TRUST MODELING FOR SECURE WEB APPLICATION DEVELOPMENT

Mahmood Doroodchi, Department of Mathematics and Computer Science, Cardinal Stritch University, USA, mdoroodchi@stritch.edu

Mohammad B. Sharifinia, Islamic Azad University, Fasa, Iran, Sharifnia@gmail.com

Azadeh Iranmehr, Information Technology (E-commerce), Shiraz University, Iran
iranmehr@gmail.com

Abstract

Trust is the main concern in securing web applications and it has to be implemented in every layer of application at the time of development. Building trust based on a proper model is an important step in designing a secure web-based system. Such model should address users’ sociological, economical, and personal expectations of trust in every layer of web applications. One approach is to make sure that data is transferred securely and trustable between the different parts of the application. Because of the importance of web services in modern web applications and the important role of message in it, our focus is in web service message security.

In this approach, the original entity authentication is identified as the first step in establishing trust. Next, the security requirements for message level of web services are listed as the key principle of an effective security design and implementation of web applications. Moreover, threat analysis of web applications and their possible solutions lead to complete the proposed model. This paper analyzes the threats that can be related to the use of web services technology and its messaging system in a web application.

Keywords: Trust model, security, risk analysis, web application.

1 INTRODUCTION

Secure web application development, similar to any other software development, includes the major step of requirement analysis. Examples of possible requirements for secure web-based software includes but not limited to “Identification and Authentication”, “Authorization”, “Non-repudiation”, and “Integrity and Privacy”.

In order to deliver such security mechanism and covering all the requirements simultaneously, it is necessary to provide a framework or a model for trust. For this purpose we formulate the “trust model” based on available security methodologies and risk analysis methods for web application development.

2 TRUST AND WEB APPLICATION

2.1 Trust

According to ITU-T X.509, Section 3.3.54, trust is defined as [10]: "Generally an entity can be said to ‘trust’ a second entity when the first entity makes the assumption that the second entity will behave exactly as the first entity expects.”

In order to define trust and trust modelling relative to security architecture, the following set of principles or elements are distinguished [1]:

M. Doroodchi, M. B. Sharifinia, and A. Iranmehr
Trust Modelling for Secure Web Application Development
Trust is a characteristic and quality of security architecture.

Trust is the balance of liability and due diligence. For example, one must decide how much effort is needed to reduce liability to an acceptable level for a particular business proposition and stated security policy. In other words, equilibrium of trust must be established.

Trust is the enabling of confidence that something will or will not occur in a predictable or promised manner. The enabling of confidence is supported by identification, authentication, accountability, authorization, and availability or in overall security elements (see section 3.1.1).

Trust is the binding of unique attributes to a unique identity, for example, accountability. This is both a qualitative and a subjective measure of expectations regarding another’s behavior and relative to a defined security policy. Essentially, a trust relationship is established when a satisfactory level of confidence in the attributes provided by an entity is achieved.

Trust is defined as a binary relationship, or set of compound binary relationships, based on individual identity or unique characteristic validation. That is, trust is the establishment of a relationship through a validation process and the subsequent use of that relationship in some transactional context.

To establish trust or confidence, there must be a binding of unique attributes to a unique identity, and the binding must be able to be tested satisfactorily by a relying entity. Upon achieving a satisfactory level of confidence in the attributes provided by an entity, the trust is established. This element of trust is commonly called authentication.

2.2 Trust modelling

A security architecture based on an acceptable trust model provides a framework for delivering security mechanisms. Trust modelling is the process performed by the security architect to define a complementary threat profile and trust model based on a use-case-driven data flow analysis [1]. The result of such study integrates information about the threats, vulnerabilities, and risk of particular information technology architecture. Further, trust modelling identifies the specific mechanisms that are necessary to respond to a specific threat profile [1]. A trust model is not the particular security mechanisms utilized within particular security architecture. Rather, it is the combination of those security mechanisms in conjunction with the security policy when they address all business, technical, legal, regulatory, or fiduciary requirements to the satisfaction of a relying entity. All valid trust models are predicated on the establishment of trust between any two entities. That is, original entity authentication must be performed first. To provide a baseline, a model for trust can be defined as follows [1]:

- A trust model identifies the specific mechanisms that are necessary to respond to a specific threat profile.
- A trust model must include implicit or explicit validation of an entity’s identity or the characteristics necessary for a particular event or transaction to occur.

