Possible attacks on XML Web Services

Esmiralda Moradian[†] *, and Anne Håkansson*[††]*,*

[†,††]*Department of Information Science, Computer Science,*
Uppsala University, Box 513, SE-751 20, Uppsala ,Sweden

Summary
Web Services make it easy for organisations to participate in real time communication. The inevitable challenge facing organisations today is to implement adequate Web Services security. The attacks on Web Services might cause halt of the entire network communication or expose confidential information in an organisation. In this paper, we present, Web Services security and security concerns together with analysis of possible attacks on SOAP [1] implementation of XML [2] Web Services over HTTP[3]. We will discuss some important questions about Web Services vulnerabilities emphasized by a study based on investigation of security problems in XML Web Services, and interviews with security experts. The result of this study is presented as possible attacks on XML Web Services.

Key words:
Web Services, SOAP security, XML security, possible attacks, vulnerabilities

1. Introduction

Information technology has changed the way commerce is conducted around the world. The Web is different today than it was just few years ago. Modern technology is setting the rules for how a company does business and how it communicates with customers and others in the new market places. Nowadays, it is used for application-to-application communication. To improve the efficiency and flexibility of Web communication, Web Services technologies have been developed.
E-business applications attract attackers that can manipulate the back end of an application where all the personal data is stored. This data often includes everything from credit card numbers to medical information. By using firewalls and/or Intrusion Detection Systems (IDS) companies are trying to protect themselves from attackers but they are unaware that their assets are exposed even through firewalls and IDS's. Firewalls isolate an organization's network system but allow two TCP ports remain open - port 80 for HTTP and port 447 for HTTPS[4] [39]. These ports are used for communication to send and receive Web pages. Combining easy access with human readable data formats and integration standards Web Services open new opportunities for attackers. Web Services are still at a very early stage of development and, hence, there are no proper studies on vulnerability of Web Services. Security continues to be a top concern and along with increased information exchange capabilities comes the significant security considerations and challenges.
This paper describes what Web Services are from a security point of view, the vulnerabilities they have today, and the possible attacks, that might aim at Web Services. The purpose of this paper is revealing various types of attacks based on the SOAP implementation of XML Web Services over HTTP. The result of this study is based on investigation of security problems in XML Web Services and interviews with security experts.
The aim of the paper is not to provide any solution on how to prevent Web Services attacks, but to emphasize what kind of security problems and attacks exist.

2. Web Services

Web Services technologies are increasingly deployed in business organizations to achieve their inter-system collaboration. Web Services promises large benefits like productivity, efficiency, and accuracy. Helping applications communicate with each other is the focus of Web Services. Incorporated into business networks, Web Services can support both internal and external business operations including E-commerce, finance, manufacturing, Supply-Chain management and Customer-Relationship management [21].
Many business organizations find Web Services as the crucial key to achieve so-called 'Real Time Enterprise', a service that simplifies information processing by using

[1] SOAP- Simple Object Access Protocol
[2] XML- extensible Markup Language
[3] HTTP- A client/server protocol used for the WWW for the exchange of HTML documents

[4] HTTPS- A variant of http used for connecting to HTTP servers through SSL

Manuscript received August, 2005.
Manuscript reviced September, 2005.

appropriate technologies. An enterprise can shorten the time it takes to deliver business information and make the relevant information available to business organizations. The shortened time can cause a quick response to significant business events, which lead to faster and better business decisions. However, the enterprise needs to have many communication channels in order to share real time information across different devices, platforms, and locations, which can be easily achieved by Web Services. Furthermore, in contrast to the high cost of private networks, Web Services can lower the cost of the system integration [21].

The adoption of Web Services enables business organizations to improve business process efficiency by reducing cost and time and to gain expansive business opportunities. The application of integrated communication makes it possible for business organizations to gain a wider group of collaborative business partners, customers, and services, which eventually brings them in new competitive service markets [21]

Web Services are not tied to a single protocol. Any Internet-based transport protocol such as SMTP (Simple Mail Transfer Protocol), HTTP (Hypertext Transfer Protocol), and FTP (File Transfer Protocol) can be implemented to exchange a Web Services request. Generally, Web Services are services offered via Web [1].

Using the SOAP protocol over HTTP sends a request to a service at a given URL. When the service receives the request, it processes the request, and returns a response. A company can be the provider of Web Services and also the consumer of other Web Services. Web Services have many advantages, such as:

- Web Services are platform- and programming language-neutral, and communications mechanism-independent [13].
- The services are modular, which allows users choose what technique they will implement [13].
- Web Services expected to reduce complexity since they incorporate encapsulation. All components in Web Services are different kinds of services [13].
- Based on XML messages, Web Services have flexible models for data interchange [13].

The main disadvantage with Web Services is that they are new. The Web services are not proven technology and they lack a clear understanding of what Web Services really are [13].

2.1 Web Services Technologies

Web services are characterized by interface and business protocols. SOAP, WSDL [1] and UDDI [2] support Web Services and are nowadays the core of Web Services. WSDL and UDDI are used for discovering of the service and the design of the implementation.

SOAP, UDDI, and WSDL are built using XML and various Internet protocols for example HTTP. Use of SOAP, WSDL, and UDDI in Web Services support systems using different domains and architectures in cooperation to implement business functions. The relationship between Web Services, SOAP, UDDI and WSDL is presented in Figure 1.

The foundation of Web Services, Extensible Markup Language (XML) has syntax flexible enough to enable the definition of service description language and protocols, and is a platform-independent. XML is widely adopted and a commonly accepted standard used for describing the different aspects of a service.

The SOAP communication protocol defines a set of rules for encoding information and a routine to represent RPC (Remote Procedure Calls) and responses. SOAP supports the exchange of information in a decentralized and distributed environment based on XML. XML and SOAP allow all kinds of systems to communicate with each other. Every SOAP message has a sender, an ultimate receiver and an arbitrary number of nodes, also called intermediaries, which process the message and route it to the receiver. The body of the message contains the information sent to receiver. The header contains any additional information, i.e. the information that can be processed by intermediate nodes.

In processing messages, the intermediate SOAP nodes that SOAP header elements are targeted at, can play one or more roles. These roles, defined in SOAP specification, are: *next, ultimateReceiver or none* [1], [13].

Every node receiving a message is the next one in the chain of nodes processing the message and if the header is targeted at the next role, every node receiving the message can process it. If the header is targeted at the ultimateReceiver role, then the ultimate receiver processes the message. If the targeted role is none it means that no receiver processes the element [1], [13].

A message body does not have a role associated to it but the message body is targeted at the ultimate receiver, who must understand the body. In SOAP, there are no requirement to return a response for a message received. However, the message body can be divided into request message body elements and response message body elements [1], [13].

