The road to Hell[1] is paved with good intentions: A story of (in)secure software development

Richard Sassoon*, Martin Gilje Jaatun†, Jostein Jensen*
*Norwegian University of Science and Technology (NTNU)
†Department of Software Engineering, Safety and Security SINTEF ICT
Trondheim, Norway
rsassoon@gmail.com, Martin.G.Jaatun@sintef.no, Jostein.Jensen@idi.ntnu.no

***Abstract*—In this paper, we present the results of a security assessment performed on a home care system based on SOA, realised as web services. The security design concepts of this platform were specifically tailored to meet new security challenges and to be compliant with legal frameworks applicable to the healthcare domain. This security design was fed as input to the development team, which implemented the system. However, our assessment revealed a software platform with severe security weaknesses and vulnerabilities, demonstrating dangers that are, or should be, well known.**

Our experience illustrates that security must be built as an intrinsic software property and emphasises the need for security awareness throughout the whole software development lifecycle.

***Index Terms*—secure software development; secure design; security assessment; security awareness; SOA**

I. Introduction

Enabling information systems to communicate via open networks such as the Internet, will always be associated with elements of risk. Mavridis et al [1] correctly state that "*Security risks cannot be entirely removed when transmitting information over the Internet*". The European Parliamentary Technology Assessment (EPTA) network has made similar considerations and specifically expressed concerns that privacy is challenged by the increase in development of ICT applications for the healthcare sector [2]. Such concerns are also raised by others, such as Ilioudis and Pangalos [3] and van der Haak et al. [4]. The privacy of personal data is continuously threatened by users with bad intentions. Healthcare systems handle sensitive personal information, and only the imagination limits the possible misuse of such; pharmaceutical industry can use it for directed marketing purposes, insurance companies can adjust their customers' insurance coverage, and information such as personal identification numbers can be used for identity thefts.

Service Oriented Architecture (SOA) is gaining popularity, and according to independent reports by Computer Economics [5] and Gartner [6] it has been adopted by more than fifty percent of companies worldwide, particularly in Europe and North America. Healthcare organisations are also starting to develop information systems based on SOA. One example of such initiative is presented by the National ICT, which is a Norwegian institution coordinating ICT initiatives in hospital health services. They recommend SOA as a way of achieving a common platform for the services within their responsibility [7]. However, SOA introduces some security concerns, as pointed out by Epstein et al. [8] and the New Rowley Group [9], such as access to services from inside and outside the organisation. Therefore, developing secure SOA based systems is not trivial and, according to Knight [10], security *"will only be part of your service oriented architecture (SOA) if you build it in yourself"*.

The following sections present experiences related to the development of a SOA-based service platform intended to support home care of elderly and people with cognitive disabilities. Information systems designed to treat sensitive personal information are subject to extensive regulation by legislation [11] [12] to ensure availability, confidentiality and integrity of personal information, and as such should be securely designed, implemented and deployed.

The rest of this paper is organised as follows: In section II we describe the background for our assessment, including a brief description of the MPOWER project. Section III describes our assessment methodology, and high-level results are given in section IV. We discuss our findings in section V, and offer some recommendations in section VI. Section VII concludes the paper.

II. Background

The security evaluation was performed on the results of a European research project, MPOWER, using its security design as a starting point. As already mentioned, the objective of MPOWER is to develop a healthcare platform that deals with sensitive health data and therefore it should comply with the European Union Directive 95/46/EC [12], which regulates the handling of private data for the member states of the EU. This and specifically the Norwegian implementation of this directive was studied in order to define a set of security requirements to be included in the security design. As part of the evaluation, several components of the WS-* specifications were reviewed, in search of proper ways to deal with security for Web Services.

[1]Hell is a small collection of houses, a hotel, and a shopping mall, situated right next to Trondheim Airport, whose main claim to fame is being the birthplace of 1990 Miss Universe winner Mona Grudt (a.k.a. "The beauty queen from Hell").

A. Security Design

In this section we give an overview of the security design elaborated for the project. We will see that a proper documentation is not enough to achieve security, and that security awareness is an important element throughout the entire development process. Table I shows the security components that were identified, and which should be implemented in order to fulfill the legal requirements for the platform. The security requirements are further discussed in Jensen et al. [13].

