifying information should not be stored or transmitted in an insecure fashion.

Consider identifying risk cases for each security requirement. Risk cases are like use cases—though perhaps more properly termed "misuse cases"—that lay out various scenarios of security failure. If you think about end-to-end processes that users go through, along with the environments in which your software will be deployed, you may think of some possible failures or issues you otherwise would have missed. You can confirm the presence or absence of these failures through specific tests.

You can learn how to run penetration tests yourself. Alternatively, you can hire a testing services provider to handle it for you. On the one hand, you might have to make a significant investment in training and books to learn how to perform penetration tests properly and therefore decide a professional external resource can do a better job. On the other hand, you might feel more comfortable having security expertise in your team and therefore decide to invest in growing it.

Thoroughly testing applications that will run in various installed environments can be a real challenge. Such tests are a combination of end-to-end process testing, compatibility testing, and penetration testing. Depending on the multiplicity of environments, users, and procedures that your application can support, such tests cost a lot of money in terms of systems and effort. To save money on setting up a large variety of test configurations in-house, consider using a testing service provider.

Your prioritized list of risks should guide the penetration test, but you should also test for other failures that you might not have thought of. Based on the failures you find, revise your list of risks. Add new risks where you find unexpected failures. Increase the likelihood and impact based on the failures you find. You might also decrease the likelihood and impact for risks that don't relate to observed failures, or relate to failures that were less important than you expected. However, be careful about assuming that a risk that isn't exploitable today won't be exploitable in future releases of the software.

Keep a list of the security problems you find and where you found them. You'll need this list to fix the problems, of course. However, I also recommend that you classify the problems in a few ways. One classification is based on the type of security flaw.[2] Another is the date on which the code was written or the version of the software in which it was introduced. Yet another is the major subsystem or component the code is part of. In addition, classify the severity (impact on the system) and priority (impact on the user) of each failure. Finally, classify each problem based on the security risks you identified earlier.

Analyze to Know Where You Stand

The security test mentioned above will find security-related failures. However, not every security bug in the code will always exhibit a security failure. In other words, it is possible to have underlying bugs that did not exhibit any symptoms during the penetration test. Therefore, to find additional problems, do a static analysis of the code.

Static analysis means going through your code to look for bugs that could cause failures. You might have input fields which are not appropriately checked for size or syntax before being handed off for processing. You might have weak error handling. You might have situations where unauthorized users can pass snippets of languages like SQL or Korn shell into the system where they would be executed. Just because these bugs didn't result in failures doesn't mean they aren't bugs, and you should look for them.

You can automate your static analysis using tools. A wide variety of static analysis tools for identifying security weaknesses in code exist, so you can probably pick one that fits your exact language, environment, needs, and budget. For a large, existing code base, these tools will identify a large number of problems. Not all of these problems are of the same severity and importance. Somehow, you'll need to focus your attention on the most important of them. Fortunately, good tools will allow you to turn particular rules on and off and tune your static analysis at a level of granularity as fine as individual lines of code. Again, your list of risks can help guide you as you determine where to focus.

Based on your static analysis, add to your list of security problems where you found each problem, and its classification.

Evaluate to Understand Where You Stand

You've gathered a lot of data in the first few steps. Time to evaluate that data. What does the data mean; i.e., what information and patterns are hiding in the data? What is a smart plan of action for improving software security?

First of all, sort the problem list by priority and severity. You will likely want to immediately fix the problems with the highest levels of priority and severity. Microsoft famously reached a point where the number of critical security bugs became so high that they embarked on a crash problem to resolve these bugs. For months, Microsoft programmers did nothing but address security bugs. You might not be in as deep a hole—or be able to spare that much effort—but you'll want to address the urgent items right away.

However, you should also do some further evaluation before wading into battle with the security bugs. Bugs do not tend to be evenly distributed across the code base, but rather tend to exist in clusters. Decades ago, IBM studied their MVS software and found that 38% of the bugs that caused problems in production lived in 4% of the modules. On an Internet appliance project, I found that 69% of the bugs we discovered during testing lived in 25% of the modules. By looking for modules with particularly high numbers of security bugs, you might find that completely refactoring one or two modules is the smartest way to improve your software's security.

As you start to think about the long-term, evaluate how many bugs arise from each kind of security flaw. This will tell you the most typical problems that you and your team face. Can you reduce the incidence of such problems through training for your developers? Better code reviews? Better design reviews? All three? After all, you don't want to be fighting a constant battle against security problems with every release, so you and your team need to learn how to create better software.

You should also evaluate the incidence of security bugs based on the age of the code in which they were found. Software tends to wear out over the years, not as physical devices do, but rather through on-going maintenance which reduces the quality of the code. In addition, older code that was written when a programming language was new—or when the team was new to the language or technology—might contain more bugs. Plan for long-term refactoring of decrepit modules that are disproportionate contributors to software insecurity.

Repair the Problems - Carefully

Any time a developer repairs a bug in software, there is a risk that might introduce a new bug. Many people call these regression bugs, because they represent some reduction in the level of software quality that was present before.

The risk of regression bugs applies to security bugs as much as any other bug. In addition, you can't assume that repairing a security bug would necessarily introduce either no bug at all or another security bug. Fixing a security bug might introduce a functionality bug. So, as you repair the security bugs, make sure you have a plan to deal with regression risk. How

2 For example, you can use the OWASP's Top Ten Web application security flaws if you are creating Web applications (www.owasp.org/index.php/Category:OWASP_Top_Ten_Project).

can you do so?

Your test team typically deals with part of the regression problem. They might have created an automated suite of regression tests for functionality, performance, reliability, or other important quality characteristics. However, waiting for the end-stage testing is not ideal, as the cost and schedule implications of dealing with a bug increase the longer that bug is in the system.[3]

So, before delivering code to the test team, the developers should use code reviews, static analysis, and automated unit tests to help manage regression risk for each change they had made to the system. Code reviews, ideally performed by at least two experts in addition to the author, should help catch many problems. Using the static analysis tool you've already invested in to check your new code is a best practice, and good static analysis tools can find many types of problems, not just security problems. Finally, the use of an automated unit testing harness will provide a framework for an automated set of tests that will allow developers to modify and refactor code with a higher level of confidence.[4]

Examine Results in the Real World

Any time you make a process change, you should monitor how those process changes affect the real world. For example, a couple years ago I was training for a marathon, but I hurt my ankle by overtraining in hills. So, I switched to a training schedule that focused on low-impact aerobic exercises like bicycling and elliptical machines while my ankle healed. Did this process change help me achieve success? Two real world measures applied:

1. Based on the symptoms in my ankle, did it heal while I continued this training regimen, and could I gradually reintroduce running to the training? The answers to both questions were "yes."
2. Was I actually able to run the marathon without pain and without re-injuring myself? Thankfully, the answer to this question was "yes" as well.

Similarly, you want to make sure that your new process reduces the number of known security bugs in your code over time, that the test team finds fewer bugs during system test execution, and that the number of security-related incidents that occur in the field gradually goes down.

You should not expect that these three numbers would go down monotonically. Some natural variation in the testing and development processes will mean the number of known bugs might go both up and down. However, the trend over the long-term (say, one year or more) should be that the average number of known security bugs in any given month has gone down.

Similarly, you might have good months and bad months—months where no field security incidents are reported and months where a rash of them are—but this might simply be a natural variation in usage patterns or seasonal usage. For example, you would expect that financial application security bugs related to fiscal-year closing operations would increase at the end of the year. However, again, the trend over the long-term should be that the average number of security incidents in any given month has gone down.

In addition to monitoring your own security bugs and failures, follow the news. The Internet and trade magazines can help you check for problems in applications similar to yours in business domain, implementation technology, or both. If you hear stories about problems that you think might constitute a risk for your application, update your risk analysis and re-evaluate accordingly.

Institutionalize Success

The last step of this process is to do everything all over again, on every single project. That's something of an overstatement, since you don't need to start from a clean slate. You will need to repeat process, though, using your existing work as a baseline:

1. Re-assess security risks.
2. Re-test the application for security failures.
3. Re-analyze the software for security bugs.
4. Re-evaluate patterns in security risks, failures, and bugs.
5. Repair with care.
6. Re-examine the real-world results.

In each of these steps, make sure you look both at new concerns related to changes to your applications and concerns you might have previously overlooked.

Institutionalizing success, the final step of process improvement, is very easy to overlook. After a big push to improve software security, you might be tempted to celebrate success, relax your guard, and gradually slip back into old practices of coding.

We recently had a client that asked us to run a penetration test of their systems. We found a number of security failures during this test, and reported our findings to the engineering team. Later in the project, shortly before release, we re-ran the penetration test. The engineers had resolved all of the failures we had found previously. However, they had also built a bunch of new stuff, which had the exact same kinds of underlying security bugs exhibiting similar security failures. My client had dealt with the manifestations of bad security practices by repairing the security bugs we had found the first time, but had not changed the bad security practices themselves.

Conclusions

Software security is an important concern, and it's not just for operating system and network vendors. If you're working at the application layer, your code is a target. In fact, the trend in software security exploits is away from massive, blunt-force attacks on the Internet or IT infrastructure and towards carefully crafted, criminal attacks on specific applications to achieve specific damage, often economic.

In this article, I laid out a seven-step process to reduce your software's exposure to these attacks.

1. Assess security risks to focus your improvements.
2. Test the software for security failures.
3. Analyze the software for security bugs.
4. Evaluate patterns in security risks, failures, and bugs.
5. Repair the bugs with due care for regression.
6. Examine the real-world results by monitoring important security metrics.
7. Institutionalize the successful process improvements.

Carefully following this process will allow your organization to improve your software security in a way which is risk-based, thoroughly tested, data-driven, prudent, and continually re-aligned with real-world results.

Biography

Rex Black - See page 70.

3 For more on the economics of defects, see my article "Testing ROI: What IT Managers Should Know," on the Library page of our Web site, www.rbcs-us.com.

4 For a detailed case study of how RBCS helped one client implement a process of code reviews, static analysis, and automated unit testing, including creation of an automated test tool framework, see my article "Mission Made Possible," written with Greg Kubaczkowsi, at the Library page of our Web site, www.rbcs-us.com.

Case Study: An Automated Software Testing Framework (ASTF) Example

by Elfriede Dustin

This book excerpt from "Implementing Automated Software Testing," Addison Wesley March 2009, by Dustin, Garrett, Gauf, provides a case study of how the authors successfully implemented an automated software testing framework (ASTF), using mostly open-source tools.

We first define our ASTF requirements and for the purpose of this case study they were as follows:

- Support applications[1] running on multiple distributed computers.
- Support applications developed in different languages.
- Support applications running on different types of OSs.
- Support applications that have Graphical User Interfaces (GUIs) and those that do not (for example, back-end message interface testing).
- Support applications that use different types of network protocols such as TCP/IP, DDS, CORBA, etc.
- Support the integration of multiple commercial testing tools from different vendors (as new or better products emerge).
- Support testing without having to install the ASTF on the same computers as the System-under-Test (SUT), i.e. the ASTF should not impact SUT applications and configurations, and should require very low footprint.
- Allow for test input data to be inserted to one or more SUTs using existing application interfaces. SUT interfaces include GUI, message-based, command-line, file-based, for example.
- Testing scenarios to be supported include individual applications/SUTs or components, and coordinated and concurrent interaction with multiple applications or components.
- Capture/document test results at both the test step and test case levels.

The framework additionally needed to provide the following capabilities and be able to:

- Select and execute one or more (batch) tests to be run at a specified time.
- Verify that the system configuration is valid prior to test execution.
- Provide the status of test execution:
- Percent complete
- Pass/fail status
- Generate a searchable electronic log of test results.

1 Note application here refers to either the SUT or AUT

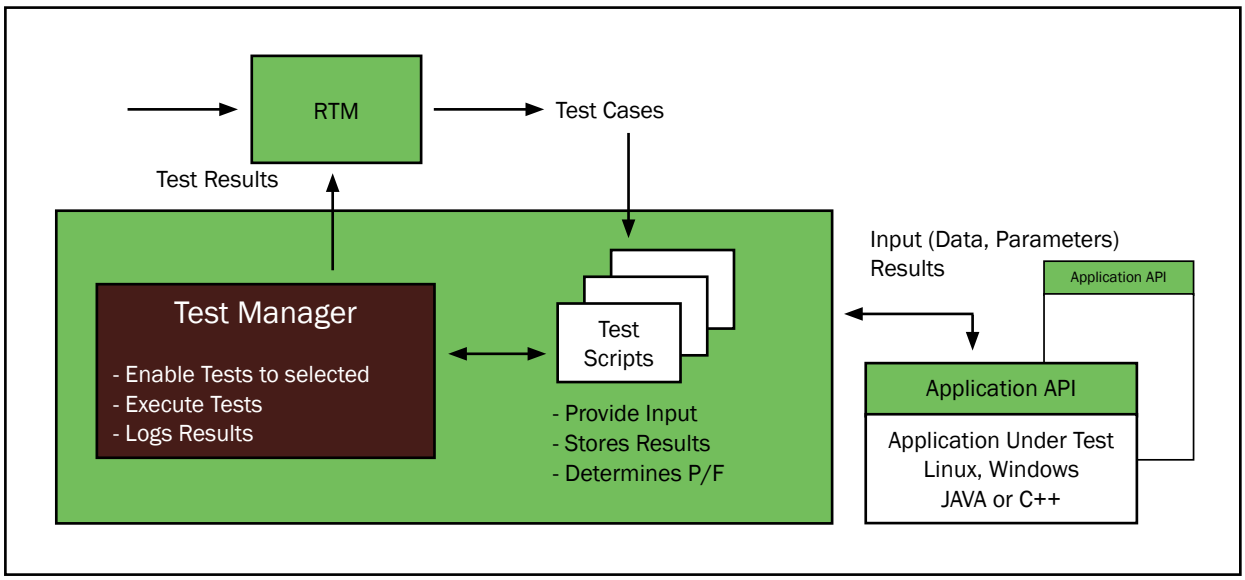


Figure 1 Automation framework concept

- Provide an automated analysis of test results and generate the appropriate reports.
- Document test results per test case to support automated maintenance of Requirements Traceability Matrix (RTM).
- Pre-fill a trouble report for review and analysis before submitting to defect tracking tool in the event a test step fails.

Figure "Automation framework concept" describes our initial high-level framework concept.

Framework Concept

Our goal was to implement open-source tools using the automation framework concept described in Figure 1. The distributed environment supported by the framework was similar to Figure "Distributed Environment supported by ASTF."

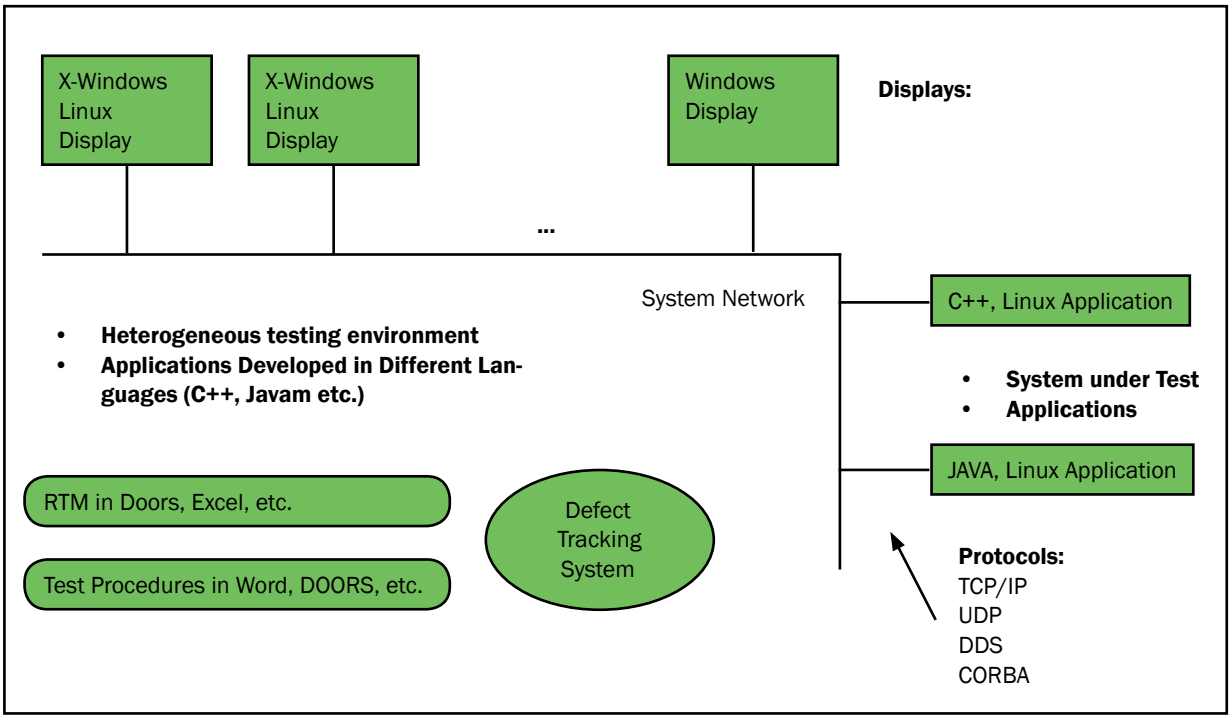


Figure 2 Distributed Environment supported by ASTF

Key Design Features

Our framework design consisted of the following key design features:

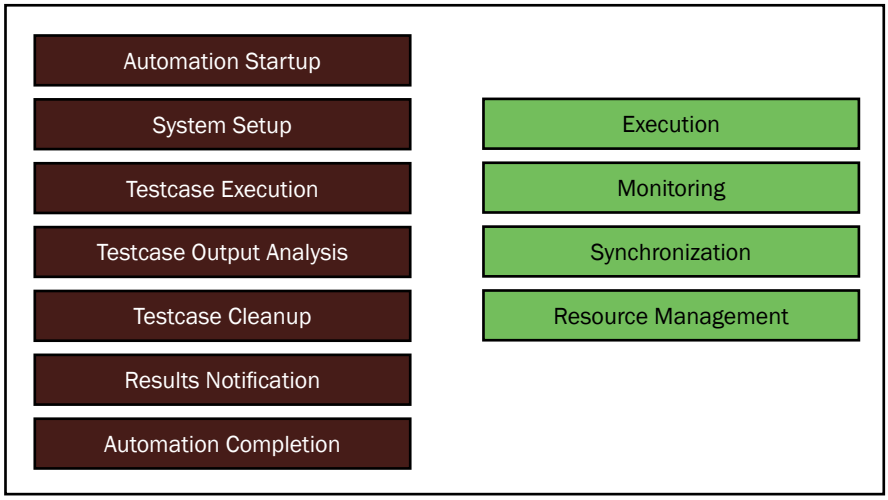
- We used a single design/approach to support testing at the system level and on individual applications.
- The goal was to leverage commercially available automation test tools and open-source tools and products to the maximum extent possible.
- We planned for ongoing insertion of automation test tools and products from multiple vendors (to be able to use the best tools and products as the market changed).
- The test suite had to be scalable and portable.

We then searched for a framework that would provide automation tasks as defined in Figure "Automation Tasks," i.e., a framework that would provide for automation startup, system setup, test case execution, test case output analysis, test case cleanup, results notification, and automation completion.

Automation Tasks

For each automated test, the test execution needed to be managed or coordinated. We found, among other tools, Software Testing Automation Framework (STAF) to meet our needs. We could manage the test case execution through STAF-provided services. How we evaluated STAF/STAX which met our requirements this automation framework.

We determined that end-to-end automation would be possible using STAF/STAX; see Figure "End-to-End Automation with STAF and STAX."



End-to-End Automation with STAF and STAX

We also determined that STAF/STAX could meet all our needs related to a central test manager (see Figure 5), central to the automation framework.

A STAF-STAX in-house design and architecture was required. See Figure 6.7 High-level ASTF architecture design in Chapter 6 of the book "Implementing Automated Software Testing" for the sample design we came up with.

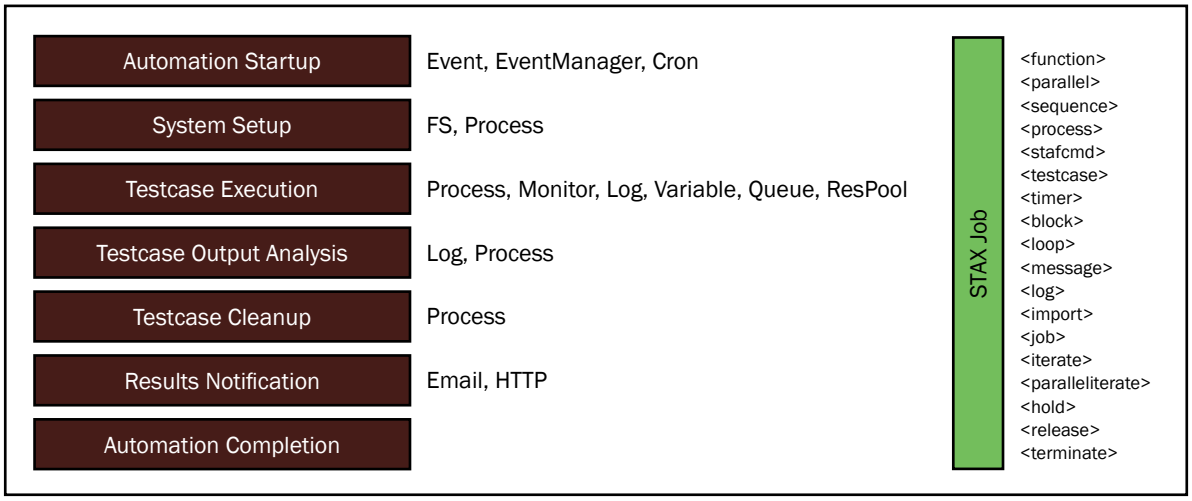


Figure 4 End-to-end automation with STAF/STAX (Ibid.)

Test Manager

The Test Manager is generally the central feature of the test automation tool. Here which test cases and the number of them to be run are selected, or the timing of when the batch test file is to run is dictated. A Test Manager as part of the ASTF should have the following minimum set of requirements:

- Allows test case development and modification, including various test data forms, test data pools, etc.
- Allows browsing of test cases to be run
- Allows selection of test cases to be run
- Allows monitoring of test case execution
- Allows for statusing of test case execution, including how many steps are complete, how many are left, how many tests have been run, and their pass/fail status
- Allows an operator to bring up test case reports

We used STAF, which comes with a test manager GUI called STAXMon. It did not provide all of the features described above, but because it's open-source we were able to add the needed features easily. See Appendix C of the book "Implementing Automated Software Testing" for more details describing how STAF/STAX met our framework needs. See also the section on Redstone Eggplant that describes how it met our capture/playback needs.

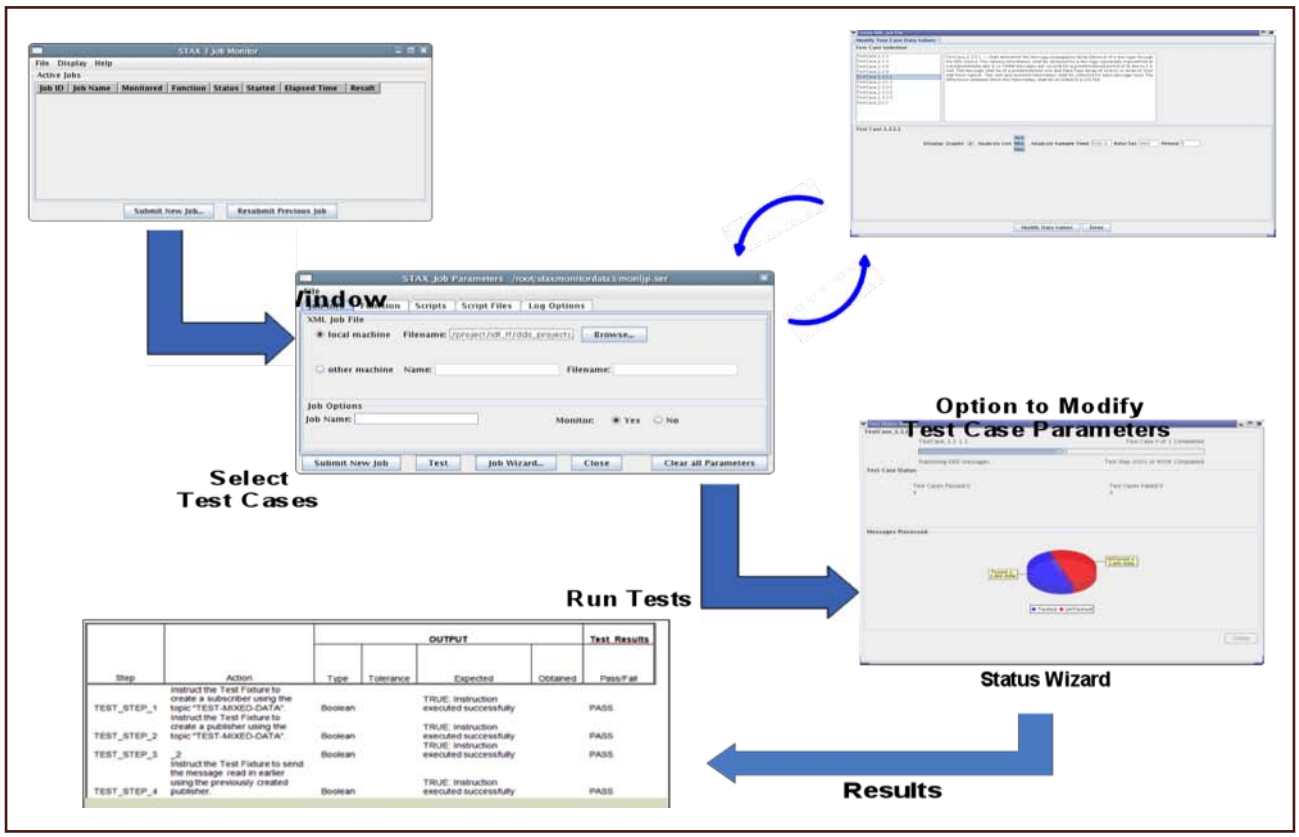


Figure 5 Test manager central to the automation framework

ISSECO®

SECURE SOFTWARE ENGINEERING

ISSECO® | **Certified Professional for Secure Software Engineering**

TRAINING

There is no doubt that application security is a vital issue for any modern application. Unfortunately there is no plug that you can press to turn it on. Application security should be implemented throughout the product life-cycle.

The ISSECO course will teach you the how and when. The course will introduce the technology but never the less the methodology. Furthermore you will learn the security logic and how it can be achieved without breaking the budget.

Contents

- View Of The Attacker
- Explain The Definition Of The Terms Hacker, Cracker And Ethical Hacker
- Show Examples Of Famous Hackers And Their Attacks
- Point Out The Different Skill Levels
- Modes Of Hacking
- Hacker Motive
- Illustrate The Process Of Hacking
- Outline Common Hacking Tools
- View Of The Customer
- Explain Why Customers Expect Secure Software
- Trust & Threat Models
- Methodologies
- Requirements Engineering
- Secure Design, Secure Coding, Secure Testing, Secure Deployment
- Hands-On Workshop
- Security Response
- Explain The Difficulties Of Fixing Security Issues Via Standard Maintenance Processes
- Code & Resource Protection

Tutor: Manu Cohen-Yashar

28.09.2009 - 30.09.2009 in Frankfurt am Main, Germany **Price: 1899,00 € (plus VAT)**

Register at http://training.diazhilterscheid.com

SELA GROUP

Díaz Hilterscheid

- For message interface testing we needed to implement additional code. Here we looked at automatic code generation: When writing automation test code (code to verify the SUT), consider autogenerating code, because it
- Provides predictable test software
- Is scalable
- Is portable
- Fixes problems in one place
- Can be reused across projects
- Cuts down on development time per project, which makes automating tests more feasible

More on Automated Test Case and Test Code Generation

We generated automated test case code and automated framework code.

For example, the automation framework code generated code that glued the SUT interface code to the test framework. A software interface could be a DDS middleware or a simple Multicast layout. We generated software interface code from various input formats, i.e., IDL, C-style headers, etc.

Additionally, while many test case generation tools are available, we didn't want to be tied to a vendor provided solution and therefore developed this functionality in house using Java. For automating test case code generation, we implemented the following: Each step in the test case procedure had associated code generated to execute its behavior. A standardized test case format facilitated automated step procedure extraction. Here XML was used to define the extracted test step information as input into the autogen (autogeneration) process.

Additionally, we enhanced an out of the box capture/playback tool with additional capability, such that as the test engineer for example "clicked" on a GUI to record a test step, automated code was generated behind the scene, based on keywords.

StringTemplates provided for a common code set as well as project-unique sets. Figure "automation code generation" shows the automation code generation concept.

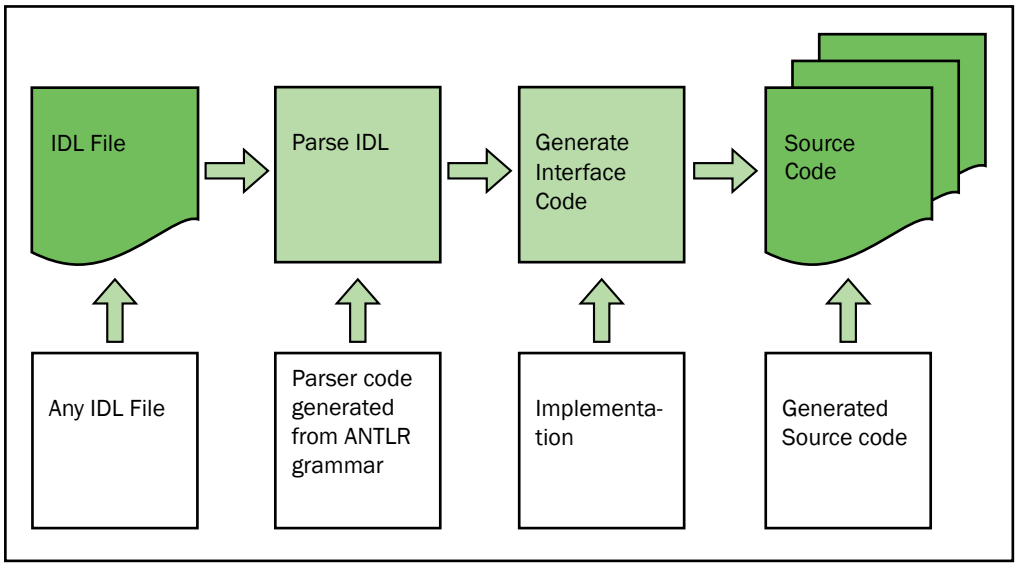


Figure 6 Automation code generation concept

Tools used for this automated code generation effort were

- **ANTLR** (www.antlr.org/)
  - Works off of grammar files; several common languages' grammar files are available for download (e.g., IDL, C)
  - Generates lexical analyzer and parser functionality in Java (other languages are supported as well)
- **StringTemplate** (www.stringtemplate.org/)
  - Uses ANTLR; developed by the same team
  - Provides pluggable templates; code can be generated in multiple languages without modification of the autogen code
- **JAXB** (https://jaxb.dev.java.net/)
  - XML to Java object binding based on XML schema
  - Generates Java classes used to interpret XML data

Results Reporting

Our goal for results reporting was to keep it as generic as possible and independent of actual decision making, whether we used a data parser as depicted in the next Figure "Automated Reporting" or an application reporting actual results from a test run.

