Apprehension of threats and countermeasures in semantic web services

Rashmi pandey¹, Suresh Kumar²

¹ M.Tech (Information Security) Scholar, Department of C S E,IP University Ambedkar Institute of Technology, Geeta Colony, New Delhi, India
² Assistant Professor, Department of CSE Ambedkar Institute of Technology, Geeta Colony, New Delhi, India

Abstract - In today’s world, data is important and confidential on World Wide Web. Many organizations have become important sources of data. Use and exchange of data and extracting information from data and use of information have become important need. The data is not only integrated from the various sources but extracting information from these data pattern and tread is also important as the database. These data sources can be managed by the database management system or by data warehouse. As increasing information is going to be difficult to properly inform the way it is, to fix this problem we need to read semantic web. Semantic web concept is govern by Tim.Berners Lee, that the W3C (World Wide Web consortium) heads [1]. Agent uses WS (web services) to use this technology to communicate with each other and carry out activities. In order to ensure the safety of the operation of web services, the semantic web services needs privacy, confidentiality, authentication and integrity among others. We will discuss various categories of vulnerabilities and find out causes of threats and defined its countermeasure.

Keywords - threats, security, SOAP, Authentication, Authorization, Cryptography

I. INTRODUCTION

Semantic web is a web of data and methods and technology to allow machines to understand the meaning of information on the World Wide Web. The Semantic Web and agent technologies are just out-of-date. The Semantic Web is described as a World Wide Web evolution in which information is available on the Web includes machine accessible semantics for increasing information processing automation and improving information system interoperability [1][2].

The Semantic Web is an extension of the current Web in which information is given a well-defined meaning and allow users to interact, to find and to share information easily with others on the web, improved enabling computers and human’s to work in collaboration. On the semantic web, we are able to express our feelings which are understandable by machines. Peoples are capable to carry out their tasks such as finding the word, reserving the books and penetrating low price products. It is the idea of having data on the Web defined and linked in a way that it can be used for more effective innovation, mechanization, integration and salvage across various applications. But regrettably, the relevant information which is present on the web is concealed amongst vast amount of immaterial information that is also available on the Web.

There are several types of data present on the web which is to be more private and confidential in nature but the privacy of the data is not preserved due to subtle attacks, malevolent users and their agents or malicious agents can disclose sensitive information or sabotage the information of others. Therefore, agents and their sustaining technologies need secure and reliable Web service.

II. SECURITY PRINCIPAL

As the communications between the Agents, Agents with the Semantic Web Server takes place over the web. These communications are more prone to attacks and have security threats. Therefore it is important to resist them by preserving the following security principles in the Semantic Web [3].

1. Confidentiality: - it is ensured that the data that is transferred from one place to another over the web should not be disclosed to others and should be private on the web.

2. Authentication: - it means that any Agent should be authenticated first before entering in the communication over the web.
3. **Integrity**: Information should not be tempered during the broadcast. It prevents any kind of amendment in the authentic information.

4. **Non-Repudiation**: Any Agent cannot be able to deny in future for having sent the information to other Agent on the web.

5. **Availability**: The entire Agent on the web should be able to perform their assigned services. Availability in the face of denial of service attacks that exploit vulnerabilities unique to Web service technologies, especially targeting core services, such as discovery service, on which other services rely.

### III. ATTACK SCENARIO

In the Semantic Web Service, XML (Extensible Mark-up Language) based SOAP (Service Oriented Access Protocol) messages are used when the clients requests to Semantic Web Server or when Semantic web server sends response (web service) messages to the clients [4][5]. The attacker or intermediary captures the transit message and flood it into the network so that the legitimate clients are not able to use the services and also changes some contents of the message or change full messages and sent it to the recipient of the web services. This type of attack does not require the malicious user to know the contents of the message [6]. Figure show the attack scenario in between the Requestor and the Receiver [7].

### IV. TYPES OF THREATS ON THE WEB SERVICES

#### A. Unauthorized Access

Web services that provide sensitive or secret information should authenticate and authorized. Weak authentication and authorization mechanism give unauthorized access of secret information to the attackers [7][8][9][10].

**Vulnerabilities**: Vulnerabilities that can lead to unauthorized access through a Web service include:

- No authentication and authorization is used
- Passwords passed in plaintext in SOAP headers
- Authentication process or mechanism which is used in the communication channel is unsecured.
- Elegant hackers
- Dictionary attack

**Countermeasures**: You can use the following countermeasures to prevent unauthorized access:

- Use password digests or hash function in SOAP headers for authentication.
- Use Kerberos tickets in SOAP headers for authentication.
- Use X.509 certificates in SOAP headers for authentication.
- Use Windows authentication.
- Use role-based authorization to restrict access to Web services.