2.2.1 Major Trust Models

Different systems have different thresholds for risk, and the choice of a trust model should be based on that threshold. Specific security solutions should map to the applicable trust model. This section discusses three primary trust models [1]:

**Direct Trust:** Direct trust exists when one performs the validation of an entity's credentials without reliance on any other entity. All entities gain trust by their association with a common entity responsible for the original entity authentication of each relying entity, always following a stated security policy. A direct trust model is found in some architecture using Public Key Infrastructure (PKI).

**Transitive Trust:** This model of trust is established when the trust attribute is transmitted through another party. Transitive trust allows the following:
• Entity A validates and trusts Entity B.
• Entity B validates and trusts Entity C.
• Entity A trusts but does not need to validate Entity C. For example, Entity A trusts Entity C, but does not perform original entity authentication of Entity C.

Such a trust model is common in distributive or peer-to-peer systems. It relies on participating entities to align their security policies that control credential validation.

**Assumptive Trust:** With this model, there is no mandatory, explicit, direct credential validation. With essentially no control over the validation process, you must either “take it, or leave it.”

### 2.3 Web Application

In essence web application is a client/server (or in general n-tier) application that interacts with users or other systems using HTTP [4]. Normally, for a user the client piece would most likely be a web browser like Internet Explorer or Firefox; whereas for another software application this would be an HTTP user agent that acts as an automated browser. The end user views web pages and is able to interact by sending choices to and from the system. The functions performed can range from relatively simple tasks like searching a local directory for a file or reference, to highly sophisticated applications that perform real-time sales and inventory management across multiple vendors, including both business to business (B2B) and business to consumer (B2C) e-commerce, workflow and supply chain management, and legacy applications. Modern applications run on distributed application servers, connecting to multiple data sources through complex business logic tiers. A generic process flow for web applications is presented in figure 2. Within this architecture for web applications, the technology of web services can be used for a variety of purposes.

SOAP, the Simple Object Access Protocol [8] was developed to enable a communication between entities in a web application. It was designed as a lightweight protocol for exchange of information in a decentralized, distributed environment and it is based on standards of HTTP and XML.

As a web-based application that is network-enabled, web applications are at risk for many security exploits. On the other hand, one of the underlying principles of security is ensuring that all entities involved in a transaction trust one another. To this end, web applications should support a trust model that can be used to enable web applications to trust the identities of entities.

### 3 SECURITY REQUIREMENTS

In this section after we explain the security mechanism under the framework of trust, we elaborate the scopes for securing messages and the most useful standard for it.

#### 3.1.1 Elements of Security

Important security elements include [3]:

- **Identification and Authentication.** Verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in an information system.
- **Authorization.** The permission to use a computer resource, granted, directly or indirectly, by an application or system owner.
- **Integrity.** The property that data has not been altered in an unauthorized manner while in storage, during processing, or in transit.
- **Non-repudiation.** Assurance that the sender of information is provided with proof of delivery and the recipient is provided with proof of the sender’s identity, so neither can later deny having processed the information.
- **Confidentiality.** Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information.
• **Privacy.** Restricting access to subscriber or relying party information in accordance with Federal law and organization policy

The problems of defining trust and trust metrics have primarily focused on public key authentication. Trust models and metrics for public key infrastructure systems address authentication between sender and receiver entities, message integrity, and data confidentiality. These are all aspects of a security and often the terms *trust model* and *security model* are used interchangeably [7].

### 3.1.2 WS-Security

WS-Security was developed to provide SOAP extensions that define mechanisms for using XML Encryption and XML Signature to secure SOAP messages. WS-Security describes enhancements to SOAP messaging to provide quality of protection through message integrity, message confidentiality, and single message authentication. These mechanisms can be used to accommodate a wide variety of security models and encryption technologies [6]. The ability of WS-Security to provide authentication (as well as confidentiality and integrity) at the SOAP message level is important for web applications to trust the messages they receive.

### 3.1.3 Where to encrypt?

There are two main places where encryption is applied to a message [5]:

- **At the “Transport level”** - where techniques such as SSL can be used to encrypt data, including messages sent over an HTTP connection. Transport level security can also be used to identify the sender of the message if client side certificates are used or if alternative methods such as name and password are provided with the message.

- **At the “Message level”** - where the content of the message is digitally signed and/or encrypted.