[1] Web Services Description Language

[2] Universal Description, Discovery and Integration

UDDI defines data structures and API's (programming interfaces) for publishing service descriptions in the registry and serves as a mechanism for discovering where specific Web Services are provided and who provides them. The information contained in UDDI is categorized in white, yellow and green pages by OASIS (Organization for the Advancement of Structured Information Standards) [33]. In the UDDI registry, a Web services description contains information about business information, service information, binding information and information about specifications of services. This information is described by different entities [1]. Schematic view of UDDI registry is presented in Figure 2.

WSDL (Web Services Description Language) describes service interfaces and specifies what data must be provided, but also what will be returned. WDSL is text-based, machine-generated and machine-processed language, usually, used during design and development of Web Services application.

In WSDL, specifications are XML documents that describe Web Services and service interfaces. WSDL specifications are characterized by an abstract part and concrete part, where WSDL documents describe logical and concrete details of Web Service.

In defining an abstract part of WSDL interface Alonso and Hartman [1], [13] points out four steps:

1) Identify and define all the data structures that will be exchanged between the applications.
2) Define messages by dividing the messages into portions characterized by name and by type.
3) Define operations[1] with four basic operations: one-way, request-response, solicit-response, and notification.
4) Group operations into port types, which define the operations and messages sent or received.

The above definitions are considered abstract because no implementation-specific information is specified. For example, the exact set of port types that service implements, the protocol used to transmit the messages is not given, nor is the encoding used for the data.

The concrete part describes aspects of the services that are determined by the service provider and contains bindings, services, and ports specifying all the information missing in the abstract part [1], [13]. Discussing the concrete part of a WSDL interface Alonso [1], Hartman [13] and other authors points out three so called constructs:

1) Interface bindings used for specifying the encoding for the messages and the underlying protocol bindings for all operations and messages in a given port type.
2) Ports for specifying the address at which the service is available.
3) Services for specifying the ports at which the service is available [1].

With the information stored in the concrete part, users know all the ports that are implemented as a group and the network address at which the functionality of a port type is implemented but also what protocols to use. The users also know about how to structure XML messages to interact with services, and what to expect when contacting the service. Figure 3 presents the components of WSDL document.

There is no centralized middleware in Web Services and, therefore, when invoking the service it is necessary to specify the address (URI) of the service and the transport protocol (HTTP). This information makes it possible to construct a client that invokes operations offered by a Web Service [13]. Using XML, SOAP, WSDL and UDDI enable faster and cheaper enterprise application integration, supply chain integration and distributed development. They support Internet-based service distribution models and makes communication between applications more easily.

2.2 Web Services Architecture

Web Services solve cross-platform interoperability issues by allowing services to work across different environments. Web Services follow a request/response model of interaction. A primary purpose of Web Services architecture is to integrate different Web Services and to open up the corporate network to the external world. The fact is that Web Services are the way to expose internal operations so that they can be invoked through the Web.

Web Services architecture is based on three components: the service provider, the service registry and the service requestor [1].

Service provider creates Web Services and defines an interface for invoking them. Then the service provider will publish the service descriptions in a service registry. Service registry (UDDI) uses the information included with the service description to catalogue each service The service requester makes a service request, which is asks the service registry to find it. The Service registry searches for the requested service and replies with a service description, indicating where to locate the service and how to invoke it. The service requester binds to the service provider by invoking the service [1].

The Figure 4 illustrates how a service discovery takes place in a Web Services environment.

Hartman [13] divides Web Services so called development cycle into three phases: publish, find, and bind phases.

During the publish phase the organization decides to offer a Web Service. When the application is developed and

[1] Operations are also called interactions or transmission primitives

available as a Web Service, the organization describes the interface to the application so the users can understand how to access it. This description can be in a form of WSDL that can be understood by Web Services development tools. Once services have been described they must be made available to organisations that are interested in using them.
During the find phase requestors can search in UDDI register for possible providers and learn about the organization that offers the service, the service offered, and the interface to the service.
The last phase in the development cycle it the bind phase. After the decision to use a service a service requestor implements the service interface.
Many traditional systems architectures can be characterized as tightly coupled. Web Services architecture is dynamic and loosely coupled. Therefore, when designing Web Services, security aspects must be considered because Web Services require high security.

2.3 Web Services Security

The number of enterprises and organizations that discover the strength of Web Services increases. Applications based on Web Services provide broad connectivity across a multitude of back-end systems. Unfortunately, the Web Services can challenge privacy and security of the corporation by opening a pipeline to its most valuable information assets, which can be used by attacker [13].

Data flows across the Internet through HTTP, which makes it possible to transmit a lot of data in a short time. Everything across HTTP is transmitted in plain text. Sensitive information transmitted across the Internet in clear text attracts attackers, who can use a network-sniffer or other tools to eavesdrop[1] on the data. There are several other mechanisms attacker can use to construct spoofing, redirection, or malformed URL attacks. Redirection attacks, for example, can cause password capturing or other security-related identity compromises [10].
Information security is a concern for most business. Web Services are targeted at distributed applications that allow business across Internet. Web Services are open and interoperable by the design. The openness that Web Services exchanges provide can cause a security breach. Many protocols are creating security problems and openings for attackers. The attackers can gain access to the data within the transactions and retrieve confidential data, and even steal sensitive information like credit card details. Unauthorised access to content, malicious modification of content and/or acceptance of incorrect or wrong information or interruption may cause a security threat to an organisation's information system [10].
According to Walid Negm[2]. The WS-Security, by itself, does not ensure security nor does it provide a complete security solution. WS-Security is a building block used in conjunction with other Web service and application-specific protocols to accommodate a wide variety of security models and encryption technologies.
Many security experts have discussed security problem regarding XML Web Services, but most of them are focusing only on technologies based on trust mechanisms such as XML signature, XML Encryption. Lack of understanding the Web Services security risks prevent many organisations from implementing Web Services [10].
Security mechanisms such as identification, cryptography, authentication, authorization, data integrity and confidentiality play an important role in providing basic level of Web Services security.

- Identifying the users of the Web Services makes it possible to track the customers and services for them.
- In Web Services cryptography is used to protect sensitive data, to ensure the integrity of messages.
- Authentication is used to identify parties that are establishing a Web Service connection. Service requestor needs to be authenticated by the service provider before sending information and vice versa.
- Authorization is used to decide what an authenticated user is allowed to do, what information users and applications can access, and which operations they may perform.
- Data integrity ensures that no one has tampered with a message after it was initially created. To provide data integrity XML encryption and XML signatures are used. The goal of digital signatures is to ensure the receiver of a message that the content have not been modified, and that the message is sent from original sender.
- Confidentiality ensures users that their data is protect from unathorized users. Confidentiality is of top priority and that the messages have isolation when engaging transactions within Web Services.

