Chapter 2

Literature Survey on Web Services Security

The evolution of Web services has facilitated the integration of business processes running in different platforms scattered across different geographical locations of the world. Along with the benefits that Web services provide for online transactions, it also poses some security threats. This chapter provides a study on the security issues in Web services, the threats that they are facing today and the possible attacks that might be aimed at Web services.

2.1 Introduction

Service Oriented Computing (SOC) [2] is a platform independent and language independent computing paradigm. SOC helps in implementing and configuring distributed software applications in a manner that provides productivity and quality with service orientation. Services are simply means for building distributed applications, where the objectives are mainly on how these applications are built and how services should function together. Service Oriented Architecture (SOA) is the standard architecture for this new computing paradigm. It defines an interaction between software agents as an exchange of messages among service requesters and service providers. Requesters are software agents that request for a service. Providers are software agents that provide services. The basic SOA includes, service provider, service requester and service registry. The interactions in SOA involves publish, find and bind operations. A service provider could be an industrv. business or a company capable of providing services. The key requirement of any service provider is interoperability, security and performance [11]. A service requester also could be a company
or a business that is need of the service, where a service registry is a place, entity or a system that helps both service provider and service requester to discover each other.

Web service is the current technology of SOC. Web service technology implements SOA. The interface WSDL, registry UDDI and the message SOAP are built using XML and communicate via Internet protocols like HTTP, SMTP etc. XML is a platform independent language and a commonly accepted standard used for describing the different aspects of a Web service. XML and SOAP allow all kinds of systems to communicate with each other [27]. From the information exchange point of view, a Web service can be regarded as a simple mechanism to send and receive messages by a node [9]. Web service is a self contained, self describing, loosely coupled, re-usable software component that can be published, discovered and interacted via Internet protocols. Web services perform functions like transformation, storage and retrieval of data, aggregation, composition, orchestration etc. [10]. This chapter presents a latest review of research and development in Web services security.

2.2 Service Oriented Computing

Service Oriented Computing (SOC) [2] provides a better way to create a new architecture that trends towards autonomy and heterogeneity. Now-a-days, the current trend in the application space is moving from tightly-coupled systems towards loosely- coupled systems. The latest evolution in this new category of systems is a new paradigm, called Web services. The Web Service paradigm is the logical evolution from object oriented systems to the systems of services. As in object-oriented systems, some of the fundamental concepts in Web services are encapsulation, message passing and dynamic binding. Service Oriented Computing (SOC) is the new emerging paradigm for distributed computing and e-business processing that is changing the way software applications are designed, architected, delivered and consumed. Services are autonomous platform independent computational elements that can be described, published, discovered, orchestrated and programmed using standard protocols for the purpose of building agile networks of collaborating business applications distributed within and across organisational boundaries. Combined with recent developments in the area of distributed systems, workflow management systems, business protocols and languages, services can provide the automated support needed for e-business integration both at the data
and business logic level. They also provide a sound support framework for developing complex business transaction sequences and business collaboration applications. Adopting the SOC paradigm has the potential to bring about reduced programming complexity and costs, lower maintenance costs, faster time-to-market, new revenue streams and improved operational efficiency. However, before the SOC paradigm becomes a reality, there is a number of challenging issues that need to be addressed including among other things, like service modelling and design methodologies, architectural approaches, service development, deployment and composition, programming and evolution of services and their supporting technologies and infrastructure. The growing trend in software architecture is to build platform independent software components, called Web Services that are available in the distributed environment of Internet or Intranet. Architecture for service-based applications has three main parts: a service provider, a service requester and a service registry. Service providers publish their services on registries so that consumers can easily find the services and invoke them. A service requester can be a customer or a client or a person who need the particular service. The service registry process can be done through a basic protocol like UDDI (Universal Description Definition Integration).

### 2.3 Service Oriented Architecture

Service Oriented Architecture (SOA) is essentially a collection of services. These services are communicated with each other. The communication can involve either simple data passing or it could involve two or more services co-ordinating some activity. SOA allows a software programmer to model programming problems in terms of services offered by components to anyone, anywhere over the network. In other words, any application residing anywhere on any computer system would be able to interact with any service anywhere over the network. SOA defines an integration between software agents as an exchange of messages between service requesters (clients) and service providers. Clients are software agents that request the execution of a service. Providers are software agents that provide the services. Agents can be simultaneously both service requesters and service providers. Providers are responsible for publishing a description of the services that they provide. Clients must be able to find the descriptions of the services that they require and must be able to bind each other. The basic SOA is a relationship of three kinds of participants: the service provider, the service discovery agency and the service requester (client). The interactions involve publish, find and
bind operations. The service provider defines a service description of the service and publishes it to the service discovery agency through which a service description is published and made discoverable. The service requester uses a find operation to retrieve the service description typically from a discovery agency and then both the service provider and service requester negotiate each other by using bind operation.