#### B. Malicious File Execution

- Code vulnerable to remote file inclusion (RFI) allows attackers to include hostile code and data, resulting in devastating attacks, such as total server compromise [11][15].

**Vulnerabilities**: -

- Developers frequently use or concatenate input with file system or allow to upload a file.
- Many frameworks allow the use of peripheral point references such as URL’s file system references
- If the data in the web is not sufficiently checked, any content of the data can be integrated, processed or invoked by the web server
- It can be unfriendly and influential.

#### C. Parameter manipulation

- Parameter manipulation may be defined as the attack in which the content of the data is changed or modified by the unauthorized person in an unauthorized way [9][10]. In a badly designed and developed web application, malicious users can modify things like prices in web carts, session tokens or values stored in cookies and even HTTP headers [13].

#### D.
Vulnerabilities:
- The message is not digitally signed by the trusted third party or by any authorized party.
- The message is not encrypted with public or private key.

Countermeasures:
- Use digitally signed message to verify or to authenticate the sender and the recipient of the message.
- Use asymmetric and symmetric algorithm to encrypt the message.
- Use message digest so that the attacker is not able to temper the message.

E. Disclosure of configuration data:
- There are two main ways in which a Web service can disclose configuration data. First, the Web service may support the dynamic generation of Web Service Description Language (WSDL) or it may provide WSDL information in downloadable files that are available on the Web server [7][9][10][12][14].

Vulnerabilities:
- Vulnerabilities that can lead to the disclosure of configuration data include [9][10][12][14]:
  - Unconstrained WSDL files are available on the web server which is downloaded by the attacker.
  - A constrained Web service supports the active generation of WSDL on the web server and allows the attacker to consume the web services characteristics in an unauthorized way.
  - Weak exception handling

Countermeasures:
- You can use the following countermeasures to prevent the unwanted disclosure of configuration data [9][10][12][14]:
  - Authorize access to WSDL files (using NTFS permissions).
  - Remove WSDL files from the Web server.
  - Disable the documentation protocols to prevent the dynamic generation of WSDL.
  - Capture exceptions and throw a Soap Exception or SOAPHeaderException that returns only minimal and harmless information back to the client.
  - Configure restricted ACLs on text-based configuration files such as Machine.config and Web.config.
  - Keep custom configuration stores outside of the Web space. This removes the potential to download Web server configurations to exploit their vulnerabilities.

F. DDOS:
- Denial of service denies legitimate users access to a server or services. The SYN flood attack is a common example of a network level denial of service attack. It is easy to launch by the attacker and difficult to track or capture. The aim of the attack is to send more requests to a server than it can handle so that the legitimate users is are not able to access the web services which is provided by the web server[19].

One of the first steps in processing a Web Service request is parsing the SOAP message and transforming the content to make it accessible for the application behind the Web Service. Especially when using namespaces, XML can become verbose and complex in parsing, compared to other message encodings. Thus, the XML parsing process allows other possibilities for a special kind of Denial-of-Service attacks, which is called Coercive Parsing attacks [17].

Vulnerabilities:
- Legitimate users behave like Attacker.
- Unencrypted channel through which the attacker will be able to capture the transmitted information.

Countermeasures:
- to prevent denial of service include [16][19][20]:
  - Apply the latest service packs.
  - Harden the TCP/IP stack by applying the appropriate registry settings to increase the size of the TCP connection queue, decrease the connection establishment period, and employ dynamic backlog mechanisms to ensure that the connection queue is never exhausted.
  - Use a network Intrusion Detection System (IDS) because these can automatically detect and respond to SYN attacks.
  - Thoroughly validate all input data at the server.
• Use exception handling throughout your application's code base.

G. Message replay: - Web service information can potentially travel through multiple intermediate servers. In the message replay attack, an attacker captures and copies a message and replays it to the Web service impersonating the client. The message may or may not be modified [7][18].

Vulnerabilities: - Vulnerabilities that can enable message replay include:

• Messages are not encrypted.
• Messages are not digitally signed to prevent tampering

Attacks: - The most common types of message replay attacks include:

• Basic replay attack. The attacker captures and copies a message, and then replays the same message to the client and impersonates the client. In this type of attack, the attacker does not require to know the content of the message.
• Man in the middle attack. The attacker captures the message that is transmitted through the communication channel and then changes some of its contents, for example, a shipping address, and then replays it to the Web service.
• Duplicate messages are not detected because no unique message ID is used

Countermeasures: - You can use the following countermeasures to address the threat of message replay:

• Use an encrypted communication channel, for example, SSL.
• Encrypt the message payload to provide message. Although this does not prevent basic replay attacks, it does prevent man in the middle attacks where the message contents are modified before being replayed.
• Use a unique message ID or nonce with each request to detect duplicates, and digitally sign the message to provide tamper proofing.