To maintain secure Web Services, all these security mechanisms should be combined and used with additional levels of security. An additional level of security is application level security that can secure transmission down to each part of the message.
For the security at application level, Galbraith [10] discusses four security concepts: Public Key Cryptography (PKC), SOAP security extensions, digital security and XML digital signature.

- Public Key Cryptography that used for web-based authentication and data encryption but also for generating and exchanging digital signatures enabling a secure

[1] Information about eavesdropping is given further ahead.

[2] Walid Negm Vice President Forum Systems Inc.

exchange of information. PKC ensures data integrity, user authentication and data confidentiality.

- SOAP security extensions are headers that are optional part in the SOAP message in an XML document. By using HTTP as transport and XML as payload for sending and receiving messages, SOAP is running on different platforms. SOAP does not require strong connection between client and the server. To define document types SOAP uses the W3C XML Schema [1], [13].

SOAP exchanges information, using messages in the form of SOAP envelope. Containing all the information to be sending each envelope is divided on two parts: a header and a body. SOAP messages may or may not have a header – it is an optional part of a SOAP message, but all SOAP messages must have a body, which is mandatory. Message body may not be modified unlike the message header [1], [13].

- Digital security ensures the integrity of operating system components, drivers and programmed objects.
- XML security focuses on message security [10].

Because of Web Services interact with different systems, devices, platforms, and locations it is difficult to detect unauthorized activity. Hence, there is no complete or absolute security, but security mechanisms slow the attackers down.

2.4 Web application attacks

As many security experts point out Web applications attacks should be taken in consideration. Many of Web application attacks might change their target on XML Web Services. Some of attacks are new and were created when Web Services were developed. To be able to identify Web Services attacks it is necessary to understand the targets and characteristic features of Web application attacks. Many of Web application attacks that can change their target are described bellow.

2.4.1 Input validation

Input validation attacks target any modifiable data parsed by the server. The server performs some function that relies on the value of the data to be within some predetermined type of range [19]. This group includes buffer overflow attack, cross-site scripting, IP-spoofing, SQL injection, network eavesdropping, dictionary attack, and data tampering.

2.4.1.1 Buffer overflow

When performing a buffer overflow attack, an attacker put a larger amount of data than expected into program variable, and by doing so, it can execute arbitrary code with the privilege of the running user- usually root. The amount of memory, reserved for the operation becomes smaller than the amount of data written to the memory. As result the extra data goes somewhere undesirable. Poorly written code, such as a program that inserts data into a buffer and does not check the size of the data being inserted, often invites buffer overflow attacks [17].

2.4.1.2 Cross-site scripting

Cross Site Scripting (also known as XSS or CSS) vulnerability is caused by the failure of a site to properly validate user input before returning it to the client's web-browser. The essence of cross-site scripting is when an attacker causes a legitimate web server to send a page to a victim's browser that contains malicious script or HTML chosen by the attacker.

The attacker inspects applications and chooses the one that does not filter the input and at which the user is authenticated. The attacker inserts a malicious code in the request, and this will be returned to the victim by that application. The malicious script runs with the privileges of a legitimate script originating from the legitimate web server. While the script runs the attacker can perform the attack on behalf of the user. The majority of XSS attacks rely on <scripts> tags. An XSS can cause significant damage by unknowingly distributing passwords and credit card numbers to unreliable sources. XSS attack targets the users of the application and not the application itself. An XSS attack sets up, for example, a Trojan horse in the victim's web browser. The browser does not know that the code is not legitimate, because the script code is downloaded from a trustworthy site. The Trojan horse is often created by client-side languages such as JavaScript, Java, VBScript, ActiveX and Flash, or takes advantage of a known vulnerability in the browser [19], [28], [32].

2.4.1.3 SQL Injection

SQL injection attack can be prevented only on application layer. The SQL injection attempts to manipulate the application's database by using raw SQL statements. The attacker can execute arbitrary commands in the database using this attack. First the attacker establishes some sort of indication regarding errors in the system and tries to identify errors. The attacker can continue with the next step and locate errors that are a result of a manipulated input. The attacker tests for vulnerabilities to exploit through, for example, entering single quotes, enumerating privileges or evaluating returned information. The attacker can choose different means to perform the attack [16], [19], [31]. The attacker can then proceed to perform the injection. Turning a SQL injection string into a valid SELECT statement will probably require some changing before the database accepts it. If SQL statement does not follow the proper structure of an SQL query, a syntax error will be generated and displayed in error messages. If

the application displays the result of the SQL query, the attacker can manipulate the statement to return other information.
Manipulating a SELECT … WHERE statement can be useful in bypassing an authentication mechanism or retrieving a value from the current table of a database.
While the basic SELECT statement is constructed to retrieve one or more rows from a single table, statement created with UNION allows the attacker to group multiple queries into a single result. However, the UNION keyword gives an attacker opportunity to create queries that retrieves other fields from the same table or fields from different tables. The attacker can access all the tables in the system and retrieve information which is not normally supplied by an application, such as credit card numbers or other sensitive and private information [16], [19], [31].

2.4.1.4 IP-spoofing

Performing IP-spoofing attack an attacker fakes IP address to deceive receiver to believe it is sent from a location that it is not actually from. If attacker gains access to the network with a valid IP address, he/she can modify, reroute, or delete data. The attacker can gain access to sensitive information or take control of the "victim" computer by installing, for example, backdoor program [3].

2.4.1.5 Network eavesdropping

Network eavesdropping attack occurs when an attacker gained access (possibly through IP spoofing or Man-in-the -middle attacks) to the data path to the specific network. Performing this attack the attacker can capture traffic and obtain usernames and passwords. Passwords and usernames that are passed in plaintext from client to server allow the attacker to collect sensitive data that passes between two active nodes on a network by eavesdropping the connection between the two nodes. The victim usually knows nothing of the attacker's presence, or that he/she, i.e. victim is being monitored [11].

2.4.1.6 Dictionary attack

An attacker systematically tests all possible passwords to perform dictionary attack. The attackers goal is to obtain passwords. To perform dictionary attack means trying every word in the dictionary until matching password is found. Most password-based authentication algorithms are vulnerable to dictionary attacks [3].

2.4.1.7 Data Tampering

Data tampering occurs when an attacker changes or modifies legitimate data, while it passes over the network. Any data sent from the client-side may be manipulated by an attacker. Successfull attack results in tampered data that reaches origin destination [16], [19], [32].

2.4.2 Denial of Service (DoS) attacks

In Denial of Service, DoS, attack the attacker's goals are to reveal information that he can use for crashing the Web application process. DoS may disable the users' computer or network where the attack can, for example, prevent data exchanging between two sites. Performing DoS the attacker may attack a router, firewall, or proxy server with the goal of making them unusable. By using proxy server, for example, an attacker can redirect his malicious traffic for own benefit. The effect of the Denial of Service attack is to prevent the service-providing computer from being able to provide the service [3], [32].