In Web service models, **Service Provider** is the owner of the Web services. It holds the implementation of the service application and makes it accessible via the web. **Service Requester** represents a human or a software agent that intends to make use of some services to achieve a certain goal. **Service Registry** is a searchable registry providing service descriptions. It implements a set of mechanisms to facilitate service providers to publish their service descriptions. Meanwhile, it also enables service requester to locate services and get the binding information.

In Web services interacting modes, **Service Publishing** is to make the service descriptions available in the registry, so that the service requester can find it. **Service Finding** is to query the registry for a certain type of service and then retrieve the service description. **Service Binding** is to locate, contact and invoke the service based on the binding information in the service description.

### 2.4 Web Service

Web services are new concept that provides flexibility and connectivity between different systems. Web Services are using SOAP (Simple Object Access Protocol), a XML based protocol for communication over the network. A Web service is a software system identified by a URL, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by internet protocols. Web service is a loosely coupled software components that interact with one another dynamically via standard Internet technologies.

The Web services stack is as follows –

1. **Communications.** A set of network protocols helps to realize the network accessibility of Web services such as HTTP, SMTP etc.
(2) **Messaging.** The messaging layer provides a document based messaging model for interaction with Web services.

(3) **Descriptions.** The description or representation layer is for describing Web services.

(4) **Discovery.** The discovery layer is for locating and publishing Web services.

(5) **Processes.** The processes layer supports more complex interactions between Web services, which enable Web service interoperation.

To compose Web services, the following things are required. These are –

1) Quality of Service (QoS)
2) Security
3) Reliability
4) Cost

This chapter identifies the main security requirements for Web services and describes how much security requirements are addressed by standards for Web services security recently developed or under development by various standardizations bodies.

**2.5 Web Service Technologies**

The technologies that form the foundations of Web services are SOAP, WSDL and UDDI. The international standard of SOAP is available in [12], where as those of WSDL and UDDI are available in [24] and [25] respectively.

**2.5.1 SOAP**

Simple Object Access Protocol (SOAP) is used for communication among different Web Services. SOAP [12] messages flow from originator to an ultimate receiver through a SOAP message path. A SOAP message consists of Soap: Envelope which contains a soap: Body element and an optional soap: Header element. The soap: Header element may contain a set of child elements.
Lines 01-06 contain the document root element named soap: Envelope. The syntax and semantics for the soap: Envelope are defined by SOAP. Line 02 contains soap: Header which is the optional part of the SOAP protocol. Soap: Header contains information for the SOAP node, the processor of the SOAP message, how to process the SOAP message. Lines 03-05 contain an element name soap: Body which is a child of the soap: Envelope element. Line 04 contains the get Quote element which is a child of the soap: Body. SOAP messages are nothing but XML documents.

2.5.2 WSDL

Web Service Description Language (WSDL) is an XML based language for describing functional properties of Web services. It aims at providing self-describing XML-based service definitions that applications as well as people can easily understand. In WSDL, a service consists of a collection of message exchange end-points. An end-point contains an abstract description of a service interface and implementation binding. The abstract description of a service contains – definition of messages that are consumed and generated by the service (i.e. input and output messages and – signatures of service operations. The implementation binding contains information about the location of a binding, the communication protocol to use (e.g. SOAP over HTTP) for exchanging messages with the service [10].
WSDL is written in XML. It is used to describe Web services. It is also used to locate Web services. A WSDL document is just a simple XML document. A WSDL document describes a Web service using the following elements [24] –

- `<type>`. The data types used by the Web services.
- `<message>`. The messages used by the Web services.
- `<portType>`. The operations performed by the Web service.
- `<binding>`. The communication protocols used by the Web services.

The Services and Ports define the location of Web services.