Our lesson learned was to report actual results only to simplify autogeneration of source code. We put comparison logic in a separate utility and were then only concerned with comparing expected results (text, integer, etc.) with the actual results from the test run, as described in Figure "Reporting Modularity"

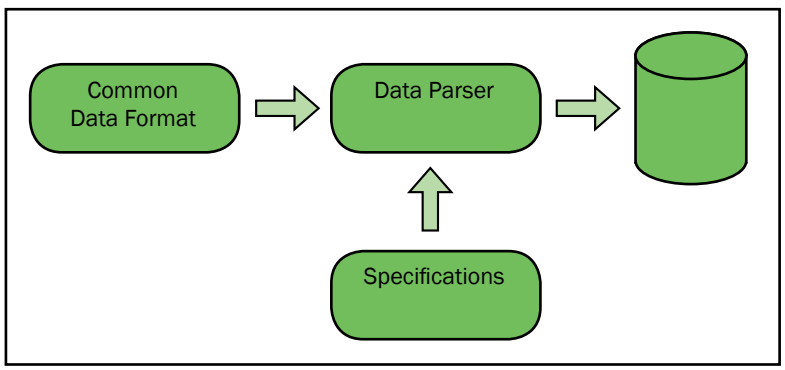


Figure 7 Automated reporting

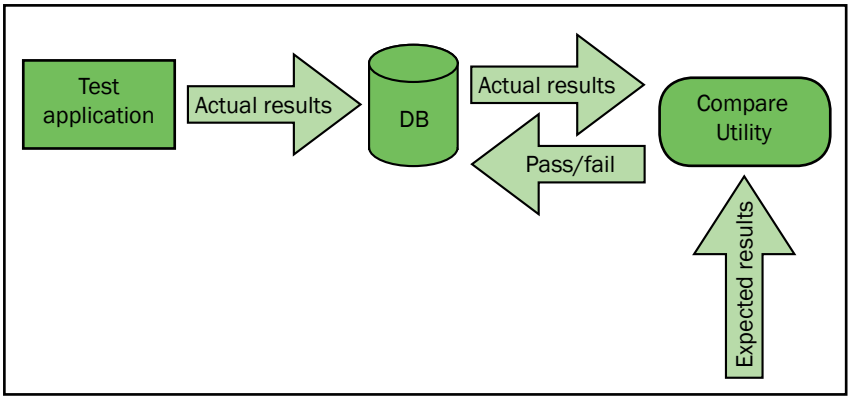


Figure 8 Reporting modularity

Automated Defect Reporting

As mentioned throughout the book it is important to define and adhere to a defect tracking lifecycle. Chapter 9 of the book "Implementing Automated Software Testing" provides the Bugzilla defect tracking lifecycle example. For our defect tracking tool for the purpose of this case study we selected Bugzilla. As part of our case study, we also automated the defect reporting. Our framework allowed for filling in most of the information needed for a software trouble report, and we used information gathered from the test run to provide a detailed trouble report. This helped developers re-create failures. We decided not to directly populate the defect tracking database with the failures, but instead reported the potential failures to an intermediate output that would allow a reviewer to review the defect before submitting it automatically, with the click of a button, to the defect tracking database. The type of information about the failed test steps we were tracking as part of this effort were

- Test case name and step number
- Test case step detailed information
- Requirement that failed
- Test data used
- Description of failure, which can include a screen print of the error and more

The option we considered was to add "Priority," however this would only work effectively if priority was considered in combination with "severity," for example a high priority test case failure could become a high priority defect, but with a low severity. Using "Priority" as a pre-defined field standalone wasn't effective, because a high priority test case run could result in a low priority defect.

Biography

Elfriede Dustin leads the efforts in automated software testing research programs at Innovative Defense Technology (www.idtus.com), a company that specializes in the design, development and implementation of Automated Software Testing Solutions. Elfriede has authored multiple books on Software Testing, most notably "Automated Software Testing," books which have been translated into various languages, has published numerous articles and regularly presents at conferences. Her most recent book "Implementing Automated Software Testing," is co-authored with IDT's president Bernie Gauf and co-worker Thom Garrett and is available in bookstores now.

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 retesting 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.

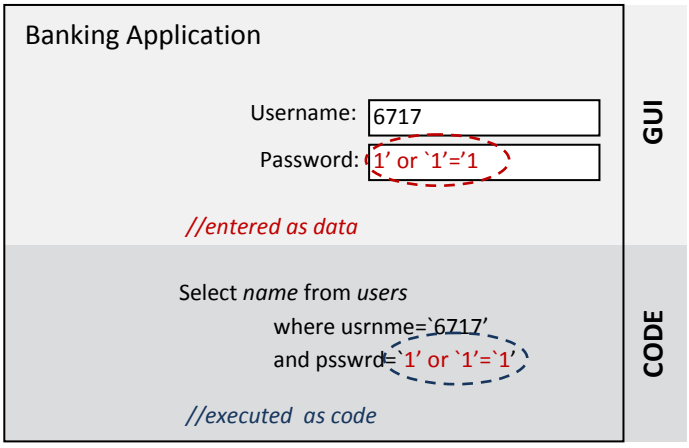


Figure 1 An SQL-Injection example

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.

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 clickjacking 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

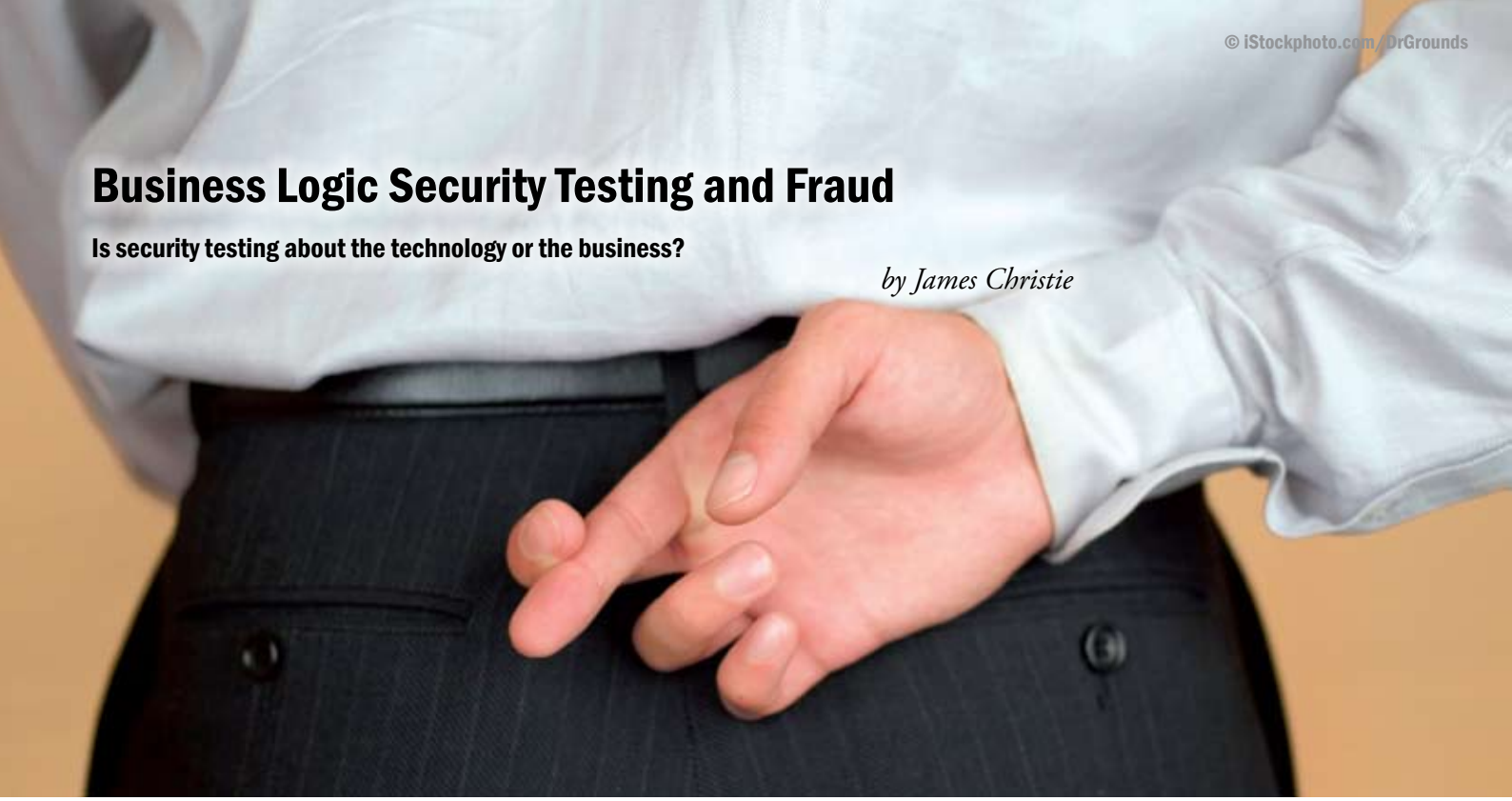
iii. http://www.owasp.org/index.php/Category:OWASP_CLASP_Project

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.



Business Logic Security Testing and Fraud

Is security testing about the technology or the business?

by James Christie

When I started in IT in the 80s, the company for which I worked had a closed network restricted to about 100 company locations with no external connections. Security was divided neatly into physical security, concerned with the protection of the physical assets, and logical security, concerned with the protection of data and applications from abuse or loss.

When applications were built, the focus of security was on internal application security. The arrangements for physical security were a given, and didn't affect individual applications. There were no outsiders to worry about who might gain access, and so long as the common access control software was working there was no need for analysts or designers to worry about unauthorized internal access.

Security for the developers was therefore a matter of ensuring that the application reflected the rules of the business; rules such as segregation of responsibilities, appropriate authorization levels, dual authorization of high-value payments, reconciliation of financial data.

The world quickly changed and relatively simple, private networks isolated from the rest of the world gave way to more open networks with multiple external connections and to web applications.

Security consequently acquired much greater focus. However, it began to seem increasingly detached from the work of developers. Security management and testing became specializations in their own right, and not just an aspect of technical management and support.

We developers and testers continued to build our applications, comforted by the thought that the technical security experts were ensuring that the network perimeter was secure.

Nominally, security testing was a part of non-functional testing. In reality, it had become somewhat detached from conventional testing.

According to the glossary of the British Computer Society's Special Interest Group in Software Testing (BCS SIGIST) [1], security testing determines whether the application meets the specified security requirements.

SIGIST also says that security entails the preservation of confidentiality, integrity and availability of information. Availability means ensuring that authorized users have access to information and associated assets when required. Integrity means safeguarding the accuracy and completeness of information and processing methods. Confidentiality means ensuring that information is accessible only to those authorized to have access.

Penetration testing, and testing the security of the network and infrastructure, are all obviously important, but if you look at security in the round, bearing in mind wider definitions of security (such as SIGIST's), then these activities can't be the whole of security testing.

Some security testing has to consist of routine functional testing that is purely a matter of how the internals of the application work. Security testing that is considered and managed as an exercise external to the development, an exercise that follows the main testing, is necessarily limited. It cannot detect defects that are within the application rather than on the boundary.

Within the application, insecure design features or insecure coding might be detected without any deep understanding of the application's business role. However, like any class of requirements, security requirements will vary from one application to another, depending on the job the application has to do.

If there are control failures that reflect poorly applied or misunderstood business logic, or the business rules, then will we as functional testers detect that? Testers test at the boundaries. Usually we think in terms of boundary values for the data, the boundary of the application or the network boundary with the outside world. Do we pay enough attention to the boundary of what is permissible user behavior? Do we worry enough about abuse by authorized users, employees or outsiders who have passed legitimately through the network and attempt to subvert the application, using it in ways never envisaged by the developers?

I suspect that we do not, and this must be a matter for concern. A Gartner report of 2005 [2] claimed that 75% of attacks are at the application level, not the network level. The types of threats listed in the report all arise from technical vulnerabilities, such as command injection and buffer overflows.

Such application layer vulnerabilities are obviously serious, and must be addressed. However, I suspect too much attention has been given to them at the expense of vulnerabilities arising from failure to implement business logic correctly. This is my main concern in this article. Such failures can offer great scope for abuse and fraud. Security testing has to be about both the technology and the business.

Problem of fraud and insider abuse

It is difficult to come up with reliable figures about fraud because of its very nature. According to PriceWaterhouseCoopers, in 2007 [3] the average loss to fraud by companies worldwide over the two years from 2005 was $2.4 million (their survey being biased towards larger companies). This is based on reported fraud, and PWC increased the figure to $3.2 million to allow for unreported frauds.

In addition to the direct costs, there were average indirect costs in the form of management time of $550,000 and substantial unquantifiable costs in terms of damage to the brand, staff morale, reduced share prices and problems with regulators.

PWC stated that 76% of their respondents reported the involvement of an outside party, implying that 24% were purely internal. However, when companies were asked for details on one or two frauds, half of the perpetrators were internal and half external.

It would be interesting to know the relative proportions of frauds (by number and value) which exploited internal applications and customer-facing web applications, but I have not seen any statistics for these.

The U.S. Secret Service and CERT Coordination Center have produced an interesting series of reports on "illicit cyber activity". In their 2004 report on crimes in the US banking and finance sector [4], they reported that in 70% of the cases the insiders had exploited weaknesses in applications, processes or procedures (such as authorized overrides). 78% of the time the perpetrators were authorized users with active accounts, and in 43% of cases they were using their own account and password.

The enduring problem with fraud statistics is that many frauds are not reported, and many more are not even detected. A successful fraud may run for many years without being detected, and may never be detected. A shrewd fraudster will not steal enough money in one go to draw attention to the loss.

I worked on the investigation of an internal fraud at a UK insurance company that had lasted 8 years, as far back as we were able to analyze the data and produce evidence for the police. The perpetrator had raised 555 fraudulent payments, all for less than £5,000 and had stolen £1.1 million by the time that we received an anonymous tip-off.

The control weaknesses related to an abuse of the authorization process, and a failure of the application to deal appropriately with third-party claims payments, which were extremely vulnerable to fraud. These weaknesses would have been present in the original manual process, but the users and developers had not taken the opportunities that a new computer application had offered to introduce more sophisticated controls.

No-one had been negligent or even careless in the design of the application and the surrounding procedures. The trouble was that the requirements had focused on the positive functions of the application, and on replicating the functionality of the previous application, which in turn had been based on the original manual process. There had not been sufficient analysis of how the application could be exploited.

Problem of requirements & negative requirements

Earlier I was careful to talk about failure to implement business logic correctly, rather than implementing requirements. Business logic and requirements will not necessarily be the same.

The requirements are usually written as "*the application must do*" rather than "*the application must not...*". Sometimes the "*must not*" is obvious to the business. It "*goes without saying*" - that dangerous phrase!

However, the developers often lack the deep understanding of business logic that users have, and they design and code only the "*must do*", not even being aware of the implicit corollary, the "*must not*".

As a computer auditor I reviewed a sales application which had a control to ensure that debts couldn't be written off without review by a manager. At the end of each day a report was run to highlight debts that had been cleared without a payment being received. Any discrepancies were highlighted for management action. I noticed that it was possible to overwrite the default of today's date when clearing a debt. Inserting a date in the past meant that the money I'd written off wouldn't appear on any control report. The report for that date had been run already.

When I mentioned this to the users and the teams who built and tested the application, the initial reaction was "*but you're not supposed to do that*", and then they all tried blaming each other. There was a prolonged discussion about the nature of requirements.

The developers were adamant that they'd done nothing wrong, because they'd built the application exactly as specified, and the users were responsible for the requirements.

The testers said they'd tested according to the requirements, and it wasn't their fault.

The users were infuriated at the suggestion that they should have to specify every last little thing that should be obvious - obvious to them anyway.

The reason I was looking at the application, and looking for that particular problem, was because we knew that a close commercial rival had suffered a large fraud when a customer we had in common had bribed an employee of our rival to manipulate the sales control application. As it happened, there was no evidence that the same had happened to us, but clearly we were vulnerable.

Testers should be aware of missing or unspoken requirements, implicit assumptions that have to be challenged and tested. Such assumptions and requirements are a particular problem with security requirements, which is why the simple SIGIST definition of security testing I gave above isn't sufficient – security testing cannot be only about testing the formal security requirements.

However, testers, like developers, are working to tight schedules and budgets. We're always up against the clock. Often there is barely enough time to carry out all the positive testing that is required, never mind thinking through all the negative testing that would be required to prove that missing or unspoken negative requirements have been met.

Fraudsters, on the other hand, have almost unlimited time to get to know the application and see where the weaknesses are. Dishonest users also have the motivation to work out the weaknesses. Even people who are usually honest can be tempted when they realize that there is scope for fraud.

If we don't have enough time to do adequate negative testing to see what weaknesses could be exploited, then at least we should do a quick informal evaluation of the financial sensitivity of the application and alert management, and the internal computer auditors, that there is an element of unquantifiable risk. How comfortable are they with that?

If we can persuade project managers and users that we need enough time to test properly, then what can we do?

CobIT and OWASP

If there is time, there are various techniques that testers can adopt to try and detect potential weaknesses or which we can encourage the developers and users to follow to prevent such weaknesses.

I'd like to concentrate on the CobiT (Control Objectives for Information and related Technology) guidelines for developing and testing secure applications (CobiT 4.1 2007 [5]), and the CobiT IT Assurance Guide [6], and the OWASP (Open Web Application Security Project) Testing Guide [7].

Together, CobiT and OWASP cover the whole range of security testing. They can be used together, CobiT being more concerned with what applications do, and OWASP with how applications work.

They both give useful advice about the internal application controls and functionality that developers and users can follow. They can also be used to provide testers with guidance about test conditions. If the developers and users know that the testers will be consulting these guides, then they have an incentive to ensure that the requirements and build reflect this advice.

CobiT implicitly assumes a traditional, big upfront design, Waterfall approach. Nevertheless, it's still potentially useful for Agile practitioners, and it is possible to map from CobiT to Agile techniques, see Gupta [8].

The two most relevant parts are in the CobiT IT Assurance Guide [6]. This is organized into domains, the most directly relevant being "Acquire and Implement" the solution. This is really for auditors, guiding them through a traditional development, explaining the controls and checks they should be looking for at each stage. It's interesting as a source of ideas, and as an alternative way of looking at the development, but unless your organization has mandated the developers to follow CobiT, there's no point trying to graft this onto your project.

Of much greater interest are the six CobiT application controls. Whereas the domains are functionally separate and sequential activities, a life-cycle in effect, the application controls are statements of intent that apply to the business area and the application itself. They can be used at any stage of the development. They are;

AC1 Source Data Preparation and Authorization

AC2 Source Data Collection and Entry

AC3 Accuracy, Completeness and Authenticity Checks

AC4 Processing Integrity and Validity

AC5 Output Review, Reconciliation and Error Handling

AC6 Transaction Authentication and Integrity

Each of these controls has stated objectives, and tests that can be made against the requirements, the proposed design and then on the built application. Clearly these are generic statements potentially applicable to any application, but they can serve as a valuable prompt to testers who are willing to adapt them to their own application. They are also a useful introduction for testers to the wider field of business controls.

CobiT rather skates over the question of how the business requirements are defined, but these application controls can serve as a useful basis for validating the requirements.

Unfortunately the CobiT IT Assurance Guide can be downloaded for free only by members of ISACA (Information Systems Audit and Control Association) and costs $165 for non-members to buy. Try your friendly neighborhood Internal Audit department! If they don't have a copy, well maybe they should.

If you are looking for a more constructive and proactive approach to the requirements, then I recommend the Open Web Application Security Project (OWASP) Testing Guide [7]. This is an excellent, accessible document covering the whole range of application security, both technical vulnerabilities and business logic flaws.

It offers good, practical guidance to testers. It also offers a testing framework that is basic, and all the better for that, being simple and practical.

The OWASP testing framework demands early involvement of the testers, and runs from before the start of the project to reviews and testing of live applications.

Phase 1: Before Deployment begins

1A: Review policies and standards

1B: Develop measurement and metrics criteria (ensure traceability)

Phase 2: During definition and design

2A: Review security requirements

2B: Review design and architecture

2C: Create and review UML models

2D: Create and review threat models

Phase 3: During development

3A: Code walkthroughs

3B: Code reviews

Phase 4: During development

4A: Application penetration testing

4B: Configuration management testing

Phase 5: Maintenance and operations

5A: Conduct operational management reviews

5B: Conduct periodic health checks

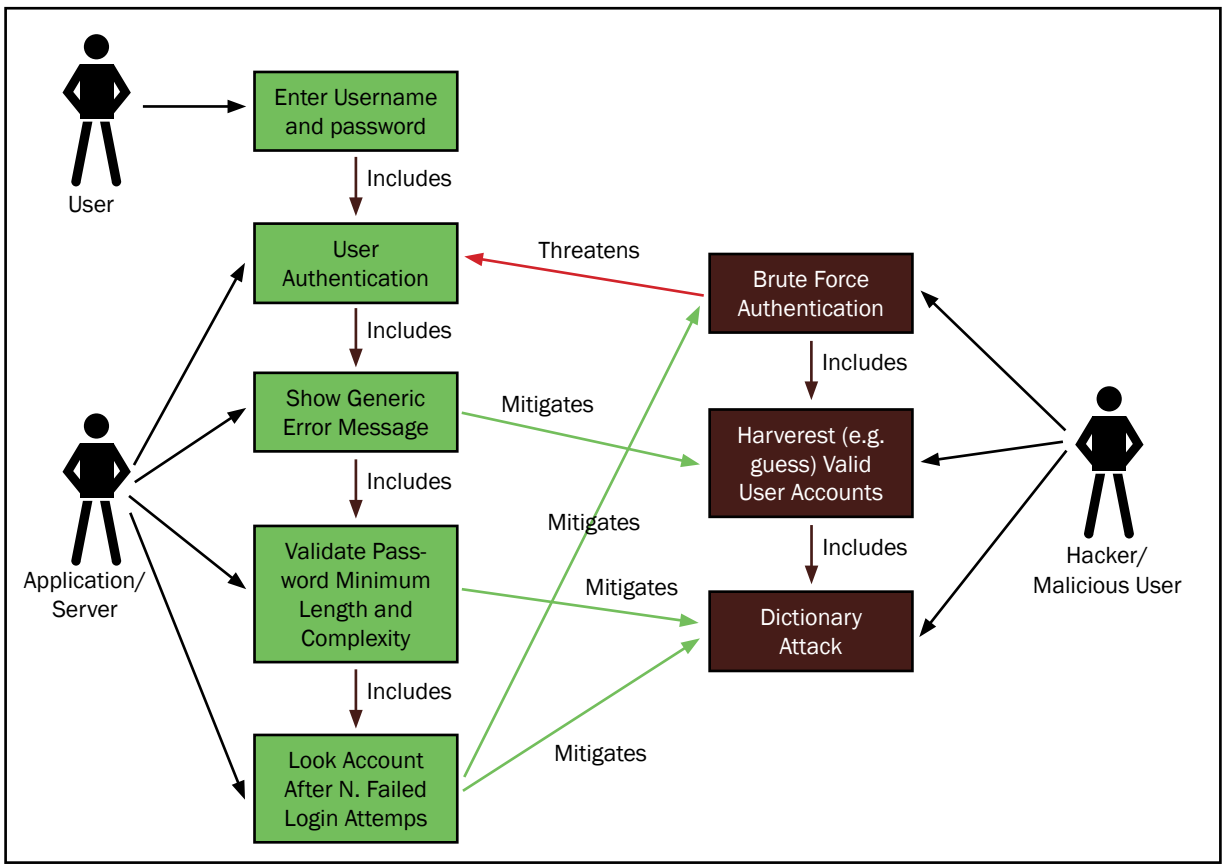
5C: Ensure change verification

OWASP suggests four test techniques for security testing; manual inspections and reviews, code reviews, threat modeling and penetration testing. The manual inspections are reviews of design, processes, policies, documentation and even interviewing people; everything except the source code, which is covered by the code reviews.

A feature of OWASP I find particularly interesting is its fairly explicit admission that the security requirements may be missing or inadequate. This is unquestionably a realistic approach, but usually testing models blithely assume that the requirements need tweaking at most.

The response of OWASP is to carry out what looks rather like reverse engineering of the design into the requirements. After the design has been completed, testers should perform UML modeling to derive use cases that "*describe how the application works. In some cases, these may already be available*". Obviously in many cases these will not be available, but the clear implication is that even if they are available, they are unlikely to offer enough information to carry out threat modeling.

The feature most likely to be missing is the misuse case. These are the dark side of use cases! As envisaged by OWASP, the misuse cases shadow the use cases, threatening them,



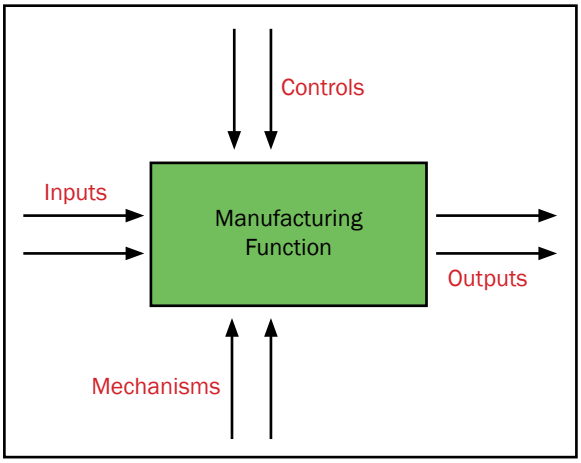
then being mitigated by subsequent use cases.

The OWASP framework is not designed to be a checklist to be followed blindly. The important point about using UML is that it permits the tester to decompose and understand the proposed application to the level of detail required for threat modeling, but also with the perspective that threat modeling requires; i.e. what can go wrong? what must we prevent? what could the bad guys get up to?

UML is simply a means to that end, and was probably chosen largely because that is what most developers are likely to be familiar with, and therefore UML diagrams are more likely to be available than other forms of documentation. There was certainly some debate in the OWASP community about what the best means of decomposition might be.

Personally, I have found IDEF0 a valuable means of decomposing applications while working as a computer auditor. Full details of this technique can be found at www.idef.com [9].

It entails decomposing an application using a hierarchical series of diagrams, each of which has between three and six functions. Each function has inputs, which are transformed into outputs, depending on controls and mechanisms.



Is IDEF0 as rigorous and effective as UML? No, I wouldn't argue that. When using IDEF0 we did not define the application in anything like the detail that UML would entail. Its value was in allowing us to develop a quick understanding of the crucial functions and issues, and then ask pertinent questions.

Given that certain inputs must be transformed into certain outputs, what are the controls and mechanisms required to ensure that the right outputs are produced?

In working out what the controls were, or ought to be, we'd run through the mantra that the output had to be accurate, complete, authorized, and timely. "Accurate" and "complete" are obvious. "Authorized" meant that the output must have been created or approved by people with the appropriate level of authority. "Timely" meant that the output must not only arrive in the right place, but at the right time.

One could also use the six CobiT application controls as prompts.

In the example I gave above of the debt being written off, I had worked down to the level of detail of "write off a debt" and looked at the controls required to produce the right output, "cancelled debts". I focused on "authorized", "complete" and "timely".

Any sales operator could cancel a debt, but that raised the item for management review. That was fine. The problem was with "complete" and "timely". All write-offs had to be collected for the control report, which was run daily. Was it possible to ensure some write-offs would not appear? Was it possible to over-key the default of the current date? It was possible. If I did so, would the write-off appear on another report? No. The control failure therefore meant that the control report could be easily bypassed.

The testing that I was carrying out had nothing to do with the original requirements. They were of interest, but not really relevant to what I was trying to do. I was trying to think like a dishonest employee, looking for a weakness I could exploit.

The decomposition of the application is the essential first step of threat modeling. Following that, one should analyze the assets for importance, explore possible vulnerabilities and threats, and create mitigation strategies.

I don't want to discuss these in depth. There is plenty of material about threat modeling available. OWASP offers good guidance, [10] and [11]. Microsoft provides some useful advice [12], but its focus is on technical security, whereas OWASP looks at the business logic too. The OWASP testing guide [7] has a section devoted to business logic that serves as a useful introduction.

OWASP's inclusion of mitigation strategies in the version of threat modeling that it advocates for testers is interesting. This is not normally a tester's responsibility. However, considering such strategies is a useful way of planning the testing. What controls or protections should we be testing for? I think it also implicitly acknowledges that the requirements and design may well be flawed, and that threat modeling might not have been carried out in circumstances where it really should have been.

This perception is reinforced by OWASP's advice that testers should ensure that threat models are created as early as possible in the project, and should then be revisited as the application evolves.

What I think is particularly valuable about the application control advice in CobIT and OWASP is that they help us to focus on security as an attribute that can, and must, be built into applications. Security testing then becomes a normal part of functional testing, as well as a specialist technical exercise. Testers must not regard security as an audit concern, with the testing being carried out by quasi-auditors, external to the development.

Getting the auditors on our side

I've had a fairly unusual career in that I've spent several years in each of software development, IT audit, IT security management, project management and test management. I think that gives me a good understanding of each of these roles, and a sympathetic understanding of the problems and pressures associated with them. It's also taught me how they can work together constructively.

In most cases this is obvious, but the odd one out is the IT auditor. They have the reputation of being the hard-nosed suits from head office who come in to bayonet the wounded after a disaster! If that is what they do, then they are being unprofessional and irresponsible. Good auditors should be pro-active and constructive. They will be happy to work with developers, users and testers to help them anticipate and prevent problems.

Auditors will not do your job for you, and they will rarely be able to give you all the answers. They usually have to spread themselves thinly across an organization, inevitably concentrating on the areas with problems and which pose the greatest risk.

They should not be dictating the controls, but good auditors can provide useful advice. They can act as a valuable sounding board, for bouncing ideas off. They can also be used as reinforcements if the testers are coming under irresponsible pressure to restrict the scope of security testing. Good auditors should be the friend of testers, not our enemy. At least you may be able to get access to some useful, but expensive, CobiT material.

Auditors can give you a different perspective and help you ask the right questions, and being able to ask the right questions is much more important than any particular tool or method for testers.

This article tells you something about CobiT and OWASP, and about possible new techniques for approaching testing of security. However, I think the most important lesson is that security testing cannot be a completely separate specialism, and that security testing must also include the exploration of the application's functionality in a skeptical and inquisitive manner, asking the right questions.

Validating the security requirements is important, but so is exposing the unspoken requirements and disproving the invalid assumptions. It is about letting management see what the true state of the application is – just like the rest of testing.