In order to selectively encrypt part of the message and provide confidentiality beyond point to point connection we have to use message level security.

In our case, securing a SOAP message at the message level is the preferred method. However, unlike Transport Layer Security this method requires that all security processing occurs at a higher level in most technology stacks. The rest of this paper deals with the message level security.

### 4 TRUST MODELING FOR WEB APPLICATION

The model proposed in this paper is based on thorough analysis of risks in web application. Just as we identified original entity authentication as the first step in establishing trust, we need to identify the starting point for defining a trust model. With the underlying principle understood, the next step is to gather security requirements for every layer of application. These requirements must be specific to each application, although they may well be similar to those of other comparable applications. In this paper our focus is message level security.

Next, we build a threat profile based on the process flow. Some risks are going to be more significant than others and will require greater effort to mitigate. This paper analyzes the threats that can be related to the use of web services technology in a web application. Finally, by identifying security mechanisms and standards to respond to the specified risks, the model is actually implemented.

#### 4.1 Process Flow in Web Applications
Web services have become a common building block in modern web applications. A web service is essentially an XML-messaging based interface to some computing resource. Using SOAP for creating and consuming messages, and using HTTP for transporting messages. Each layer of the web service's stack represents one of the fundamental functional areas of a web service instance in web applications. These layers are depicted in the following diagram [9].

![Web service stack](image)

**Figure 1: Web service stack**

A web service application may include several logical layers incorporating functions such as application business logic. The details of each layer of the web service stack are [9]:

**Data Layer** - The data layer translates the application specific data into the model chosen for the specific web service. The data layer includes the functions necessary to support flexible data typing.

**SOAP Message Layer** - The SOAP message layer is the infrastructure that processes SOAP messages, dispatches them, and may optionally fulfill Quality of Service requirements. On the sending side the message layer writes SOAP messages, based on the data model. On the receiving side the message layer processes the SOAP messages and dispatches requests to the correct application or method.

**Transport Layer** - The transport layer sends and receives messages.

In this document, the assumed web services communication model is SOAP over HTTP. Basic SOAP interactions are asynchronous and unidirectional, but can be combined to implement request/response processes, or even more sophisticated interactions.

Next to the originating and receiving node of a web service, intermediate nodes can be defined, as shown in Figure 2. Those intermediate nodes can process the SOAP message, and add extra information to the message (such as a signature on a part of the message).

![Process flow of SOAP message](image)

**Figure 2: Process flow of SOAP message**

The remainder of this paper will focus on how to develop trust model for secure web application. As the importance of web service and exchanging message between them in web application the main emphasis is on message level security when developing a trust model. It is not intended to provide a fully secure system design and should not be taken as a template for a production level security design. This model is useful in developing a customized trust model for a particular web application development. Different applications have different security requirement, and consequently their trust model can not be the same.
4.2 Determining Security Requirements

Security requirements vary from system to system depending on the system specifications, the deployed systems, and the operations that are carried out on them. In this paper, the message level security requirement for web applications is analyzed. Therefore, security requirements are specified for message integrity, authentication and confidentiality. Security requirements are initially investigated by asking the following questions:

- **Message Integrity.** Would message alteration by a third party be harmful?
- **Authentication.** Does the receiver care where the message is originated from?
- **Confidentiality.** Would a third party gain some information from the disclosure of message content?

4.2.1 Message Integrity

Message integrity is required to ensure that messages have not been altered in transit. Typical alterations to a message could include:

- Altering the originating user's identity
- Altering the identity of the application sending the message
- Altering data in the message
- Altering configuration information in the message

4.2.2 Authentication

Authentication is required to allow the receiver to determine where the message has originated from. In practice the recipient of a message will often authenticate the sender of a message by first checking whether the data in the message has been signed (using public certificate) with a private key (for message integrity), and later by checking the credentials in that public certificate to determine the identity of the sender.

The recipient will also need to check if you can trust the certificate issuer, and may also need to compare the data in the content of the message that identifies the sender, either in the SOAP header or in the payload, with the identity as stated in the public certificate.