- *Port/Endpoint*. The Port contains the location of a Web service and the binding used for service access. The Port defines the connection point to a Web service.
- *Service*. The service contains the Web service name and a list of ports. It contains the collection of related endpoints. More details are available in [24].

```xml
<definitions>
    <types>
        definition of types ------------
    </types>
    <message>
        definition of message -----------
    </message>
    <portType>
        definition of a port -----------
    </portType>
    <binding>
        definition of binding ----------
    </binding>
</definitions>
```

*Figure 2.5 (b): The Main Structure of a WSDL Document*
2.5.3 UDDI

Universal Description Discovery and Integration (UDDI) is an XML-based registry for Web services. It is a place where businesses register and search for Web services. It is a directory for storing information about Web services. UDDI communicates via SOAP. UDDI uses WSDL to describe interfaces to Web services. A UDDI registry service is a Web service that manages information about service providers, service implementations and service metadata. Service providers can use UDDI to advertise the services they offer. Service consumer/requester can use UDDI to discover services according to their requirements and to obtain the service metadata needed to consume those services [25]. It defines an interface for advertising and discovering Web services. The UDDI information model identifies three types of information. These are –

- **White Pages.** White pages contain general information such as business name (i.e. a service provider's name) and contact information (i.e. service provider's phone no.).

- **Yellow Pages.** Yellow pages contain meta-data that can be used to effectively use to locate businesses and services based on classification schemes.

- **Green Pages.** The green pages contain service access information including service descriptions and binding templates. A binding template represents a service end-point (i.e. a service access interface) [10].

In UDDI registry, a Web service description contains business information, service information, binding information and information about specifications of services [27]. More details are available in [25].
2.6 Security

The Web services security dimensions have been identified as, secure messaging, resource protection, negotiation of contracts, trust management and security properties. These dimensions encompass the security elements in a Web services environment which can be discussed broadly in the general security framework consisting of -

1) Authentication: Who is attempting to gain access. Authentication is concerned with the establishment of the proof of identities of entities in a system. The entity may be a user, a process or a service. Masquerading is a standard attack in authentication mechanism.

2) Authorization: Which resources is this requester allowed to access. Once an entity has been authenticated, the next issue is to ascertain which operations the entity is allowed
to do and on what resources. The authorization mechanism deals with granting and revoking privileges of authenticated entities.

3) **Confidentiality:** Can the data be read while in transit or storage. The issue of confidentiality specifies that only the sender and the intended recipient should be able to access the content of the message. An authorized person should not be able to access a message. It is achieved by encryption and decryption of messages. An encryption algorithm is used to convert plaintext into cipher text. There are two types of encryption in general use: symmetric and asymmetric encryption. In symmetric encryption, the decryption key is the same as the encryption key and in asymmetric encryption, the decryption key is not same as the encryption key. Eavesdropping is a standard attack in confidentiality.

4) **Non-repudiation:** Can a sender deny having sent a message. There are situations when a user sends a message and later on repudiates it. Repudiation may be on sending, receiving or on the time of sending or receiving the message as well.

5) **Availability:** Is this system vulnerable to Denial of Service (DoS) attack. The issue of availability states that resources, services should be available to authorized parties at all times. Denial of Service (DoS) is a standard attack on availability.

6) **Integrity:** Has the data or system been tempered with. When the content of a message is changed during transmissions than the integrity of the message is lost. Data integrity relies on mathematical algorithms known as hashing algorithms. A hashing algorithm takes a block of data as input and produces a much smaller piece of data as output. This output is sometimes called a digest of the data. If the data is a message, it is called a message digest. MD-5, SHA-1 and SHA-2 are the standard hashing algorithms.

7) **Auditing:** Is there a record of requester access to data. Security auditing is defined as involving the recognition, recording, storage and analysis of information related to security-relevant-activities. Audit records are intended to be examined to determine which security relevant activities took place and whom (which user) is responsible for them [28]. In computer security systems, audit trail is a sequential record of system resource usage. This includes user login, file access and also other various activities.
2.6.1 Authentication of Web Services

The requester, provider and registry in SOA need to be authenticated during composition, binding and execution of Web services. When a new service is composed by the service provider, the descriptions of that service are provided by the registry, and then the registry is to be authenticated by the requester. Web services may be vulnerable to man-in-the-middle attack, masquerading attack during composition, binding and execution. Various authentication techniques are discussed as follows:

1) **Single-Sign-On.** Once the end-user has been authenticated by its attributes login-id and password, it need not be authenticated again for communicating with the other services. Once the end-user sign on to a Web site and then a SOAP request is produced on the user’s behalf, the route among multiple Web services is established using user’s login-id and password. This functionality is known as Single-Sign-On (SSO) [15].