A Communit

over 110,000 C

Are you on th

www.is

*March 2009


y Worldwide!
Certifications*
e right track?
tqb.org
ISTQB®
International Software
Testing Qualifications Board

References

[1] British Computer Society's Special Interest Group in Software Testing (BCS SIGIST) glossary. http://www.testingstandards.co.uk/glossary.htm

[2] Gartner Inc. "Now Is the Time for Security at the Application Level", 2005. http://www.sela.co.il/_Uploads/dbsAttachedFiles/GartnerNowIsTheTimeForSecurity.pdf

[3] PriceWaterhouseCoopers, "Economic crime- people, culture and controls. The 4th biennial Global Economic Crime Survey", 2007.

[4] US Secret Service "Insider Threat Study: Illicit Cyber Activity in the Banking and Finance Sector", 2004. http://www.secretservice.gov/ntac/its_report_040820.pdf

[5] CobiT 4.1, IT Governance Institute, 2007. http://www.isaca.org/

[6] CobiT IT Assurance Guide, IT Governance Institute, 2007.http://www.isaca.org/

[7] OWASP Testing Guide, V3.0, 2008. http://www.owasp.org/index.php/Category:OWASP_Testing_Project

[8] Gupta, S. "SOX Compliant Agile Processes", Agile Alliance Conference, Agile 2008. http://submissions.agile2008.org/files/CD-966.pdf

[9] IDEF0 Function Modeling Method. http://www.idef.com/IDEF0.html

[10] OWASP "Threat Modeling", 2007. http://www.owasp.org/index.php/Threat_modeling

[11] OWASP Code Review Guide "Application Threat Modeling", 2009. http://www.owasp.org/index.php/Application_Threat_Modeling

[12] Microsoft," Improving Web Application Security: Threats and Countermeasures", 2003. http://msdn.microsoft.com/en-us/library/ms994921.aspx

Biography

James lives in Perth in Scotland . He is currently working as a consultant through his own company, Claro Testing Ltd (www.clarotesting.com).
James has 24 years commercial IT experience, mainly in financial services, with the General Accident insurance company and IBM (working with a range of blue-chip clients) throughout the UK and also in Finland.
His experience covers test management (the full life-cycle from setting the strategy, writing the test plans, supervising execution, through to implementation), information security management, project management, IT audit and systems analysis.
In 2007 he successfully completed an MSc in IT. His specialism was "Integrating usability testing with formal software testing models". He is particularly interested in the improvement of testing processes and how the quality of applications can be improved by incorporating usability engineering and testing techniques.
James is a Chartered IT Professional and a professional member of the British Computer Society (MBCS). He holds the ISEB Practitioner Certificate in Software Testing and is a member of the Usability Professionals Association.

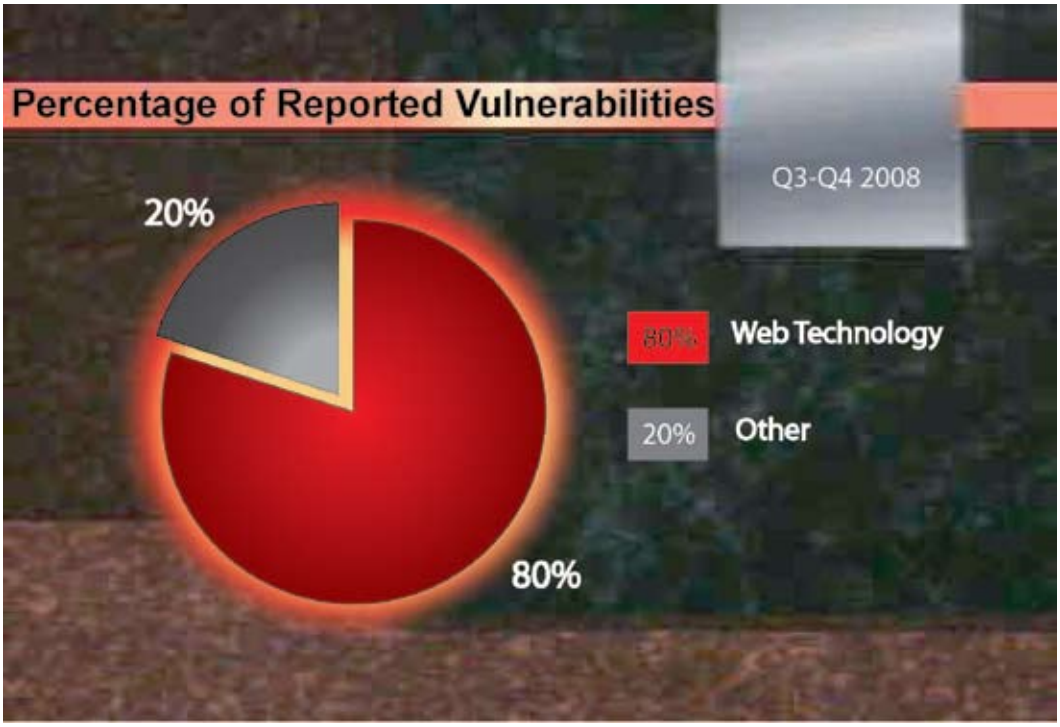
Demystifying Web Application Security Landscape

by Mandeep Khera

U.S. Government passes the stimulus package and includes $355M for cyber security. Hacking against Government sites including electric grid intensifies. New regulations for privacy are being passed or proposed across Europe, Asia, and Americas (http://lastwatchdog.com/senate-bill-mandates-strong-federal-role-internet/). Hacking against government agencies and corporations across the globe continues at a faster rate than ever. From social networking sites like Facebook and Twitter to large corporations like Heartland to government agencies like Utilities and Pentagon – no one has been spared. Even countries are using cyber wars as the latest lethal weapons against one another.

A lot of these attacks are occurring at the Web application layer meaning through the Web sites.

Hackers are getting smarter. They are no longer trying to attack the network layer which has gotten a lot more secure over the last few years with a vast majority of organizations deploying firewalls and Intrusion Detection Systems (IDSs). The low-hanging fruit for hackers are Web applications. Why? Because that's where the vulnerabilities are. Our research shows that Web application vulnerabilities continue to dominate amongst the total published vulnerabilities, as is evidenced from the chart below.



Source: Cenzic Application Security Trends Report – Q3-Q4, 2008

In spite of all the attacks, regulations, and all the hype, why aren't companies and governments doing something about it? One reason is that most people still don't quite understand what Web application security means or they don't believe they'll ever get hacked. In this article, we are going to try and demystify the application security landscape and also bust some of the myths around this space.

Before we look at Web application security, let's define a Web application. Most consumers and even a lot of IT professionals don't realize that Web sites that allow us to do business transactions online are powered by Web applications. And, in some cases there are hundreds or even thousands of these applications acting as the engine for Web sites. Simply put, a Web application is a software program that's written in a browser-supported language like HTML, and is accessed over the browser. This is different from the older ways of applications that used to have fat clients which accessed the server.

Technically, the fact that these Web applications are vulnerable is not a new phenomenon. The fact is that they have always been vulnerable since the early days of the Web in the late 90s. However, hackers started exploiting these vulnerabilities in the early 2000s as networks got more secure and hackers realized that most Web sites were wide open for hacking.

So, how did these applications get deployed with so many vulnerabilities? First, most develop-

ers were not trained to think about security when developing applications. During the client-server era, there weren't any public facing applications, so no one thought it would be a big issue. The Web made everything open which was great for business, but with everything good comes something bad. In this case, we got the hackers. Secondly, even developers, who were trained in security and really understand security issues, suffer from lack of time. Most developers are extremely busy and under a time crunch to get the applications out in time. They don't have time to even do unit testing, forget about security testing.

There are a lot of vendors touting Web application security since it's become the buzz phrase of the decade. At a high level, there are really only two broad categories of solutions in this space as follows:

- **Web Application Security Vulnerability Management** – One can view this as testing of Web applications just like testing of applications for functional and performance features. There are a number of options available to do this type of testing including manual pen testers – internal or external, Web application scanners (black box or dynamic testing) done through the Web interface just like a hacker, Web application scanning in Software as a Service/managed service, source code scanning (scanning of raw code to find vulnerabilities). Most experts and analysts agree that the best approach to start with is dynamic or black box testing, because you get the biggest bang for the buck. If you do not have expertise in-house, a SaaS solution is the best one to start with. Some vendors like Cenzic offer both Software and SaaS solutions to help customers transition from one to the other if appropriate. Manual pen testers can add further value by helping customers with the processes and remediation information as well as address security holes that cannot be found through automation like social engineering types of vulnerabilities.
- **Web application firewall** – Web application firewalls are similar to network firewalls, except that they focus on the application layer (layer 7 in the OSI model) and can block traffic based on known vulnerabilities if configured properly. Firewalls are either in software or appliance formats. WAFs can be good for cases where companies don't have the time to fix all the vulnerabilities in time. However, WAFs do require proper configuration of rules or they might have false positives and block good traffic, which can be devastating for a business.

All the other technologies that are touted as Web application security such as encryption, identity management, PKI certificates, etc. are necessary technologies in their own right, but should not be confused as Web application security.

Besides technology, organizations need to make sure that they have good processes in place and that proper accountability of application security is clear throughout the enterprise. If InfoSec is responsible for testing for vulnerabilities, clear rules need to be established for developers to fix the vulnerabilities based on priorities. Web application security is not a one time event, and it needs to have an ongoing focus with continuous testing and remediation.

Even with all the hype around Web application security and the various solutions available, most companies are not doing anything or enough about it. Besides pure inertia, there are many myths that are holding companies back from moving forward on these initiatives. Here are the 10 common myths:

- *I have SSL, so my Web sites are secure*: Well, Secure Socket Layer (SSL) has its place in helping provide some protection to the consumers while they are conducting transactions online. However, it does nothing to protect hackers from hacking into Web sites. SSL is a good technology that assures the consumer that the server he/she is connecting to is a valid server, and it encrypts the communication between the browser and the server. So, the SSL lock symbols on most of the sites can be misleading.
- *I have a network firewall, so that should be enough*: Network firewalls were created to keep the bad guys out at the network layer. They were not designed to protect traffic at the Web application layer. NW Firewalls do not validate user inputs to an HTML application, and they don't detect maliciously modified parameters in a URL request.
- *I have an Intrusion Prevention System, so that should prevent me from all intrusions*: IDS and IPS are good devices, but again at the network layer. These devices were created to block access and do not understand injections at the Web application level. Hackers have access to the same forms and fields that consumers do, and unless you use IPS to block all your Web traffic, it's not going to help you thwart hacker attacks.
- *My data is encrypted, so why should I care if someone attacks*: While data encryption is a solid best practice to follow, it's not enough to protect Web applications. Once hackers come in through the Web site and are able to exploit the application, they can access the data and decrypt later.
- *I don't have any public-facing Web applications*: Insider threats are bigger than ever. Even if you don't have external-facing applications, you still have to protect your employee data on internal applications.
- *I can test my Web application once a year and I'll be fine*: Every month there are 400+ new application-related vulnerabilities and hackers know about them. Also, every time you make any change to a Web application, you have to make sure that there are no new vulnerabilities. Application testing has to be a continuous process with at least monthly if not weekly testing.
- *I have never been hacked, so I am fine*: First of all, gone are the days when hackers used to hack to gain fame. Now, most Web hacking is done by organized criminals and in some cases by government-sponsored organizations. These guys don't want you to know that you are being hacked. There are more and more companies who are finding out that they were being hacked for over a year before they discovered the attacks. Also, do you really want to take a chance to see if you get hacked?
- *I only have commercial applications from large vendors, so it's not my problem*: You are responsible for all the customer and employee information, even if you didn't create the application. You need to make sure that you test all the applications as well as any plug-ins that you wrote for those commercial applications. Notify your commercial vendor and put pressure on them to release a patch so you have recourse in case of a breach.
- *I have never been audited* – You have to protect your Web applications to secure your most important asset – customer information. If your applications are secure, you'll pass the audit and comply with regulations. The reverse is not necessarily true.
- *Application Security is painful to implement* – Although it's more difficult to secure Web applications than the network layer and desktops, there are many easy solutions to get your process jump-started. Like all initiatives, once you get going, the road gets less bumpy.

All indications are that cyber attacks at the Web application layer will continue to rise in the coming months and years. With 80% of vulnerabilities in Web applications and over 75% of attacks happening through the Web sites, the question is not IF you will get attacked, the question is WHEN you will get attacked.

There are a lot of good resources available to help you get going with your initiatives including OWASP, SANS, and many other free Webinars from various vendors like Cenzic. There's a lot of help available to move you along the process. You need to take that first step.

Biography

Mandeep Khera has over 24 years of diversified experience in marketing, engineering, business development, sales, customer services, finance and general management.

Mandeep has been with Cenzic, an application security software company, since 2003 as the Chief Marketing Officer driving the strategic, product, and outbound marketing functions. Prior to joining Cenzic, Mandeep managed marketing functions for VeriSign's Managed Security Services product line, as the Vice-President of Marketing for Maaya, a Web-Services Software company, as Vice-President of Worldwide Marketing for UCMS, an eCRM software company and as the Vice-President of Marketing and Engineering for Embarcadero Systems Corporation, a logistics management software company.

Previously he was with Hewlett-Packard for 11 eleven years in various positions including as a general manager of a software business unit.

Mandeep is a graduate of Harvard Business School's Leading Product Development program and Northwestern University's Executive Development Program. He holds a Bachelor of Commerce with honors from New Delhi.

ISTQB® Certified Tester-Foundation Level

in Paris, in French

Test&Planet : votre avenir au présent.

Venez suivre le Cours « Testeur Certifié ISTQB – Niveau fondamental » à Paris, en français !

Test&Planet, centre de formation agréé DIF, vous propose ce cours D&H en exclusivité.

Apprenez à mieux choisir Quoi, Quand et Comment tester vos applications. Découvrez les meilleurs techniques et pratiques du test logiciel, pour aboutir à une meilleure qualité logicielle à moindre coût.

Prochaines dates disponibles :

6-8 Juillet 2009

7-9 Septembre 2009

Possibilités de cours sur mesure.

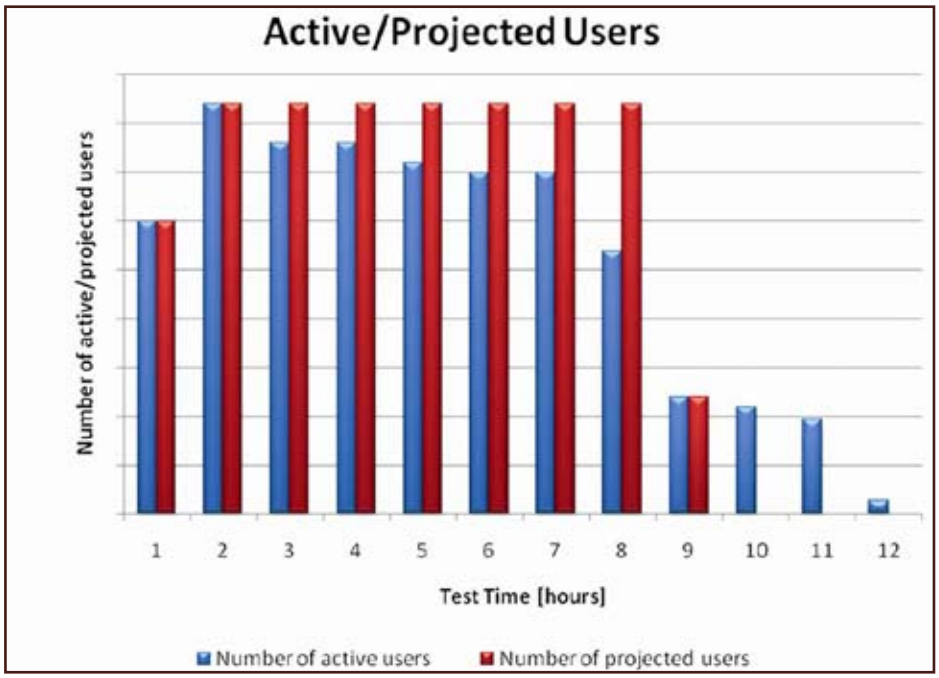
Pour vous inscrire ou pour plus d'information : www.testplanet.fr

Customer Success Story - Advertorial

by Vladan Konstantinovic

Overview

Hypo Alpe-Adria-Bank a.d. Belgrade has been conducting business in Serbia since 2002. As a part of Hypo Alpe-Adria-Bank International, the most successful regional bank in the Alpe Adriatic region, it is today among the top 5 most important financial institutions in Serbia, with a presence in all large industrial centers, enabling orientation towards long-term investment financing with its stability and significant financial potential, which has positioned Hypo Alpe-Adria-Bank a. d. Belgrade as one of the most important banks in the development of the Serbian economy.



Challenge

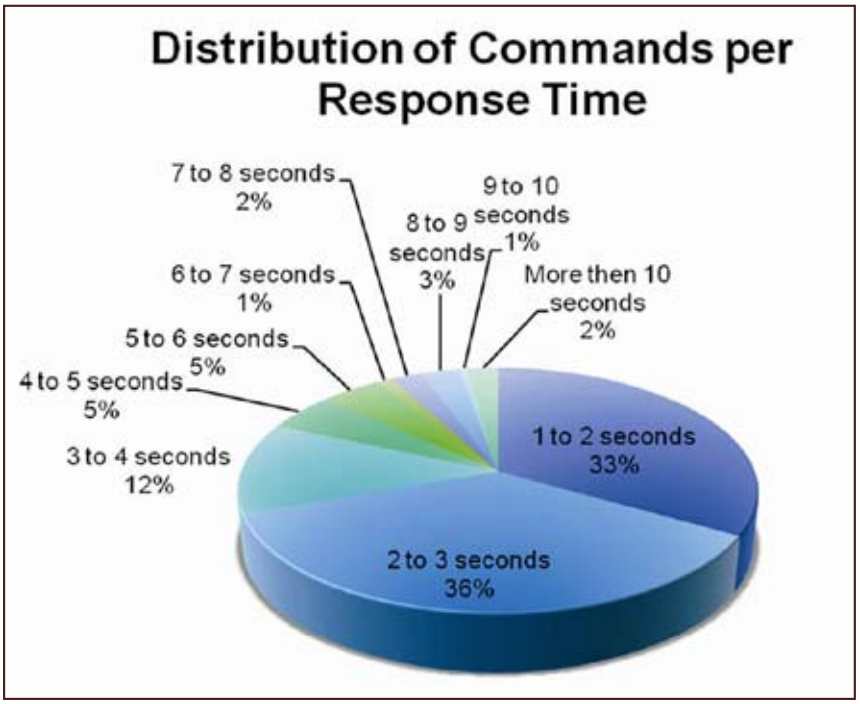
Hypo Alpe-Adria–Bank a.d. Belgrade needed to implement a completely new front-end system that was to be deployed throughout all branches in Serbia. The new system needed to support the daily business of all branches in the country, and it had to be capable of handling high numbers of simultaneous online users. Mission-critical for the bank was to have a strong, robust and reliable application that can cope well with great workloads on a daily basis.

The bank needed to test the bandwidth of the new system before it went live for daily usage. PSTech has provided the bank with an effective and reliable solution.

PSTech Solution

By using the LoadXpert performance testing tool, PSTech developed a solution implementing a robust test methodology supplied by LoadXpert, that supports complex usage scenarios and data processing, secure protocols and a distributed server farm integrated with sophisticated infrastructure.

With an integrated traffic recorder within the LoadXpert tool, PSTech was able to capture and parameterize the encrypted traffic between web browser and application servers. Captured traffic was later shaped into a library consisting of test commands, test cases and user archetypes. Based on usage scenarios defined by the bank, PSTech generated different load scripts to be used by a load generator component within the LoadXpert tool, in order to produce the required application load.



This solution was able to successfully generate the application load on demand by emulating workload of real-life bank tellers and managers with a virtual user technology implemented in LoadXpert tool. During each test, PSTech was able to monitor in real time all relevant performance indicators and metrics using the LoadXpert Performance Monitor component. After each test, LoadXpert reports provided PSTech with all relevant information about the application performance during the test.

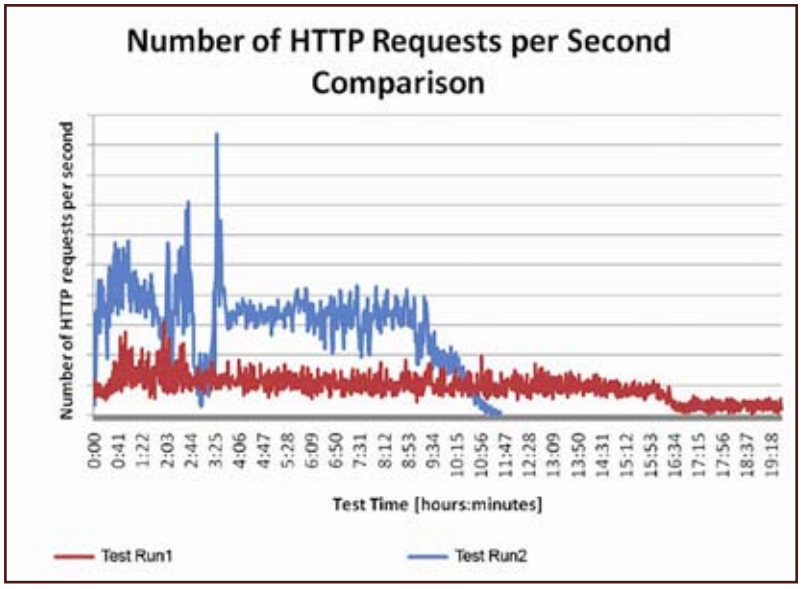
After several rounds of testing, critical issues were discovered that needed extra attention, which in the end resulted with a greatly improved application response time and increased capacity to handle hundreds of users.

Along with metric analysis of the application response under load, the LoadXpert tool also pinpointed bottlenecks within the application and its test bed that needed to be improved before going live. This sort of information was very valuable since it unveiled possible issues before they happen in the real world.

Results

Our efforts resulted in:

- Average page response time reduced by 30%
- Workload throughput increased by 40%
- Dramatically decreased percentage of functional errors during high load, by a factor of 10
- Increased supported number of users by 30%
- Increased workload capacity (requests per second) by 250%



Customer Quote:

"The implementation of a new front-end system was of crucial importance for the future development of our bank. We wanted to make sure that all aspects of the delivered solution function according to our ambitious specifications. In order to do this we had two options. Our first option was to buy expensive test tools and make huge investments in human resources in terms of new recruitments and trainings. The other option was to outsource performance, stress and load testing. Since we are a financial institution and not an IT company, we chose the second option, not only because we wanted to avoid huge investments in something that is not part of our core business, but also because of the expert know-how that we expected from our outsourcing partner. We decided to go with PSTech, and they showed a level of professionalism and expertise much beyond our expectations. With the help of PSTech, we improved all of our critical parameters. During the project, they not only performed tests, but also consulted with us by proposing solutions for improvements. For future projects of this kind,in which we require support, PSTech will be our first choice."

Jelena Kozić, Project Manager and Head of Product Management Department, Retail Sector, Hypo Alpe Adria Bank Belgrade

About PSTech:

- Innovative ICT provider of fully integrated business solutions and advanced services to local and global markets, enabling our customers and partners to explore their business development and optimization through our expert knowledge and experience since 1996.
- Specialized in CRM and ERP solutions, applications development and sustaining, database management, Auto ID, Asset control/ tracking and rich media, based on open source, Microsoft, Cisco and Apple platforms.
- Specialized in Performance Testing Services and Tools via unique package branded as LoadXpert with world-wide promotion on CeBIT 2009.
- Specialized in long-term software outsourcing and applications development, testing, implementation and sustaining based on leading technologies – IP telephony, UC, audio/video/web conferencing.

Biography

Vladan Konstantinovic is a full-time Resource Manager and a seasoned Quality & Assurance Manager, with more than 8 years of experience in software testing currently running a team of 16 testing experts. His career and experience include projects in areas of functional testing, test automation and performance test. His areas of expertise include web-based solutions, with an outstanding experience in rich media, unified communication and voice/video/web conferencing. He is a trainer and author of various training tutorials and materials in use for engineer training purposes.

How to conduct basic information security audits?

by Nadica Hrgarek

Managing information security has become more complex and the numbers of internal and external security threats are increasing. Conducting an information security audit makes good business sense to assess threats and security risks to information systems, to develop risk mitigation strategies and to ensure that identified security risks remain within acceptable levels. The purpose of this article is to provide guidance on how to conduct basic information security audits as part of the internal audit programme.

Introduction

Every business relies on the continuous and safe operation of information technology to maximize effective achievement of organizational goals. Today's organizations face a number of challenges in managing information security. Software vulnerabilities, unauthorized intrusions, increased spam delivery, infection of systems and data by viruses, worms or Trojan horses (backdoors), phishing attacks, denial of service (DoS) attacks, computer-assisted fraud, website defacements, economic espionage, laptop or mobile hardware theft, insider threats, non-compliance (e.g. security policy not followed by employees) and other threats to information security are just a few of many challenges for organizations. It is very important to note that information security is a continuous management process that requires ongoing attention and a well established risk management process.

To protect valuable IT assets such as computers and laptops, servers, networks, and sensitive data and to detect security risks/threats, companies should conduct regular information security audits. However, many companies ignore this important fact and do not perform their own basic information security audits for various reasons, such as lack of qualified auditors, lack of information security awareness and communication by top management, information security requirements and procedures are not established, costs to involve/hire IT security specialists being too high, etc. Another problem is that most internal auditors and IT employees are not familiar with information security audits.

Organizations that have implemented a quality management system in conformance with international standards (e.g. ISO 9001, ISO 13485, etc) can take advantage from internal audits when performing basic information security audits. The international quality management standards ISO 9001:2008 and ISO 13485:2003 require conducting internal audits (clause 8.2.2).

Internal audits should be conducted at defined intervals by independent and skilled internal auditors for management review and internal purposes. All elements of a quality management system (e.g. design control, process and production controls, corrective and preventive actions, software, management review, document and record control, etc) should be audited. The results of the internal audit activity should lead to continuous improvements, process innovations, risk mitigation, corrective and/or preventive actions, corrective action closures, lessons learned, observations, etc.

ISMS

ISO/IEC 27001:2005 is an information security management framework which helps organizations to establish a documented information security management system (ISMS). The purpose of ISMS is to ensure adequate and appropriate security controls that adequately protect information assets. ISO/IEC 27001:2005 standard can be used to assess compliance of the ISMS by internal and external parties. Clause 6 of this standard requires performing internal ISMS audits at planned intervals (at least once a year) to determine whether security objectives, controls, processes and procedures of the ISMS are compliant, efficient and executed as planned.

Internal ISMS audits shall be conducted by independent, competent and adequately trained auditors. Internal information security auditors should have the following knowledge and skills: communication and report writing skills, sufficient knowledge needed to perform technical security testing (i.e. knowledge in utilizing the penetration tools selected to detect vulnerabilities), working knowledge of application programming, network engineering skills, system administration skills, risk management skills, knowledge to interpret the requirements of information security standards in the context of an ISMS audit, skills to conduct an internal audit in accordance with ISO 19011:2002, etc.

The three core elements of information security (also known as the CIA triad) according to ISO/IEC 27001:2005 are:

- **Confidentiality** of information – protecting access to it from unauthorized users or systems.
- **Integrity** of information – safeguarding the accuracy and completeness of information and processing methods. Examples of integrity incidents are: an unauthorized user deletes a record from a database, changes a statement in a source code, executes an application, etc.
- **Availability** of information – ensuring information and associated assets are available to authorized users or systems when requested or needed. Examples of common availability incidents are: autho-

rized users are not able to access a website because of a DoS attack (e.g. ping of death, SYN flooding), loss of data processing capabilities as a result of natural disasters (e.g. fire, flood, earthquake, hurricane, etc) or human actions (e.g. bombs or strikes).

ISO/IEC 27002:2005 standard identifies 11 domain areas, 39 control objectives and 133 best practice security controls which help organize security policy and procedures. The eleven domain areas of ISO/IEC 27002:2005 are: (1) information security policy, (2) organizational security, (3) asset classification, (4) personnel/human resources security, (5) physical and environmental security, (6) communications and operations management, (7) access control, (8) information systems acquisition, development and maintenance, (9) information security incident management, (10) business continuity management, and (11) compliance with legal requirements.

Information security audit process

Security of information, computer systems, networks and the people who manage them are the focus of the information security audits. An information security audit is usually performed by competent auditors through interviews, vulnerability assessments, examinations of OS settings, analyses of network shares and historical data to identify all possible security risks. The information security audit should be built on past audit results to help refine the security policy and correct deficiencies which are discovered through the audit process. Table 2 lists some sample key questions that any information security audit should address to protect information assets against manipulation and destruction, to preserve availability, confidentiality and integrity of information.

Penetration tests should be complemented with information security audits and other security measures. These tests simulate an attack on an organization's systems and network; check if there are any improperly configured systems or other vulnerable systems. Penetration testing and other security tools (Table 3) allow auditors to discover the vulnerabilities. However, it is important to emphasize that some penetration testing tools can reproduce real attacks, which could cause systems crashing and compromise computer systems or network if they are not protected. Therefore, before conducting any penetration test on systems or networks, it is recommended to have consent from management. If internal resources are being used to perform penetration tests, those resources shall be independent and experienced penetration testers.

The following websites provide a review and brief description of currently available vulnerability assessment tools: www.securitywizardry.com, sectools.org.

Company assets to consider when conducting information security audits may include, but are not limited to:

- information assets (data files, user manuals, product specifications, procedures and work instructions),
- paper documents (contracts, procedures, guidelines),
- software assets (application software, system software, source code),
- hardware assets (computers, laptops, web servers, routers and networking equipment, printers, company smartphones/PDAs, scanners),
- personnel assets (users, administrators, developers),
- service assets,
- location assets.

The internal information security audit usually contains the following audit activities:

- *Planning and preparing the audit* – Appointing the audit team leader; defining audit objectives, scope and criteria; selecting an appropriate audit team; conducting a document review prior to the on-site audit activities (e.g. security policy and other applicable procedures, standards); preparing the audit plan; preparing work documents (e.g. audit checklist, audit sampling plans, forms for recording information); selecting appropriate audit tools and environment.
- *Conducting the on-site audit* – Conducting the opening meeting; review of documentation (e.g. hardware/software inventory, network architecture, incident logs, user account policy, password policy, backup policy, audit trails, etc); conducting interviews; review of location and environment controls (e.g. server room, fire extinguisher, fire protection door, etc); conducting technical assessment (e.g. running static and dynamic tools, firewall testing, checking system logs, checking system against known vulnerabilities, searching for privileged programs, checking all configuration files of running processes, checking extra network services, code review of non-standard programs, using social engineering techniques, etc); generating the audit findings; preparing the audit conclusions; conducting the closing meeting.
- *Generating, approving and distributing the audit report.*
- *Conducting audit follow-up activities* – Verification of completion and effectiveness of corrective/preventive actions arising from the internal audit.