H. Probing attacks: - WS Probing attacks can be subdivided into two subclasses: WSDL Scanning and Parameter Tampering [21][22].

• WSDL Scanning. The WSDL document includes all of the operations that are available to the consumer; it is straightforward for a hacker to run through all of the operations with different message request patterns until a reach is identified.
• Parameter Tampering. The parameters of an operation are described within a WSDL document; the hacker can again “play around” with different parameter patterns in order to access unauthorized information. WSDL as an advertising mechanism for Web Services describes the methods and parameters used to access a specific Web Service, and in this way exposes the Web Service to possible attacks by providing a potential attacker initial information about how to access a specific web service[24].

Additionally, improper exceptions/errors handling by service hosting platform may provide important information about the internal service structure that can be used by attacker [24].

Vulnerability: -

• WSDL documents contain all functions that are available to consumers; they become an easy target for attackers [16].
• WSDL documents also have information about function parameters; hence, attackers can try to submit various parameters such as special characters in order to crash the implementation of WS [16].
• UDDI and UBR are used to find WS; however, they can also point to targets and provide all information needed to attack WS [16].
Attackers do not need to scan the Web in order to find vulnerable WS; they just go to any UBR and find targets. Such UDDI and UBR (UDDI Business Registry) attacks are rather dangerous because of difficulties to detect them [16].

I. CDATA Field Attacks: XML/XPath/SQL Injection Attacks:

Injection flaws, particularly SQL injection, are common in web applications. Injection occurs when user-supplied data is sent to an interpreter as part of a command or query. The attacker's hostile data tricks the interpreter into executing unintended commands or changing data ensuring that all inserted data is properly escaped[16][22][23].

Vulnerability: - vulnerability can occur when a poorly-written program uses user-provided data in a database query without first validating the input[25].

* Incorrectly filtered escape characters
* Incorrect type handling
* Blind SQL injection

Countermeasures: - countermeasures to prevent SQL injection include [25]:

* Perform thorough input validation. Your application should validate its input prior to sending a request to the database.
* Use parameterized stored procedures for database access to ensure that input strings are not treated as executable statements. If you cannot use stored procedures, use SQL parameters when you build SQL commands.
* Use least privileged accounts to connect to the database.

J. Cryptography

Most applications use cryptography to protect data and to ensure it remains private and unaltered. Top threats surrounding your application's use of cryptography include[19][20]:

* Poor key generation or key management
* Weak or custom encryption

Poor Key Generation or Key Management: -Attackers can decrypt encrypted data if they have access to the encryption key or can derive the encryption key. Attackers can discover a key if keys are managed poorly or if they were generated in a non-random fashion.

Countermeasures to address the threat of poor key generation and key management include:

* Use built-in encryption routines that include secure key management. Data Protection application programming interface (DPAPI) is an example of an encryption service provided on Windows 2000 and later operating systems where the operating system manages the key.
* Use strong random key generation functions and store the key in a restricted location — for example, in a registry key secured with a restricted ACL — if you use an encryption mechanism that requires you to generate or manage the key.
* Encrypt the encryption key using DPAPI for added security.
* Expire keys regularly.

Weak or Custom Encryption: -An encryption algorithm provides no security if the encryption is cracked or is vulnerable to brute force cracking. Custom algorithms are particularly vulnerable if they have not been tested. Instead, use published, well-known encryption algorithms that have withstood years of rigorous attacks and scrutiny.

Countermeasures that address the vulnerabilities of weak or custom encryption include:

* Do not develop your own custom algorithms.
* Use the proven cryptographic services provided by the platform.
* Stay informed about cracked algorithms and the techniques used to crack them.

K. Sniffing attacks
Sniffing or eavesdropping is the attack in which the attacker monitoring the traffic on the network or the communication channels for capturing the information such as plaintext passwords, PIN or configuration information of the system. In a simple packet sniffer, an attacker can easily read all the plaintext traffic. Also, the attackers can fissure the packets which are encrypted by the hashing algorithms and can decrypt the information that is to be confidential.