4.2.3 Confidentiality

Confidentiality is required to conceal sensitive information in messages. Not all parts of messages are necessarily sensitive, and in some cases a message may not be considered sensitive at all. Thus, there may be no need for confidentiality. Some parts of the message that are typically considered sensitive include:

- The Soap Body – this could contain information such as order data, which could aid competitors
- The Signature – in some cases the body of the message will contain predictable variations, making it subject to guessing attacks. To prevent this the signature data should also be encrypted

4.3 Risks and solutions

Security decisions must always be made with an understanding of the threats facing the system to be secured. While there is enormous number of security standards and technologies available for securing web applications, there may not be adequate or just necessary ready-to-use tools for a particular organization or an individual service and often it is required to customize the solutions. For that reason, it is important to understand the threats that face each web application so that organizations can determine which threats their web applications must be secured against and which standards they must use to decrease their risks. As mentioned before because of the importance of the message in the
web application, this paper emphasizes on the general risks and threats exposing messages in most web applications.

SOAP message itself faces the following security challenges:

- SOAP does not perform any authentication between SOAP endpoints or intermediaries, so there is no way to verify the origin of a SOAP message.
- SOAP does not provide a mechanism for ensuring data integrity or confidentiality either at rest or during transit.
- SOAP does not provide a mechanism for detecting resubmitted SOAP messages.

Therefore, messages in web applications may face many risks depending on the application. In general, according to WS-I, the main threats facing web services are [2]:

- **Message alteration.** An attacker inserts, removes or modifies information within a message to deceive the receiver
- **Loss of confidentiality.** Information within a message is disclosed to an unauthorized individual
- **Falsified messages.** Fictitious messages that an attacker intends the receiver to believe are sent from a valid sender
- **Man in the middle.** A third party sits between the sender and provider and forwards messages such that the two participants are unaware, allowing the attacker to view and modify all messages
- **Principal spoofing.** An attacker constructs and sends a message with credentials such that it appears to be from a different, authorized principal
- **Forged claims.** An attacker constructs a message with false credentials that appear valid to the receiver
- **Replay of message.** An attacker resends a previously sent message
- **Replay of message parts.** An attacker includes portions of one or more previously sent messages in a new message
- **Denial of service.** An attacker causes the system to expend resources disproportionately such that valid messages cannot be met.

Each organization based on their needs and goals has to prioritize the above risks. For example, in some instances messages need not to be kept confidential and loss of confidentiality is not a concern. Later, based on the assessed priority particular solutions can be adopted.

In the following, some standards that can be used to protect web service’s message against many of the threats are listed:

- **W3C XML Encryption.** Used by WS-Security to encrypt messages and provide confidentiality of part or all of a SOAP message
- **W3C XML Signature.** Used by WS-Security to digitally sign messages and provide message integrity and sender authentication
- **WS-Security Tokens.** Allows messages to include credentials to aid receivers in determining whether or not the message sender is authorized to perform the requested action. Supported token types include:
  - **Username/password.** The most common credentials in web applications
  - **OASIS SAML Assertion.** Asserts that the sender has been authenticated and/or supply attributes associated with the sender
  - **IETF X.509 certificate.** Coupled with XML Signature, a receiver can verify that the CA issued the certificate used to sign the SOAP message
  - **ISO Rights Expression Language.** Used to provide public key information, attributes of those keys, as well as information about the sender’s license
  - **IETF Kerberos token.** Allows web services to exist in a Kerberos domain.
- **W3C WS-Addressing IDs.** Allows the message sender to supply a unique identifier for the message
- **IETF SSL/TLS.** Secures the HTTP protocol over which SOAP messages are sent and received
- **SSL/TLS with client authentication.** Requires both the sender and receiver to authenticate with one another before securing the HTTP protocol
- **IETF HTTP authentication.** Allows usernames, passwords (via HTTP Basic) or password digests (via HTTP Digest) to be sent as part of the HTTP header.

Table 1 illustrates which standards provide protection against these threats. As the table shows, SSL/TLS and WS-Security, through XML Encryption and XML Signature, provide similar protections against threats. Application developers should select one of these solutions depend on the risks that they recognize in risk analysis process during application development.

<table>
<thead>
<tr>
<th>Threat</th>
<th>Message Alteration</th>
<th>Loss of Confidentiality</th>
<th>Falsified Message</th>
<th>Man in the Middle</th>
<th>Principle Spoofing</th>
<th>Forged Claim</th>
<th>Replay of Message Parts</th>
<th>Replay of Message</th>
<th>Denial of services</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML Encryption</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>XML Signature</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>WS-Security Tokens</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>WS-Addressing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSL/TLS</td>
<td>X</td>
<td>X</td>
<td>X'</td>
<td>X'</td>
<td>X'</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SSL/TLS with client certificates</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>HTTP Authentication</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Threat mitigated only for provider messages to requester, not for requester messages to provider.

