ISSUES AND CHALLENGES IN WEB SERVICES IN THE 21ST CENTURY

Ms. Priyanka Yadav*

Mrs. Karuna Nidhi Pandagre*

Abstract: Web Services Security (WS-Security) is the emerging security standard designed to address these issues. Web services are a widely touted technology that aims to provide tangible benefits to both business and IT. Their increasing use in the enterprise sector for the integration of distributed systems and business critical functions dictates the need for security assurance yet there is currently no security testing methodology specifically adapted to applications that implement web services. Web Service Enhancement (WSE) allows you to implement message level security solutions including authentication, encryption and digital signatures. In this paper we analyze the threats and security issues that can be related to the use of web services technology in a web application.

Keywords: Ws-Security, WSE, Digital signatures, authentication.

* Asst. Prof. Career College, Computer Department, Barkatullah University Bhopal (M.P.)

1 INTRODUCTION

Web services are used by an increasing number of companies as they expose products and services to customers and business partners through the Internet and corporate extranets. Microsoft has released Web Services Enhancements (WSE) 2.0 for Microsoft .NET 1.1 and WSE 3.0 for .NET 2.0, which supports WS-Security and a related family of emerging standards. The security requirements for these service providers are of paramount importance. In some cases, primarily intranet or extranet scenarios where you have a degree of control over both endpoints, the platform-based security services provided by the operating system and Internet Information Services (IIS) can be used to provide point-to-point security solutions. However, the message based architecture of Web services and the heterogeneous environments that span trust boundaries in which they are increasingly being used pose new challenges. These scenarios require security to be addressed at the message level to support cross-platform interoperability and routing through multiple intermediary nodes.

2 ISSUES IN WEB SERVICES

Quality of service (QoS) is a combination of several qualities or properties of a service, such as:

Availability: is the percentage of time that a service is operating.

Security: properties include the existence and type of authentication mechanisms the service offers, confidentiality and data integrity of messages exchanged, no repudiation of requests or messages, and resilience to denial-of service attacks.

Response time: is the time a service takes to respond to various types of requests. Response time is a function of load intensity, which can be measured in terms of arrival rates (such as requests per second) or number of concurrent requests. QoS takes into account not only the average response time, but also the percentile (95th percentile, for example) of the response

time.

Throughput: is the rate at which a service can process requests. QoS measures can include the maximum throughput or a function that describes how throughput varies with load intensity.