2.4.3 Session Management attacks

Weak session management results in vulnerabilities in the application that leads to attacks on session management. During the session management attacks, the attacker is able to manipulate a session token, which can be a cookie, URL parameter, session ID, or any obfuscated value that is passed between the application and browser. Analysing tokens gives the attackers an opportunity to identify vulnerabilities in the application [16], [19].

2.4.3.1 Session replay

Successful performed session hijacking attack opens an opportunity for attacker to perform session replay attack. The attacker bypasses authentication by sniffing URL parameter or Cookie value. Using the hijacked token attacker replays that token to application. By replaying a request, attacker gain access to the system under a false identity, i.e. under identity of legitimate user [16], [19].

2.4.3.2 Man-in-the-middle

When an attacker intercepts messages between two hosts i.e. client and server the Man-in-the- middle attack can occur. Both client and server assume that they are communicating with each other, but the communication between the sender and recipient is actually flowing through the attacker. The attacker is free to modify the content of the messages and sent them to the original recipient. What happens is the recipient receives the message that he/she thinks comes from the sender and acts on it. Then recipient sends the message back to the sender, which goes through the attacker. Unfortunately, neither sender nor recipient knows that they have been attacked [8].

2.4.4 Parameter tampering

The aim of parameter tampering attacks is to modify parameters sent between the user and the application.
An attacker may change the URL parameters or mode parameter. The parameter tampering includes Query string manipulation, Form field manipulation, HTTP header manipulation [32].

2.4.4.1 HTTP header manipulation

HTTP header consists of control information that is passed between the client and server. HTTP header request sends from clients and HTTP header response sends from server. Web browsers do usually not allow header modification, but an attacker can write his/her own program to perform the HTTP request [38].

2.4.4.2 Form field manipulation

Hidden form fields are fields added to an HTML form, which are not displayed in the client's browser. They are sent back to the server in plain text using the HTTP Post protocol, when the form that contains them is submitted. An attacker could manipulate both pre-selected (check boxes), free form, and hidden form fields with purpose to submit desirable values. This could be done by saving the page using "view source", "save", editing the HTML and re-loading the page in the web browser [11].

2.4.4.3 Query string manipulation

When a user makes a request, which can be of several different types (Get, Post, Head, Put, Trace and others), a HTTP command associated with a method that tells the server which type of action has to be performed, specifies. Get and Post methods are most commonly used methods. The Get method is designed for getting information and the Post method is designed for posting information. When a URL directory is typed in a browser the Get method is used. The Get method can include some of its own information, which passed as a sequence of characters appended to the request URL in what 's called a query string. Users can manipulate the query string values, because as it was mentioned above they are displayed in the browser's URL address field [11], [32], [19], [38].

3. Result from the study

In this section, we present the result of an investigation of Web Services together with interviews through e-mails with a couple of security experts. The result has been categorized as of possible attacks on XML Web Services, which is presented in Table1.
Breaking up the typical application paradigm into SOAP with loosely coupled endpoint applications, security becomes an important concern.
Web Services traffic flows through port 80 and 443, which simplifies the traffic between multiple entities, but it also creates a new security problems. As was mentioned earlier, the SSL is insufficient when it comes to Web Services security. Nonetheless, the SSL was not designed to secure Internet-based Web Services [26]. Web Services allows corporate resources to be accessible to outsiders, i.e. uninvited parties, e.g., business partners, suppliers. These uninvited parties can result catastrophic system failures and data loss [13], [19], [16].

Web Services security must be flexible enough because of different Web Services users have different security concerns. According to Hartman [13] and other security experts, neither firewalls nor security technologies based on PKI (Public Key Infrastructure) are useful for Web Services security, because firewalls lets HTTP traffic through Web Services request via HTTP pass through them. Most firewalls can recognize SOAP messages but view it as well-formed HTTP traffic. Security technologies based on PKI are inadequate for securing chains of applications connected by Web Services. The modular nature of Web services presents a security challenge, says Pete Lindstrom[1].
Many authors and security experts agree on that it is very likely that attacks on Web Services will be similar to the ones on Web application but more sophisticated.
The problem is that the XML messages contain, in a plain text form, a concentrated set of data interesting for attackers. Attackers have more information available through WSDL files and UDDI entries, which provide detailed information enabling attackers to gain entry. Messages have an XML format, which clearly show the data elements.
By using XPath queries attackers can extract a complete XML document. XML Path Language (XPath) is a query language that searches for, locates and identifies parts of XML document. XPath is used to identify the elements of an XML document to which the digital signature applies. The XPath is used to query XML databases. XPath models an XML document as a tree of nodes. To locate and identify nodes in an XML document, path expressions are used. Location path can be absolute or relative and consists of one or more location steps, which are separated by the slash. The XPath contains a function library, with Node Set, String, Number and Boolean Functions and

[1] Pete Lindsrom research director at Spire Security

supports four types of expressions, i.e., Numerical, Equality, Relational and Boolean Expressions [13], [41].
The XPointer[1] is built on top of the XML Path Language and uses XPath expressions to identify the particular point in XML document or part of an XML document that an XLink[2] links to.
Web Services require a security that offers consistency to all applications and services. Openness of Web Services contradicts security. On the other hand Web Services is simple and easy to implement, drastically reducing cost and time of development.

3.1 SOAP security and security problems

SOAP does not define any security mechanism for Web Services [1]. SOAP interfaces are software APIs that can expose a lot of functionality through port 80, for example, different critical operations.
As it was mentioned before, SOAP security extensions are headers, which are optional part in the SOAP message in XML document.
SOAP headers provide security information and protect the individual SOAP messages that are transmitted. A security element of the SOAP message header targets at a specific role, where each role can have at most one security element, but security header element can be presented many times in a SOAP message, i.e. there can be multiple security elements in the header.
Message security information targeted at different receivers must appear in different security header elements. If security header element does not target a specific actor, i.e. receiver, it can be consumed by anyone. Security element contains all security information relevant to the role, but also can contain several types of sub elements such as: UsernameToken KeyInfo,, referenceList, SecurityTokenReference, BinarySecurityToken, Signature, and EncryptedData [1], [10], [13].
According to OASIS [33] various types of threats can target SOAP message, for example, modification of SOAP message. SOAP header that can be included in the SOAP message might be vulnerable. If an attacker obtains SOAP message e.g. an implementation error, he/she can modify data, delete header instructions and insert new header instructions.
The body of the SOAP message contains the information the sender sends to receiver. It is possible to guarantee that the intended receiver gets the message, and not an unauthorised third party.