An audit report should provide a complete, accurate and concise record of the audit. It should be prepared by the lead auditor and signed by the audit team members. Audit reports increase top management awareness of security issues and assist top management in decision-making processes. An audit report usually includes the following information:

- Audit reference number,
- Date of audit,
- Identification of lead auditor and audit team members,
- Executive summary,
- Audit criteria,
- Audit findings (e.g. non-conformities, observations),
- Recommendations for the audit findings (i.e. corrective, preventive or improvement actions),
- Audit conclusions,
- Appendices.

Conclusion

There are many benefits in performing information security audits: ensuring business continuity, minimizing business damage (e.g. preventing financial and availability losses, avoiding image loss), improved enterprise security, better risk management process, gaining deeper knowledge of different aspects of security, measuring compliance with current security policies and procedures, etc. Security audits should be performed at regular intervals or after any significant infrastructure or software changes.

Due to the global economic downturn, IT security budgets are tight and top management needs to understand how information security audits help to detect security threats, improve safeguarding of assets and ultimately decrease costs. Top management support and commitment to the information security is one of the key success factors in any effective information security project.

To obtain perfect security is not possible, and therefore the costs of information security should be commensurate with the business needs and security risks of any computer system.

Number	Title
ISO/IEC 13335-1:2004	Information technology – Security techniques – Management of information and communications technology security – Part 1: Concepts and models for information and communications technology security management
ISO/IEC 18028-1:2006	Information technology – Security techniques – IT network security – Part 1: Network security management
ISO/IEC 18028-2:2006	Information technology – Security techniques – IT network security – Part 2: Network security management
ISO/IEC 18028-3:2005	Information technology – Security techniques – IT network security – Part 3: Securing communications between networks using security gateways
ISO/IEC 18028-4:2005	Information technology – Security techniques – IT network security – Part 4: Securing remote access
ISO/IEC 18028-5:2006	Information technology – Security techniques – IT network security – Part 5: Securing communications across networks using virtual private networks
ISO/IEC 18045:2008	Information technology – Security techniques – Methodology for IT security evaluation
ISO/IEC 27000:2009 *under development*	Information technology – Security techniques – Information security management systems – Overview and vocabulary
ISO/IEC 27001:2005	Information technology – Security techniques – Information security management systems – Requirements
ISO/IEC 27002:2005	Information technology – Security techniques – Code of practice for information security management
ISO/IEC FCD 27003 *under development*	nformation technology – Security techniques – Information security management system implementation guidance
ISO/IEC FCD 27004.2 *under development*	Information technology – Security techniques – Information security management – Measurement
ISO/IEC 27005:2008	Information technology – Security techniques – Information security risk management
ISO/IEC 27006:2007	Information technology – Security techniques – Requirements for bodies providing audit and certification of information security management systems
ISO/IEC WD 27007 *under development*	Information technology – Security techniques – Guidelines for information security management systems auditing

Table 1: Overview of common ISO/IEC standards for IT/information security

Security Area	Sample audit question
Logical security	• Have all custom-developed applications been written with security in mind? • Have custom-developed applications been tested for security flaws?
	• Have the operating systems and commercial applications been updated with the appropriate security patches? • Have the security patches been tested before deployment?
User identification and authentication	• Are user names (IDs) unique and linked to real persons? • Is there a defined number of consecutive unsuccessful attempts to login?
User password management	• Are passwords in place? (e.g. minimum password length, password expiration, password reusability, disabled passwords that are no longer valid) • Are the users informed and asked to follow good security practices in selection and use of passwords? • Are all passwords changed regularly, especially the system administrator's? • How difficult are passwords to crack?
Virus protection	• Is anti-virus software installed on all computers? • Are personal computers regularly scanned for malware (e.g. computer viruses, logic bombs, spyware/adware)? • Is a procedure for automatically updating the anti-virus software in place?
Access control	• Are there access control lists (ACLs) in place for network assets to control who has access to shared data? • Are there access control lists in place for applications and information? • Are there access control lists in place for mobile computing and teleworking? • Are the user access rights reviewed at regular intervals and revised, if necessary? • Are there audit logs to record who has accessed data? • Are the audit logs reviewed?
Data backup and recovery	• How is backup media stored? • Who has access to backup media? • Is backup media up-to-date? • At what frequency are backups taken and tested? • Is there a method for performing a restore of the data?
Configuration and application change management	• How are configurations and code changes documented? • Do all system changes go through a formal change control process? • Have changes been tested before being placed into production? • How are records reviewed? • Who conducts the reviews?

Security Area	Sample audit question
User support	• Is user documentation (user manuals, online help, etc) available and up-to-date? • Have users been trained in the proper use and security of applications they use? • Is there a process for user improvement requests?
Disaster recovery	• Is there a disaster recovery plan in place? • Has a disaster recovery plan been reviewed and approved? • Are disaster recovery teams established to support disaster recovery plan? • Are recovery plans regularly tested? • Is there at least one copy of company's data and application software stored in a secure, off-site location? • Does a hardware maintenance contract exist with a supplier?
Information security policy	• Is there a documented security policy in place? • Who is the owner of the information security policy? • Who is responsible for its review according to a defined review process? • Has security policy been communicated to all employees and contractors? • Is there a clear-screen policy in place? • Is there a clear-desk policy to ensure that employees secure confidential files when they are not working on them?
Security awareness and training	• Is there security awareness and is an appropriate information security training program in place? • Have all copies of software been properly licensed and registered?
Physical and environmental security	• Are systems left logged in while employees are away? • Has physical protection against external and environmental threats (e.g. fire, flood, earthquake, explosion, electromagnetic interference) been designed and applied? • Has physical security for offices, rooms and facilities been designed and applied? • Are there physical entry controls to protect secure areas and to ensure that only authorised personnel are allowed access? • Are there physical protection and guidelines for working in secure areas? • Has equipment been protected from power failures and failures caused by other utilities? • Has equipment been correctly maintained to ensure availability and integrity? • Are network servers physically secure in a separate area?

Table 2: Sample generic information security audit checklist

Name of Tool	URL	Description
Nessus®	www.nessus.org	Remote network vulnerability scanner
SAINT®	www.saintcorporation.com	Network vulnerability scanner
IBM® Internet Scanner	www.iss.net	Network vulnerability scanner
Retina®	www.eeye.com	Network security scanner
QualysGuard® Suite	www.qualys.com	Tools for vulnerability management, policy compliance, PCI (Payment Card Industry) compliance, and web application scanning
CORE IMPACT Pro	www.coresecurity.com	Automated penetration security testing software
SATAN	www.porcupine.org/satan	Security administrator tool for analyzing networks
Nmap (Network Mapper)	nmap.org	Free and open source utility for network exploration and security auditing (port scanner)
John the Ripper	www.openwall.com/john	Multi-platform password hash cracker for detection of weak passwords
Crack	ftp.cerias.purdue.edu/pub/tools/unix/pwdutils/crack/	Password cracker
Tiger	www.nongnu.org/tiger	Unix security audit and intrusion detection tool
COPS (Computer Oracle and Password System)	www.nongnu.org/tiger ftp.cerias.purdue.edu/pub/tools/unix/scanners/cops/	System monitoring tool
Foundstone	www.foundstone.com	Many free tools and resources: forensic tools, Foundstone SASS (Software Application Security Services) tools, intrusion detection tools, scanning tools, stress testing tools, etc

Table 3: Overview of commonly used information security audit tools

Basic information security terms

An **asset** is anything that has value to the organization. [ISO/IEC 13335-1:2004] Assets are subject to many kinds of threats.

A **threat** is a potential cause of an unwanted incident, which may result in harm to a system or organization. [ISO/IEC 27001:2005] An example of a threat could be the accidental deletion of system data.

Vulnerability is defined as a weakness of an asset or group of assets that can be exploited by one or more threats. [After ISO/IEC 27001:2005] Vulnerabilities can be found in software, information systems, network protocols and devices, etc. If vulnerability is not managed, it will allow a threat to materialize. Examples of vulnerability are: unpatched software, weak passwords, lack of access control, no firewall installed, insufficient security training, unlocked doors and windows, shared accounts, programming input validation errors, etc.

A **risk** is the potential that a given threat will exploit vulnerabilities to cause loss or damage to an asset or group of information assets and thereby cause harm to the organization. It is measured in terms of a combination of the probability of an event and the severity of its consequences. [After ISO/IEC 13335-1:2004]

Information is an asset which, like other important business assets, has value to an organization and consequently needs to be suitably protected. [ISO/IEC 27002:2005] Information can be in various forms: printed or written on paper, stored electronically, transmitted by post or e-mail, shown on corporate videos, spoken in conversation or exists as knowledge acquired by individuals.

Information security is preservation of confidentiality, integrity and availability of information; in addition, other properties, such as authenticity, accountability, non-repudiation, and reliability can also be involved. [ISO 27002:2005]

Industrial espionage is unauthorized collection of confidential, classified or proprietary documents.

Biography

Nadica Hrgarek holds a B.Sc. in information systems and a Master of Science in information science from the University of Zagreb, Croatia.
Since March 2007 she has been a member of the RA/QA department at MED-EL Elektromedizinische Geräte GmbH (www.medel.com), a hearing implant company located in Innsbruck, Austria. Nadica is currently working as a Senior QA Specialist – Quality Improvement. Her responsibility covers all aspects of quality improvements ranging from coordination of corrective and preventive actions, non-product software validation support, conducting training, internal and supplier audits, etc.
Nadica is a certified ISTQB tester (full advanced level), ISO 9000 internal and lead auditor and has conducted more than 20 internal, product, process and supplier audits.
She is co-founder and Head of the Advisory Board of the Croatian Testing Board (CTB), which was founded in 2008. She is also a member of the German Association for Software Quality and Training (ASQF).

Advanced Software Test Design Techniques, Decision Tables and Cause-Effect Graphs

by Rex Black

Introduction

The following is an excerpt from my recently-published book, *Advanced Software Testing: Volume 1*. This is a book for test analysts and test engineers. It is especially useful for ISTQB Advanced Test Analyst certificate candidates, but contains detailed discussions of test design techniques that any tester can—and should—use. In this first article in a series of excerpts, I start by discussing the related concepts of decision tables and cause-effect graphs.

Decision Tables

Equivalence partitioning and boundary value analysis are very useful techniques. They are especially useful when testing input field validation at the user interface. However, lots of testing that we do as test analysts involves testing the business logic that sits underneath the user interface. We can use boundary values and equivalence partitioning on business logic, too, but three additional techniques, decision tables, use cases, and state-based testing, will often prove handier and more powerful techniques.

In this article, we start with decision tables. Conceptually, decision tables express the rules that govern handling of transactional situations. By their simple, concise structure, decision tables make it easy for us to design tests for those rules, usually at least one test per rule.

When I said "transactional situations," what I meant was those situations where the conditions—inputs, preconditions, etc.—that exist at a given moment in time for a single transaction are sufficient by themselves to determine the actions the system should take. If the conditions are not sufficient, but we must also refer to what conditions have existed in the past, then we'll want to use state-based testing, which we'll cover in a moment.

The underlying model is either a table (most typically) or a Boolean graph (less typically). Either way, the model connects combinations of conditions with the action or actions that should occur when each particular combination of conditions arises.

If the graph is used, this technique is also referred to as a cause-effect graph, because that is the formal name of the graph. However, it's important to keep in mind that any given decision table can be converted into a cause-effect graph, and any given cause-effect graph can be converted into a decision table. So, which one you choose to use is up to you. I prefer decision tables, and they are more commonly used, so I'll focus on that here. However, I'll show you how the conversion can be done.

To create test cases from a decision table or a cause-effect graph, we design test inputs that fulfill the conditions given. The test outputs correspond to the action or actions given for that combination of conditions. During test execution, we check that the actual actions taken correspond to the expected actions.

We create enough test cases that every combination of conditions is covered by at least one test case. Frequently, that coverage criterion is relaxed to say, we cover those combinations of conditions that can determine the action or actions. If that's a little confusing, the distinction I'm drawing will become clear to you when we talk about collapsed decision tables.

With a decision table, the coverage criterion boils down to an easy-to-remember rule of at least one test per column in the table. For cause-effect graphs, you have to generate a so-called "truth table" that contains all possible combinations of conditions and ensure you have one test per row in the truth table.

So, what kind of bugs are we looking for with decision tables? There are two. First, under some combination of conditions, the wrong action might occur. In other words, there is some action that the system is not to take under this combination of conditions, yet it does. Second, under some combination of conditions, the system might not take the right action. In other words, there is some action that the system is to take under this combination of conditions, yet it does not.

Consider an e-commerce application like the one found on our RBCS Web site, www.rbcs-us.com. At the user interface layer, we need to validate payment information, specifically credit card type, card number, card security code, expiration month, expiration year, and cardholder name. You can use boundary value analysis and equivalence partitioning to test the ability of the application to verify the payment information, as much as possible, before sending it to the server.

So, once that information goes to the credit card processing company for validation, how can we test that? Again, we could handle that with equivalence partitioning, but there are actually a whole set of conditions that determine this processing:

Does the named person hold the credit card entered, and is the other information correct?

Is it still active or has it been cancelled?

Is the person within or over their limit?

Is the transaction coming from a normal or a suspicious location?

The decision table in Table 1 shows how these four conditions interact to determine which of the following three actions will occur:

Should we approve the transaction?

Should we call the cardholder (e.g., to warn them about a purchase from a strange place)?

Should we call the vendor (e.g., to ask them to seize the cancelled card)?

Take a minute to study the table to see how this works. The conditions are listed at the top left of the table, and the actions at the bottom left. Each column to the right of this left-most column contains a business rule. Each rule says, in essence, "Under this particular combination of conditions (shown at the top of the rule), carry out this particular combination of actions (shown at the bottom of the rule)."

Notice that the number of columns—i.e., the number of business rules—is equal to 2 (two) raised to the power of the number of conditions. In other words, 2 times 2 times 2 times 2, which is 16. When the conditions are strictly Boolean—true or false—and we're dealing with a full decision table (not a collapsed one), that will always be the case.

Conditions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Real account?	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
Active account?	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	N	N	N	N
Within limit?	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
Location okay?	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
Actions																
Approve?	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Call cardholder?	N	Y	Y	Y	N	Y	Y	Y	N	N	N	N	N	N	N	N
Call vendor?	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Table 1: Decision Table Example (Full)

Did you notice how I populated the conditions? The topmost condition changes most slowly. Half of the columns are Yes, then half No. The condition under the topmost changes more quickly but more slowly than all the others. The pattern is quarter Yes, then quarter No, then quarter Yes, then quarter No. Finally, for the bottommost condition, the alternation is Yes, No, Yes, No, Yes, etc. This pattern makes it easy to ensure you don't miss anything. If you start with the topmost condition, set the left half of the rule columns to Yes and the right half of the rule columns to No, then following the pattern I showed, if you get to the bottom and the Yes, No, Yes, No, Yes, etc., pattern doesn't hold, you did something wrong.

Deriving test cases from this example is easy: Each column of the table produces a test case. When the time comes to run the tests, we'll create the conditions which are each test's inputs. We'll replace the "yes/no" conditions with actual input values for credit card number, security code, expiration date, and cardholder name, either during test design or perhaps even at test execution time. We'll verify the actions which are the test's expected results.

In some cases, we might generate more than one test case per column. I'll cover this possibility in more detail later, as we enlist our previous test techniques, equivalence partitioning and boundary value analysis, to extend decision table testing.

Collapsing Columns in the Table

Notice that, in this case, some of the test cases don't make much sense. For example, how can the account not be real but yet active? How can the account not be real but within limit? This kind of situation is a hint that maybe we don't need all the columns in our decision table.

We can sometimes collapse the decision table, combining columns, to achieve a more concise—and in some cases sensible—decision table. In any situation where the value of one or more particular conditions can't affect the actions for two or more combinations of conditions, we can collapse the decision table.

This involves combining two or more columns where, as I said, one or more of the conditions don't affect the actions. As a hint, combinable columns are often **but not always** next to each other. You can at least start by looking at columns next to each other.

Conditions	1	2	3	5	6	7	9
Real account?	Y	Y	Y	Y	Y	Y	N
Active account?	Y	Y	Y	N	N	N	-
Within limit?	Y	Y	N	Y	Y	N	-
Location okay?	Y	N	-	Y	N	-	-
Actions							
Approve?	Y	N	N	N	N	N	N
Call cardholder?	N	Y	Y	N	Y	Y	N
Call vendor?	N	N	N	Y	Y	Y	Y

Table 2: Decision Table Example (Collapsed)

To combine two or more columns, look for two or more columns that result in the same combination of actions. Note that the actions must be the same for all of the actions in the table, not just some of them. In these columns, some of the conditions will be the same, and some will be different. The ones that are different obviously don't affect the outcome. So, we can replace the conditions that are different in those columns with the dash character ("-"). The dash usually means either I don't care, it doesn't matter, or it can't happen, given the other conditions.

Now, repeat this process until the only further columns that share the same combination of actions for all the actions in the table are ones where you'd be combining a dash with Yes or No value and thus wiping out an important distinction for cause of action. What I mean by this will be clear in the example I present in a moment, if it's not clear already.

Another word of caution at this point: Be careful when dealing with a table where more than one rule can apply at one single point in time. These tables have non-exclusive rules. We'll discuss that further later in this section.

Table 2 shows the same decision table as before, but collapsed to eliminate extraneous columns. Most notably, you can see that columns 9 through 16 in the original decision table have been collapsed into a single column.

I've kept the original column numbers for ease of comparison. Again, take a minute to study the table to see how I did this. Look carefully at columns 1, 2, and 3. Notice that we can't collapse 2 and 3 because that would result in "dash" for both "within limit" and "location okay." If you study this table or the full one, you can see that one of these conditions must **not** be true for the cardholder to receive a call. The collapse of rule 4 into rule 3 says that, if the card is over limit, the cardholder will be called, regardless of location. The same logic applies to the collapse of rule 8 into rule 7.

Notice that the format is unchanged. The conditions are listed at the top left of the table, and the actions at the bottom left. Each column to the right of this left-most column contains a business rule. Each rules says, "Under this particular combination of conditions (shown at the top of the rule, some of which might not be applicable), carry out this particular combination of actions (shown at the bottom of the rule, all of which are fully specified)."

Notice that the number of columns is no longer equal to 2 raised to the power of the number of conditions. This makes sense, since otherwise no collapsing would have occurred. If you are concerned that you might miss something important, you can always start with the full decision table. In a full table, because of the way you generate it, it is guaranteed to have all the combinations of conditions. You can mathematically check if it does. Then, carefully collapse the table to reduce the number of test cases you create.

Also, notice that, when you collapse the table, that pleasant pattern of Yes and No columns present in the full table goes away. This is yet another reason to be very careful when collapsing the columns, because you can't count on the pattern or the mathematical formula to check your work.

Cause-Effect Graphs

When collapsing a decision table, a cause-effect graph can help you make sure you don't accidentally collapse columns you shouldn't. Some people like to use them for test design directly, but, as I mentioned earlier, I'm not too fond of trying to do that.

The process for creating a cause-effect graph from a decision table or decision table from a cause-effect graph is straightforward. To create a cause-effect graph from a decision table, first list all the conditions on the left of a blank page. Next, list all the actions on right of a blank page. Obviously, if there are a lot of conditions and actions, this will be a big page of paper, but then your decision table would be big, too.

Now, for each action, read the table to identify how combinations of conditions cause an action. Connect one or more conditions with each action using Boolean operators, which I show in Figure 1. Repeat this process for all actions in the decision table.

If you happen to be given a cause-effect graph and want to create a decision table, first list all the conditions on the top left of a "blank" decision table. Next, list all the actions on the bottom left of the decision table, under the conditions. Following the pattern shown earlier, generate all possible combinations of conditions. Now, referring to the cause-effect graph, determine the actions taken and not taken for each combination of conditions. Once the actions section is fully populated, you can collapse the table if you'd like.

In Figure 1, you see the cause-effect graph that corresponds to the example decision tables we've looked at so far. You might ask, "Which one, the full or collapsed?" Both. The full and the collapsed are logically equivalent, unless there's something wrong with the collapsed version.

At the bottom left of this figure, you see the legend that tells you how to read the operations. Let's go clockwise from the top left of the legend.

We have simple causality: If A is true, B will occur, or, in other words, A causes B.

We have negation: When A is not true, B will occur, or, not A causes B.

We have AND operation: When A1 and A2 are both true, B will occur, or A1 and A2 causes B.

We have OR operation: When A1 or A2 is true, B will occur, or A1 or A2 causes B.

Let's look at the connection between conditions and actions. The solid causality lines, together with an AND operator, show that all four conditions must be met for the transaction to be approved.

The dashed causality lines, together with negation operators and an OR operator, show that, if the account is not real or the account is not active, we will call the vendor.

The dotted causality lines are a bit more complicated. First, we combine the "within limit" and "location okay" conditions, with negation operators and an OR operator, to create an intermediate condition of "Limit or location

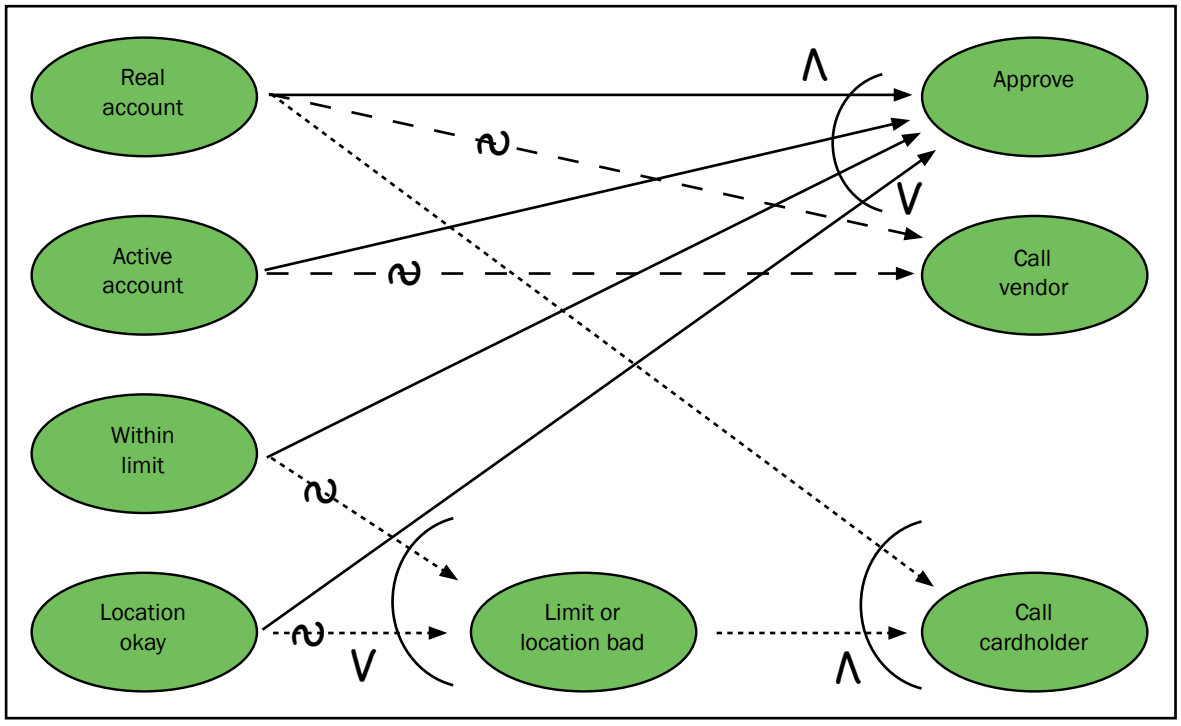


Figure 1: Cause-Effect Graph Example

bad". Now, we combine that with the "real account" condition to say that, if we have an over-limit or bad location situation, and the account is real, we will call the cardholder.

Combining Decision Table Testing with Other Techniques

Let's address an issue I brought up earlier, the possibility of multiple test cases per column in the decision table via the combination of equivalence partitioning with the decision table technique. Let's refer back to our example decision table shown in Table 1, specifically column 9.

We can apply equivalence partitioning to the question of, "How many interesting—from a test point of view—ways are there to have an account not be real?" As you can see from Figure 2, this could happen six potentially interesting ways:

- Card number and cardholder mismatch
- Card number and expiry mismatch
- Card number and CSC mismatch
- Two of the above mismatches (three possibilities)
- All three mismatches.

So, there could be seven tests for that column.

How about boundary value analysis? Yes, you can apply that to decision tables to find new and interesting tests. For example, "How many interesting test values relate to the credit limit?"

As you can see from Figure 3, equivalence partitioning and boundary value analysis show us six interesting possibilities:

- The account starts at zero balance
- The account would be at a normal balance after transaction
- The account would be exactly at the limit after the transaction
- The account would be exactly over the limit after the transaction
- The account was at exactly the limit before the transaction (which would ensure going over if the transaction concluded)
- The account would be at the maximum overdraft value after the transaction (which might not be possible)

Combining this with the decision table, we can see that would again end up with more "over limit" tests than we have columns—one more, to be exact—so we'd increase the number of tests just slightly. In other words, there would be four within-limit tests and three over-limit tests. That's true unless you wanted to make sure that each within-limit equivalence class was represented in an approved transaction, in which case column 1 would go from one test to three.

Non-Exclusive Rules in Decision Tables

Let's finish our discussion about decision tables by looking at the issue of non-exclusive rules I mentioned earlier. Sometimes more than one rule can apply to a transaction. In Table 3, you see a table that shows the calculation of credit card fees. There are three conditions, and notice that zero, one, two, or all three of those conditions could be met in a given month. How does this situation affect testing? It complicates the testing a bit, but we can use a methodical approach and risk-based testing to avoid the major pitfalls.

Conditions	1	2	3
Foreign exchange?	Y	-	-
Balance forward?	-	Y	-
Late payment?	-	-	Y
Actions			
Exchange fee?	Y	-	-
Charge interest?	-	Y	-
Charge late fee?	-	-	Y

Table 3: Non-exclusive Rules Example

To start with, test the decision table like a normal one, one rule at a time, making sure that no conditions not related to the rule you are testing are met. This allows you to test rules in isolation—just like you are forced to do in situations where the rules are exclusive. Next, consider testing combinations of rules. Notice I said, "consider," not "test all possible combinations of rules." You'll want to avoid combinatorial explosions, which is what happens when testers start to test combinations of factors without consideration of the value of those tests. Now, in this case, there are only eight possible combinations—three factors,

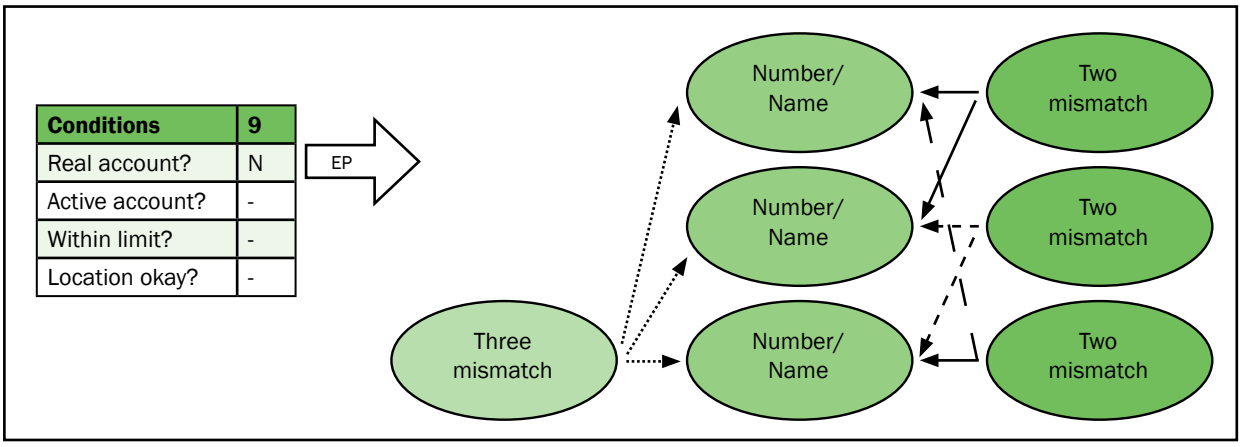


Figure 2: Equivalence Partitions and Decision Tables

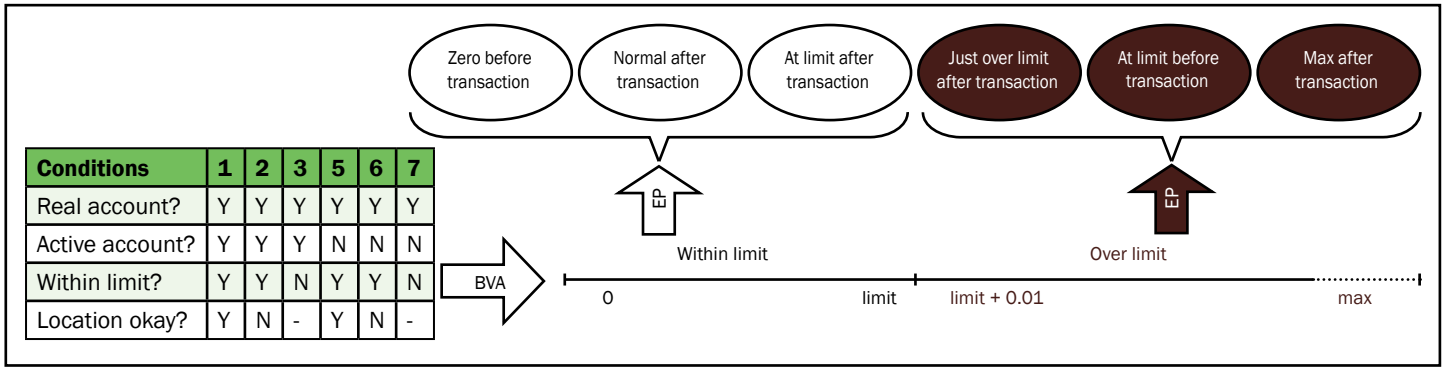


Figure 3: Boundary Values and Decision Tables

two options for each factor, 2 times 2 times 2 is 8. However, if you have six factors with five options each, you now have 15,625 combinations.

One way to avoid combinatorial explosions is to identify the possible combinations and then use risk to weight those combinations. Try to get to the important combinations and don't worry about the rest. Another way to avoid combinatorial explosions is to use techniques like classification trees and pairwise testing (see my books *Advanced Software Testing: Volume 1* or *Pragmatic Software Testing* for a further discussion on those techniques).

Conclusion

In this article, I've shown how to apply decision tables and cause-effect graphs to the testing of sophisticated and complex internal business logic in applications. Decision tables are a great way to test detailed business rules in isolation, especially for transactional types of situations. However, we will need to look at two additional techniques, use cases and state-based test techniques, to deal with the full range of internal business logic testing we need to do. I'll address those techniques in the next two articles in this series.