2) **Federated Trust.** Once the end-user has been authenticated using its attributes login-id and password, the route among multiple Web services may be established on the basis of their trust relationships. This mechanism is known as federated trust [16].

3) **WS-Trust.** To route among multiple Web services the trust relationships must be established among different Web services. The trust relationships among multiple Web services can be either direct or through the discovery agency.

4) **WS-Routing.** SOAP does not specify the actual message path along which a SOAP message is to travel. The message path consists of several intermediaries that a SOAP message will visit. SOAP needs to rely on the transport protocols and follow their message path models. Therefore, it is impossible to specify which intermediaries the SOAP message can visit when it travels to its destination. WS-routing enables SOAP path to be specified as a SOAP message header. It consists of a sequence of references to the intermediary SOAP nodes. When a message arrives at an intermediary node, the node will remove the reference thereof and send the message to the next node.
based on the specified order. An intermediary node can also incorporate new nodes or remove existing nodes to change the message path [7].

5) Multi-hoping.

![Diagram of Multi-Hoping]

**Figure 2.6 (a): An Example of Multi-Hoping**

From the above figure it is assumed that,

\[
\text{service}_A \text{ authenticates service}_B \Rightarrow \text{service}_A \text{ trusts service}_B, \\
\text{service}_B \text{ authenticates service}_C \Rightarrow \text{service}_B \text{ trusts service}_C, \\
\text{Therefore, } \text{service}_A \text{ authenticates service}_C \Rightarrow \text{service}_A \text{ trusts service}_C.
\]

The above example shows the case of multi-hoping, which means that the route through multiple Web services. This gives the concept of WS-routing where the SOAP messages are to route through the multiple Web services. The concept of WS-routing is not same with Single-Sign-On and Federated-trust, because the former considers only the SOAP messages has to route among multiple Web services. So far security challenges are concerned, mechanisms like Single-Sign-On and Federated-Trust may be designed on the top of WS-routing.
2.6.2 Authorization of Web Services

Once an entity has been authenticated, the next work is to ascertain which operations the entity is allowed to do and on what resources. The authorization mechanism deals with granting or revoking access of resources. The standard for authorization in Web service domain is XACML.

**XACML.** Data are more easily accessible today, by more people, so control of access is much more important. Due to this, there are regular requirements for the protection of data. The Extensible Access Control Markup Language (XACML) is an XML based language which is required for expressing the rules needed to make authorization decisions. XACML includes access control policy language which defines the set of subjects that can perform particular actions on resources and the other one is request/response language, which is a way to express queries about whether a particular access should be “granted” or “revoked” [17].

The main objects that XACML deals with are attributes. Attributes are named values of known types. Specifically, attributes are the characteristics subject, resource, action or environment in which the access request is made [17]. Attributes are used in XACML to aid in creating access control policies. Attributes refer to individual properties of the subject, resource, action or environment that are applicable to the access request such as the subject’s user name or environment’s current time.

**Access Control.** Access control is generally defines as the prevention of unauthorized use of a resource which includes the prevention of use of a resource in an unauthorized manner. As confidentiality is a basic requirement for every interaction, access control is considered as an important security mechanism. Various authorization models are developed, the most notably known as Role Based Access Control (RBAC) and Attribute Based Access Control (CBAC) [18].

1) **RBAC.** RBAC (Role Based Access Control) is an authorization mechanism which is based on some roles that are applicable for authorized users. RBAC simplifies security management by providing a role hierarchy structure. Many organisation uses flexible platform independent XACML to support RBAC. RBAC on a Web service platform should be implemented for the administrator, developers and any other privileged accounts that will be required for the Web service to operate. The Web service platform
must be configured to enforce separation of roles that means not allowing a user assigned to one role to perform functions exclusively assigned to another role [16].

2) **ABAC.** ABAC (Attribute Based Access Control) provides a mechanism for representing either a user's or application's access profile through a combination of the following attribute types –

   a) **Subject Attributes (s):** This is associated with a subject (user or application) that defines the identity and characteristics of that subject.

   b) **Resource Attributes (r):** This is associated with a resource such as a Web service, system function or data.

   c) **Environment attributes (e):** This attribute type describes the operational, technical and situational environment or context in which the information access occurs.