Component	Description
Access Control	Access control includes the Authentication and Authorisation services. The Authentication service includes operations to verify a user's credentials, and initialises a process for issuing security tokens. The Authorisation service determines which operations and data an authenticated user can access, allowing access to resources only to legitimate, authorised users. Authorisation in MPOWER is based on a Role-Based Access Control (RBAC) scheme, extended by a Context-Based protection[2].
User Management	The user management service provides operations to administer users of the system, such as adding and deleting users, updating users roles and retrieving user information.
Role Management	The role management service enable the administrator to manage the roles of the system. Administrator may e.g. add/delete roles, and assign/delete users to/from roles
Access Management	The access management service includes operations to manage the permissions and access profiles associated with the access control system.
Token Management	The token management service is used by the Authentication and Authorization services to issue and validate security tokens, which are used for access control purposes.
Public Key Infrastructure	The PKI service provides an interface for general management of certificates, such as issue, renew, revoke and verify certificates.
Audit	The audit service provides an interface enabling other services to log data that needs to be stored and retrieved for future audit purposes.
Encryption	The encryption mechanism describes functionality related to confidentiality and integrity protection of data.
Secure Storage	The secure storage mechanism describes functionality for storing data securely and for retrieving secured data from storage.
Secure Communication	Secure communication is the mechanism describing functionality needed to secure data being transmitted between two endpoints.

TABLE I
MPOWER SECURITY COMPONENTS. ADAPTED FROM [14]

B. System Environment

The expected environment of the system is presented in figure 1, where we can visualise the most important components, and which helps us understand the tests and results. As we will see in section IV, what we expect is not what we experience.

[2]Context information is important for limiting a role's access to a subset of target objects, depending on context elements associated to the user that was assigned the role [15]. Such elements include the relationship this user has with the requested information, e.g., a user with a *doctor* role that should have access only to his patients' data.

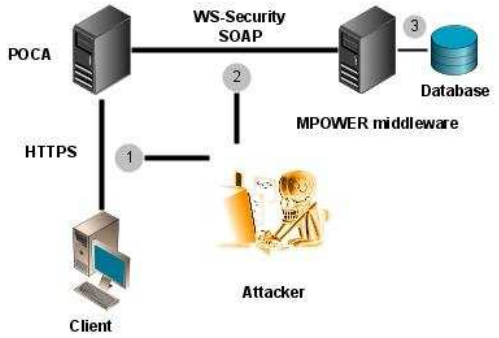

Fig. 1. Expected System Environment

C. Project Characteristics

The MPOWER project was funded by the European Commission's 6th Framework Programme, and thus had to satisfy a number of constraints. The project partners had to be diverse in geographical location and represent industry, academia and research institutions. EU research proposals all follow a set template, and it is implicitly assumed that the work will be organised in "Work Packages".

Although an agile Scrum approach was chosen for the project, the different work packages were established in a conventional manner, and developed independently. A separate security WP was a part of the plan, which should have been a good approach, since this would contribute to setting focus on the security aspects of the project. Unfortunately, due to a serious lack of continuity of key members in the project, the security design was delayed for a long period and it had to be elaborated in parallel with the implementation, and thus the implementation of the security mechanisms started before the security design was finished.

Other early project decisions contributed to the delay of the security design and the resulting parallelism mentioned above: No threat modeling was employed and no security requirements were thought of. The latter had to be done, finally, when the security design document was being developed.

The development process followed a Scrum approach for the most part. The security design process, however, ended up having more in common with a traditional waterfall approach, which may have contributed to the security work falling out of synch with the rest. In line with the chosen Scrum approach, a backlog of functional requirements was maintained. Somehow, only the functional results of the security design made it out of the backlog (e.g. the authentication and token management services were implemented), leaving most non-functional security aspects alone in the dark.

III. ASSESSMENT METHODOLOGY

Every organisation should have some standard methods for performing its development activities, including security tests of its applications. A poorly conducted test, i.e., one that does not find critical vulnerabilities, could expose the

company's assets, e.g., causing leakage of sensitive data and, consequentially, financial loss. As Herrman points out [16], the disclosure of private information makes the company liable for not complying to government and industry regulations, and to lawsuits from clients or customers, besides the damage to its credibility.