**Vulnerability:**
- Unencrypted channel
- Unencrypted message, message digests and hash function cannot be used.

**Countermeasures:**
To help prevent sniffing include [19][26]:
- Use strong physical security and proper segmenting of the network. This is the first step in preventing traffic from being collected locally.
- Encrypt communication fully, including authentication credentials. This prevents sniffed packets from being usable to an attacker. SSL and IPSec (Internet Protocol Security) are examples of encryption solutions.
- Active monitoring by IDSs for sniffing signatures (e.g. lost or delayed packets) and using encrypted traffic over all network connections [26].

V. Edifice to Secure ASP.NET Applications
Using Authentication, Authorization, and Secure Communication Process classified by the Microsoft

Web services often need to be able to authenticate their callers (other applications). One authentication technique is to require callers of the Web service to present a client certificate. There are two common situations where client certificates are used:
- An ASP.NET Web application calls a Web service that uses certificate authentication.
- An application running under interactive user context, such as a Windows Forms application, calls a Web service that uses certificate authentication.

The code that makes the Web service call is the same in both of these scenarios, although the way you install the certificate on the client computer may differ. You can install a client certificate in one of two certificate stores:
- The local machine store
- The user store

When a user installs the client certificate to access a particular Web service, by default this certificate is installed in that user's store.

**Local Machine Store**

When you install a client certificate in the local machine store, it is only accessible by the administrators group and the client who installed it. Therefore you have to install the access control list to grant the access to the authorized client. If you have multiple ASP.NET applications on the same Web server, you can run each application in a separate application band, using a different identity for each application, and grant access to the client certificate only to the identities that need it.

**Step 1: Install a Client Certificate in the Machine Store**

If you have a client certificate in a PKCS#12 (.pfx) file, you can use the Microsoft Windows HTTP Services (Win HTTP) certificate configuration tool to install the certificate and grant access to additional accounts, such as the Network Service account.

To install the certificate from a .pfx file
1. Download the WinHttpCertCfg.exe tool from Microsoft Download Center at http://go.microsoft.com/fwlink/?linkid=20506 and install it on the client computer.
2. Run the following command from a command window. Note that by default the WinHttpCertCfg utility is located in \program files\windows resource kit\tools.

   WinHttpCertCfg.exe -i PfxFile -c LOCAL_MACHINE\MY -p password

   In the above command, replace PfxFile with the name of your .pfx file. Replace password with the password of the .pfx file. If the .pfx file is not password protected, omit the -p argument.

**Step 2: Configure Access to the Certificate**

In this step, you configure the ACL on the certificate in the local machine store to allow the client application to access it. If your client is a Windows Forms application, you must grant access to the user account which is running the application. If your client is an ASP.NET application and you have not configured a custom account to run ASP.NET, you must grant access to the Network Service account. If you have configured a custom account to run ASP.NET, you must grant access to the custom account.

Use the following command to grant access to a specific user account:
WinHttpCertCfg.exe -g -c LOCAL_MACHINE\MY -s "Issued_To_name" -an "Account_name"

Replace Issued_To_name with the name to which the certificate was issued. Replace Account_name with the name of a local machine or domain account. This search string is not case sensitive. The string is used to find the first enumerated certificate with a subject name that contains this substring.

Step 3: Install the Root Certificate of the CA

If you have not already installed the root certificate of your CA, you must do so. The Microsoft Windows operating system includes the root certificates of many external CAs pre-installed in the Trusted Root Certification Authorities certificate store.

To check if the CA root certificate is installed
1. Run the Microsoft Management Console (MCC). On the File menu, click Add/Remove Snap-in.
2. Select the Certificates add-in. When asked to select which certificates the add-in will manage, select Computer account and then Local machine.
3. When you have added the Certificates snap-in, in the left pane of the MMC snap-in, expand Certificates (Local Computer).
4. Expand Trusted Root Certification Authorities, and then click Certificates.
5. Confirm that your CA certificate is listed.

If the CA certificate is not listed, you must install it. If the CA root certificate has been issued to you in a certificate file (for example, a .cer, .der, or .pfx file), perform the following steps:

To install the CA root certificate from a file
1. In the left pane of the MMC snap-in, expand Certificates (Local Computer).
2. Right-click Trusted Root Certification Authorities, and then click Import.
3. Use the Certificate Import wizard to import the certificate from the file.

VI. CONCLUSION

Security has become a prime concern in every field so as in Semantic Web Services. Threats which can violate the security of web services are examined in this paper and countermeasures to handle such threats are presented in the IV section. Moreover, day by day new threats are emerging and it is difficult to detect them. Therefore there is a requirement of prevention method which can detect and withstand all emerging attacks along with the defined attacks.

ACKNOWLEDGEMENT

We want to extend thanks to our “Prof. Suresh Kumar” for providing us all infrastructure and facilities to work in need and to do this research work.

REFERENCES