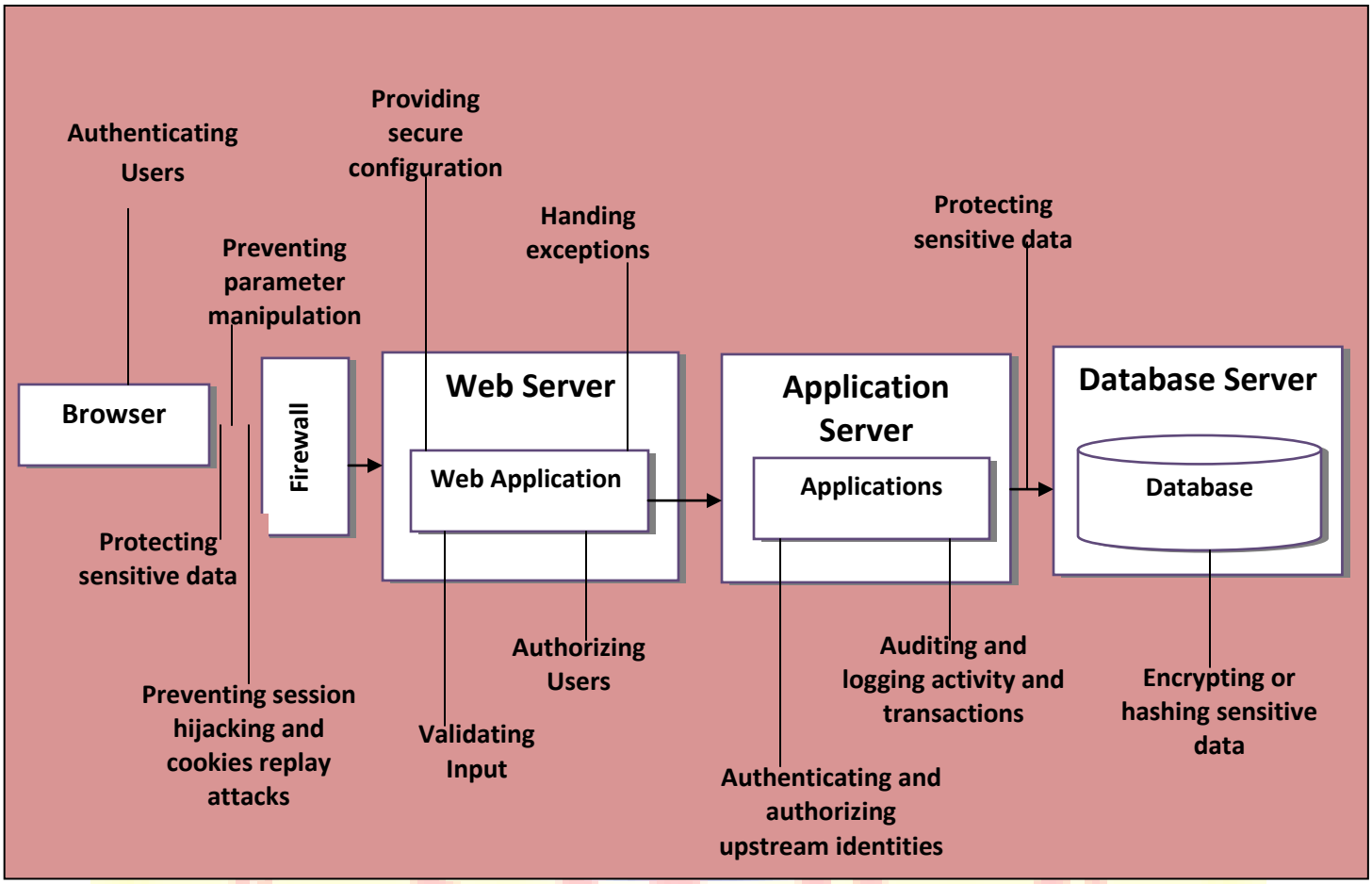
3 MAIN WEB SERVICES THREATS

Web services are a more and more common building block in modern web applications. Threat analysis of a web application can lead to a wide variety of identified threats. Some of these threats will be very specific to the application; others will be more related to the underlying infrastructural software, such as the web or application servers, the database, the directory server and so forth.

A web service is essentially an XML-messaging based interface to some computing resource. The web services protocol stack consists of:

- Some transport layer protocol, typically HTTP.
- An XML-based messaging layer protocol, typically SOAP [9]
- A service description layer protocol, typically WSDL [10]
- A service discovery layer protocol, typically UDDI [11]

Figure 3.1: Issues in web applications



I Unauthorized Access

Web services that provide sensitive or restricted information should authenticate and authorize their callers. Unauthorized Access is when a person who does not have permission to connect to or use a system gains entry in a manner unintended by the system owner. The popular term for this is "hacking".

Vulnerabilities

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

- No authentication used
- Passwords passed in plaintext
- Basic authentication used over an unencrypted communication channel

Countermeasures

You can use the following countermeasures to prevent unauthorized access:

- Use password digests
- Use Kerberos tickets
- Use X.509 certificates
- Use Windows authentication.
- Use Digital Certificates

Parameter Manipulation

Manipulating the data sent between the browser and the web application to an attacker's advantage has long been a simple but effective way to make applications do things in a way the user often shouldn't be able to. 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. No data sent to the browser can be relied upon to stay the same unless cryptographically protected at the application layer. Cryptographic protection in the transport layer (SSL) in no way protects one from attacks like parameter manipulation in which data is mangled before it hits the wire. Parameter tampering can often be done with:

- Cookies
- Form Fields
- URL Query Strings

- HTTP Headers

Example of Cookies manipulation from a real world example on a travel web site modified to protect the innocent (or stupid).