Biography

With a quarter-century of software and systems engineering experience, Rex Black is President of RBCS (www.rbcs-us.com), a leader in software, hardware, and systems testing. For over a dozen years, RBCS has delivered services in consulting, outsourcing and training for software and hardware testing. Employing the industry's most experienced and recognized consultants, RBCS conducts product testing, builds and improves testing groups and hires testing staff for hundreds of clients worldwide. Ranging from Fortune 20 companies to start-ups, RBCS clients save time and money through improved product development, decreased tech support calls, improved corporate reputation and more. As the leader of RBCS, Rex is the most prolific author practicing in the field of software testing today. His popular first book, Managing the Testing Process, has sold over 35,000 copies around the world, including Japanese, Chinese, and Indian releases. His five other books on testing, Advanced Software Testing: Volume I, Advanced Software Testing: Volume II, Critical Testing Processes, Foundations of Software Testing, and Pragmatic Software Testing, have also sold tens of thousands of copies, including Hebrew, Indian, Chinese, Japanese and Russian editions. He has written over thirty articles, presented hundreds of papers, workshops, and seminars, and given about thirty keynote speeches at conferences and events around the world. Rex has been President of the International Software Testing Qualifications Board and is a Director of the American Software Testing Qualifications Board.

Load Testing In 10 Steps

by Shai Raiten

Most software companies perform load testing on their products. Load testing is one of the most important testing types today. When the Amazon website crashed for two hours on 6/6/2008, it would have been difficult to calculate the exact amount that outage cost Amazon, because customers could come back later for purchases. However, the stock closed down 4.6 percent at $80.63. Everyone thought: "*They know how to handle load*".

In this article, I'm not going to talk about load testing tools or advanced testing techniques, but rather, I would like to talk about 10 basic steps that are the foundation for creating a good, precise and powerful load test suite.

Step 1 - Identify Objectives

The purpose of this step is to identify and write the performance objectives of your application. The key question you should ask yourself is:

"How should my application behave under load?"

The main parameters we should consider are:

Response time - The time that the application would take to display a certain output or perform a certain calculation. Example: the product catalog must be displayed in less than 3 seconds.

Throughput – The rate of successful message delivery over a communication channel. Example: the system must support 100 requests per second.

Resource utilization - A frequently overlooked aspect, resource utilization defines how much resource your application consumes in terms of CPU, memory, disk I/O, and network I/O.

Maximum user load - Determine how many users can run on your testing hardware configuration.

This is probably the most important step!

Step 2 - Identify Key Scenarios (or Profiles)

What are scenarios?

Scenarios are anticipated user paths that generally incorporate multiple application activities.

How do you identify scenarios?

Key scenarios are those for which you have specific performance goals or those that have a significant performance impact. These scenarios represent business activity of users over time.

For example: Open 'about window' will take less resource than perform 'buy' action.

The "Buy" action, as opposed to "Open about window", will involve multiple actions like: SQL, Credit Card validation, IIS.

Step 3 - Identify the Workload

Identify the distribution / ratio of the work - For each key scenario, identify the distribution / ratio of the work. The distribution is based on the number of users executing the scenario (according to their profile).

For an existing application this information can be provided from IIS log/counters as described in step 1.

For a new application this information can be based on market research, historical data, market trends and prototypes.

Calculate the users load per scenario - Based on the previous data, calculate the maximum possible concurrent users for the application. Using the work distribution for each scenario, calculate the % user load per key scenario. For example, the distribution of load for a key scenario could be similar to that shown in the following table.

Scenario	% of users	Sum
Login	60	600
Registration	10	100
Buy	30	300
Total	**100**	**1000**

Step 4 - Identify Metrics

Metrics are a derivative of your performance objectives. They are used to measure your application's real-time performance in comparison with your performance objectives. In addition, they also help you to identify problems and bottlenecks within your application.

Network-specific metrics: This set of metrics provides information about the overall "health" and efficiency of your network, including routers, switches, and gateways.

System-related metrics: This set of metrics help you identify the resource utilization on your server. The set includes CPU, memory, disk I/O, and network I/O metrics.

Platform-specific metrics: Platform-specific metrics are related to software that is used to host your application, such as the .NET Framework common language runtime and ASP.NET-related metrics.

Application-specific metrics: These include custom performance counters embedded in your application code that monitor the application's "health". You might use custom counters to determine the number of concurrent threads waiting to acquire a particular lock or the num-

ber of requests queued to make an outbound call to a Web service.

Service level metrics: Service level metrics can help to measure overall application throughput and latency, or they might be tied to specific business scenarios.

Metric	Accepted Level
% Process Time	No more than 75%
%Physical Memory	No more than 55%
Network utilization	Less than 65%
Total # of Threads	**No more than 50 per process**

Step 5 – Pick Load Test Tool

Before writing tests, we need to pick the right load testing tool.

In order to select the proper tool for our application, we will need to do some research. Load testing tools require specific knowledge, and each tool has its advantages and disadvantages.

Everyone can learn how to use a load testing tool. However, different tools are better for particular purposes, so selecting the right tool can have significant impact on our testing process.

Most tools have very good functionality and advanced features. One of the first parameters you would use for comparing tools is the user limit (or lack thereof) and the scalability of that limit. As a rule of thumb, try to avoid tools that cap the amount of users simulated via licensing (more users = more money).

For example, Team System Test Edition has no user Limit, so we can simulate as many users as our hardware allows us to.

Personally, I use Microsoft's Team System; if you have picked this tool, you may contact me for assistance.

Step 6 - Create Test Cases

- What is a test case?
  - A group of activities involved in a scenario/user profile.
  - The test cases are created based on the scenarios and the profile mix identified in the previous steps.

Each test case should include the expected results in such a way that each test case can be marked as a 'pass' or 'fail' after execution.

Test Case: Search phone owner

ID	Description	Expected Results
1	Open Application	Application is up.
2	Perform login with user: Demo and password: 123	User details: Name: Demo User Last: Demo User City: Tel Aviv
3	Search phone number: 555-123456	User details: Name: Demo User Last: Demo User City: Tel Aviv

Create an automated test and set a specific load configuration for it.

Example: 400 users for 1 hour

And load test expected results for Test Case: Search phone owner:

Category	Expected Results
Avg. response time	Lower than 8 sec.
Resource utilization - Processor time - Available Memore	 - Lower than 75% - Lower than 55%

Step 7 – Prepare\Understand Your Load Environment

Our setup environment should duplicate the architecture of our production environment as closely as possible.

It is important to simulate the load on environment similar to our production environment, because even the smallest hardware or configuration difference can have a big impact on our results.

Creating a testing environment which exactly duplicates the production environment can be hard and isn't always feasible, but we need to do our best, because this environment will help us estimate the load results of our product.

We must understand the hardware limits of our environment and find the bottlenecks before starting the test.

Example: I have created a load environment with 2 computers that will run 10,000 concurrent users, the ISP provides 1Mbps.

Each user will open the browser and perform a search for a dynamic word in www.live.com.

Opening 5000 browsers on one machine?

I've never tried it, but I'm sure that CPU usage will be100% for a while.

When CPU usage is 100% the computer is not creating the needed load, and we will get false information, so this is our first bottleneck.

Do you think 1Mbps is enough bandwidth for 10,000 users?

No! 10,000 users cannot work together on 1Mbps bandwidth. This is the second bottleneck.

Step 8 – Run it Step by Step

Begin load testing with a small number of users distributed against the user profile, and then increase the load incrementally. It is important to have sufficient time between each step, so that the system has enough time to stabilize before the next set of user connections executes the test case.

Incrementing the number of users slowly will make it easier to find the exact point/threshold where the system crashes or hangs due to load. Starting the tests with a large number of simulated users will prevent us from detecting that point/threshold efficiently.

Run the load test in cycles. Each cycle should achieve a certain load increment, and should have analysis and fixing time in between. Check the metrics for each cycle, and document them, so you can show evidence that the load was achieved already.

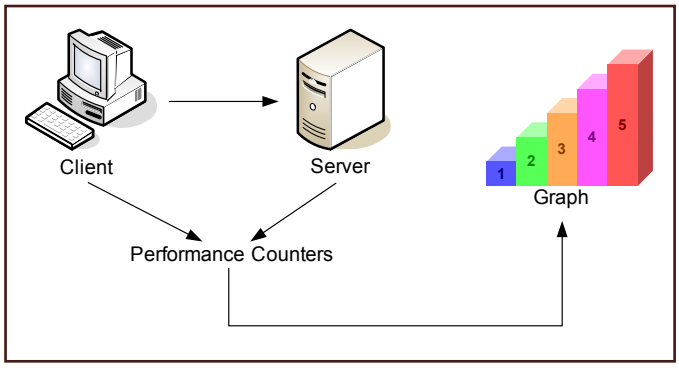
Step 9 - Run

After successfully implementing step 8, the system is considered stable, and we can run the full load test as mentioned in the preplanned workload.

Before running the test, we must make sure we are monitoring both the computer running the load test and the computer taking the load. This will help us find bottlenecks on both sides of the test.

Team System Test Edition can perform such dual monitoring.

If your load tool doesn't have this ability, you can use Windows Performance Monitor (Perfmon). The disadvantage of using Perfmon is that you will have two or more separate graphs which you need to analyze.



Step 10 - Analyze and Evaluate the Results

After each run, and of course after the full run, we analyze the results and check against the metrics and make sure our objectives were achieved.

We save the results of each run and compare

them with other runs. This way we may notice improvement or deterioration of the performance and load of the application.

Make sure you can share your load and performance test results and evaluations. Performance and load testing is a serious discipline, but it also needs to be understood by other key business disciplines such as business operations, senior management and maybe finance.

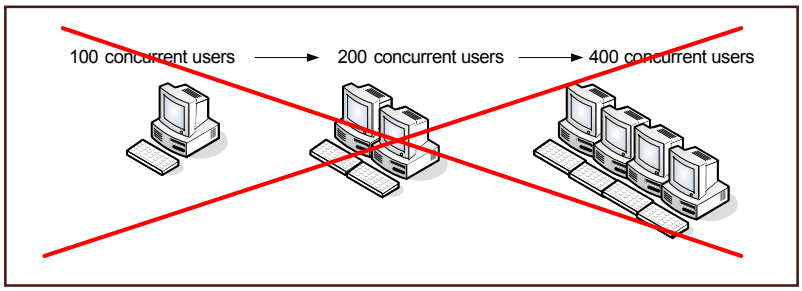
Try to simplify things, so it will be understood by those specific populations: It helps to have some kind of graphical reporting ability (in the tool) that allows us to share our test results with other parts of our organization.

My Rule is - *The results from one environment aren't necessarily equal to the results of another.*

Can you say that  ==  I say maybe…

X10

Example: The production environment has 10 computers and the load testing environment has 2 computers. The load test showed that 2 computers can hold more than 1000 users, so 10 computers can hold approximately 5000 users? ***NO!***



There is no doubt that 10 computers are stronger than 2, but we need to take into consideration that we have different hardware and configurations for each environment that can affect our results.

So why perform load testing if the results are inconclusive to production?

Load Testing helps you to **estimate** the behavior of the application under load.

Summary

That being said, I'm sure that for expert performance test engineers the above rules of thumb are obvious – you have tried them in trial and error situations. For all the other testers, who would like to know how to start a performance and load test and get things done, I hope I have explained in simple enough words what the basic step by step actions are that I suggest should be followed, in order to create a good, precise and powerful load test suite. Good luck!

Resources

Team System Team Edition – http://msdn.microsoft.com/en-us/teamsystem

Windows Performance Counters - http://msdn.microsoft.com/en-us/library/aa373083(VS.85).aspx

Shai's Blog – http://blog.microsoft.co.il/blogs/shair

Biography

Shai Raiten is Team System MVP, currently working for Sela Group as ALM senior consultant and trainer specialized in Microsoft technologies, especially Team System and .NET technology. Shai has 6 years experience that revolves around Software/Hardware QA, Test Management, Automation and load testing. In the past he worked as load expert for a big outsourcing company.

Testing the Enterprise Security: Anti-Spam and Anti-Virus

by Dr. Marian Ventuneac

Enterprise Anti-Spam and Anti-Virus solutions are widely used to protect corporate e-mail servers against various external threats including spamming, viruses, spyware, and phishing attacks. Usually claiming a high rate of malicious message filtering (between 95-99%), it is hard to argue that its main purpose is realized. However, no comprehensive benchmarking on how such security solutions stand against internal attacks is currently available. Relying on various commercial and open-source technologies (Microsoft .NET, MySQL, PHP, Linux, Apache HTTP server, etc.), the majority of Anti-Spam and Anti-Virus enterprise solutions employ Web-based applications to allow remote configuration, administration and management of spam-quarantined e-mails. While Web-based applications are often found to be vulnerable to a wide variety of security vulnerabilities (including SQL Injection, Cross-Site Scripting, Denial of Service, Privilege Escalation, etc.), such enterprise security solutions make unfortunately no exception.

This paper highlights the need of vendor-certified security testing for Anti-Spam and Anti-Virus enterprise solutions, in order to protect it against internal attacks. In a structured effort to benchmark and potentially improve various enterprise security products, the author's recent research done in collaboration with Data Communication Security Laboratory from University of Limerick, (Ireland) is presented. Various security vulnerabilities identified in high-profile enterprise Anti-Spam and Anti-Virus products commercialized by vendors such as Marshal8e6 [1], Barracuda Networks [2], and Symantec [3] are discussed, while the implications of vulnerabilities exploitation and the risks for the enterprise are analyzed.

Anti-Spam and Anti-Virus Solutions Overview

Anti-Spam and Anti-Virus enterprise products for protecting e-mail servers are commercialized as software-based solutions, hardware appliances or security virtual appliances. Software-based filtering solutions require pre-installed and configured operating systems running on hardware or virtual appliances. Such solutions are built for specific network infrastructures and servers, often come with a lengthy list of hardware and software requirements, and significant effort could be required to configure and integrate them within the existing infrastructure.

The security integrated hardware and software solutions are considered to be more robust, high-performance, scalable and cost-effective than the traditional software-based solutions. Such products are deployed as hardware appliances, embed hardened operating systems and various pre-configured commercial and open-source security software, are control-locked by the vendors and require limited configuration to be done by customers for integration purposes.

Security virtual appliances bundle virtual machines for allocating the required physical hardware resources (CPU, memory, disk storage, network interface) with hardened operating system and a variety of pre-installed and pre-configured security software packages, including firewalls, anti-spam, message filtering, and web filtering solutions. With the proliferation of virtualization and cloud computing, security virtual appliances are quickly adopted by companies worldwide, mainly due to the increased availability, minimal dependency on the hardware hosting the virtualization platform, minimal required configuration and cost-effectiveness.

Tested Enterprise Solutions

Representative Anti-Spam and Anti-Virus solutions from Marshal8e6, Barracuda Networks and Symantec were tested as part of this work.

MailMarshal SMTP, a Gateway Security product from Marshal8e6, is an enterprise Anti-Spam and Anti-Virus software solution providing anti-spam, e-mail threat protection, content security, policy enforcement and data leakage prevention services. Running on Windows platforms, the product provides a Web-based interface built on Microsoft .NET framework for remote access and management of quarantined e-mail from both inside and outside the enterprise.

Multiple security products from Barracuda Networks tested include Barracuda Spam and Virus Firewall, Barracuda Web Filter, Barracuda IM Firewall and Barracuda Message Archiver. Built on a hardened Linux kernel, Barracuda Spam and Virus Firewall is a hardware appliance, with various open-source and commercial software solutions for Anti-Spam and Anti-Virus protection of e-mail servers. Additionally, the product is recommended for protecting the enterprise against spoofing, phishing, spyware and Denial of Service (DoS) attacks. Barracuda Web Filter is an integrated solution for web filtering, providing protection against spyware and malware, content filtering and web access control. Barracuda IM Firewall provides powerful management and monitoring functionality for internal and external instant messaging services, while Barracuda Message Archiver is an effective enterprise-class e-mail archiver.

Symantec Brightmail Gateway Virtual Edition is an enterprise virtual appliance built on a hardened RedHat Linux, providing protec-

tion for e-mail and instant messaging against spam, viruses and malicious content attacks. Running on VMware virtualization platforms, the Symantec solution is one of the newest enterprise security products being commercialized as a virtual appliance.

Testing Methodology

The increased availability of the Anti-Spam and Anti-Virus enterprise solutions allows prospective customers and security researchers to evaluate the provided functionality. However, attackers also have the opportunity to test, discover, exploit and even trade security vulnerabilities affecting such products. Furthermore, the proliferation of the security virtual appliances allows potential attackers to avail of fully-functional, safe and controlled environments for identifying security vulnerabilities and to devise efficient, low-footprint attacks targeting the virtual appliances, as well as hardware appliances and software solutions using the vulnerable code base.

While the installation and configuration of each of the above enterprise security solutions varies greatly, the testing methodology used to evaluate its security is pretty much the same, aiming to discover and exploit vulnerabilities in the provided configuration and management Web-based applications. The devised testing methodology uses a generic security test plan to check the robustness and effectiveness of built-in security controls providing security functionality for achieving data confidentiality, integrity and availability. For example, checks were performed to determine if suitable authentication and authorization controls were built for the tested solutions. Furthermore, custom tests were devised for underprivileged authenticated users to attempt gaining higher privileges through escalation of privileges, to potentially gain full administrative control of the software or the appliance. Additionally, the security assessment focused on how authenticated attackers could potentially discover and exploit various security vulnerabilities, such as those identified by the OWASP Top 10 Vulnerabilities Classification [4]:

- Cross Site Scripting (XSS)
- Injection Flaws (SQL Injection)
- Malicious File Execution
- Insecure Direct Object Reference
- Cross Site Request Forgery (CSRF)
- Information Leakage and Improper Error Handling
- Broken Authentication and Session Management
- Insecure Cryptographic Storage
- Insecure Communications
- Failure to Restrict URL Access

A black-box testing approach was exercised, using a range of manual and open-source automated security testing solutions including WebScarab, Paros Proxy, Burp Suite, and techniques such as parameter fuzzing, request and response manual and automated manipulation, scripts manipulation and alteration of hidden fields, session attributes analysis, URL spidering and crawling, etc. Where security vulnerabilities were discovered, custom exploits were built to assess the potential damages and the risk an enterprise is exposed to when an attacker successfully compromises the product security.

For such identified vulnerabilities, a responsible disclosure process was followed to notify the vendors of the affected products. Providing recommendations and collaboration with the vendors to identify the causes, other affected versions and products, and coordination of the public release of security advisories were an important aspect of the work, as well as increasing the security awareness with the enterprise on the identified security vulnerabilities.

Vulnerability Analysis

Various security vulnerabilities were identified as being exploitable for each of the analyzed products. Such vulnerabilities include cross-site scripting, cross-site request forgery, SQL injection, and escalation of privileges, as presented below.

Cross-Site Scripting (XSS)

Undoubtedly, Cross-Site Scripting (XSS) vulnerabilities were the most often encountered and successfully exploited in all the tested products.

Two persistent XSS vulnerabilities identified in MailMarshal SMTP 2006 [5] allow injecting malicious JavaScript and HTML code using the "list of blocked senders" and "list of safe senders" functionality. When an internal user A exploits the vulnerability and uses it against another user B via the "delegated spam management" functionality, this could expose user B session information, to enable unauthorized access to user B intranet, or to install malicious files or Trojans on user B computer.

A significant number of XSS vulnerabilities were found in all the tested Barracuda hardware appliances [6], and could be exploited by manipulating parameters of index.cgi resource. Thus, the Search Based Retention Policy functionality of Barracuda Message Archiver allows persistent XSS attacks, while manipulation of various parameters in IP Configuration, Administration, Journal Accounts, Retention Policy, and GroupWise Sync functionality allow multiple reflected XSS attacks. Additionally, error reporting, search functionality and manipulation of hidden parameters could allow an attacker to perform a multitude of reflected XSS attacks against Barracuda Spam Firewall, IM Firewall and Web Filter products.

Similarly, Symantec Brightmail Gateway virtual and hardware appliances were identified as being vulnerable to multiple reflected XSS attacks [7], by manipulating parameters of the edit.do, PatternFlow$viewReadOnly.flo, ComplianceFlow$edit.flo, saveSpamSettings.do and saveSpamSettings.do resources, as shown below:

```
url_placeholder/edit.do?userID=%3Cscript%3Ealert(
%27xss%27)%3C/script%3E

url_placeholder/PatternFlow$viewReadOnly.
flo?patternId=<STYLE>@import"javascript:alert('xss
')";</STYLE>

url_placeholder/ComplianceFlow$edit.
flo?complianceFolderId=<img%20
src="javascript:alert('xss')">

url_placeholder/saveSpamSettings.do?...&updateTab
=""><script>alert('xss')</script>&…

url_placeholder/runUtil-
ity.do?selectedHost=<img%20
src="javascript:alert('xss')">
```

Cross-Site Request Forgery (CSRF)

Multiple Cross-Site Request Forgery (CSRF) vulnerabilities were discovered and could be exploited in several of the tested products, such as MailMarshal SMTP and Barracuda Message Archiver. By injecting custom JavaScript and HTML code, an attacker could potentially target administrative or management functionality from the same or other applications used by the user(s) being attacked.

SQL Injection

A SQL Injection vulnerability was identified using parameter fuzzing techniques in Barracuda Spam Firewall products [8], allowing an internal attacker to successfully compromise the appliance security. The discovered SQL Injection vulnerability allows an internal attacker to inject arbitrary SQL code into the query used to filtering the user accounts in Users->Account View section. The value of pattern_x parameter (where x = 0..n) can be manipulated for this purpose, once filter_x parameter is set to 'search_count_equals' value.

Using the above vulnerability, exploits were devised to identify the used database engine (MySQL), the database version, database user, database name, etc:

```
url_placeholder&pattern_0=if(version() like
concat(char(52),char(37)),5,0)

url_placeholder&pattern_0=if(user() like
concat(char(114),char(37)),5,0)

url_placeholder&pattern_0=if(database()
like concat(char(99),char(37)),5,0)
```

Gaining such information allows an attacker to devise more powerful exploits, such as the following Denial of Service (DoS) exploit employing the MySQL benchmark() function:

```
url_placeholder&pattern_0=if(rand(benchmark
(x,sha1(y))),5,0).
```

Provided the values of x and y placeholders are large enough, this attack could effectively consume the resources of the hardware appliance (CPU and memory), thus severely affecting the availability of the appliance.

Escalation of Privileges

Three major vulnerabilities were recently identified as exploitable in Symantec Brightmail Gateway hardware and virtual appliances [9], allowing an underprivileged authenticated user to access protected user and system information, use resources requiring administrative privileges for altering appliances settings, and to gain complete administrative privileges.

By manipulating the value of userID parameter of edit.do resource, an attacker could enumerate all the valid accounts configured for the appliance:

```
url_placeholder/administrator/edit.
do?userID=x
```

where x is any value between 1 and the maximum number of user accounts n. This allows harvesting user information, such as user name IDs and e-mail addresses.

An internal underprivileged attacker could access the resources used to initially configure the security appliances, thus compromising the appliance's network and monitoring settings:

```
url_placeholder/setup/
SiteSetupAppliance$exec.flo?flowId=0
```

Furthermore, the security of the Symantec appliances was severely compromised by elevating the privileges of a user with no rights to manage user accounts, to gain full administrative control of the attacked appliances. Discovered by using parameter fuzzing techniques (which proves very useful once again), the exploitation of the identified vulnerability allows an attacker to successfully create a new user account with full administrative privileges.

Assessing the Risk for the Enterprise

Having the benefit of being trusted entities, internal attackers could devise and effectively use custom attacks targeting individuals or a large number of employees of the enterprise. Considering that most of the existing auditing controls are not designed to identify and log such attacks, network and system administrators would not have enough information to detect it and take preventive measures.

Using the Common Vulnerability Scoring System (CVSS) standard [10], the potential risk for an enterprise associated with the identified vulnerabilities was evaluated as follows:

- XSS and CRSF classified as low to medium risk (CVSS score 3.5 - 4.3)
- SQL Injection classified as of medium risk (CVSS score 6.5)
- Escalation of Privileges classified as of high risk (CVSS score 9)

The identified XSS vulnerabilities could be exploited by an internal attacker to achieve user(s) information disclosure, session hijack, access to Intranet servers, installing Trojans on other users' desktops, or injecting malicious JavaScript code to request inappropriate materials 'on behalf' of the user(s) being attacked.

The CRSF attacks could be used by an attacker to access administrative and account management functionality, usually accessible only to the attacked user. This could lead to compromised user account integrity, or elevation of privileges.

Exploiting the identified SQL Injection vulnerability in Barracuda Spam Firewall could lead to database information disclosure and effective Denial of Service attacks. Disclosing database information could be used to devise database version-specific attacks, based on published security vulnerabilities. Provided that the database process runs with root or other administrative privileges, this information could be used to devise more powerful attack vectors to potentially extract sensitive content from available databases and tables. When the DoS attack is successfully performed, this could affect the product functionality and services availability.

Furthermore, the exploitation of the high-risk escalation of privileges vulnerabilities identified requires low-complexity exploits, and could completely compromise the confidentiality, integrity and availability of the services provided by the appliance.

Conclusions

In the author's opinion, security testing performed either by the vendors or by enterprises using the affected products would most likely have identified the majority (if not all) of the vulnerabilities discussed in this paper. While several vulnerabilities are affecting versions of the products used worldwide for years, it is imperative that security of such products is thoroughly tested. Provided that Anti-Spam and Anti-Virus security solutions vulnerable to similar attacks to those described in this paper are deployed as part of enterprise security infrastructures, this will most likely lead to compromised enterprise security, often quantifiable by damaged reputation and financial losses.

References

[1] Marshal8e6, http://www.marshal8e6.com/

[2] Barracuda Networks, http://www.barracudanetworks.com/

[3] Symantec, http://www.symantec.com

[4] Open Web Application Security Project, http://owasp.org

[5] CVE-2008-2831, http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2008-2831

[6] CVE-2008-0971, http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2008-0971

[7] CVE-2009-0063, http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2009-0063

[8] CVE-2008-1094, http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2008-1094

[9] CVE-2009-0064, http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2009-0064

[10] Common Vulnerability Scoring System, http://www.first.org/cvss/

Biography

Dr. Marian Ventuneac is a Senior Technical Consultant and Lead Security with Perot Systems (Ireland). In a client facing position, Marian was responsible with the functional and technical design, development and delivery of large-scale technology based business solutions, providing Lead Security specialist skills across numerous projects and pre-sales activities.

Being a Computer Science graduate of Technical University of Cluj-Napoca (Romania), Marian was offered in 2001 a Master research scholarship on Applied Security Frameworks, with Data Communication Security Laboratory (DCSL), University of Limerick (Ireland).

Software Test Automation: Frame Your Own Requirements

by Suri Chitti

You might think of doing test automation for many reasons. You need to look at your reasons closely. Do your reasons look like text book examples such as 'I want to avoid having my testers do repetitive tasks that can easily be automated'? Then ask yourself this question: 'If I can avoid having my testers do repetitive tasks that can easily be automated, will it yield a tangible benefit to my test process?' You should have a sound statement of purpose for test automation. It should be something like this. 'Over the next two years, my key web-based product 'WebClassRoom' will go through two major releases. Each release will have several patch cycles. I need an automated test system that can regression test a patch version in 1 week. This will add value to my patch testing process.'

This statement alone is far from being all that you need from test automation. You should completely articulate what you need and expect from the automated test system. These should be your test automation requirements. Traditionally, you would begin by looking at two things: a functional test automation tool that would allow you to translate your functional test cases into automated code and/or a tool that could simulate a number of users to generate a load and test your application for performance-related factors. This approach is more likely to jade your requirements in terms of the features available on commercial tools and make you end up with a list of generic expectations like cost reduction, avoiding manual tedium, increasing test coverage etc. These are goals that may or might not be realized and should not be your requirements

You should rather look at your test process to come up with your requirements. You do not want to have an automated test system that aims to meet generic goals but stands separate from your test process. Neither should you have two test processes – one manual and one automated. You should have a single test process and your automated test system should plug into it. Here are some examples of process-centric requirements: You might be using a test management tool to streamline your test process. You would be forgetting something if you were not expecting the automated system to integrate with your tool. Your application might need substantial configuration before you execute test cases on it. You should expect your automated test system to automate the configuration before running actual tests. You might have a certain pattern of running tests that you would like to be replicated in your automated test system. You might be in possession of a certain automation tool that you would like to be utilized in your automated system. This is fair because tools are expensive, and instead of leaving the tool decision to the design stage you might consider its use as a requirement if you possess it. There are a number of reasons why you could end up having a certain tool without having been actively involved in the tool acquisition process. These could be strategic alliances or tie-ups with vendors (decisions made at a higher level of management), mergers with other companies (thereby acquiring their tools), and tool purchases made by other groups in your organization (and not utilized). The tool you possess may not be best suited to your test process or applications. However, most tools can be made to work in most circumstances with the right skills and effort. There are expectations that are feasible from automated runs but not so feasible from manual testing. For example, you might be tempted to look at and retrieve errors from the application or database logs following an automated run, knowing fairly well that the activity on the application is constant and intended in automated runs. Further, you may want to find out the specific actions that caused the errors. You might want to have your automated system do some cleanup or backup following a run. You might want to have your results formatted in a particular way and sent to key stakeholders. You might want to have defects to be automatically logged to a tracking system (though such features are specific utilities offered by some tools and may not be worthwhile building from scratch in your environment, in the absence of such tools). You might have a practical need to have test data parameterized, say if it is not uncommon for you to test patch versions delivered to specific clients when running tests with their specific data. You might have requirements about the usability of the automated system, particularly when the end users may not be skilled automation test engineers.

At the end of this important exercise, you will have a list of requirements that are centered around your test process. Having these requirements is not sufficient to decide whether you will go for automation or not. Assessments, estimations, feasibility and ROI considerations, etc will have to follow. However, it is an important and first step towards successful test automation. Test automation implemented around process-centric requirements is more likely to add value to your testing. Being sure of what you want will keep you ahead of the game, whichever way you choose to implement the automation – whether you wish to do it in-house or seek the services of a vendor specializing in providing automated test solutions. Going to a vendor with a list of specific requirements that tell more about your process is much better than going with a wish list of generic benefits.

Biography

My name is Suri Chitti. I am a software test professional. I have many years of experience in the industry and have worked on and managed several software test automation projects. I have implemented many commercial, freeware and open-source tools that aid software testing. I have built several custom tools and interfaces between different types of tools. I have worked for major companies like HP, Manhattan Associates and Tesco PLC that operate in verticals like storage area networking, supply chain management and retail. I have implemented projects in several countries. Besides testing, I enjoy swimming and watching the sea and the sky. I studied engineering and obtained a master's degree.

Database Auditing

by Craig Steven Wright

Databases are the proverbial keys to the kingdom for the majority of organizations. They hold most of the Intellectual Property that has value and are the end goal of many attacks. In the past, the database was commonly protected inside the organizational firewall. Today in a web 2.0 world, the database is there for all to access any time they wish. This includes attackers and others who should not have access.