According to NIST [17], the use of a consistent, documented and repeatable security testing methodology allowsfor benefits such as:

- Structured security testing, minimising testing risks, where the results are properly documented for future reference, providing the means to assure the test has been conducted as it should be.
- Facilitate the transition of new assessment personnel.
- Effective planning of resources (e.g., staff, hardware, software) to use for performing the assessments, thus reducing overall costs and time to conduct them.

A. Reviewed Methods

Our security assessment was inspired by the following recognised methodologies and guidelines:

- The *Open Source Security Testing Methodology Manual (OSSTMM)* [18], [19], which provides a methodology for a thorough security test, referred to as an OSSTMM audit.
- The *Technical Guide to Information Security Testing and Assessment - NIST-SP800-115* [17]. This document provides guidelines for planning and conducting an information security assessment via technical testing and examination techniques.
- The *OWASP Testing Guide* [20], created by the The Open Web Application Security Project (OWASP). This guide covers the scope of testing, the principles for a successful testing process, and the necessary testing techniques. The focus is the integration of testing in the software development life cycle.
- The *SIFT Web Services Security Testing Framework* [21] is a methodology specifically designed for testing web services security.

B. Chosen Method

Our objective was to demonstrate the level of security that a SOA-based healthcare system can achieve, but without the illusion of uncovering every possible flaw, as this would be unrealistic. By following some recommendations from the methodologies and guidelines presented above, we defined a custom approach to perform the security testing on the MPOWER platform. This method consists of the same three phases defined by NIST, adapted and simplified, to suit our purpose:

- **Planning**: In order to plan what kind of tests should be part of our assessment, we looked at the security components, outlined in section II-A, which should offer an acceptable level of security if well implemented. This, and knowing that the system is implemented via web services, served as the basis for selecting test cases associated to relevant threats. Incremental changes to the test cases played an important part in the process, as new details of the implementation were unraveled.
  During this phase we had to characterise:
  - Test target: The assessment would be performed on a proof of concept application (POCA), that makes use of the MPOWER middleware services. The tests would be conducted in a controlled environment, with no restrictions regarding scope or techniques to be employed.
  - Objective: The main objective was to verify if the MPOWER security components offered the desired security level or fell short of their intent. Security problems discovered in the POCA implementation might be considered as a consequence of the assessment.
  - Techniques: A mix of techniques would be used to perform the tests, the main one being penetration testing and, to a lesser extent, code and configuration reviews. As McGraw [22] tells us, passing a penetration test does not guarantee that an application is free of vulnerabilities.
    It is important to point out that the tester had direct access to the necessary resources (i.e., source code, WSDL, application server, database).
  - Scope and limitations: The scope of the tests was not as extensive as it could have been, leaving out, e.g., operating system specific flaws, wireless transmissions, physical security and social engineering.
    The focus would be on the functionalities provided by the MPOWER platform and the observed security when using them. The test cases better define the intended scope, and since the assessment was performed in a controlled academic environment, there were no restrictions on executing tests that could crash the application, provoking a denial of service.
  - Tools: Based on the selected test cases, the following tools were chosen:
    - WebScarab and Burp Suite: Their proxy functionality allows for tampering and replay attacks, besides analysis of HTTP(S) traffic.
    - Wireshark: Used to inspect packets going through a specified network interface (including the loopback).
    - soapUI: Having its focus on Web Services testing, it was a natural choice. It allows for WSDL inspection and service invocation, besides a good support for WS-Security.
    - WSFuzzer: This tool has the objective to automate SOAP-based web services penetration testing by dynamically generating several attack vectors, for fuzzing parameters.
  - Constraints: The time frame was quite limited, so the focus had to be on the most important test cases. Such factors could negatively influence the end result.

- **Execution**: Having established the test cases, these were put into practice in order to identify vulnerabilities.
- **Post-Execution**: The results generated during the execution phase were analyzed and presented together with the respective countermeasures. General recommendations based on these findings were proposed.

IV. Overall Results

This section presents the test cases which uncovered issues, together with a short explanation, as well as some other considerations based on our results. As said before, the basic security mechanisms, expected and shown in figure 1, were not active and therefore many attacks were possible.