ABAC policy rules are generated as Boolean functions of s, r and e attributes and shows that whether a subject (s) can access a resource (r) in a particular environment (e) which is written as –

\[
\text{Rule X : can_access (s, r, e) \leftarrow f(ATTR (s), ATTTR (r), ATTTR (e))}
\]

ABAC generates with XACML, which relies on policy-defined attributes to make access control decisions [16].

**XACML Architecture** [23]. The XACML architecture specifies the implementation of a Policy Enforcement Point (PEP), a Policy Access Point (PAP), a Policy Decision Point (PDP), a Policy Information Point (PIP) and a Context Handler. Each of these are devoted to one specific task that is access control process. The PEP receives access requests from the requester and forwards them to the PDP which is responsible for the evaluation of attributes and which will take decision whether access is "granted" or "deny". The PIP supplies the attributes of subject, resource, action or environment to the PDP which are relevant for "allow" or "deny" access decision on resources. The access policy is provided by the PAP that stores and maintains the access rules. The XACML architecture also employs a Context Handler which manages a repository for all attribute definitions and their corresponding
implementations. The Context Handler is responsible for obtaining and supplying the requested values. The Context Handler shall return the values of attributes that match the attribute designator or attribute selector. If no attributes from the request context match then the attribute shall be considered missing. The authorization decision arrived at by the PDP is sent to the PEP with some obligations. The PEP fulfils the obligations based on the authorization decision sent by PDP which is either "permits" or "denies" access [19].

The data flow model of XACML architecture shown in Figure 2.6 (b) is as follows –

The PAP writes policies and policySets and makes them available to the PDP. The access requester sends a request for access resources to the PEP. The PEP sends the request for access to the Context Handler. The Context Handler converts the access request to an XACML request and sends it to the PDP. The PDP requests the attributes of subject, resource, action and environment from the Context Handler. The Context Handler requests attributes from the PIP. The PIP obtains the requested attributes and returns them to the Context Handler. The Context Handler sends the requested attributes and resource to the PDP. The PDP evaluates the policy. The PDP returns the response context including the authorization decision to the Context handler. The Context Handler returns the response to the PEP. The PEP fulfils the obligations. If access is permitted, then the PEP permits access to the resource; otherwise, it denies access [18], [19].

Figure 2.6 (b): Data Flow Diagram for XACML Architecture
2.6.3 Confidentiality of Web Services

XML encryption is mainly ensuring confidentiality to encrypt the XML data. XML encryption is the way to handle requirements for security in data interchange applications. XML encryption does not introduce any new cryptography algorithms or techniques for securing message. RSA, SHA-1, MD5 algorithms may still be used for actual encryption of message [20].

2.6.4 Non-repudiation of Web Services

WS-Security can provide non-repudiation of both the SOAP messages and its contents. WS-security supports signing the SOAP header, to ensure that the recipient and sender of the SOAP message have not changed since the message was sent [16]. Non-repudiation can be achieved by logging individual messages for later retrieval.

2.6.5 Availability of Web Services

The issue of availability states that, resources, services should be available to authorized users at all times. Availability is a major issue in Web service security. One of the means of denying availability is due to Denial-of-Service (DoS) attack. A DoS attack aims to use up all the resources of a service so that it is unavailable to users [20].

2.6.6 Integrity of Web Services

Integrity in Web services is mainly concerned with the WSDL files, which describes, defines or publish the functionalities of a Web service. When a service requester communicates with the registry for a suitable service, WSDL file is the basis of selecting the service during composition. XML signature is the technology that can be used for message integrity [16].
2.6.7 Audit Records and Mechanism

Records are necessary to enable a resolution if a party to a transaction denies the occurrence of the transaction or if other dispute arises; also to trace user access, behaviour and to enable system integrity verification [8].

2.6.8 Security Policy

A security policy [18] regulates the rules, requirements and mechanisms for Web service transactions in a large distributed environment. A security policy includes the following –

1) How the sender is authenticated, that is what mechanism is used with what parameters and in what value ranges.

2) Within the SOAP message, which XML elements are encrypted using what algorithms and key sizes and for what particular recipients or the recipient roles.

3) Within the SOAP message, which XML elements are integrity protected using what mechanisms with which algorithms and key sizes.

Enable implementers to define a security policy and enforce it across various platforms with varying privileges.