Database systems are both the most overlooked and the most crucial areas in need of securing. Most of the reason for both security and compliance comes down to information stored on databases, and in many instances all the critical information held by a company will be found on its database. This is not to state that other systems are not important, but that databases, though often overlooked, form the keystone of our information systems. A basic knowledge of SQL is assumed throughout this section. In any event, it is important to have someone involved with a security review who understands and knows how the database system is configured.

At the same time as we are opening database access, we are doing less to protect them than ever. There is light at the end of the tunnel. NIST, DASA and the Centre for Internet Security have detailed guidelines for securing these database systems.

Database Security

Database security is about both the specifics of the database itself and also the system and network it runs on. There are some general areas that you may also want to address when testing a database. These include:

- Policies and procedures
- Patches
- Operating system security
- Setup files
- Service privileges
- Physical security
- Change control
- Disaster recovery
- Separation and restriction of production, test and development environments
- Scripts, jobs or batch files
- The storage of user names and passwords in an unencrypted format
- Application patch level
- User and role rights
- Configuration parameters

It is also necessary to check that processes and control are in place to restrict the use of default and simple passwords.

Principles for Developing a Database Review Strategy

When testing databases in order to ensure that they are secure: test generally, then specifically. In any security reviewing involving a database, start with gathering the system configuration. Some of the key checks include:

- Protect the audit trail – Has the organization protected the audit trail so that audit information cannot be added, changed, or deleted without being recorded and logged?
- Audit normal database activity - The process of gathering historical information about particular database activities that may be reviewed as a baseline. Knowing the baseline provides a starting point to find changes that are out of the ordinary.
- Audit only pertinent actions – In order to avoid cluttering the meaningful information with useless audit information, audit only the targeted database activities. It is all too easy to let the scope of an engagement grow. Due to time and cost limitations, this will only hurt the process in the long run and leave the organization less secure.
- Archive audit records and purge the audit trail – After you have collected the required information, archive audit records that are of interest and purge the audit trail of this information. Maintaining a historical trail is useful and will help future reviews.

Check Triggers

Database triggers are procedural code that is automatically executed in reaction to selected events on a particular table, row or field in a database. A security test of a database should check that these are used and where. Triggers need to be set to fire when events that are defined in policy occur.

System triggers

System triggers allow the activation of controls that start when system events take place. These events can include:

- The start- up and shutdown of the database,
- Logon and logoff from users,
- Privileged access, and
- The creation, altering and dropping of schema objects.

Using autonomous transactions also allows a log to be written for the above system events. Any comprehensive review of a database security should check what (if any) system triggers exist and ensure that these are aligned with the policy of the organization. An example trigger would be sending an alert if a user with administrative access has been added to the database.

Update, delete, and insert triggers

Defense in depth requires an understanding of the users' actions at multiple levels. This is not just access to the database, but access at the detailed row level for selected events and

where there is sensitive data. Database triggers need to be written to capture changes at the column and row level.

Where data is extremely sensitive and any and all changes must be recorded, the database can be configured to write entire rows of data detailing a change to the data (who, what, where and why). This can be done both ahead of and subsequent to the modification of data being made with a write of information to a log table in the database and to an alternate location. This class of logging is extremely resource-intensive. It requires that at least as many extra records are written and stored as the planned change (and at times more).

Fine-grained audit and review

Fine-grained audit is also commonly based on internal triggers that react when selected SQL code is parsed. This approach allows the reviewer to perform access reviews to the row and column level – not only for changes –but as well for read statements.

System logs

Databases generate numerous log files. Many of them providing useful information that can assist in an audit or review of the database. The alert log (for instance) can be used to provide evidence of database start-up and shutdown events. More crucially, it will provide details of structural changes (such as adding a data file to the database or changes to the schema).

Validate database access

Check to find out who has access to the database (and even what tables, rows and fields they have access to). Checking access requires that the audit verify access location and time (where and when). Logon failures should also be checked with seemingly legitimate access at out of the ordinary or anomalous times (such as access to a local payroll system at 3am on a Sunday morning).

Auditing changes to the database structure

Production databases should **NEVER** allow **ANY** user to alter the schema structure. Changes should only be done (such as for upgrades) at definite times (that are logged and approved through change control). All other changes should be regarded as suspicious. Any privileges allowing this must be reviewed carefully. An examination of the database logs for evidence of structural changes can uncover evidence of invalid or unauthorized use of the database.

Monitor any use of system privileges

It is one thing to check the configuration of a database; it is another all together to validate that access has been the same as a configuration file over time, or indeed if the database is reacting as it should. Logging to a separate system is critical for this reason. If the DBA and system administration function lie with the same person, it is possible to remove evidence of changes to the system.

Separate logs provide the capacity to check if either an attacker or a rogue DBA has made any authorized changes to the database.

Log and record data changes to objects

These requirements are very application and installation specific. This is where the security tester needs to know what they are doing and why. This type of review needs to be purposeful and objective. It is easy to exceed the scope of an object access audit, and in this event it is also possible for the testers to breach the law themselves (for instance in gaining an unauthorized view of health information).

Failed log-on attempts

Check for attempts to gain unauthorized access to the database (and ensure the logs are available).

Attempts to access the database with non-existent users

This could be an attempt to bypass the controls in place over the system.

Attempts to access the database at unusual hours

Check for any attempts to access the database outside of working hours in environments where this is feasible. Otherwise, validation of access patterns over time may be completed using a baseline.

Check for users sharing database accounts

Non-repudiation hinges on not sharing accounts and access. Shared accounts are the anathema of a secure system, and there is no compliance regime that allows this practice.

Multiple access attempts for different users from the same terminal

Check if multiple database accounts have been used from the same terminal. This can indicate compromised access or shared access.

Views

A view is a subset of a database that is presented to one or more users. A view is created through the querying of one or more database tables, producing a dynamic result table for the user at the time of the request. As a result, a view is always based on the current data in the base tables. The main advantage of a view is that they may be built to present only certain data. They can be used to restrict the columns and rows that are presented to the user rather than the full table. This prevents the user from viewing other data in the table that may be considered confidential.

Views may be granted to a user without giving the user access to the base tables. Consequently the user cannot directly access the table and find out the other information that they contain. Even in large databases it is essential to take a sample of various views in the database and ensure that the select statements that are used to create the view do not call excessive data. Unfortunately, this is a business-derived process and cannot be simply integrated into tools. The analysis of views is complex, because it is derived from business rules. And as business rules will vary between organizations, and even between departments within an organization, it is not possible for a single tool to automatically check all possible view states.

Remember, a view may be bypassed by some users or if the database is accessed in an unexpected manner. Always consider how appropriate the use of a view is and never trust this as being the sole source of database security.

Integrity Controls

Integrity controls aid by protecting data from unauthorized use and update. CASE tools can be used to take samples of the integrity controls used across a database and ensure that these match the business requirements. Integrity controls can be used to limit the values a field may hold and also the actions that may be performed on the data. They may also trigger the execution of other procedures. For instance, integrity controls may be used to place an entry into a log to record access to tables. In this way user access may be recorded. It is possible to record information across different tables.

One way of monitoring changes to a database even from the administrative staff would be to have tables with restricted access. These tables could be mirrored on another database and accessible only by security and audit staff. An example of this would be to record all changes made by the database administrator to such a table and have them as a record for posterity. One form of integrity control is a domain. A domain is a method of creating a user-defined data type.

When a domain is defined, any field may be assigned to that domain as its data type. An advantage of a domain is that if it ever changes, it can be changed in one place, the domain definition, and all fields within this domain will be changed automatically. Next, a single check clause may be used within a constraint on various fields. If the limits of the check were to change, a DVA would have to find every instance of the integrity control and change it in place separately. A check would enable this to occur, or be logged, or have other controls automatically.

Assertions are constraints that enforce certain database conditions. Assertions are checked automatically by the DBMS when transactions are run that can involve tables or fields where assertions exist. Assertions are often extremely complex and involve detailed investigations against business rules. Unfortunately, it is generally not possible to use tools to check assertions.

Next, database triggers are also effective in adding security controls to a database. A trigger can include an event, condition and action. Triggers may be more complex than an assertion, but will allow the database to automatically prohibit inappropriate actions, automatically start handling procedures using stored procedures or other processes, or write a row to a log file. This may be used to reflect infor-

mation about the user and transaction that has been created. This log may then be displayed in a format that can be read by humans or using automated procedures and tools. Like any stored procedure domains and triggers can be used to enforce controls for all users and all database activities.

These controls do not have to be coded into each query or program. This makes it difficult for individual users or even malicious code to circumvent controls around the database. Even with assertions, triggers and stored procedures on a database, other forms of integrity control are necessary. It is still not possible to stop all malicious or unauthorized access to a database. As such, a change audit process is still necessary. To do this, all user activity should be logged and monitored. The reason for this is to check that all policies and constraints are being enforced across the database.

The difficulty in this method is that every database query and transaction needs to be logged to record the characteristics of all data use. It is essential that all modifications to the database include who accessed the data, the time the data was accessed and, if a program or query was used to run this, what that query or program was. It is also essential to log the network address or location where the request was generated from. There are also other parameters depending on the business and database structure that may be used to aid an investigation of a suspicious data change. The problem with this sort of structure is that it creates extra tables, extra maintenance.

This additional cost often puts people off. However, the savings in the long run and the increased ease with which databases may be verified can make it worthwhile.

Authorization Rules

Authorization rules are controls which are incorporated into the data management systems to restrict access to data and may also restrict the actions taken by the users when they are accessing the data. For instance, an authorization rule could be used to restrict a user with a particular user name and password to read any record in the database but not to modify those records.

In Oracle the privileges are:

- **Select**. This gives the user the capability to query the object.
- **Insert**. This gives the user the capability to insert records into the table or view.
- **Update**. This updates records in the table or view.
- **Delete**. Delete enables the user to delete records from a table or view.
- **Alter**. This allows the user to alter the table.
- **Index**. This allows the user to create indexes on a table.
- **References**. This enables the user to create foreign keys that reference the table.
- **Execute**. The execute privilege allows a user to execute a procedure, package or function.

Data Access Auditing

Data access auditing is a surveillance control. By monitoring access to all sensitive information contained within the database, suspicious activity can be brought to the reviewer's awareness. Databases commonly structure data as tables containing columns (think of a spreadsheet, only more complex). Data access auditing should address six questions:

1. Who accessed the data?
2. When was the data accessed?
3. How was the data accessed? (What computer program or client software was used?)
4. Where was the data accessed from (i.e. the location on the network or Internet)
5. Which SQL query was used to access the data?
6. Was the attempt to access data successful? (And if yes, how much data was retrieved?)

The evidence available to the reviewer is provided:

- Within the client system (this may be infeasible – such as in web-based commerce systems),
- Within the database (including the logs produced by the database that are sent to a remote system), or
- Between the client and the database (such as firewall logs, IDS/IPS devices and host based events and logs).

Auditing within the client entails using the evidence available on the client itself. Client systems can hold a wealth of database access tools and the logs that these create. These logs may contain lists of end-user activity that a user has performed on the database. In respect of web based systems, the web server itself may be treated as a client of sorts.

To obtain an adequate audit trail from client systems alone, all data access must have occurred using client tools under the control of the organization conducting the audit. In the event that data access can transpire using other means, it is rare that sufficient evidence will be available. This option by itself is the entirely worst option available to the reviewer, but it can provide additional evidence in support of the other methods. This is chiefly used in the event of a forensic investigation.

CASE (Computer Aided Software Engineering) Tools

CASE tools can be a great aid to auditing database systems. CASE or Computer-Assisted Software Engineering tools not only help in the development of software and database structures, but can be used to reverse-engineer existing databases and check them against a predefined schema. There are a variety of both open-source and commercial CASE tools. In this chapter we'll be looking at Xcase (http://www.xcase.com/).

Many commercial databases can run into the gigabyte or terabyte in size. Standard command line SQL coding is unlikely to find all of the intricate relationships between these tables, stored procedures and other database functions. A CASE tool on the other hand can reverse-engineer existing databases to produce diagrams that represent the database. These can first of all be compared with existing schema diagrams to ensure that the database matches the architecture that it is originally built from and to be able to quickly zoom in on selected areas.

Visual objects, colors and better diagrams may all be introduced to further enhance the reviewer's capacity to analyze the structure. Reverse-engineering a database will enable the reviewer to find out the various structures that have been created within the database. Some of these include:

- The indexes,
- Fields,
- Relationships,
- Sub-categories,
- Views,
- Connections,
- Primary keys and alternate keys,
- Triggers,
- Constraints,
- Procedures and functions,
- Rules,
- Table space and storage details associated with the database,
- Sequences used, and finally the entities within the database.

Each of the tables will also display detailed information concerning the structure of each of the fields that may be viewed at a single glance. In large databases a graphical view is probably the only method that will adequately determine if relationships between different tables and functions within a database actually meet the requirements. It may be possible in smaller databases to determine the referential integrity constraints between different fields, but in a larger database containing thousands of tables there is no way to do this in a simple manner using manual techniques.

When conducting an audit of a database for compliance purposes, it is not just security functions such as cross-site scripting and sequel injection that need to be considered. Relationships between various entities and the rights and associated privileges that are associated with various tables and roles also need to be considered. The CASE tools allow us to visualize the most important security features associated with a database. These are:

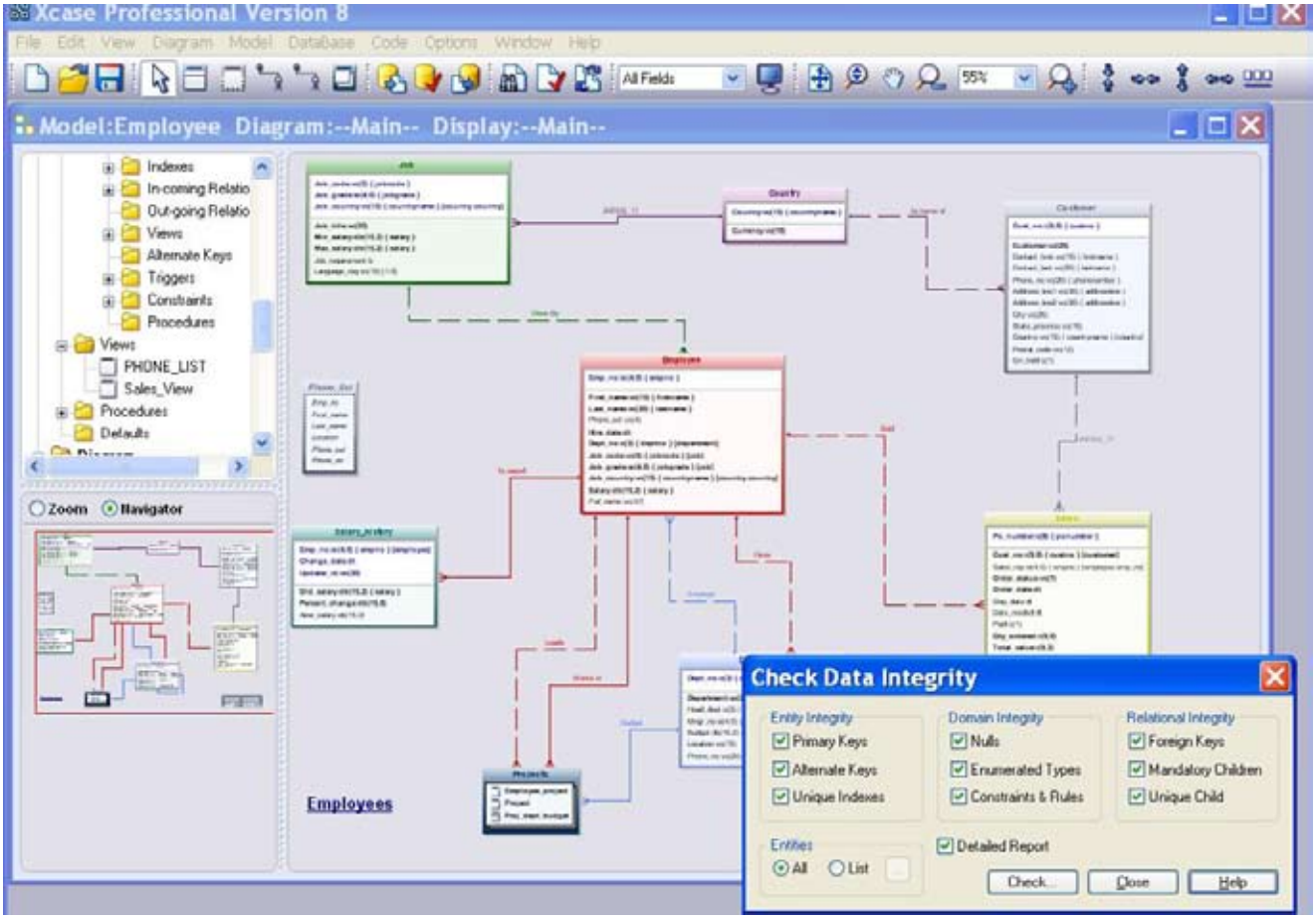


Fig 1 Display database schema.

1. Schemas restrict the views of the database for users,
2. Domains, assertions, checks and other integrity controls defined as database objects, which may be enforced using the DBMS in the process of database queries and updates,
3. Authorization rules. These are rules which identify the users and roles associated with the database and may be used to restrict the actions that a user can take against any of the database features such as tables or individual fields,
4. Authentication schemes. These are schemes which can be used to identify users attempting to gain access to the database or individual features within the database.
5. User-defined procedures which may define constraints or limitations on the use of the database,
6. Encryption processes. Many compliance regimes call for the encryption of selected data on the database. Most modern databases include encryption processes that can be used to ensure that the data is protected.
7. Other features such as backup, check point capabilities and journaling help to ensure recovery processes for the database. These controls aid in database availability and integrity, two of the three legs of security.

CASE tools also contain other functions that are useful when auditing a database. One function that is extremely useful is model comparison.

Case tools allow the reviewer to:

- Present clear data models at various levels of detail using visual objects, colors and embedded diagrams to organize database schemas,
- Synchronize models with the database,
- Compare a baseline model to the actual database (or to another model),

Case tools can generate code automatically and also store this for review and baselining. This includes:

- DDL Code to build and change the database structure
- Triggers and Stored Procedures to safeguard data integrity
- Views and Queries to extract data

The reviewer can also document the database design using multiple reporting options. This allows for the printing of diagrams and reports and the addition of comments to the reports and user-defined attributes to the model.

Data management features allow the reviewer to validate the data in the database being reviewed against the business rules and constraints defined in the model and generate detailed integrity reports. This can be extended further to access and edit the data relationally using automatic parent/child browsers and lookups and then to locate faulty data subsets using automatically generated SQL statements. These provide valuable sources of errors and help in database maintenance – making the audit all the more valuable.

Model comparison involves comparing the model of the database with the actual database on the system. This can be used to ensure change control or to ensure that no unauthorized changes have been made for other purposes. To do this, a baseline of the database structure will be taken at some point in time. At a later time the database could be reverse-engineered to create another model, and these two models could be compared. Any differences, variations or discrepancies between these would represent a change. Any changes should be authorized changes and if not, should be investigated. Many of the tools also have functions that provide detailed reports of all discrepancies.

Many modern databases run into the terabytes and contain tens of thousands of tables. A baseline and automated report of any differences, variations or discrepancies makes the job of auditing change on these databases much simpler. Triggers and stored procedures can be stored within the CASE tool itself. These can be used to safeguard data integrity. Selected areas within the database can be set up such as honeytoken styled fields or views that can be checked against a hash at different times to ensure that no-one has altered any of these areas of the database. Further, in database tables, it should not change. Tables of hashes may be maintained and validated using the offline model that has stored these hash functions already. Any variation would be reported in the

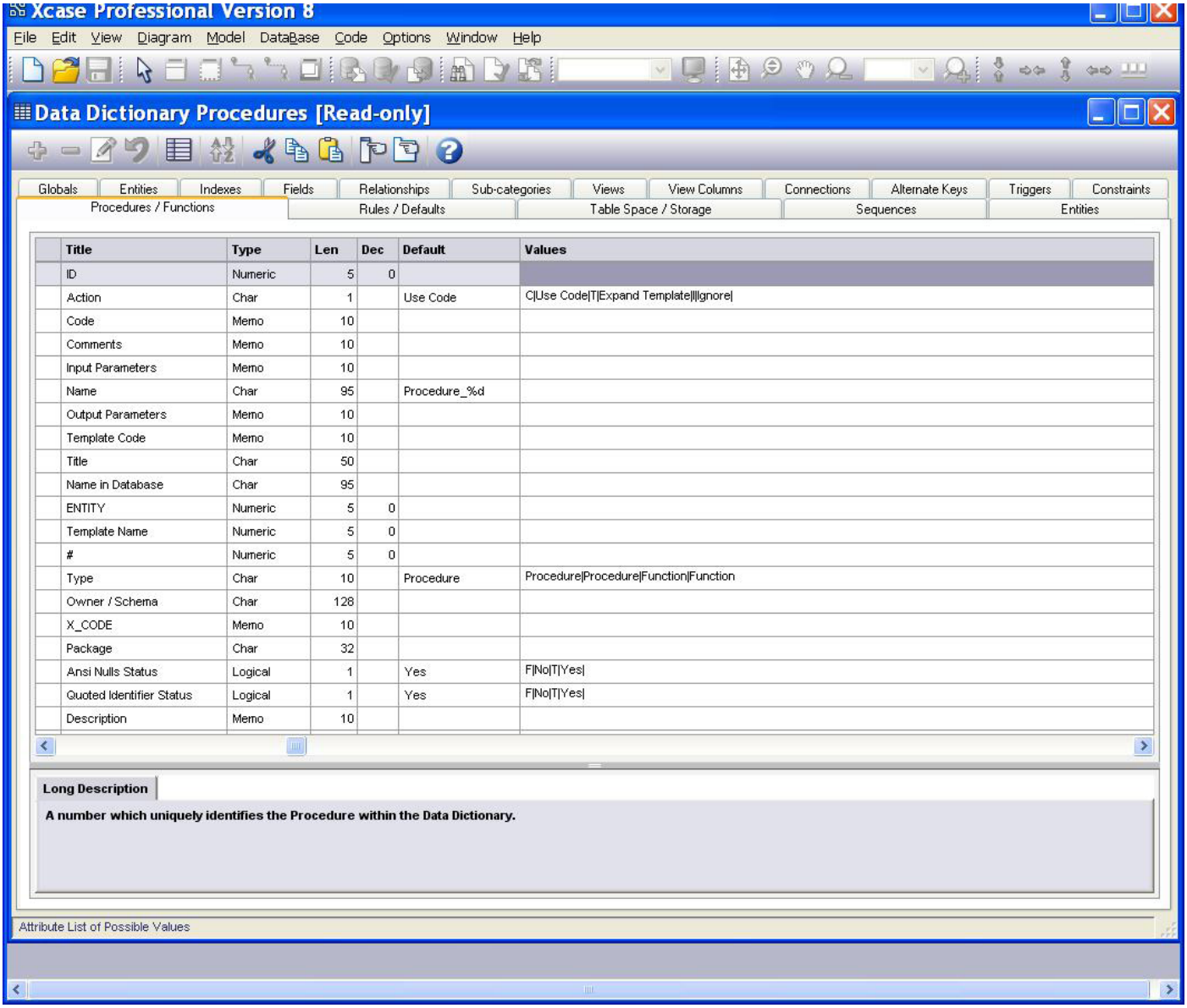


Fig 2 Reverse Engineer existing databases into presentation quality diagrams in minutes.

discrepancy report.

Next, the capability to create a complex ERD or Entity Relationship Diagram in itself adds value to the audit. Many organizations do not have a detailed structure of the database, and these are grown organically over time with many of the original designers having left the organization. In this event, it is not uncommon for the organization to have no idea about the various tables that they have on their own database.

Another benefit of CASE tools is their ability to migrate data. CASE tools have the ability to create detailed SQL statements and to replicate through reverse-engineering the data structures. They can then migrate these data structures to a separate database. This is useful as the data can be copied to another system. That system may be used to interrogate tables without fear of damaging the data. In particular, the data that has migrated to the tables does not need to be the actual data, meaning that the reviewer does not have access to sensitive information but will know the defenses and protections associated with the database. This is useful as the reviewer can then perform complex interrogations of the database that may result in damage to the database if it was running on the large system. This provides a capability for the reviewer to validate the data in the database against the business rules and constraints that have been defined by the models and generate detailed integrity reports. This capability gives an organization advanced tools that will help them locate faulty data subsets through the use of automatically generated SQL statements.

Vulnerability Assessment Tools

Any database sits on top of another operating system. As such tools such as NMAP may be used to check for open ports on the database system and determine if there are other services running on the host. This is important as standards (such as PCI-DSS) call for the restriction of other services to the host allowing only those that are necessary. This means that the database has to be a bastion.

That is the system needs to be built for purpose and should not be shared with other applications. Next vulnerability and assessment tools ranging from Nessus through to commercial assessment tools such as CORE IMPACT may be used to check the database for a variety of vulnerabilities. Nessus for instance has a variety of plug-ins associated with Oracle, Microsoft SQL and My SQL databases. These plug-ins allow Nessus to check for vulnerabilities associated with these particular database systems as well as also checking for application vulnerabilities and operating system vulnerabilities that may be associated with the system and may affect the database. Further, many of the database vendors also provide free tools. Microsoft SQL server comes with the SQL server analyzer. This product looks at the best practices for the SQL database and can analyze against these best practice statements.

Local Security

The security of the database overall is only ever as good as the security of the system it resides on. Anyone with physical access to the host or administrative access to a system can compromise a database. At the least copying of the data is possible – stories of people who have purchased hard drives from organizations that have not wiped the data on the drives and that have sold them via eBay only to have recovered data are a near daily occurrence.

No database can be considered compliant with

any standard if the system it is running on is also not adequately secured. As such, always ensure that the host the database is running on is secured. Disable any unnecessary services and patch the system on a regular basis. By regular – this also means often.

Creating a Checklist

The most important tool that you can have is an up-to-date checklist for your system. This checklist will help define your scope and the processes that you intend to check and validate. The first step in this process involves identifying a good source of information that can be aligned to your organization's needs. The integration of security checklists and organizational policies with a process of internal accreditation will lead to good security practices and hence effective corporate governance.

The first stage is to identify the objectives associated with the systems that you seek to audit. Once you're done, this list of regulations and standards that the organization needs to adhere to may be collated. The secret is not to audit against each standard, but rather to create a series of controls that ensure you have a secure system. By creating a secure system you can virtually guarantee that you will comply with any regulatory framework.

The following sites offer a number of free checklists that are indispensable in the creation of your SQL database audit framework.

CIS (The Center for Internet Security)

CIS provides a large number of Benchmarks for both the Operating Systems and also applications. CIS offers both Benchmarks and also a number of tools that may be used to validate a system. The site is: http://www.cisecurity.org. CIS currently has configuration benchmarks for the following database applications:

- Oracle Database 8i
- Oracle Database 9i/10g
- MySQL
- Microsoft SQL Server 2005
- Microsoft SQL Server 2000

SANS

The SANS Institute has a wealth of information available that will aid in the creation of a checklist as well as many documents that detail how to run the various tools.

The SANS reading room (http://www.sans.org/reading_room/) has a number of papers that have been made freely available:

- GSNA Audit Gold Papers
- GSOC Oracle Gold Papers
- General Tools papers (http://www.sans.org/reading_room/whitepapers/tools/)

SANS Score (Security Consensus Operational Readiness Evaluation) is directly associated with CIS.

NSA, NIST and DISA

The US Government (through the NSA, DISA and NIST) have a large number of security configuration guidance papers and benchmarks.

NIST runs the US "National Vulnerability Database" with the Microsoft SQL Security Checklist from DISA (http://iase.disa.mil/stigs/checklist) and a generic database checklist.

Summary

Database security consists of a number of key categories, all of which need to be tested. These include:

- Server Security (the process of limiting the access to the database server).
- Database Connections (such as local access and remote network connectivity to the database using authentication and authorization).
- Table Access Control (Table access control is related to an access control list restricting access to the database tables).
- Restricting Database Access (Firewalls and network segmentation).

Ensuring that your database is secure is a key aspect of organizational security.

Biography

Over the years Craig has personally conducted and managed in excess of 1,600 IT security- related engagements for more than 180 Australian and international organizations in both the private and government sectors.

As a strong believer in life-long learning, Craig has qualifications in Law, IT, Mathematics and Business. However, his driving focus is research and development in the security and risk arena. He is the first person to have obtained multiple GSE certifications (Malware and Compliance) and, presently, the only one in the Southern Hemisphere to hold a GSE Certification. He is continuing his quest and will sit the third and final GSE Certification this year (2009) at the SANS conference in September.

Craig designed the architecture for the world's first online casino (Lasseter's Online) in the Northern Territory; as well he has, in the past, designed and managed the implementation of many of the systems that protect the Australian Stock Exchange.

To add to these accomplishments, he has authored IT security-related books and articles as well as designed a new university program for Charles Sturt University in New South Wales, Australia which will offer a Master in Digital Forensics. This program will commence in 2010 and be offered as an on-campus and distance education program.

EuroSPI²2009

European Systems & Software Process Improvement and Innovation

16th EuroSPI Conference

Systems and SW Process Improvement Practice and Case Studies - Lessons to be Learned

2nd - 4th September 2009

University of Alcalá, Alcalá de Henares, Madrid, Spain

http://2009.eurospi.net

Programme 2nd - 4th September 2009

Tutorial Day

4 Stream Conference Day

Mixture of Industry and Research Session

SPI Manifesto Event

EU Certificates Day

Exhibition

Experts from Industry and Research from 24 Countries

Includes also Software Testing Key Notes, Tutorials and Presentations

EuroSPI²2009 Key Notes

Fighting a Chameleon - Engaging Diverse Challenges on the Road to High Maturity
Jan Pries Heje, Roskilde University, Denmark

Software Disasters - Understanding the Past, to Improve the Future
Patricia McQuaid, California Polytechnic State University, USA

New Developments in ISO 15504
Alec Dorling, International SPICE Project, UK

Project-Based Test Automation

by David Harrison

This article sets out a project-based experience of Automated Testing in the context of software developed with the Java-Swing programming framework. In terms of business domain, the software represented an advanced costing tool for use by underwriters in the re-insurance business. However, we focus here on the *pattern of solution* more than the domain-specific re-insurance aspects.

This article is based on material from the forthcoming book "*Automated Functional Testing for Java-Swing*", to be published by the author in April, 2009.