As shown in Table 1, there are no standards that protect against DoS attacks. Many of the availability techniques used by high volume web applications, such as load balancing, clustering and replication, can be used to aid in preserving availability.

Web applications also face threats associated with all software: defects in a web applications implementation may lead to exploitable vulnerabilities. Web applications, are remotely accessible, so attackers can take advantage of the web application's availability to probe it for potential exploits. As with any remotely accessible service, it is important that web applications be implemented securely and that traditional network security tools and techniques be used to limit access to the web applications to only those networks and systems that should have legitimate access.
4.4 Data Origin Authentication and Identification

Verifying that a message has not been changed does not mean that it is authentic. Data origin authentication is the first step in developing trust model. Data origin authentication is the corroboration that the source of the data received is as claimed. To make a claim of origination requires some kind of identification. Identification refers to the act of presenting an identifier to a system so that the system can recognize a system entity and distinguish it from other entities.

In this process flow each system entity provides authentication by including an X.509 security token in combination with XML signatures over the body and one or more header elements. The X.509 certificate contains the public key for signature validation and a set of values identifying the subject who possesses the associated private key. The public key and subject values are each signed by the certificate authority.

At a minimum, each implementation should support the ability to perform the following checks for an authentic message:

- The certificate expiration date has not passed
- The certificate has not been revoked, e.g. by the owner reporting it as lost.
- The certificate was issued by a trusted certificate authority
- The XML Digital Signature is verified using the certificate’s public key. This is considered proof that the originator of the message holds the private key.

The following additional authentication with X.509 certificates may be implemented within WS-Security infrastructure and/or application logic:

- **Infrastructure.** Message senders are authenticated only if the certificate in the message matches a certificate in the receiver’s certificate store.
- **Application Logic.** Additional data may be stored within a certificate that needs to be available to the application to allow authentication decisions based on the contents of certain fields within the X.509 certificate.

At a minimum, each implementation should support the ability to ensure the signing certificate does actually exist in the receiver’s certificate store. Additional application level validations can also be implemented, but do not need to be. Without such validation, any unknown party could create a signature that is verifiable. Similarly, if the certificate fails any of its checks, then it should not be treated as valid. However note that:

- Checking for expiry dates should be made based on when the message was created and signed. Most certificates eventually expire. However if the message was created and signed before the certificate expired, then the message is still valid
- Rejecting a message because a certificate has been revoked should also take into account when the certificate was revoked as messages signed before the certificate was revoked are also likely to be valid
- Some organizations use expired certificates to sign messages, although this is not a good practice.

Deciding to reject a message is really a question of the policy that an organization wants to adopt based on the risks that follow from accepting a message that is not authentic. This will vary from application to application and business to business.

5 CONCLUSIONS

There are many security challenges in application development due to the fact that security mechanisms are not incorporated into the applications during the analysis and development phase. Thus, a key principle of effective security design and implementation is that security should be built into every layer of a solution rather than added as an afterthought. One of the underlying principles of security is ensuring that all entities involved in an application's transaction trust one another. To this
end, developer should identify assets to be protected, analyze possible attacks, and decide about the necessary protection levels, security requirements and authentication methods. As mentioned before, elaborating a predefined and exactly formulated trust model for all applications is impossible due to variation in nature of problems. In other words, each application has its own assets to be protected, its own threats to be prevented, and its own methods among its entities. However, web service security standards provide required properties to develop robust, secure, and reliable trust model for web applications. However, developers still have to analyze their applications from the threat level point of view and select the proper security solution. In particular, they should select entities that must be authenticated and identified, and define the proper authentication method. Defence-in-depth through security engineering, secure software development, and trust modelling can provide much of the robustness and reliability required by these applications during applications development.

Because sensitive data is being transmitted over the web, generally encryption should be required by developers at certain points. In web applications, usually a direct trust model seems to provide better control feature since the type of information a client is allowed to access must be strictly controlled. This trust model incorporates the basic security principle: data is accessible only on a need-to-know basis.

6 REFERENCES

[9] Scott Werden, Colleen Evans, Marc Goodner, WS-I Usage Scenarios, version 1.01, 2003