2.7 XML Security

XML documents are easily accessible which contributes lack of security in XML. Without knowing XML, one can decipher information easily and pick out sensitive information such as – name, address, credit card no. etc. This is the reason, why XML data is vulnerable. Two techniques broadly used to secure XML message are – XML encryption and XML digital signature [27].
2.7.1 XML Encryption

The XML encryption enables encryption of an entire XML document or specific parts of an XML document. It is possible to encrypt a complete XML file or an element of XML file, non-XML data and 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 stored in the XML document. 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 [27]. XML Encryption allows XML syntax to be encrypted to ensure data confidentiality. The encrypted syntax is replaced by an <EncryptedData> element containing the ciphertext of the encrypted data as content. The XML encryption also defines an <EncryptedKey> element for any transportation purposes. An XML data is encrypted with a randomly generated symmetric key, which itself is encrypted using the public key of the message recipient. In SOAP messages, the <EncryptedKey> element – if present – must appear inside the security header [20].

2.7.2 XML Digital Signature

XML digital signature allows XML syntax to be digitally signed to ensure integrity or to proof of authenticity [20]. The XML digital signature standard defines a schema discussed in Figure 2.7 for capturing the result of a digital signature operation applied to XML data or other data types. XML signatures add authentication, data integrity and non-repudiation to the data that they sign. A fundamental feature of XML signature [14] is the ability to sign only specific portions of the XML tree rather than the complete document. XML signature is an evolving standard for digital signatures [22].

The XML digital signature schema discussed in Figure 2.7 is a 2-tuple signature, where the signed element is referenced by its URI under the <Reference URI?> element [26]. XML signature schema uses URI to identify resources, algorithms and semantics. The <SignedInfo> element contains the signed data and specifies what algorithms are to be used. The <SignatureMethod> and <CanonicalizationMethod> are used by the <SignatureValue> element and are included in <SignedInfo> to protect them from tempering. The <SignatureMethod> is the algorithm that is used to convert the canonicalized <SignedInfo>
into the <SignatureValue>. It is a combination of a digest algorithm and a key dependent algorithm and possibly other algorithms, for example RSA, SHA-1, MD-5 etc. The <CanonicalizationMethod> is the algorithm that is used to canonicalize the <SignedInfo> element. Each <Reference> element includes the <DigestMethod> and the resulting <DigestValue> calculated over the identified (by URI) data object. It also may include <Transforms> that produce the input to the digest operation. One or more <Reference> elements specify the resource being signed by URI reference and any <Transforms> to be applied to the resource prior to signing. <DigestMethod> specifies the Secure Hash Algorithm (SHA-1) before applying the one-way hash function. <DigestValue> contains the result of applying the hash algorithm to the transformed resources. The <SignatureValue> element contains the digital signature value that is an encrypted digest of the SignedInfo. The signature generated with the parameters specified in the <SignatureMethod> element of the <SignedInfo> element after applying the algorithm specified by the <CanonicalizationMethod>. <KeyInfo> (optional) indicates the key to be used to validate the signature. Since, <KeyInfo> is outside of <SignedInfo>, if the signer wishes to bind the keying information to the signature, a <Reference> can easily identify and include the <KeyInfo> as part of the signature. The <Object> element (optional) contains any other information to support the signature [22].

Figure 2.7 [13]: The Structure of XML Digital Signature
From the above discussion, it can be noted that, XML digital signature and XML encryption solves many problems, but they do not solve all the problems. XML signature involves many algorithms and the strength of the signature depends on the used digest algorithms [27].

2.8 Secure Socket Layer

SSL (Secure Socket Layer) is a technology that is used to protect companies from Web Service Security (WSS) attacks. SSL used in encryption technique, which are in turn used to implement for data protection. SSL creates a secure tunnel in between source and destination computers based on public key encryption technique. A common protective measure is to send messages over a secure connection that is using SSL. For example, an SSL connection between two computers may be sufficient for simple applications. For multiple Web services, complete message or individual part of messages may be encrypted and signed to protect the confidentiality and integrity of Web service messages [21].

It is therefore very important to have a tool to guarantee that the message was not modified in transit and that each party can be sure that no other fraudulent party is participating. One of the first counter-measures is the usage of SSL.

1) SSL allows the client to determine that the service is the real service that the client is supposed to communicate with. The SSL server (i.e. the service provider) sends his certificate, which is checked by the client (application).

2) SSL guarantees message integrity. All traffic between the two TCP endpoints is protected from forgery.

3) SSL encrypts all traffic, so that the (even plaintext XML) messages cannot read by unauthorized parties.

4) SSL is widely available and understood by all major browsers. Also, SSL can easily be configured for Web services and commercial applications usually support it as well.