Structure of Project QA Activities

The structure of the Quality Assurance (QA) activities in this project-based development setting, can be summarized as shown in Figure 1, below:

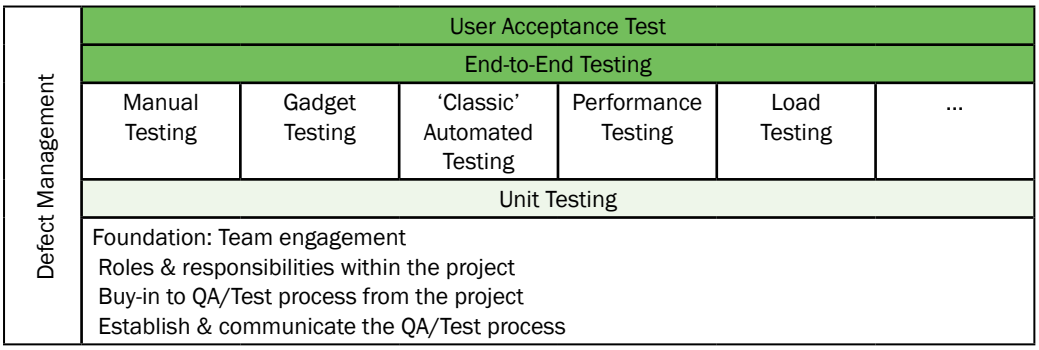


Figure 1 – QA/Test Pattern

In this pattern you can see that the 'classic' automated testing, of the type that we will address here, is only *one part* of the overall Quality Assurance effort. The target of our QA work on this project was, from the outset, to develop a strategy which could be termed Agile, in that it fitted in with the iterative build cycle on the project, as well the highly seasonal nature of the development/deployment process itself.

Let's take a look at each of the parts of this pattern in a little more detail:

Foundation

This part is really crucial to the success of a project-based QA/Test effort. Its role in the pattern is to set out the fundamental project issues which must be achieved in order to maximise the success of the overall QA effort. This naturally affects also the possible successful outcome of any automated testing effort. For an automated testing effort to be given at least some chance of success, the QA/Test team needs strong support from both the IT as well as the business parts of a project. The collaboration from the IT group is particularly important for the pattern presented here, as we need IT involvement in specializing the application slightly to enable access to the internals of the application.

In order for good collaboration to happen, such things as team engagement and having a clearly established set of roles & responsibilities, is *essential*. Having a clear technical process set out is one thing, but this must be communicated to *all project members* by the QA workstream.

Unit Testing

A key part of the process, underpinning all the efforts that follow, is that of Unit Testing. Typically, the developers will use a standard unit testing framework, which for Java would most likely be JUnit[1].

Defect Management

This part of the QA/Test process reflects that part which turns testing into a genuine Quality Assurance process. As hinted in the Foundation part of the pattern, it is important for a development project to recognize the importance of defects, not just as negative things to be got rid of, but as *extremely positive*, naturally occurring items of project life. Indeed, defects are to be aggressively sought within a project if the *deeply negative* characteristics of them are to be avoided that of their impact on the user community if they are deployed to the productive environment.

Manual Testing

Yes – Manual Testing. This is always a theme in the project-based QA/Test story, particularly for projects that are at the very beginning of their development. In these cases, it is quite usual that a large part of the functionality of the software will have to be manually tested. This arises from the unfortunate fact of project life that software in its initial development phase is usually highly volatile in terms of its user interface design, thus precluding automated testing.

Gadget Testing

This part of the testing process is a catch-all for the testing which uses specialist executables that fulfill specialist testing tasks. This form of testing would be used if the software being

1 http://www.junit.org

developed has a mathematical result, for example, which needs to be asserted. A specialist tool which, using known input data, causes the calculation to be performed and then compares the result to a known expected result, would be a good example of a "Gadget".

Classic Automated Testing

This form of testing is the subject of this article. Suffice it to say here that this testing is characterized by exercising the built software through its User Interface (UI). We add here that, from the outset, the task of test automation was seen as essentially a coding task. With this mindset, we looked for "efficiency" in just the same way as any software development workstream would. Often test automation is viewed in quite the opposite way, the coding aspects being set to the background or hidden behind simplistic definitional "spreadsheets" and the like. Taking on board the mental model of "coding" releases us from seeking the false dawn associated with alternative approaches and endlessly struggling with the fundamentals in order to achieve some sort of successful, repeatable outcome.

Performance Testing

In this form of testing we seek to establish that the software meets the performance metrics defined for it, usually per business operation or functionality. There are a number of commercial and open-source tools to help in this area.

In our approach we have used our approach to perform simple performance testing, establishing performance metrics for "Search" from distant global locations.

Load Testing

This form of testing is concerned with ensuring that the software performs as expected under concurrent user load. Often this type of testing is mandated when the software architecture is Client-Server. The Server-side will need to be highly resilient and fail-safe, as the number of "logged-in" users grows and performs specified key business tasks. As in the case of Performance Testing, there are a number of tools on the market for performing this type of test.

Where's the "Agile" Part?

With project-based testing, especially where development is performed in a relatively short time frame, its important that the QA/Test approach and any Automated Testing approach takes account of this project feature. For test automators its important to have a strong technical basis, on which to incrementally extend the range of functionality that is tested; and for this to happen, we need to look for agility in the same way that a software development team might. The interplay between the QA/Test and development workstreams must be based upon iterative builds and a constant flow of new and expected fixed defects at the build points, which in turn underlines the vital role Defect Management has in achieving an Agile process.

This rapid cycling of defects to-and-fro between QA and development is the hinge of our Agile approach, leading to strong defect reduction over the full project cycle. Alternative strategies, such as V-Model/Waterfall cannot render this kind of positive dynamic.

Which Tool & Why

As the product under development was Java-Swing based, the decision was taken to adopt QF-Test[2]. There are a range of tools that could have been chosen, so why this one?

This tool has the following *striking* characteristics which make it stand out from the crowd:

- strong and intelligent connection with the Java VM, object creation and destruction are events that get announced over this connection
- has first-class Exceptions reflecting things happening in the test project itself as well as what is happening in the Software Under Test (SUT)
- has a powerful, extensible Name Resolving architecture, which allows a wide range of name replacements to be performed for cases that are very common, but quite challenging, in real-life software
- has the Jython language as an extension to its built-in programming paradigm (e.g. can instantiate objects from SUT), which enables some very elegant tactics to be employed to get at objects of the UI and their properties
- embraces the use library structure in test projects, which allows a structure to be introduced as between project-specific and generic concerns
- allows the development of data-driven tests using its built-in programming paradigm, which allows a very wide set of data to be used in tests. This represents a significant feature of "scaling-up" in automated testing.

Taking Exception

One of the powerful characteristics of QF-Test is its use of *first-class exceptions* which relate directly to what's happening in the SUT as well as in the test project itself. Being able to code/script tests using real exceptions makes the eventual design very well patterned as well as effective.

What's in a Name?

One of the major difficulties encountered when attempting to automate testing a real-world project context, is the ability to reliably assign names to objects that appear in the UI of the Software Under Test (SUT). QF-Test contains a *very effective and extensible* Name Resolving architecture. A range of visual elements in the SUT of the project in question demanded that the automation tool had a strong capability in dealing with object naming.

2 www.qfs.de

Talking to Objects

The project for which the automated testing solution was developed, involved an advanced graphical UI, in which the "real estate" of visual objects extended beyond the physical viewport of the screen. The extent of this real size was governed by a range of characteristics within the model being displayed. The objects viewed by the user have associated editing dialogs, which naturally are things we as test automators need to get at to perform assertive testing. The resolution of this problem critically rested on the provision of a special object within the main application frame, which contained methods which could be invoked by Jython to open the editor panel for an object specified by its "path" within the overall visual structure, and thus allowed us to gain access to these otherwise off-screen visual objects. The overall visual structure is also available by means of these built-in object methods.

Generic Controls

In order to meet our target of Agile test automation, an early sub-project in the QA workstream, was the development of a Generic Library, which would enable the software-specific workflows to have tests "coded" as fast as possible. We could benefit (as in normal software development practice) from "code" re-use and general applicability, depending only upon the values of parameters. The importance of such a Generic Library cannot be overestimated in our achievement of an Agile automation process.

Where are we?

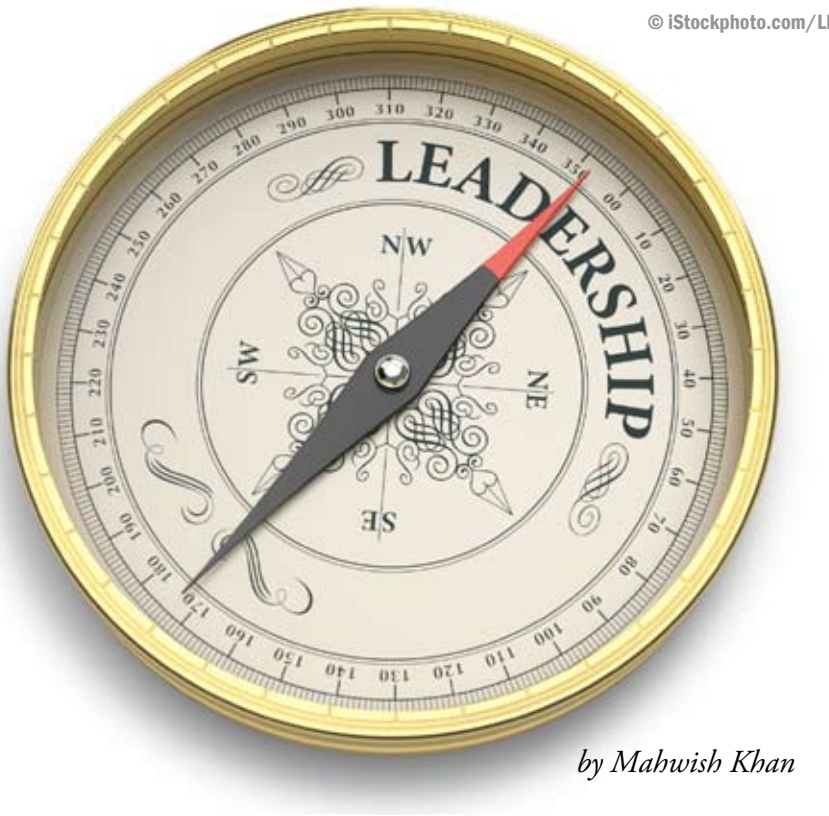
The pattern of solution for test automation offered here, embracing both the detail of how we perform this often very challenging task, as well as its relationship and fit with other parts of the overall project landscape, has now been in place for a number of years. The pattern takes full account of the fundamental challenges of test automation - which usually render the outcome in alternative approaches to rather less like testing and more like driving the software as a benign and careful "user" - is today fully part of the target project.

The software has undergone, from inception, 4 years of successful global deployment. Test automation was begun with the development of the Generic Library capability as a key precursor to the main task, in version 2 of the software.

The automation project itself is in the process of being handed over to the central support groups for them to use as part of their ensuring that maintenance changes will not adversely affect the correct operation of a crucial business tool which plays a central role in the Underwriters day-to-day activities.

Biography

David Harrison works as an independent software QA/Test Manager – currently at SwissRe, Zurich, Switzerland within the tools development group. This group has the mandate to develop and deploy reinsurance costing tools to the actuary and underwriter desktop. He can be contacted at dharrison_ch(a-t)yahoo(dot)co(dot)uk.
This article is based on material from his forthcoming book: "*Automated Software Testing for Java-Swing; A Pattern of Solution*", to be published in June, 2009.



Software Configuration Management-SCM

by Mahwish Khan

The first Law of Software Engineering is:

"No matter where you are in the system life cycle, the system will change, and the desire to change it will persist throughout the life cycle."

(Bersoff, et al, 1980)

Change is inevitable and software is affected by change from when it starts to be built. Even though everyone knows about the high rate of change, change initiatives fail at an alarming rate. This is because most initiatives fail to consider how the changes will affect the system. As change increases, so does the level of confusion among software engineers working on a project, and confusion arises if changes are not analyzed before they are made, recorded before they are implemented, reported to those with a need to know, or controlled in a manner that will improve quality and reduce error.

"There is nothing permanent except change". *(Heraclitus)*

Role of Change:

All projects have a main objective of wanting to change something, from a manual to an automated system. In fact, systems are upgraded or replaced, presumably to provide better or greater functionality, ease of use, reduction of operating costs, etc. As a project is executed, changes to the initial project plan and products are a natural occurrence. There are some sources of change:

- **Requirements:** The longer the delivery cycle, the more likely it is to happen.
- Changes in funding
- Technology advancement
- Solutions to problems
- Scheduling constraints
- Customer expectations
- Unanticipated or unexpected opportunities for an improved system

Some of these changes may appear as options, while others may be mandated from above or by circumstances, as in loss of funding. In addition, many development efforts involve a progressive evolution or elaboration of capabilities and requirements.

Now the point arises why SCM is important. Why we are giving so much attention to change management? So the answer is: where there is a change, there is a configuration and configuration needs to be managed, because if we don't control the change, it will control us and that's never good. It's very easy for a stream of uncontrolled changes to turn a well-run software project into chaos. That's why SCM is an essential part of good project management and solid software engineering practice.

Software Configuration Management is an umbrella activity that is applied throughout the software process. Because change can occur in any phase of the software development life cycle. Before we proceed, it is important to have a clear understanding of SCM and its activities, which are developed to:

1. Identify the need of change
2. Control change
3. Ensure that the change is being properly implemented, and
4. Report changes to others who may be involved.

Purpose of **S**oftware **C**onfiguration **M**anagement:

The goals of SCM are in general:

- **Configuration Identification:** What code are we working with?
- **Configuration Control:** Controlling the release of a product and its changes.
- **Status Accounting:** Recording and reporting the status of components.
- **Review:** Ensuring completeness and consistency among components.
- **Build Management:** Managing the process and tools used for build.
- **Process Management:** Ensuring adherence to the organization's development process.
- **Environment Management:** Managing the software and hardware that host our system.
- **Team Work:** Facilitate team interactions related to the process.
- **Defect Tracking:** Making sure every defect has traceability back to the source.

SCM Functional Areas:

The discipline of software configuration management defined for software projects is to ensure that a sound SCM process is implemented. SCM is comprised of four major activities: SC Identification, SC Control, SC Status Accounting and SC Auditing. All SCM activities fall within bounds of these functions.

These terms and definitions change from standard to standard, but are essentially the same. Let's look at the detail of each of the major functions.

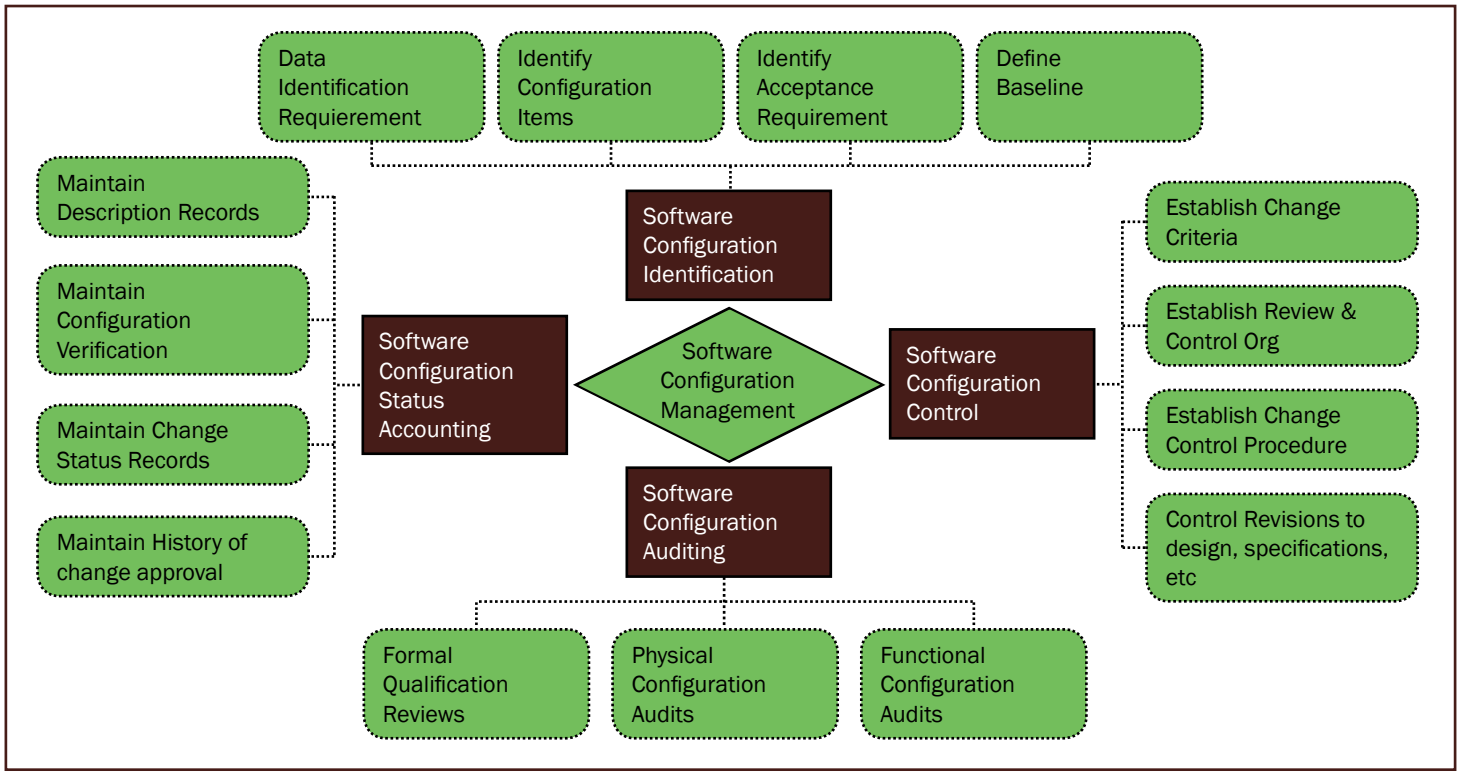


Fig: 1.1 Major Functions of SCM

Software Configuration Identification:

The process of establishing a baseline from which changes are made. Software Configuration Identification involves:

- Identifying the structure of the software
- Uniquely identifying individual components
- Making them accessible in some form

The goals for SCI are:

- To create the ability to identify the system components throughout the SDLC
- To provide traceability between the software & related SCIs.

The SC Identification Activity includes:

- Selecting items to be placed under SCM control.
- Developing the software hierarchy.
- Creating an identification scheme that shows the software hierarchy.
- Uniquely identifying the various revisions of the software product.
- Defining relationships and interfaces between the various software products.

The **software configuration item** can be defined as a work product (hardware and/or software) or information that has an end-user purpose. These attributes are recorded in the configuration documentation and baseline. The **Baseline** is prepared by combining one or more software configuration items that have been formally reviewed and agreed upon and serve as a basis for further development. Baselining an attribute forces formal configuration change control processes to be effected in the event that these attributes are changed. When an item is baselined, it becomes frozen. The term frozen is to be understood in the context that the respective item can only be changed by creating a new version. **Versions** ensure repeatability and the ability to produce any version of the software at any given time. Version Control and Baselines are primarily methods of Software Configuration Item.

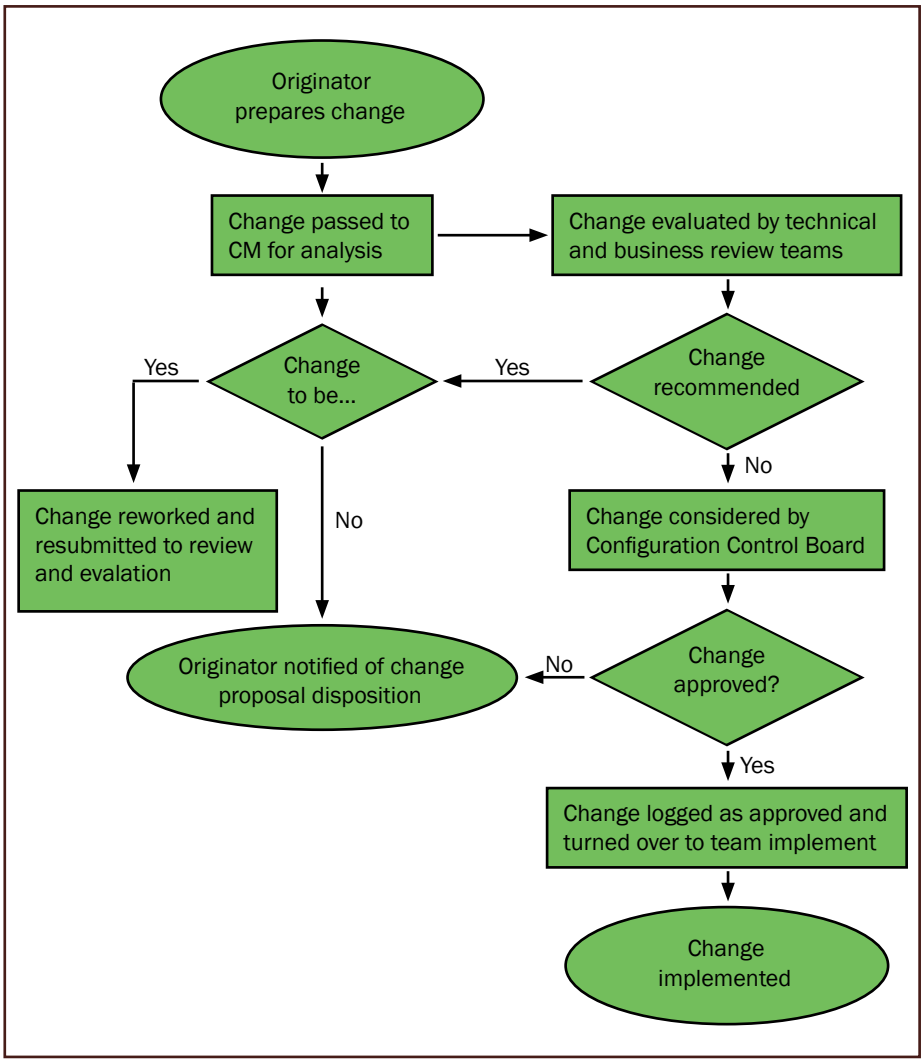


Fig: 1.2: Change control procedure

Software Configuration Change Control refers to the methods to be implemented for the management and technical control. It is a set of processes and approval stages required to change a configuration item's attributes and to re-baseline them.

It is helpful to have a group to make change decisions, the Change Control Board. It is not necessary for all projects, but certainly for larger projects, to have a formal Change Control Board whose responsibility it is to review and approve/disapprove changes. It is the CCB's responsibility to provide the mechanism to maintain orderly change processes.

Software Configuration Status Accounting refers to recording and reporting on the configuration baselines associated with each configuration item to all parties concerned: management, client, technical departments - information and their status. Reports include: transaction log, change log and item delta report, and the reports can be common as of resource usage, change in process, file revision history, etc. Basically, the main issue is acquisition and maintenance of all information concerning a project's status and that of its parts.

Software Configuration Auditing is a process to verify that the baseline is complete and accurate, that changes made are recorded, that recorded changes are made, and that the configuration items used in a version or build are documented in the configuration correctly. Effective CM requires regular evaluation of the configuration. This is done through the auditing function. Functional and physical are the types of audits which are performed. In order to achieve functional and performance attributes of a configuration item, a functional configuration audit is done, while a physical configuration audit ensures that a configuration item is installed in accordance with the requirements of its detailed design documentation.

Summary:

Software configuration management is an umbrella activity that is applied throughout the software process. SCM identifies controls, audits, and reports modifications that regularly occur while software is developed and after it has been released to a customer. All information produced as part of software engineering becomes part of the software configuration. The configuration is organized in a manner that enables orderly control of change.

References:

I. Software Configuration Management from Software Quality Assurance: Principles and Practices by Nina S. Godbole

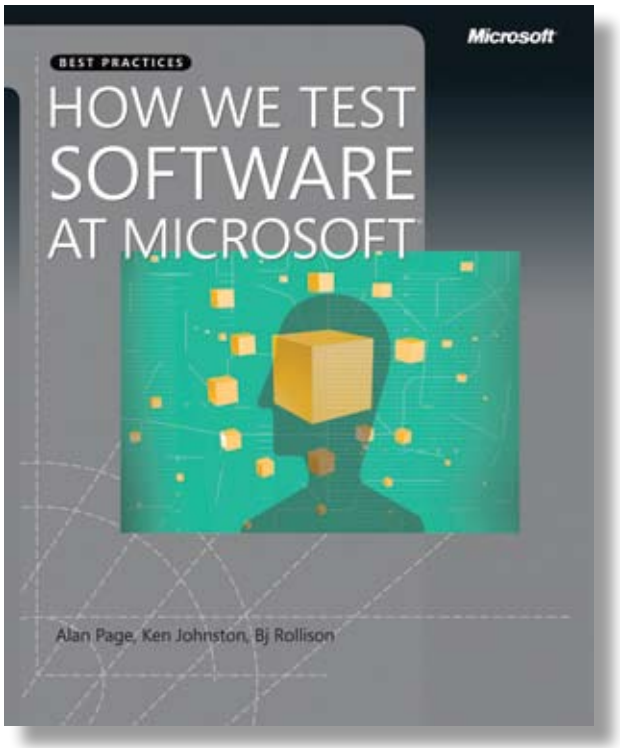
II. Software Configuration Management from Software Engineering, A practitioner Approach by: Roger S. Pressman.

Biography

I, Mahwish Khan, have worked in Quality Assurance and the testing of web applications and software systems for more than two years. Being a QA expert, I have been involved in testing and change management activities. In my recent position, I am responsible for testing of web applications.

INTRODUCING

- Discover how Microsoft implements and manages the software-testing process company-wide
- Gain expert insights on effective testing techniques and methodologies
- Shares such facts as the number of test machines at Microsoft, how the company uses automated test cases, and bug statistics
- Answers key testing questions, such as who tests what, when, and with what tools
- Describes how test teams are organized, when and how testing gets automated, testing tools, and feedback



HOW WE TEST SOFTWARE AT MICROSOFT
ISBN: 9780735624252

ABOUT THE AUTHORS

Alan Page has been a software tester for more than 14 years, including more than 12 years at Microsoft Corporation.
Ken Johnston is group manager for testing in the Microsoft Office business group.
Bj Rollison is a Test Architect in the test excellence organization at Microsoft.
Authors' website: http://www.hwtsam.com

Microsoft® Press

Buy the book! **http://www.microsoft.com/learning/en/us/books/11240.aspx**

Align for Good Test Design

Implementing Test Design Techniques Together

by Richard van der Pols, Andrew Jong, & Jeanne Hofmans

The implementation of test design techniques in an organization usually has a lot more impact than initially expected. Most test professionals that come in touch with test design techniques start off with a lot of enthusiasm and are eager to implement them in the projects they are involved in …. only to find that there is more to the implementation of test design techniques than simply letting the testers apply the technique.

With this article the authors, all of them test consultants at Improve Quality Services in The Netherlands, will provide an overview of what impact the implementation of test design techniques can have on the various stakeholders in an organization (and thus in a project). The overview provided is based on practical experiences in multiple domains (e.g. finance, medical and transport) with stakeholders like project management, resource management, test management, testers and customers/users. For each of these stakeholders, there are benefits coming from the implementation of the test design techniques, such as more predictable quality, clearer risk mitigation, clearer priorities and better control (see [3] for a list of benefits). However, there are not only benefits: various types of cost can also be expected. Naturally, costs means money involved, but also means resources and skills, workshops and training, quality of the test basis, necessary process improvements and creating test awareness.

This article is applicable to implementing all types of test design techniques: specification-based (black box, also known as behavioral techniques), structure-based (white box or structural techniques) and experience-based. Specification-based techniques include both functional and non-functional techniques (i.e. quality characteristics). More on test design techniques can be found in [2], [3] and [4].

The Initiative

The initiative to implement test design techniques often starts with the professional tester who wants to bring into practice what he or she has learned in a testing course. Attending this course can be the result of career-planning (possibly from within his organization), or based on a management decision, trying to improve the quality and/or effectiveness of testing based on success stories from other projects or companies. Maybe even the testing process itself has to be improved because there have been complaints from the customer about the effectiveness of testing, e.g. too many major defects in production or during acceptance testing. Keep in mind that the purpose of testing is to contribute to the quality of the product. Test design techniques are one of the many means by which this fundamental objective can be achieved.

We even have seen cases where the customer takes the initiative and requires that the supplier can show that test design techniques have been applied. Though this is mostly seen in safety-critical products, this might become common practice in the future, as customers too are getting more and more aware of the fact that the quality of their products and/or services is also determined by the quality of testing.

This paper starts with the testers' viewpoint. The test manager sends some of his testers on a testing course. He believes that using test design techniques will not only benefit the overall product quality, but also help to better control the test project.

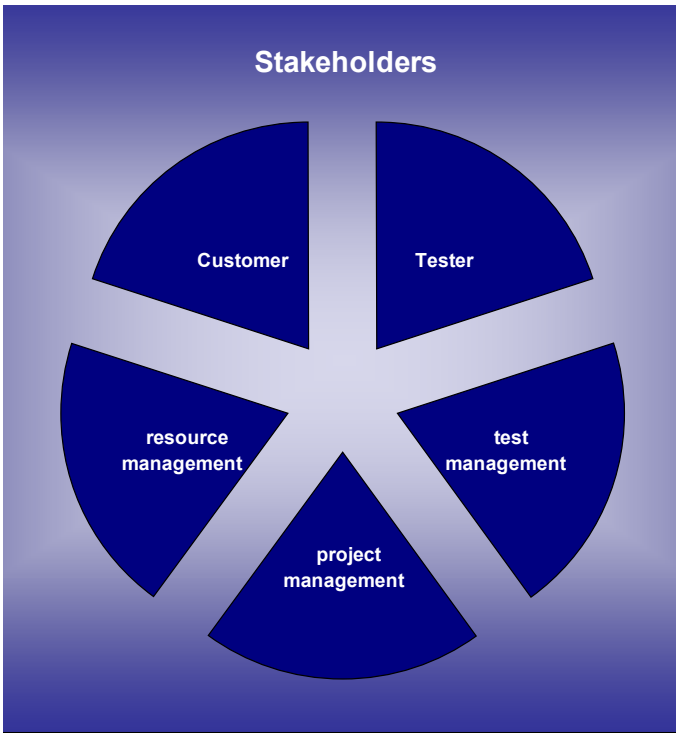


Figure 1: Stakeholders for good test design

Tester

When testers come back from a course about test design techniques, they are usually very enthusiastic to put in place what they have learned. When promoting the test design techniques, however, they usually encounter resistance from an area they would least expect it: the work floor. Possible reasons for such resistance are:

- Other testers on the team haven't yet been introduced to test design techniques and fear that they will miss tests if they only have to apply the techniques.
- Testers on the team think test design techniques are too cumbersome and time consuming and that just creating test cases on the fly is much faster, and thus more test cases can be made in the same amount of time.
- It's not the right time for change. Other tasks have a higher priority. Testers on the team think that time spent on designing test cases could better be used for other beneficial test tasks, such as test data preservation, test environment maintenance, implementing test tools or an improved simulation environment.