A. Summary of Risks

Table II gives an overview of the threats that were found by applying the specific test cases.

Test Case	Results
WSDL Scanning	Internal operations that do not need authorisation can be invoked directly via SOAP messages.
Replay Attacks	Any kind of replay attack is possible, with or without modification, at any time.
Parameter Tampering	Requests can be modified on the fly, without detection.
Forced Browsing	Users can access interfaces and functionalities related to roles they do not have.
Cross Site Scripting (XSS)	It is possible to recover session cookies and then hijack users' sessions.
XML Injection	Can trigger XSS attacks.
Test the username/-password authentication scheme	Login credentials are transmitted over insecure channels and passwords are stored in cleartext in the database.
Test the transmission of security tokens Test the effectiveness of the security token	The tokens may be captured by an attacker wishing to send direct SOAP requests, and can also be modified in an attempt to trigger other operations not indicated.
Session Management	It is possible to steal session cookies and impersonate users for long periods of time.

TABLE II
Summary of Risks

B. Looking Further

The assessment proved that the POCA and the MPOWER platform are vulnerable to common attacks targeting web applications. Considering the *OWASP Top 10 web application vulnerabilities* [23], seven of them are present in our case study system:

1) Cross Site Scripting (XSS)
2) Injection Flaws
3) Information Leakage and Improper Error Handling
4) Broken Authentication and Session Management
5) Insecure Cryptographic Storage
6) Insecure Communications
7) Failure to Restrict URL Access

Based on our observations, we can infer that SOA-based systems in general are expected to suffer from the same problems if security is not treated properly. While this is not surprising, the fact that an organisation that is concerned with data confidentiality and integrity does not implement basic security mechanisms, makes us wonder how many other similar cases that are completely vulnerable.

Even though we evaluated a healthcare system, we can extrapolate the results to other domains since the vulnerabilities found are not specific. Therefore, the findings presented are relevant when considering the development of secure applications, based on SOA or not. Problems related to disclosure of personal information will cause a loss of users' trust and make the organisation(s) behind the applications liable to lawsuits, no matter if it happens in the healthcare domain or any other.

V. Discussion

Although our dedicated testing environment and access to all necessary resources made it easier to execute the attacks (e.g., via proxying or sniffing), we have to bear in mind that a skilled attacker could perform the same actions on the system, only needing more preparation to do so. Just having the possibility for an attack is enough reason to be worried about the security of a system. In a production environment the system will be exposed, and if we haven't been able to weed out all the security flaws, we should expect some form of attack to be successful. Thus, the vulnerabilities found have to be mitigated; assuming that they won't be found by the hackers is bordering on the naïve.

It may seem that we are dealing with an old topic here, but we have to consider that, even though there is a careful planning and security design, problems exist and should not be discovered too late. A proper security validation has to be performed periodically during the software development life cycle to make sure that there are no loose ends in any module or their integration.

We also have to take into account that both POCA and the MPOWER platform are prototypes of an ongoing research project, and are still not ready for production. Even that being the case, it is not an excuse for the probable relaxed focus on the security aspects, considering that the formal plans maintained a high security posture. Nevertheless, the costs for fixing the issues at this point in the project are certainly higher than if the assessment was performed earlier, or if security testing had been part of the secure development lifecycle (SDLC).

We can also wonder about the project aspects that may have influenced the security achieved and perceived. Is SCRUM the problem? Is it Waterfall? Or is it simply communication problem? As the original project plan did not comprise testing, no problems could be discovered and associated to a particular moment in time. What happened is due to a combination of factors and no single element can be pointed as the sole culprit.

Although agile methods make it difficult (or impossible?) to comply with the stringent documentation requirements of, e.g., the Common Criteria [24], several authors have argued that agility and security need not be inversely proportional measures. Beznosov [25] opines that the agile XP methodology can provide "good enough" security, while Wäyrynen et

al. [26] claim that the solution to achieving security in an XP development is simply to add a security engineer to the team. Siponen et al. [27] advocate a solution that more or less can be summed up as ”think about security in every phase”. Although this may be a result of their choice of example agile method (Feature-Driven Development), we are not entirely convinced that simply declaring security aspects to be yet another feature will result in secure software.