[1] XPointer -XML Pointer Language
[2] XLink - XML Linking Language. XLink allows links in XML documents in order to create and describe links between resources

This is supported by encrypting the whole message, i.e., from point to point. However, if the header contains some information about SOAP server, for example details of the server location, attackers can gain access to the information that could lead them to guess about the function of the application and gain the information about the SOAP endpoint [22]. According to a CERT Vulnerability Note (VU # 736923) Oracle Application Server 9iAS installs with SOAP, allowed anonymous users to deploy and remove SOAP services. Consider that the attackers could create and deploy their own SQL statements. This could open the database behind the SOAP service for penetration and allow the attackers to insert a back door into the system [22]. There is another problem in SOAP messages, namely routing. It is impossible to specify which intermediaries should be visited by a SOAP message en route to its destination [1], [10]. If the attackers compromise one of intermediate stations they can perform man-in-the-middle attack.
Protecting and authenticating SOAP message from being disclosed or modified, XML encryption and digital signatures of body, headers or both of them are often used [10].

3.2 XML security

XML describes the structure of electronic documents by specifying the tags. XML and SOAP challenges Web Services security, because they are being decentralized in architecture, allowing connection across multiple organizations.
XML Signature defines the processing rules and syntax to wrap message integrity, message authentication, and user authentication data inside an XML format.
XML documents easy access contributes to the lack of security in XML. Without knowing XML, one can decipher information easily and pick out sensitive information such as: name, address, and credit card number. This is the reason why XML data is vulnerable. While this information sits on either end of a business transaction, it is left completely unsecured, exposing critical data to theft and tampering [26]. SAML (Security Access Markup Language) is a promising standard that encodes authentication and authorization information in XML format. The SAML is an XML vocabulary that defines the syntax necessary to exchange authentication information between applications. SAML can be used to pass credentials off to multiple systems and, thus, can be used for single sign-on solutions.
Two techniques, broadly used to secure XML message are XML encryption and XML digital signature.
The XML encryption enables encryption of an entire XML document or specific parts of XML document. For example, it is possible to encrypt a complete XML file or

any single element of it, already encrypted element, non-XML data, only the contents of an XML element.
An element or contents of an element is the smallest part that can be encrypted. The only required element <CipherData> contains the encrypted content and can be represented in two ways: either an encrypted content stored in the XML document or at an external location and referred from a child element of <CipherData>. The <EncryptionMethod> element is used to specify the encryption algorithm and the key size. The key information provides how to decrypt cipher data and is stored in <KeyInfo> element [10], [13].
With XML encryption, the data is encrypted instead of the plain text data in the XML document where the encrypted data is also identified. When digital signature is used the signed data is not replaced but an additional structure is created.

XML digital signature provides flexibility to sign only specific part of XML document and allows attaching multiple signatures that apply to different parts of the documents. An XML signature consists of two required elements SignedInfo and SignatureValue. SignedInfo contains the information about the data object that is actually signed and SignatureValue contains the digital signature value that is an encrypted digest of the SignedInfo. There are also two optional elements in an XML signature, i.e., KeyInfo and Object. KeyInfo provides the information needed by the receiving application to validate the signature. Object contains any other information to support the signature.
There are different algorithms used for XML signature among which message digest (SHA-1[1], MD-5[2]), message authentication code (MAC [3]), but also signature and transform algorithm [10], [13].
While XML encryption, XML digital signature and different XML digital signature standards are important techniques to protect XML documents an examination of them lies outside the scope of this investigation.
XML signature and XML encryption solve many problems, but they do not solve all the problems. XML signature involves many algorithms and the strength of the signature depends on, for example, the used digest algorithms and the strength of the key but also the strength of authentication [10].

[1] SHA-1 – Secure Hash Algorithm produces a 160 bit-long digest
[2] MD-5 – Message Digest Algorithm produces a 128 bit-long digest
[3] MAC – Message Authentication Code authenticates the source of a message

3.3 Analysis

According to Walid Negm, Vice President, Forum Systems Inc. [26] the main difference between Web applications attacks and Web Services threats is that Web application threats target user-to-machine applications while Web services threats target machine-to-machine applications.
According to some security experts the three-tiered architecture behind Web applications makes them vulnerable to attacks based on bad programming habits like SQL injection but also makes them less vulnerable because logic is hidden beneath a presentation layer that often does some filtering. That means that the presentation layer provides a very restrictive calling syntax (HTTP POSTs with string arguments), and therefore the range of attacks is limited. In addition, the Web applications are typically built to handle a much greater load, which makes the denial of service harder.
Web services provide direct access to back-end systems, which are usually very valuable. Security experts agree about the fact that the Web Service transactions tend to be of much higher value than web applications and if a CRM[4] or ERP[5] system is attacked with, for example, a DoS attack, the business could lose millions of dollars.
According to some security expert there are some web application attacks that are not applicable to Web Services. Web services are layered on top of HTTP and rely on URI addressing. To be on the safe side, security experts recommend considering traditional Web application vulnerabilities. At the same time, they point out that attacks like cross-site scripting are attacks that specifically focused on the browser being the client. In Web Services, the custom application is the client.
There are possibilities for attackers to perform an attack if they are attacking before the message has been encrypted and if there are vulnerabilities in XML encryption and XML signature, which attackers can use at their advantage. Some of security experts say that if an attacker would compromise a client, for example, if the attack is from an employee within the company, i. e. internal attacks the attacker would have access to the private key used to encrypt a message. With the use of the private key, the attacker can send "valid" messages that are actually malicious. Some of them consider this to be a question of trust and recommend specifying the clients that are trustworthy and use authentication to verify that a message is from one of these clients.
Digital signatures alone do not provide message authentication. The attackers can record and resend signed message. In other words, there is a possibility for attackers

[4] CRM- Customer relationship management
[5] ERP – Enterprise Resource Planning

to perform replay attacks. Unfortunately, XML encryption and XML signature do not solve all Web Services security problems. For example in signed encrypted document, the digest value of a signed resource appears in clear text, which can lead to plain-text-guessing attacks. Recursive processing allowed by XML encryption specification can cause denial-of -service attacks [10].
The main use of encryption is to avoid third parties from viewing the data. An XML gateway can be configured to receive such encrypted data, decrypt it and validate that there is no malicious code inside. Walid Negm Vice President Forum Systems Inc. recommends following solution. The solution is to tear apart the SSL session at a certain point in the network path and inspect the SOAP messages including scanning binary attachments for viruses. If the SOAP message is encrypted by using W3C XML Encryption, the Web Services Firewall may also have to decrypt the message to check for threats.

OASIS [33] points out some security considerations regarding SOAP messages. Some of these are protection of security tokens (integrity), proper use of digital signature and encryption, unprotected passwords using that can lead to dictionary attacks, replay attacks, man-in-the-middle attack, and plain text attacks.