5) SSL has one disadvantage, which is that encryption and other security functions are performed only by the two TCP endpoints. If the message needs to be secured further,
then an end-to-end protection like the usage of XML signature and XML encryption in the application layer is required.

2.9 WS-Security

<table>
<thead>
<tr>
<th>WS-SecureConversation</th>
<th>WS-Federation</th>
<th>WS-Authorization</th>
</tr>
</thead>
<tbody>
<tr>
<td>WS-Security Policy</td>
<td>WS-Trust</td>
<td>WS-Privacy</td>
</tr>
</tbody>
</table>

WS-Security : SOAP Message Security

Table 2.9: The WS-Security Family

The WS-Security related specifications define the required policies to properly secure the WS interactions. The task of WS-Security is to set the constraints and capabilities of a WS and they do not have intention to substitute any security technologies. On the contrary, WS-Security related specifications enlarge and merge existing security infrastructure and correctly define how these can be used in an interoperable way [11]. WS-Security is achieved by defining header elements to be included in SOAP messages. WS-Security does not provide a complete security framework for Web services. However, it provides mechanism for ensuring single-message security within SOAP. In WS-Security, message integrity of SOAP messages provided using the XML signature specification in conjunction with security tokens. In WS-Security, message confidentiality uses the XML encryption specification in conjunction with security tokens [10]. The key aspects of WS-Security layers are as follows –

1) **WS-Security/SOAP Message Security.** It is made on the SOAP specification and specifies how to sign and secure SOAP messages.

2) **WS-Trust.** To route among multiple Web services, the trust relationship must be established among different Web services. The trust relationships among multiple Web services can be either direct or through the discovery agency [16].
3) *WS-SecureConversation*. It is built based on WS-security and WS-trust to specify how WS can manually manage and authenticate security contexts. It includes describing how service requesters can authenticate WS, as well as how WS can authenticate messages from service requesters.

4) *WS-Security Policy*. It specifies a generic format through which to describe security capabilities and requirements for SOAP message senders and receivers.

5) *WS-Federation*. It is made on all previous specifications to specify how to broker and manage heterogeneous, federated trust contexts.

6) *WS-Privacy*. It is built based on WS-Security, WS-Security Policy and WS-Trust to specify a model by which organizations using WS can indicate preferences as well as conformance to particular privacy policies.

7) *WS-Authorization*. It specifies how to access policies for WS are specified and managed using a flexible authorization language (XACML) and format [11].

### 2.10 Threats and Attacks

Security decisions must always be made with an understanding of the threats facing the system to be secured. The possibility of an attack is known as threat. The common threats of Web services are as follows [8] –

1) *Message alteration*. In this threat, message information is altered by inserting, removing or modifying information created by the originator of the information.

2) *Loss of Confidentiality*. This type of threat makes information within the message visible to unauthorized participants.

3) *Falsified messages*. This type of threat occurs when an attacker constructs new messages and sends them to a receiver who believes that the message has originated from a third party other than the sender.

4) *Principal spoofing*. A message is sent which appears to be from another principal. For example, Alice sends a message which appears as if it is from Bob.
5) **Forged claims.** A message is sent in which the security claims are forged in an effort to gain access to unauthorized information. For example, a security token which was not exactly issued by the specified authority.

6) **Replay of message parts.** A message is sent which includes portions of another message in an effort to gain access to unauthorized information.

7) **Replay of message.** A whole message is resent by an attacker.

The typical requirements for a secure system are integrity, confidentiality and availability. Any action targeting at the violation of one of these Web service security properties is called an attack. The possibility of an attack is known as vulnerability. Web services may be vulnerable to man-in-the-middle attack, masquerading attack, eavesdropping attack, Denial-of-Service (DoS) attack etc. during composition, binding and execution. Some other attacks on Web service applications like SQL injection or parameter tempering which also use non-valid messages focusing on integrity. The list of attacks that facing by Web services are [27]

1) **Buffer Overflow Attack.** This type of attack arises when the amount of reserved for the operation becomes smaller than the amount of data written to the memory. 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.

2) **Cross Site Scripting (XSS or CSS).** In this attack, the attacker inserts malicious code in the request and this will be returned to the victim by that application. While the script runs the attacker can perform the attack on behalf of the user. The majority of XSS or CSS attacks rely on `<scripts>` tags. An XSS attack sets up, for example, a Trojan horse in the victim's Web browser. The Trojan horse is often created by client-side languages such as JavaScript, Java, and VBScript or takes advantage of a known vulnerability in the browser.