In an earlier contribution, Poppendieck [28] argues that agile methods (specifically: XP) are just as suitable as traditional development methods for developing safety-critical applications. It may not follow immediately that “safe” software is also “secure”, but the former is required to pass auditing procedures that should be customisable to suit requirements for the latter.

Was the idea to implement a security work package a good one? Work packages are typically enough unto themselves, evolving on their own while ignoring other WPs. Is it better to implement security in every work package? We have to consider that there are re-usable security “components”, and these are probably best developed in a separate work package. Furthermore, a separate security WP gives security the proper attention/focus, avoiding a project falling into the usual trap: *”We’ll take care of security AFTER everything else works”*.

McGraw argues that security needs to be in focus from the beginning [29], as our experience demonstrate, and that the focus should continue during the whole project. The fact that the security design was delayed and, therefore, other components were developed without considering the security work package, set the stage for a big hole in the platform (one glaring example was hard-coded values in a web service that ended up conflicting with the security design). Communication problems among project members intensified the issues, by not bringing word about the integration of the results from the security WP and the consequences related to their (non) use. According to Howard and Lipner [30], there is a need for a *security push* in the whole organisation, or project groups, in order to focus on security alone and find problems.

Security requirements were not part of the project requirements. Partly using a Model-Driven Development (semi-agile) approach, the system design was based on models/diagrams, such as use cases, from where functional requirements were derived. The use cases in question did not cover security, and thus no security requirements were generated (we would have expected some obvious ones, such as confidentiality-protection of a doctor-patient message). Employing misuse cases would have been good idea in this setting, but they were voted down early in the project.

As we can see, there are many subtleties to be considered, which could affect the end result of a project. If any of those are out of synchrony with the others, bad things are bound to happen.

VI. General Recommendations

This practical experience on evaluating the security of such a platform is important to enlighten system designers and developers with regard to the security features of a complex project, and should be exposed to interested parties as a wake-up call since it demonstrates that common vulnerabilities are there for a reason: not enough attention was put into security during the software development life cycle.

The literature is abundant with methods to tighten the security of an application, but our intention here is just to give an idea of what can be done and why. Table III shows some recommendations.

Test Case / Threat	Countermeasures
WSDL Scanning	Do not make a WSDL file publicly available, only to trusted parties. If not possible and the internal operations cannot be removed from the application, at least hide these operations and give them non-intuitive names, i.e., not easily guessable. Another option is to restrict access to these operations via an XML Firewall [31].
Replay Attacks	Use of a signed message freshness identifier, such as a WS-Security timestamp [32], would prevent such attacks as an end-to-end complement to the point-to-point SSL/TLS. Since Web Services can communicate with other services, message layer security (provided by WS-Security) is essential to avoid intermediary manipulation.
Parameter Tampering	Ideally, both SSL/TLS and WS-Security should be implemented to avoid this attack. Building from the above recommendation, figure 1 illustrates that modifications to the requests can be made in both links 1 and 2 (data in transit), but there are cases when we have to consider the possibility of data tampering in intermediary nodes, which justifies the use of WS-Security. As this consideration applies to other recommendations described here, the approach to follow also depends on the specific scenario (e.g., are there any intermediary nodes that can manipulate data if only SSL/TLS is implemented?).
Forced Browsing	The RBAC module and its utilisation by the POCA should be revised to prevent unauthorised access to interfaces and operations. The access should be granted by the middleware services, and not the POCA, by analyzing the security token, for example.
Cross Site Scripting (XSS) XML Injection	Input sanitisation via regular expressions, for example, are a common way to restrict the input received. Shanmugam et al. [33] and Steel et al. [34] propose similar approaches to validate the input through regular expressions.
Test the username/password authentication scheme	First, use of SSL/TLS together with WS-Security, to prevent eavesdropping. Second, store a salted hash of the password, avoiding to keep it as cleartext, and attacks that make use of pre-computed hash tables (Rainbow Tables).
Test the transmission of security tokens Test the effectiveness of the security token	SSL/TLS and WS-Security provides the necessary confidentiality and integrity protection for the security token, which guarantees a proper transmission and avoids its modification.
Session Management	In order to prevent the session cookie from being stolen, SSL/TLS should be used.