To attack XML Web Services attackers search vulnerable components, i.e., the target in XML document, which is the input. Sometimes standards create some new types of threat. Consider that attackers send message that has thousands of Web Services security elements and imagine receiving and trying to validate several 1000 of signatures. In these situations, the standards make the system less secure rather than more secure.
Providing direct access to program logic, Web Services according to some security experts need to be written with great care and must be checked against malformed input to make sure it's data is not malicious, i.e. a number isn't much bigger then expected. Typically, developers tend to focus on getting the successful case working rather than worry about the hundreds of possible failure cases. Depending on what kind of service it is, the service could be vulnerable to, e.g., SQL injection and XPath injection.
As it was mentioned earlier, WSDL describes the interface used by Web Services. The WSDL often contains lots of information on how the service is used and how it can be implemented which makes it easy to design attacks and exploit possible failure cases. Extra functionality to computing environments providing by Web Services increases security risks.
Web applications attacks that may be applicable on XML Web Services and new possible attacks are presented below.

Identity attacks

Identity attacks consist of authentication and authorization attacks such as dictionary attacks, IP spoofing, data tampering, and message eavesdropping attacks. These attacks are old, i.e., Web application attacks that may change their target on XML Web Services, which is also argued by Lily [29].

Session attacks

Session attacks change the target on Web Services. Performing session attacks the attackers may capture messages or insert false instructions.

Replay attacks

Replay attacks may be a serious problem for organisations because different processes within organisation can be vandalized if an attacker success to perform this attack. According to Demchenko and Lindstrom [23], [30] the attacker captures encrypted message and sends it repetitively, which results in overloading of Web Service.

Man-in-the-middle attack

SOAP message passes multiple intermediate systems but it is impossible to specify the intermediaries that should be visited by SOAP message during the route to its destination.
Demchenko [23] points out that compromising one of intermediate station open possibilities for attackers to perform man-in-the-middle attack by inserting fake/bluff routing instructions so that message travels to malicious location and from there the attacker can send malicious instruction to original destination.

Parsing attacks

Parsing attacks including recursive payloads, oversized payloads, schema poisoning, are aimed on XML parser.

Recursive payloads

According to Lindstrom [30], attackers attempts to stress and break an XML parser by sending mass amounts of nested data. This opens an opportunity for attacker to create an XDoS attack against the XML parser.

Oversized payloads

Sending unlimited file size allows attackers according to overload the parser that may create and execute an XDoS attack. Lindstrom [30]

Schema poisoning

XML Schemas that provide formatting instructions for parsers when interpreting XML documents are used for all of the major XML standard grammars coming out of

OASIS[1]. Because these schemas describe necessary pre-processing instructions, they are susceptible to poisoning, says P. Lindstrom [30]. Attackers compromise XML schema and replaces it with similar, but a modified one. When XML schema is compromised, attackers can manipulate data processed by the application. The successfully performed schema poisoning may lead to XML Denial-of-Service attack [30].

Overflow attacks

Overflow attack is aimed at the service endpoint and SOAP engine through Web server. Lily [29] says that attackers send parameters longer than the program can handle, which can causes the service to crash. One example of a strange request is a username with more characters than expected.

Code attacks

Malicious code carried by XML messages passes through port 80. Code attacks intended to affect applications that run Web Services. Successfully performed Code attacks give the opportunity for attackers for example to extract the whole XML database.

SQL Injection

SQL Injection attack occurs when malicious SQL statements are inserted into XML in order to disrupt the back-end system, according to J. Lily [29]. Trying to force a SOAP endpoint, i.e., server to do something it wasn't meant to do. For example retrieve data it is not authorized to access, or even destroy data through SQL Injection and manipulate content within a SOAP message. This results in receiving endpoint and consumes excessive resources, i.e., buffer overflow, and crashes or becomes unresponsive.

XPath Injection

An XML document has no access control or privilege system associated with it [45]. By performing XPath injection there are possibilities for attackers to extract the whole XML database.

XDoS attack

Denial-of-Service attack is when an attacker is trying to prevent legitimate users from accessing a service. A single backend server receives more requests than it can handle.
The attackers flooding the service with a large amount of requests. When a legitimate user sends a request the service is unable to respond because of all requests that the attacker has sent. In traditional DoS attack the attackers goal is to disable a user computer or network. XDoS changes the target, and tries to take down an important service, so that legitimate users cannot use it. XDoS attacks consume resources by forcing a message to do useless work. The result from a successful XDoS attack is a programming error that causes a service to enter an infinite loop, consume all available resources and, ultimately, shut it down [29].

WSDL attacks

WSDL attacks include WSDL scanning and Parameter tampering. Performing WSDL attacks the attackers analyse and misuse WSDL information and tamper with parameters within WSDL documents.

WSDL Scanning

Web Services Definition Language (WSDL) describes logical and concrete details of Web Service. A WSDL document contains information that describes how to use parameters and information about details of methods, types of I/O and parameters of methods. Lindstrom [30] points out that through scanning the WSDL document an attacker may reveal sensitive information like types, messages, operations, port types, bindings, and guess other methods.

Parameter tampering

According to Lindstrom [30], attackers tamper parameters within a WSDL document in order to retrieve unauthorized information. If so, attacker can inject malicious code into XML parameters.

Internal attacks

Usually attackers are associated with an outsider. Well, the fact is that many attacks come from within organisations. According to FBI over 70% of theft happens from inside, behind the firewalls performed by someone that was trusted [26].
According to security experts, firewalls cannot provide any protection if attacks are performed within the firewalls by employees or former employees that have access to confidential information and are familiar with internal systems.
To minimise risks for attacks from insiders, security experts' argue that least privilege techniques should be used and only absolute necessary for the job access, should be granted.
It is not possible to provide total or absolute security. However, it is possible to identify the attackers' move in advance and this gives an opportunity for security experts to prevent many attacks that, otherwise, could cause destruction of many organisations.

[1] OASIS - Organization for the Advancement of Structured Information Standards

4. Further work

Security needs to be implemented in an early stage of an application design and/or system development process. This is needed due to the difficulties for application developers to sufficiently understand emerging technologies and their security mechanisms within a limited time and bring them in the process of Web application development. It is necessary to increase awareness and knowledge of what kind of security problems does exist in system development process, in order to prevent problems and remedy problems in early stage of system development process and prevent the security problems to occur.

5. Tables and Figures

5.1 Tables and Figures

The relationship between Web Services, SOAP, UDDI and WSDL is presented in Figure 1.