3) **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 an attacker gains access to the network with a valid IP-address, he/she can modify, reroute or delete data.
4) **SQL Injection.** 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.

5) **Network Eavesdropping.** This 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. To performing this attack, the attacker can capture traffic and obtain usernames and passwords.

6) **Dictionary Attack.** An attacker systematically tests all possible passwords to perform dictionary attack. Here, the attacker’s goal is to obtain passwords. To perform dictionary attack, the attacker trying every word in the dictionary until matching password is found. Most password-based authentication algorithms are vulnerable to dictionary attack.

7) **Data Tempering.** Data tempering attack occurs when an attacker changes or modifies valid data, while it passes over the network. Any data sent from the client-side may be manipulated by an attacker. Successful attacks results in tempered data that reaches to the receiver.

8) **Denial-of-Service (DoS) Attack.** In DoS, the attacker’s goal is to disclose information that he can use for crashing the Web application process. Performing DoS, the attacker may attack a router, firewall or proxy server with the goal of making them unusable. The effect of DoS attack is to stop the service-providing computer from being able to provide the service.

9) **Man-in-the-Middle Attack.** When an attacker intercepts messages between client and server, the man-in-the-middle attack can occur. Both client and server assume that they are communicating each other, but the communication between the sender and the receiver is actually flowing through the attacker. The attacker is free to modify the content of the messages and sent them to the receiver. The receiver receives the message that she thinks that the message has come from the sender and acts on it. Then the receiver sends the message back to the sender, which goes through the
attacker. Unfortunately, neither sender nor receiver knows that they have been attacked.

**10) Parameter Tempering.** The aim of parameter tempering attacks is to modify parameters sent between the user and the application. An attacker may change the URI parameters or mode parameter. The parameter tempering includes Query string manipulation, Form field manipulation and HTTP header manipulation.

More details of these attacks on Web services are available in [27]. This chapter have not provided any solution on how to mitigate these attacks but they have focused on the source of generating possible attacks on SOAP message implementation of XML Web services.

Some of the new attacks are summarized below which are belongs to the category of DoS attacks focusing on availability [20]. These attacks are -

1) **Oversize Payload.** In Web services, the oversize payload attack is caused due to the high memory consumption of XML processing. The total memory usage caused by processing one SOAP message is much higher than the message size. This is due to the fact that, most Web services implement a tree-based XML processing model.

2) **XML Injection.** An XML injection attack tries to modify the XML structure of a SOAP message or any other XML document by inserting content – e.g. operation parameters containing XML tags. Such attacks are possible if the special characters "<" and ">" are not escaped properly.

3) **SOAPAction Spoofing.** The optional HTTP header field “SOAPAction” can be used for operation identification. The SOAPAction field value is often used as the only qualifier for the requested operation. The SOAPAction spoofing attack is happened when the server actually ignored the SOAPAction value and invoked another operation when a message is sent to the service.

4) **WSDL Scanning.** If the Web service is created using common Web service framework tools, the generated WSDL contains all operations. In this case, an external client gains knowledge of the internal operations and can invoke them. For example, an Internet shop system needs methods for placing an order for customers (sendOrder) and for administrating the orders (adminOrders). If both sendOrder and adminOrder
operations are offered by one Web service, an attacker with the knowledge of sendOrder can easily find the administration method also.

5) **Metadata Spoofing.** A Web service client retrieves all information regarding a Web service invocation (i.e. message format, network location, security requirements etc.) from the metadata documents provided by the Web service server. Currently, this metadata usually is distributed using communication protocols like HTTP or mail. These circumstances open new attack possibilities aiming at spoofing these metadata. The most relevant attacks in this category are WSDL spoofing and security policy spoofing. More details of these attacks are available in [20].

### 2.11 Chapter Summary

Web service has emerged today as an enabling technology for business transactions across different platforms and languages where business processes are encoded and running. It is the technology which enables distributed computing mainly in the form of workflows. Security is a major concern for social acceptance over and above the platform independent and the language independent of the technology. This chapter mainly reviews the security aspects of Web services. It is seen that the security concerns are similar to other web applications in general; however Web services have specific security issues as well. The specific security challenges are mainly due to the adoption of XML in different components of Web services. The present study includes recent threats and attacks specific to XML-based Web services.