TABLE III
SUMMARY OF RECOMMENDATIONS

VII. Conclusion

We have presented the results of a security assessment performed on an eHealth platform, demonstrating that just having proper documentation/design covering data privacy and integrity is not enough to construct a secure system. We

have discussed many factors that may have contributed to this malaise, but the main lesson learned is that it is necessary that every person involved in such a project is aware of the consequences of not thinking about, implementing and testing security from the beginning. Only then will it be possible to achieve more secure systems.

ACKNOWLEDGMENT

This paper is based in part on a MSc thesis at the Norwegian University of Science and Technology [35].

REFERENCES

[1] I. Mavridis, C. Georgiadis, G. Pangalos, and M. Khair, "Access control based on attribute certificates for medical intranet applications," *J Med Internet Res*, vol. 3, no. 1, p. e9, Mar 2001. [Online]. Available: http://www.jmir.org/2001/1/e9/

[2] EPTA, "ICT and Privacy in Europe, Experiences from technology assessment of ICT and Privacy in seven different European countries) (accessed 12-03-2009)," URL: http://epub.oeaw.ac.at/ita/ita-projektberichte/e2-2a44.pdf, 2006.

[3] C. Ilioudis and G. Pangalos, "A framework for an institutional high level security policy for the processing of medical data and their transmission through the internet," *J Med Internet Res*, vol. 3, no. 2, p. e14, Apr 2001. [Online]. Available: http://www.jmir.org/2001/2/e14/

[4] M. van der Haak, A. C. Wolff, R. Brandner, P. Drings, M. Wannenmacher, and T. Wetter, "Data security and protection in cross-institutional electronic patient records," *International Journal of Medical Informatics*, vol. 70, no. 2-3, pp. 117 – 130, 2003, mIE 2002 Special Issue. [Online]. Available: http://www.sciencedirect.com/science/article/B6T7S-48WJWDF-2/2/9c027e5d8b9b2ee0e15c0da43e83df85

[5] Computer Economics, "SOA Adoption Surges (accessed 16-02-2009)," URL: http://www.computereconomics.com/article.cfm?id=1423&tag=rbspot, January 2009.

[6] Gartner, Inc, "Gartner Says the Number of Organizations Planning to Adopt SOA for the First Time Is Falling Dramatically (accessed 16-02-2009)," URL: http://www.gartner.com/it/page.jsp?id=790717, November 2008.

[7] Nasjonal IKT, "Tjenesteorientert arkitektur i spesialhelsetjenesten (SOA for specialized health services)," Available at http://www.nasjonalikt.no/Publikasjoner/Tjenesteorientert_arkitektur_i_spesialisthelsetjenesten_hovedrapport_full_v1_0e.pdf, 2008.

[8] J. Epstein, S. Matsumoto, and G. McGraw, "Software security and SOA: danger, Will Robinson!" *Security & Privacy, IEEE*, vol. 4, no. 1, pp. 80–83, 2006.

[9] New Rowley Group, Inc, "The Challenge of Securing SOA," Available at ftp://ftp.software.ibm.com/software/uk/flexible/wp/the_challenge_of_securing_soa.pdf, 2006.

[10] W. Knight, "Security – built-in or bolted-on to the soa?" *Infosecurity Today*, vol. 2, no. 2, pp. 38 – 40, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/B7GWT-4G4NCXC-H/2/64f624da4a619a4e760cb256fca643bb

[11] J. H. Celine van Doosselare, Petra Wilson and D. Silber, "ehealth..... but is it legal?" *Eurohealth*, vol. 13, pp. 1 – 4, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/B7GWT-4G4NCXC-H/2/64f624da4a619a4e760cb256fca643bb

[12] Commission of the European Communities, "Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995: On the Protection of Individuals with Regard to the Processing of Personal Data and on the Free Movement of such Data," *Official Journal of the European Communities L 281, 23 November 1995, p. 31*.