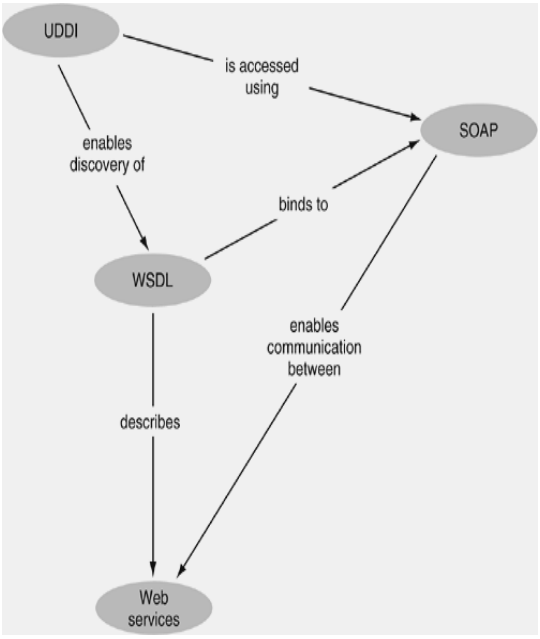

Figure 1. The relationship between Web Services, SOAP, UDDI and WSDL (Source: Erl. [24] Online documentation

A schematic view of UDDI registry is presented in Figure 2.

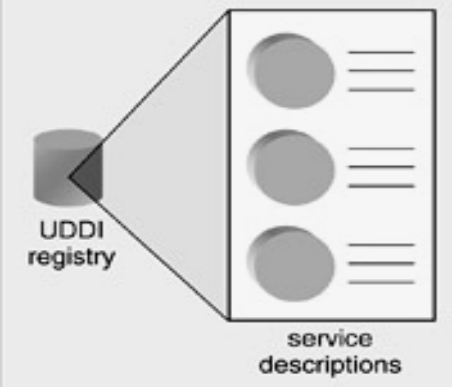

Figure 2. UDDI registry (Source: Erl [24] Online documentation)

Figure 3 presents the components of WSDL document.

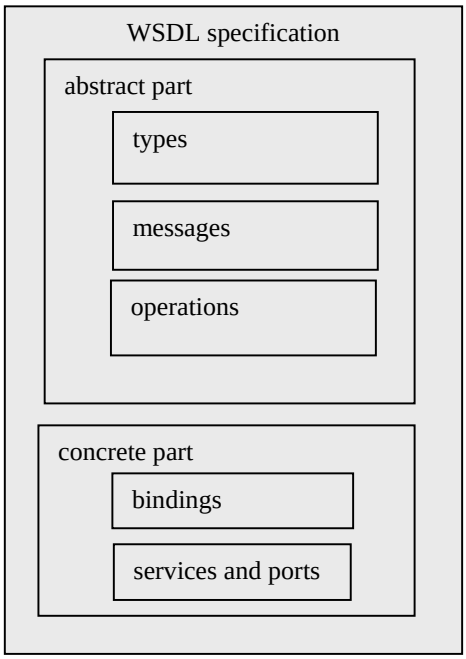

Figure 3. Components of WSDL document (Source: Alonso. [1] p. 167).

Figure 4 illustrates how service discovery take place in a Web Services environment.

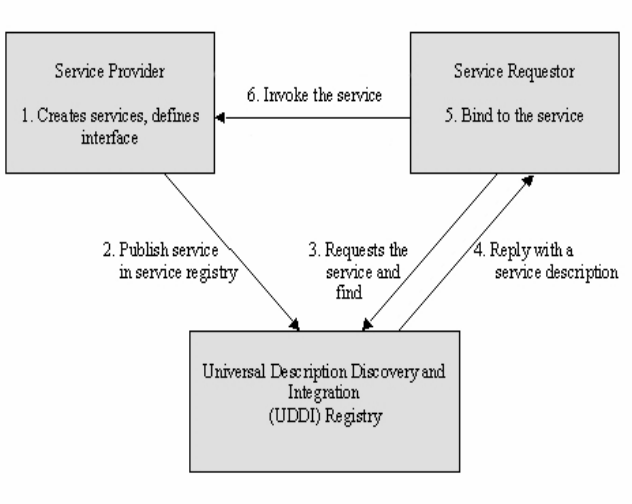

Figure 4. The Service Discovery Process

Table 1: An overview of possible attacks on XML Web Services

	Attacks and Defences		
Attacks categories	Attacks	Description	Countermeasures
Identity attacks	Dictionary attacks	By obtaining passwords, usernames an attacker acts as legitimate user and gains access to the service	Cryptographic technology: digital certificates, digital signatures, SSL.
	IP spoofing	By pretending to be a service an attacker entice a client to make requests to his directory	
	Message eavesdropping	An attacker gathering sensitive or other information through analysing Web Services messages	
	Data tampering	An attacker can modify legitimate message with illegal data. Success attack results in tampered data that reaches origin destination	
Session attacks	Replay attacks	To overload Web Service an attacker stoles messages and sends stolen message repetitively	Using message identifier or serial number control that message is used only once.
	Man-in-the-middle	An attacker inserts fake routing instructions so that message travels to malicious location. Then the attacker sends malicious instruction to original destination.	Cryptographic technology
Parsing attacks	Recursive payloads	An attacker attempt to stress and break an XML parser	Good XML parsing should be used
	Oversize payloads	Unlimited file size allows an attacker to overload parser	Check parser for abnormal conditions (large element and attribute names)
	Schema poisoning	An attacker compromises XML schema and replaces it with similar, but modified one	Schema validation
Overflow attacks	Buffer overflow attacks	This attack is aimed on SOAP engine through Web server. An attacker sends larger input than the program can handle, which can cause the service crash	Perform input validation

	Code attacks	SQL Injection	An attacker inserts and executes malicious SQL statements into XML	Input validation, exception control
		XPath Injection	An attacker forms SQL-like queries on an XML document using XPath to extract an XML database	User input must be sanitize Single and double quotes characters should not be permitted
	XDoS attack	XML Denial-of-Service attack XDoS	An attacker trying to prevent legitimate users from accessing a service by flooding the service with thousands of requests	Reject traffic from an address when it is clear that it has initiated an attack
	WSDL attacks	WSDL scanning	Through information revealed from WSDL file an attacker guesses methods	Should not have any leakage, SSL, only necessary methods
		Parameter tampering	An attacker tampers parameters in order to retrieve unauthorized information	Cryptographic technology
	Internal attacks		This attacks performs within the firewalls by an employee or former employee that usually have access to confidential information and are familiar with internal system of the organisation.	Least privilege techniques should be used and only absolutely necessary for the job access should be granted

Acknowledgment

The authors would like to express their cordial thanks to security experts for their valuable advices

References

[1] Alonso, G. Casati, F. Kuno, H. Machiraju, V. “Web Services: Concepts, Architectures and Applications”. Springer, 2004.

[2] Alvarez, G. Petrovic, S. “A new taxonomy of Web attacks suitable for efficient encoding. Computer & Security”, Vol.22(5), 2003, p435-449.