To overcome this resistance and make a successful start with the implementation of test design techniques, the testers that went to the test course have to lead by example. They should start by making test designs for the test items (e.g. requirements or use cases) assigned to them and sharing this experience with other team members; even talking with them at the coffee-machine. Don't get us wrong: we're not solely proposing a bottom-up approach. The view of the authors is that every party involved in testing has to contribute if they want to use test design techniques successfully. But if the initiative merely lies with the testers, this is an approach that we have quite often found to be very effective.

For their part, the testers could begin with analyzing a small set of requirements (also in relation to each other) and determine:

- Which requirements are more important?
- Which requirements will be more error-prone?
- Which requirement has a high impact for the customer in case of software failure?
- What type of requirement is it?

It is worth mentioning that there are tools and techniques that support the categorization of requirements in term of risks (e.g. PRISMA – The PRoduct Risk MAnagement - method, see [1]). The risk levels will influence the test approach and thus the choice of test technique to be used. The decision table test design technique is very thorough, but takes more time during the design phase. For requirements with low risk, the costs of applying such a technique are often too high.

For the selected requirements, it is important to know that different test design techniques put different requirements on the test basis. The opposite is also true: the way the requirements are written also partly determines the test design techniques to be used. For example: if the requirement uses a UML state transition diagram, this would beg for a State Transition Test (STT), or if it uses a use case diagram, then a Use Case Test (UCT) will be most appropriate. If only written organizational process descriptions are used, a Process Cycle Test (PCT) may work (e.g. at acceptance test level). For other techniques, there are also some rules which apply, e.g. for boundary testing the requirements must clearly state the boundaries. And vice versa, if a requirement is considered to be extremely important, it may be beneficial to supplement it with a state transition diagram so that the state transition test can be used.

As requirements put constraints on the test design techniques that can be used and vice versa, we suggest involving designers and programmers in the test design process to enhance the use of test design techniques. Have a meeting in which you explain the test design technique to designers and programmers; let them know what you are going to do. Designers may start to think differently about making designs. Developers and architects will learn about test design techniques, which can assist them in making more effective unit and integration tests. Just be careful not to use the same techniques in every test level. One of the strengths of test design techniques lies in differentiation: Use different test design techniques at different levels.

Once a start has been made, the one thing left to do is convince others that, from a tester's point of view, it has more advantages to use techniques than to not use them. This is a challenging task to take on alone; how this works in a team will be explained in more detail in the paragraphs describing the points of view of test managers and project managers. As any new way of working will encounter some form of resistance, this needs to be mitigated by management. Data can be collected to convince management. Some people will only be convinced by showing the quantitative data and results.

The testers should aim to prove that defects are found earlier in the development life cycle, because the requirements are more intensely studied (analogous to the review process). Another benefit is that the average quality of the produced test cases increases. Test design techniques especially help the (junior) tester who has less experience in the domain at hand.

Test Management

In the chapter "The Initiative" we mentioned a test manager who sent some of his testers on a testing course. He notices that they are enthusiastically starting to use the test design techniques. In spite of some initial resentment the other testers seem interested as well. The test manager decides that this is the time to organize a practical workshop with all testers, in which they will learn how to use test design techniques within their company. In this workshop the requirements are discussed in terms of which test design techniques can be applied (why and how) as well as the test design specifications that are already being used.

The impact of such a workshop is quite substantial. First of all, the test design techniques are actively being used by all participants. To make sure there is a long-term effect, management commitment is essential. Thus the test manager must actively promote the use of the test design techniques and seek the support of project management and resource management. The first step is to document the use of test design techniques in the company's test policy and test strategy. The test manager can set up the test policy together with resource management and other stakeholders.

It is crucial to celebrate successes. When the defect detection percentage (DDP) (see the paragraph on project management) has improved, or when yet another team starts to use test design techniques, then this should be communicated. Communication should not only be through test progress reports, but also via management and departmental meetings. Within most companies, there are platforms for internal presentations. Make use of this possibility! The task of the test manager is to keep promoting the benefits! If the benefits are not clear to project management or resource management or even the testers, the implementation of test design techniques is doomed to fail.

The benefits for project management or resource management are a more predictable quality and better control and therefore higher efficiency. To optimize the benefits, different test design techniques are to be used in different test levels to make the overall test projects more effective. With every technique different types of defect are targeted. For testers, the benefit is that they have more certainty that they are doing the right thing and that test design techniques support (junior) testers in deriving meaningful test cases. If a manager has questions about the thoroughness of a test, the tester can provide the test design specification. A test design provides much more overview much faster than the large pile of detailed test cases that used to be given to the test manager. The test manager probably did not even study them.

To understand the maturity of the test organization and the steps to take next, the test manager can use test improvement models (e.g. TMMi, TPI) as a guideline. Based on this, the test manager can determine where and when the implementation of test techniques will be most profitable.

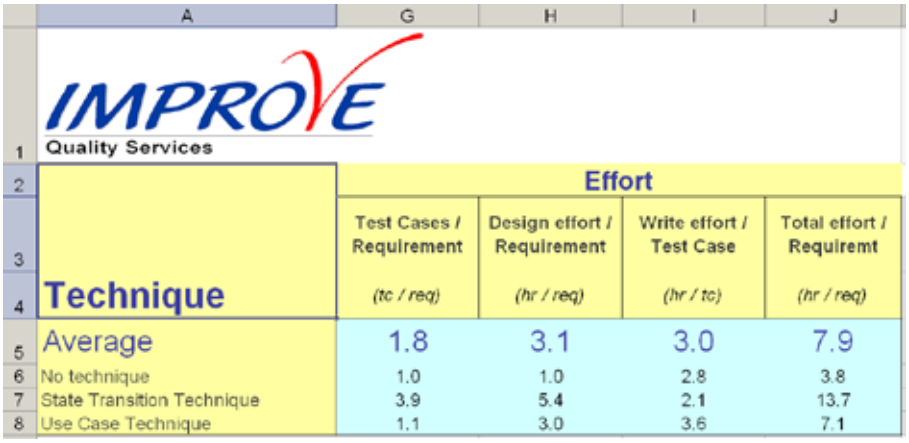
The implementation of test design techniques requires a formally organized project, a sort of improvement project (sometimes even a project within a project). The test manager should lead this project and get a budget for this.

Project Management

The role of a project manager may not be that

obvious when implementing test design techniques. However, the project manager is crucial to making the implementation successful. He is a driving force, as the project manager can act as sponsor. By making room in the budget for training and scheduling test design as a project activity, the success rate increases significantly.

Using test design techniques initially costs (additional) time and effort. A manager would not be a good manager if he did not also want to have clear insight into the possible benefits. One way is to study the test literature on the subject. Another way is to collect metrics during the implementation project. The Defect Detection Percentage (DDP) is a relatively simple metric to collect. The DDP determines the effectiveness of testing for each consecutive stage of the (test) project, including the production stage (see [5]). Besides using DDP, it is relatively easy to collect metrics related to the effort spent in the past when applying test design techniques, and then use these metrics for more accurate estimations in the current project. For example, see figure 1 for metrics from a Healthcare environment we worked in, which were used as the basis for a successor project. They resulted in quite accurate estimations at an early stage of the project.



IMPROVE Quality Services

Technique	Effort: Test Cases / Requirement (tc / req)	Design effort / Requirement (hr / req)	Write effort / Test Case (hr / tc)	Total effort / Requiremt (hr / req)
Average	1.8	3.1	3.0	7.9
No technique	1.0	1.0	2.8	3.8
State Transition Technique	3.9	5.4	2.1	13.7
Use Case Technique	1.1	3.0	3.6	7.1

Figure 2: Effort metrics from a Healthcare environment

Initial investment will be necessary, but this investment can be justified later by the collected metrics. Budget, however, is not the only factor. Some other prerequisites are that configuration management, project management and especially requirements engineering should have some degree of maturity. The testers can help the requirements engineers to improve the quality of the requirements, sometimes even by adjusting the way requirements are organized to support the use of a certain test design technique. More and more test engineers also take courses on requirements, such as the IREB Certified Professional for Requirements Engineering (see www.improveqs.nl). Quite often, this results in a process where requirements get better through the interaction between requirements engineers, designers and testers. The project manager should have an active role in facilitating and streamlining the interaction and communication between these disciplines.

Resource Management

Resource management (including QA) can have a large influence as well as a facilitating role. Resource management can decide that the use of test design techniques is part of the company's test policy and test strategy. The test policy imposes actions upon test management, as well as project management.

A not so obvious role is the facilitating role. Resource management can facilitate in providing test tools and templates to support the implementation and application of test design techniques. The incentive for resource management to organize these assets is that suitable tooling and templates make test design techniques part of the organization's standard test process. Once the use of design techniques no longer depends on some individual testers and managers, the implementation is truly successful.

In the short term, organizing these supporting assets saves time. Testers do not encounter the same problems with impractical templates again and again. A good template guides the tester instead of causing time-consuming activities and discussions. Tooling can also save a lot of time, but only if the tool is suitable for the job. Selecting the tooling is sometimes a project on in its own right, but for several organizations well worth the effort. Examples of tools to be used for designing tests using techniques are CTE, BTWIN, Casemaker and several tools for pair-wise testing. But also when a more general test tool like HP Quality Center is introduced, the effect on the implementation of test design techniques is considerable. To streamline the process, a project will help greatly.

Should the current template be adjusted, or is a new template introduced? Probably some test templates are already in place. Most test teams we have encountered use some kind of test specification template. But that template usually does not differentiate between test design techniques, it is a shell in which physical (high level) and logical (low level) test cases can be entered. A template based on IEEE 829, for example, contains the design steps.

With respect to tooling, the following needs to be considered: none of the test management tools support test designs. To automatically generate test cases using a test design technique, additional tooling must be used. Unfortunately, most of these tools support only one design technique. For a differentiated test approach using several test design techniques, several tools must be used.

Customer

Though the testers may convince the other testers in the team and convince management of using test design techniques, and have an even "higher quality" system for delivery, this is not where the application of test design techniques stops. Because as customer or client, one can also benefit from the use of test design techniques: when applying them in the process of creating acceptance test cases, even before the system actually gets developed. In this period, the customer is already intensively involved in improving the requirements, because when designing test cases, the requirements are studied in a very structured manner. This is much cheaper than realizing after the product has been delivered that the requirements initially were wrong.

Let's start with the most obvious use: acceptance tests. Such tests are created by the user to validate that the supplier has actually developed what was requested. And for these tests, test design techniques may very well be used. A technique like process cycle test (PCT), for example, can help validate whether the system supports the everyday organizational processes. A technique like use case test (UCT) can provide support by creating scenarios in which the system is used and how it should behave. Another advantage is that the scenarios can also be used as a basis to create more detailed work instructions and user manuals.

Even though the customer will not be building the actual system, he is still "building" a business. The quality of his business will have significant impact on the business success; "error-prone tools and procedures will cost a lot, both in money and reputation".

Using test design techniques will help identifying undesired situations and structuring the way of thinking.

Besides using test design techniques for acceptance testing, they can also be of great support during requirements engineering. In our everyday work, one of the things we quite often see is that when an organization starts using test design techniques, they not only find more and different defects in their products, but also find a lot of inconsistencies in the requirements for the systems they build. This is especially true in environments where product families or evolutionary systems are made, requirements seem to age and ultimately the system itself becomes the specification.

Here too, applying test design techniques can be of great help. By applying test design techniques to your requirements, you'll look at the various aspects of the requirements in a structured way. In doing so, you might find situations you would not have considered otherwise. Situations which can, for example, lead to additional development time because

Testing & Finance 2009

The Conference for Testing & Finance Professionals

June 22nd - 23rd, 2009 in Bad Homburg (near Frankfurt am Main, Germany)

The international well-known speakers are amongst others:

Prof. Dr. Schulte-Mattler, Dr. Mike Bartley, Dorothy Graham, Rex Black, Erik van Veenendaal, Graham Bath, Vipul Kocher, Manu Cohen-Yashar, Hans Schäfer, Alon Linetzki, Yaron Tsubery

www.testingfinance.com

Exhibitors

CaseMaker® Compuware hp® ib-bank-systems iSQI® International Software Quality Institute

logica PureTesting Testing Thought Leadership Saxonia Systems A vision of technology. SELA GROUP

Supporting Organisations

VOB SERVICE dpunkt.verlag Díaz Hilterscheid te testing experience

Testing & Finance 2009

The Conference for Testing & Finance Professionals

June 22nd - 23rd, 2009 in Bad Homburg
(near Frankfurt am Main, Germany)

Please fax this form to +49 (0)30 74 76 28 99 or send an e-mail to info@testingfinance.com.

Participant

Company: _____
First Name: _____
Last Name: _____
Street: _____
Post Code: _____
City, State: _____
Country: _____
Phone/Fax: _____
E-mail: _____

Billing Address (if differs from the one above)

Company: _____
First Name: _____
Last Name: _____
Street: _____
Post Code: _____
City, State: _____
Country: _____
Phone/Fax: _____
E-mail: _____
Remarks/Code: _____

☐ 1 Day **450,- €** (plus VAT)

☐ 2 Days **850,- €** (plus VAT)

☐ Yes, I like to join the social event on June 22nd, 2009.

Included in the package: The participation on the exhibition, at the social event and the catering in course of the event.

Notice of Cancellation

No fee is charged for cancellation up to 60 days prior to the beginning of the event. Up to 30 days prior to the event a payment of 50% of the course fee becomes due and up to 15 days a payment of 100% of the course fee becomes due. An alternative participant can be designated at any time and at no extra cost.

Settlement Date

Payment becomes due no later than the beginning of the event.

Liability

Except in the event of premeditation or gross negligence, the course holders and Díaz & Hilterscheid GmbH reject any liability either for themselves or for those they employ. This also particularly includes any damage which may occur as a result of computer viruses.

Applicable Law and Place of Jurisdiction

Berlin is considered to be the place of jurisdiction for exercising German law in all disputes arising from enrolling for or participating in events by Díaz & Hilterscheid GmbH.

Date Signature, Company Stamp

the design has to be adjusted in respect to this particular situation. Or worse, situations for which your supplier chooses his own interpretation and leaves you with a system that does not do the job, but still is according to specification. And then the hassle starts…

Conclusion

In this paper we have covered the actions to be taken by five different stakeholders when implementing test design techniques. Together these actions form a more holistic approach. Every discipline within an organization has to contribute in order to make the test design implementation project successful. For each of these disciplines we have provided guidelines and possible benefits (and surely there will be a lot more than the ones we have discussed, just give it some thought …).

Keep in mind that management commitment is essential in any change management process. Getting them on board requires effective communication about the benefits through the use of metrics. Collect these metrics and lead by example!

In addition, study the test improvement models to get a feeling on where your organization stands regarding the maturity of its test process and where it should go as a next step. It is easier to discuss the direction, if you know where you are and where you're going.

References

[1] Practical Risk-Based Testing, Product RIsk MAnagement: the PRISMA® method, E. van Veenendaal, June 2006. The article is to be found on www.improveqs.nl.

[2] ISTQB Foundations of Software Testing, Dorothy Graham, Erik van Veenendaal, Isabel Evans and Rex Black.

[3] The Testing Practitioner, Erik van Veenendaal.

[4] A Practitioner's Guide to Software Test Design. Lee Copeland.

[5] How to measure test effectiveness using DDP (Defect Detection Percentage), Grove Consultants, December 2003

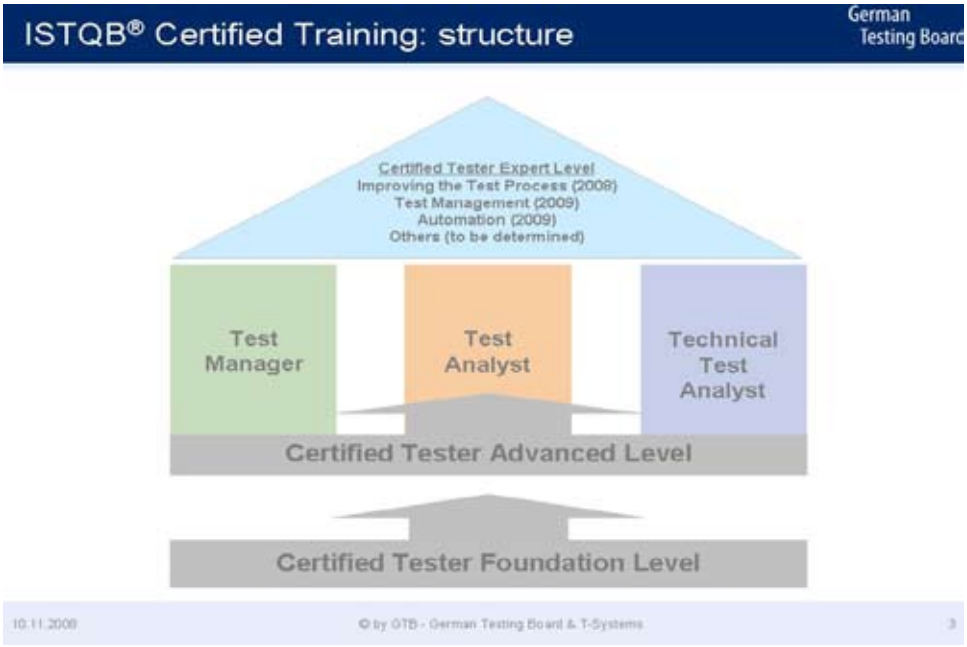
Biography

All authors work as test consultants at Improve Quality Services, a leading Dutch test and quality management consultancy and training company.

Richard van der Pols (CTAL) has a Master of Science degree in Information Sciences and has worked for over 18 years within both development and testing teams in embedded and administrative industries.

Andrew Jong has a Master's degree in (medical) Information Sciences and is an experienced software test analyst for various companies in the financial services industries.

After finishing her Master's degree in Software Technology, Jeanne Hofmans has worked as test consultant in the areas of both administrative and embedded systems.



The new ISTQB® Certified Tester Advanced Level
Focus on practical know-how

by Professor Mario Winter

The new ISTQB® Certified Tester Advanced Level syllabus is now finding its way into practical use. By September 2009 at the latest, all German-language course providers must either have switched to the new advanced level syllabus or have successfully completed re-accreditation. This is not just a rewrite of the "old" syllabus that has been in use since 2003 but a fundamentally revised, updated, intensified and expanded offering for software testers who want to go much further than the basic Foundation Level know-how, be it as test managers, as test analysts or as technical test analysts.

The International Software Testing Qualifications Board (ISTQB®) approved the new ISTQB® Certified Tester Advanced Level syllabus at the end of 2007 and from September 2009 at the latest all training providers must offer only courses based on this new version. What attendees will first notice is the much more extensive content and detail of the new syllabus with its integrated learning objectives. While the 2003 syllabus that is now on its way out consists of 38 pages, the new syllabus runs to 114. But the new Advanced Level testers will not just have to learn more. In the seminars a significantly larger and more intensive practical part awaits them. The ratio of classic knowledge teaching to practical exercises is about 50–50 in seminars that now take three instead of five days. It goes without saying that experienced testers are still entitled to sit the examination without attending a seminar. Extensive literature recommendations will be found on the websites http://www.german-testing-board.info/de/buchempfehlungen.shtm, www.istqb.org

Advanced content for advanced testers

That makes the advanced level of the Certified Tester schedule even more advanced than it already was. The wider scope and strengthening of the practical part set it apart from the basic training yet more clearly than previously (cf. chart). The new Advanced Level incorporates inter alia wide-ranging experience and best practices contributed by the now (as of March 2009) 42 national and regional boards of the ISTQB®. At the same time the working party's international membership has helped achieve a greater degree of standardisation. National procedures that previously differed have been harmonised. This standardisation mainly benefits multinational software projects such as in connection with IT offshoring. As a result, test managers can use the same specialised terminology across continents.

Analytical thinking is in demand

The greater practical orientation of the new ISTQB® Certified Tester Advanced Level is also reflected in extended learning objectives. The previous syllabus mainly sought to ensure that graduates were able to identify and carry out the work required and to provide a degree of practical orientation. The new syllabus additionally promotes and requires a cognitive command and the ability to analyse the different situations in which advanced testers and test managers find themselves.

Here are two examples. In the seminar learners are intended to use the equivalence class analysis technique as they work through a scenario. That is a classic implementation example of the kind already included at Foundation Level. The following scenario, in contrast, requires more advanced abstraction and analytical skills. Certain key product and project figures indicate that a project has fallen behind schedule. What is the explanation for this delay, and what steps must be taken to get the project back on schedule? Context-based exercises of this kind increasingly characterise training for the new Advanced Level and are accordingly to be found in the final exams.

The new content

The new ISTQB® Certified Tester Advanced Level distinguishes between three profiles: the test manager, the test analyst and the technical test analyst. In the past, the focus has been mainly on the predominantly functionally oriented test manager. In the future, training will include much more content and more techniques geared toward the other two specialised areas. All three areas are now anchored in the training as areas of equal value and are clearly distinguished from each other, being taught and examined separately.

The ISTQB® has set the new Advanced Level more clearly apart from the Foundation Level across the board. Each chapter taught has clear learning objectives. In addition, the new Advanced Level distinguishes between specific forms of software systems. Attend training courses and you will get to know some aspects of testing embedded systems such as those that are used in the automotive environment.

The test manager

Training to become a test manager offers inter alia the following new and improved content:

- How can the benefit of testing be measured and shown for the business objectives in question?
- What are the specifics of distributed testing in the context of either outsourcing or insourcing?
- What is Failure Modes and Effects Analysis (FMEA) and how can it be deployed?
- What can test management accomplish in specific contexts when testing non-func-

tional properties?

- Which test management documents must be drawn up and what content must they cover?
- Which processes are available for monitoring progress and for risk-oriented testing?

Test analyst and technical test analyst

Advanced Level now distinguishes between four different categories of test procedures: specification- and structure-oriented and error- and experience-based techniques.

Advanced Level takes its greatest step forward, however, by drawing a clear line between functionally and technically aligned test analyses. While the former concentrate on specification-oriented and on error- and experience-based test procedures, the technical test analyst is acquainted with all test procedures except for a few special specification-oriented techniques.

In addition, the new syllabus offers the following new content:

- Testing of quality characteristics as defined by ISO 9126, such as functionality, reliability or maintainability
- Additional review procedures: management reviews and audits
- Additional error management procedures
- Adopting additional process standards with a focus on improving test processes
- New procedures for automating testing
- New types of tool for automating testing
- Recommendations for the industrialisation of testing processes

Practically oriented

The Advanced Level Certified Tester in its new form has developed into practical training that covers the specific, day-to-day requirements of industry. Its new and clearly defined coverage of the different work areas of testing is oriented to the development that commercial software quality assurance has undergone in recent years. The separate view of functional and technical testing in particular fulfils to the highest degree the requirements of software projects today.

Now that this experience and these best practices have been incorporated into an international standard the new-look training boosts the status of testing activity in general and thereby takes software testing further toward the standing of a profession for highly qualified specialists. In this way it is not just the software industry that will benefit from testers who are more professional and more practice-oriented in their approach. Conversely, software testers who have undergone this training can look forward to further and more highly qualified career paths.

The new Advanced Level – Key data

When does it happen?
The first course providers in Germany are already holding seminars based on the new Advanced Level. By September 2009 all course providers must have fully converted their seminars to the new syllabus and completed their re-accreditation. From September 2009 all ISTQB® Certified Tester Advanced Level examinations will be based on the new syllabus.

What does the exam look like?
The individual test manager, test analyst and technical test analyst modules are taught and examined separately. All examinations are based on multiple-choice test format.

What preconditions must be fulfilled?
To sit the exam you must already hold the Certified Tester Foundation Level certificate and have at least 18 months of career experience as a software tester.

Do existing certificates retain their validity?
Advanced Level certificates issued on the basis of the old syllabus retain their validity. Holders of these certficates are naturally required to learn the new and expanded content of the 2007 syllabus.

How long do the training courses take?
For the new Advanced Level the duration of seminars has increased from three to five days per module (such as test manager) because the training covers much more content and includes a significantly higher share of practical work

Why do seminars for the new ISTQB® Certified Tester Advanced Level take five days instead of three days as hitherto?
There are two main reasons:
1) First, the course content has been increased considerably. In the scope and depth of what is taught, the new Advanced Level has become a fully-fledged training qualification for an advanced tester with different focal points.
2) Second, training courses are now even more practical in orientation. Course content is more strongly oriented toward everyday project scenarios, and the number of practical exercises has increased significantly. Theory and practice now each account for roughly half the time spent training. All of this is only feasible if training is given over a longer period.

Why are there three different course specialisations at the new Advanced Level?
Especially as a more highly qualified activity, the tester per se virtually no longer exists. In recent years his work has extended to different specialisations. The old Advanced Level syllabus already made a clear distinction between test management and functional testing. In addition, the tasks that face many IT projects require technically oriented testing specialists – for the technical integration of test environments, for example. ISTQB® training now caters for this demand, which mainly comes from business. It distinguishes between three roles, those of the test manager, the test analyst and the technical test analyst.

Does the expansion of course content not jeopardise the standardisation of certified tester training?
Quite the reverse. A wider range of experience and best practices from more countries than ever before has found its way into the new syllabus. The ISTQB® and national boards such as the GTB have harmonised and standardised these different approaches in years of work with the result that today's new ISTQB® Certified Tester Advanced Level is a standard that can be used better internationally than ever before. It has become an ideal instrument for companies to simplify communication in cross-border IT projects.

Biography

Mario Winter is professor at the Faculty of Computer Science and Engineering Science of Cologne University of Applied Sciences, and a member of the Software Quality Group. From 1983 to 1987, he was engaged in industrial and scientific software projects, and between 1994 and 2002, he was research fellow at the FernUniversität Hagen. Currently he is spokesman of the German Special Interest Group in Software Testing, Analysis, and Verification of the German Informatics Society (GI-TAV) and a founding member of the German Testing Board (GTB). His teaching and research focus is on software development and project management, especially on the model-based development and quality assurance of software.

Masthead

EDITOR
Díaz & Hilterscheid
Unternehmensberatung GmbH
Kurfürstendamm 179
10707 Berlin, Germany

Phone: +49 (0)30 74 76 28-0 Fax: +49 (0)30 74 76 28-99 E-Mail: info@diazhilterscheid.de

Díaz & Hilterscheid is a member of "Verband der Zeitschriftenverleger Berlin-Brandenburg e.V."

EDITORIAL
José Díaz

LAYOUT & DESIGN
Katrin Schülke

WEBSITE
www.testingexperience.com

ARTICLES & AUTHORS
editorial@testingexperience.com

350.000 readers

ADVERTISEMENTS
sales@testingexperience.com

SUBSCRIBE
www.testingexperience.com/subscribe.php

PRICE
online version: free of charge
print version: 8,00 €

ISSN 1866-5705

Index Of Advertisers

CaseMaker	65	Parasoft	57
Díaz & Hilterscheid GmbH	2, 6, 22-23, 30, 42, 47, 96-97, 103, 104	PSTech	32
eXept	19	PureTesting	103
expo:QA	101	Quality First Software GmbH	54
ImproveQS	21	RCBS	78
ISCN	85	RSI	70, 103
iSQI	11, 35	SELA	17, 103
ISTQB	52-53	Testing Experience	8, 91
Kanzlei Hilterscheid	88	Test Planet	13, 58, 103
Microsoft	92		

Training with a View

Díaz Hilterscheid

also onsite training worldwide in German, English, Spanish, French at
http://training.diazhilterscheid.com/ training@diazhilterscheid.com

"A casual lecture style by Mr. Lieblang, and dry, incisive comments in-between. My attention was correspondingly high. With this preparation the exam was easy."
Mirko Gossler, T-Systems Multimedia Solutions GmbH

"Thanks for the entertaining introduction to a complex topic and the thorough preparation for the certification. Who would have thought that ravens and cockroaches can be so important in software testing"
Gerlinde Suling, Siemens AG

Kurfürstendamm, Berlin © Katrin Schülke

15.06.09-18.06.09	Certified Tester Advanced Level - TESTMANAGER	German	Berlin
22.06.09-24.06.09	Certified Tester Foundation Level	German	Munich
29.06.09-02.07.09	Certified Tester Advanced Level - TESTMANAGER	German	Frankfurt
06.07.09-08.07.09	**Testeur Certifié ISTQB - niveau fondamental**	**French**	**Paris**
15.07.09-17.07.09	IREB - Certified Professional for Requirements Engineering - Foundation Level	German	Berlin
20.07.09-22.07.09	Certified Tester Foundation Level - Kompaktkurs	German	Hamburg
20.07.09-24.07.09	Certified Tester Advanced Level - TEST ANALYST	German	Berlin
27.07.09-30.07.09	Certified Tester Foundation Level	German	Düsseldorf/Köln
27.07.09-30.07.09	Certified Tester Advanced Level - TESTMANAGER	German	Berlin
03.08.09-06.08.09	Certified Tester Advanced Level - TESTMANAGER	German	Dresden
10.08.09-12.08.09	Certified Tester Foundation Level - Kompaktkurs	German	Berlin
18.08.09-21.08.09	**Certified Tester Foundation Level**	**English**	**Lissabon**
24.08.09-26.08.09	Certified Tester Foundation Level - Kompaktkurs	German	Frankfurt
31.08.09-03.09.09	Certified Tester Foundation Level	German	Berlin
31.08.09-03.09.09	Certified Tester Advanced Level - TESTMANAGER	German	Hamburg
07.09.09-09.09.09	Certified Tester Foundation Level - Kompaktkurs	German	Stuttgart
07.09.09-11.09.09	Advanced Level Testing - Technical Test Analyst	German	Berlin
07.09.09-09.09.09	**Testeur Certifié ISTQB - niveau fondamental**	**French**	**Paris**
14.09.09-18.09.09	Certified Tester Advanced Level - TEST ANALYST	German	Düsseldorf/Köln
16.09.09-18.09.09	Certified Professional for Requirements Engineering - Foundation Level	German	Berlin
21.09.09-24.09.09	**Certified Tester Foundation Level**	**English**	**Berlin**
28.09.09-02.10.09	Certified Tester Advanced Level - TESTMANAGER	German	Berlin
12.10.09-14.10.09	Certified Tester Foundation Level - Kompaktkurs	German	Berlin
26.10.09-29.10.09	Certified Tester Foundation Level	German	Frankfurt
02.11.09-06.11.09	Advanced Level Testing - TECHNICAL TEST ANALYST	German	Berlin
09.11.09-12.11.09	Certified Tester Foundation Level	German	Berlin
16.11.09-18.11.09	Certified Tester Foundation Level - Kompaktkurs	German	Hamburg
18.11.09-20.11.09	Certified Professional for Requirements Engineering - Foundation Level	German	Berlin
23.11.09-27.11.09	Certified Tester Advanced Level - TEST ANALYST	German	Berlin
30.11.09-04.12.09	Certified Tester Advanced Level - TESTMANAGER	German	Berlin
07.12.09-10.12.09	Certified Tester Foundation Level	German	Stuttgart
14.12.09-16.12.09	Certified Tester Foundation Level - Kompaktkurs	German	Berlin
14.12.09-17.12.09	Certified Tester Foundation Level	German	Düsseldorf/Köln

- subject to modifications -