[13] J. Jensen, I. Anne Tøndel, M. Gilje Jaatun, P. Håkon Meland, and H. Andresen, "Reusable Security Requirements for Healthcare Applications," in *ARES 2009: Proceedings of the Fourth International Conference on Availability, Security, and Reliability*. IEEE Computer Society, 2009.

[14] J. Jensen, *Security Middleware Design - MPOWER Project Deliverable D5.2*. MPOWER Consortium, 2008.

[15] A. Kumar, N. Karnik, and G. Chafle, "Context sensitivity in role-based access control," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. 3, pp. 53–66, 2002.

[16] M. Herrmann, "Security strategy: From soup to nuts," *Information Security Journal: A Global Perspective*, vol. 18, no. 1, pp. 26–32, 2009.

[17] K. Scarfone, M. Souppaya, A. Cody, and A. Orebaugh, *NIST Special Publication 800-115: Technical Guide to Information Security Testing and Assessment*. NIST, 2008.

[18] P. Herzog, *OSSTMM 3 LITE - Introduction and Sample to the Open Source Security Testing Methodology Manual*. ISECOM, 2008.

[19] ——, *OSSTMM 2.2 - Open Source Security Testing Methodology Manual*. ISECOM, 2006.

[20] OWASP, *OWASP Testing Guide v3.0*. The OWASP Foundation, 2008.

[21] C. Wong and D. Grzelak, *A Web Services Security Testing Framework - Version 1.0*. SIFT Pty Limited, 2006.

[22] B. Arkin, S. Stender, and G. McGraw, "Software Penetration Testing," *IEEE Security & Privacy*, vol. 3, no. 1, pp. 84–87, 2005.

[23] OWASP, "OWASP Top 10 2007 (accessed 03-05-2009)," URL: http://www.owasp.org/index.php/Top_10_2007.

[24] F. Keblawi and D. Sullivan, "Applying the common criteria in systems engineering," *IEEE Security and Privacy*, vol. 4, no. 2, pp. 50–55, 2006.

[25] K. Beznosov, "eXtreme Security Engineering: On Employing XP Practices to Achieve "Good Enough Security" without Defining It," in *Proceedings of the First ACM Workshop on Business Driven Security Engineering (BizSec)*, 2003.

[26] J. Wäyrynen and M. Boden and G. Bostrøm, "Security engineering and eXtreme programming: An impossible marriage?" in *Extreme Programming and Agile Methods - Xp/ Agile Universe 2004, Proceedings*, ser. Lecture Notes in Computer Science, vol. 3134. Springer-Verlag Berlin, 2004, pp. 117–128.

[27] M. Siponen, R. Baskerville, and T. Kuivalainen, "Integrating security into agile development methods," in *Proceedings of Hawaii International Conference on System Sciences*, 2005.

[28] M. Poppendieck and R. Morsicato, "XP in a Safety-Critical Environment," *Cutter IT Journal*, vol. 15, pp. 12–16, 2002.

[29] G. McGraw, *Software Security: Building Security In*. Addison-Wesley Professional, 2006.

[30] M. Howard and S. Lipner, *The Security Development Lifecycle*. Redmond, WA, USA: Microsoft Press, 2006.

[31] M. Jensen, N. Gruschka, R. Herkenhoner, and N. Luttenberger, "SOA and Web Services: New Technologies, New Standards - New Attacks," in *ECOWS '07: Proceedings of the Fifth European Conference on Web Services, 2007.*, Nov. 2007, pp. 35–44.

[32] A. Nadalin, C. Kaler, R. Monzillo, and P. Hallam-Baker, *Web Services Security: SOAP Message Security 1.1 (WS-Security 2004) - OASIS Standard incorporating Approved Errata*. OASIS Open, 2006.

[33] J. Shanmugam and M. Ponnavaikko, "A solution to block cross site scripting vulnerabilities based on service oriented architecture," *Computer and Information Science, ACIS International Conference*, vol. 0, pp. 861–866, 2007.

[34] C. Steel, R. Nagappan, and R. Lai, *Core Security Patterns: Best Practices and Strategies for J2EE^{TM}, Web Services, and Identity Management*. Prentice Hall PTR, 2005.

[35] R. Sassoon, "Security in soa-based healthcare systems," Master's thesis, Norwegian University of Science and Technology (NTNU), 2009.