[3] Amrit, T. “Web Security”. Digital Press [tm], 1999.

[4] Awad,Elias M, Ghaziri, Hassan M. “Knowledge management” Prentice Hall, 2004.

[5] Bar-Gad, I. Reshef, E. “Web application security”. White paper, Perfecto Technologies, 2000.

[6] Connolly, T. Begg, C. “Database systems”. Addison Wesley 3rd edition, 2002

[7] Conre-Murray, A. “Emerging technology: Protect Web applications from abuse and misuse”. Network Magazine, July 2003.

[8] Fitzgerald, J. Dennis, A. “Business data communications and networking”, John Wiley and Sons, 2002

[9] Forristal, J. Shipley, G. “Vulnerability assessment scanners”, Network Computing, jan 2001.

[10] Galbraith, B. Hankisson, W. Hiotis, A. Janakiraman, M. D.V., P. Trivedi, R. Whitney,D. “Professional Web Services Security”, Wrox Press, 2002.

[11] Garfinkel, S. Spafford, G. “Web Security, Privacy & Commerce”, 2nd Edition. O’Reilly, 2002.

[12] Gollmann, D. “Computer Security”. JohnWiley &Sons, 1999.

[13] Hartman, B. Flinn, Donald J. Beznosov, K. Kawamoto, S. "Mastering Web Services Security". Wiley 2003

[14] Hunter, J. "Java Servlet programming", O'reilly 2nd edition, 2001.

[15] ITS Rapport, "Terminology for information security". 1994

[16] McClure, S. Shah, S. Shah, S. "Web Hacking: Attacks and Deffence". Addison Wesley, 2002.

[17] Scambray, J. McClure, S. Kurtz, G. "Hacking exposed: Network security secrets and solutions". Osborne/McGraw-Hill, 2001.

[18] Scambray, J. Shema, M. "Hacking Exposed: Web Applications". Osborne/McGraw-Hill, 2002.

[19] Shema, M. "HackNotes: Web security portable reference". McGraw-Hill/Osborne, 2003.

[20] Stallings, W. "Network security essentials: applications and standards". Prentice Hall, Inc, 2000.

[21] Clabby, Joe. "Web services: A business level executive overview". Online documentation, 2004. http://www.intel.com/cd/ids/developer/asmo-na/eng/20035.htm?prn=Y Date: January 2005

[22] Clewlow, C. "SOAP and security". White paper, 2002. http://www.qinetiq.com/home/markets/security/securing_your_business/information_and_network_security/white_paper_index.Par.0013.File.pdf Date: December 2004

[23] DemchenkoY. "Attacks on Web Services and Grids", White paper, 2004. http://www.uazone.org/demch/analytic/draft-grid-security-incident-02.pdf Date: January 2005

[24] Erl, T. "Introduction to Web Services technologies: SOA, SOAP, WSDL and UDDI", 2004 Online documentation http://www.informit.com/articles/article.asp?p=336265 Date: January, 2005

[25] Faust,S. "SOAP Web Services Attacks". White paper, SPIDynamics, Inc., 2003. http://www.ebusinessforum.gr/content/downloads/SOAP_Web_Security.pdf Date: January 2005

[26] Forum Systems.Inc, "SSL:Not enough for today's Web Services" White paper June 2002. http://www.forumsys.com Date: October 2004

[27] Gutiérrez, C.Fernández-Medina, E. Piattini, M. "Web Services security: Is the problem solved?" http://www.auerbach-publications.com/dynamic_data/3132_1887_Web%20services%20security.pdf Date: January 2005

[28] Jason R. "Cross-site scripting vulnerabilities". Technical report, Carnegie Mellon Software Engineering Institute, 2001. http://www.cert.org/archive/pdf/cross_site_scripting.pdf Date: December 2004

[29] Lily, J. "Tips and tricks: Web Services attacks and defenses", White paper, 2004 info@reactivity.com Date: December 2004

[30] Lindstrom, P." Attacking and Defending Web Services", White paper, 2004 http://www.spiresecurity.com Date: December 2004

[31] Maor, O. Shulman, "A. Blindfolded SQL Injection", Online documentation, 2003. http://www.imperva.com Date: December 2004

[32] Microsoft Corporation. "Improving Web Application Security Threats and countermeasures.pdf", 2003 http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/threatcounter.asp Date: January 2005

[33] OASIS. "Web Services security: SOAP message security 1.0." White paper, 2004. http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0 Date: January 2005

[34] Ollman, G. "Web based session management" http://www.technicalinfo.net/papers/WebBasedSessionManagement.html Date: January 2005

[35] Segal,O. "Web application forensics: The uncharted territory". White paper, Sanctum,Inc., 2002. http://www.sanctuminc.com/pdf/WhitePaper_Forensics.pdf Date: January 2005

[36] Shah, S. "Top Ten Web attacks". White paper, 2002 http://www.net-square.com Date: November 2004

[37] SPI Dynamics. "Complete Web Application security-Phase1-building Web application security into your developing process". Online documentation, 2002. http://www.spidynamics.com Date: December 2004

[38] The Open Web Application Security Project. "A guide to building secure web applications". Version 1.1.1. Online documentation, 2002. http://www.owasp.org Date: December 2004

[39] "The XML Security Gap: It's Bigger than you think". White Paper, March 2003 info@reactivity.com Date: December 2004

[40] "Web application attacks". Online documentation. http://www.onixnet.com/Understand%20Web%20App%20Attacks.pdf Date: January 2005

[41] W3Schools http://www.w3schools.com/ Date: January 2005

[42] http://www.w3c.org/ Date: January 2005

[43] http://www.oasis-open.org/home/index.php Date: January 2005

[44] http://www.cert.org/tech_tips/denial_of_service.html Date: January 2005

[45] http://www.watchfire.com/resources/blind-xpath-injection.pdf Date: 2005

Esmiralda Moradian, Degree of Master in Computer Science, Department of Information Science, Division of Computer Science, Uppsala University, Sweden. Moradian is working as a teacher and is doing research at the Department of Information Science, Division of Computer Science, Uppsala University. Moradian research interests include Data Security, Information security, Network Security and Web Security.

Anne Håkansson, Ph D in Computer Science, is a Director of Studies at the Department of Information Science, Division of Computer Science, Uppsala University, Sweden. Since 1993 Håkansson has been doing research within knowledge-based systems, especially, in the knowledge acquisition process by using graphic representation and visual modelling. Recently, the research has incorporate data and information security in the software engineering process emphasizing awareness and preventing attacks in the software engineering cycle. Håkansson's research interests lie in Data Security, Information security, Network Security, Software engineering, Knowledge-Based Systems, Knowledge Acquisition, Graphic Representation, Visualisation, Unified Modeling Language (UML), Artificial Intelligence, and User-Centred Design.