Cookie: lang=en-us; ADMIN=no; y=1 ; time=10:30GMT ;

The attacker can simply modify the cookie to; Cookie: lang=en-us; ADMIN=yes;

y=1; time=12:30GMT;

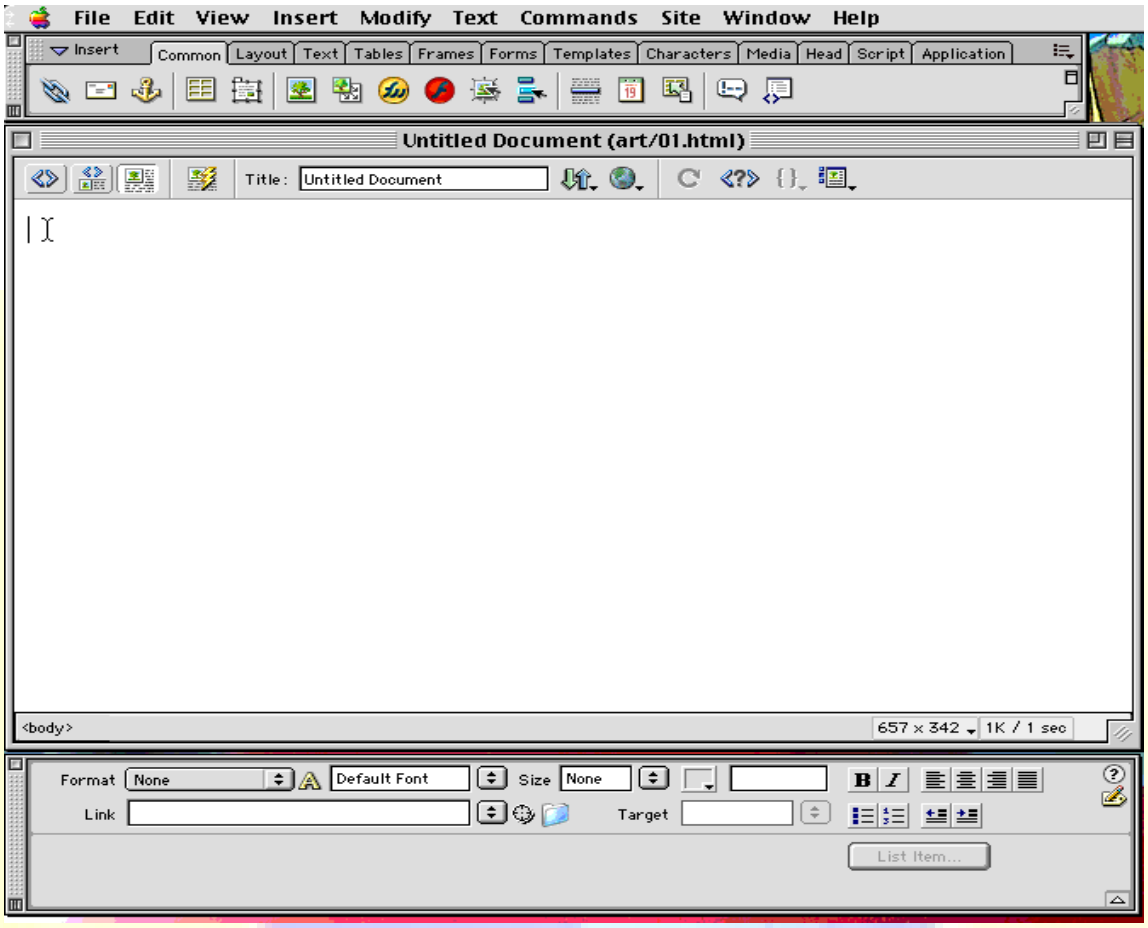
II HTTP Header Manipulation

HTTP headers are control information passed from web clients to web servers on HTTP requests and from web servers to web clients on HTTP responses. Each header normally consists of a single line of ASCII text with a name and a value. Sample headers from a POST request follow.

Host: www.someplace.org Pragma: no-cache Cache-Control: no-cache User-Agent:Lynx/2.8.4dev.9

Content-type: application/x-www-form- urlencoded Content-length: 49

Often HTTP headers are used by the browser and the web server software only. Most web applications pay no attention to them. However some web developers choose to inspect incoming headers, and in those cases it is important to realize that request headers originate at the client side, and they may thus be altered by an attacker. As an example an application uses a simple form to submit a username and password to a CGI for authentication using HTTP over SSL. The username and password form fields look like this.



Some developers try to prevent the user from entering long usernames and passwords by setting a form field value maxlength=(an integer) in the belief they will prevent the malicious user attempting to inject buffer overflows of overly long parameters. However the malicious user can simply save the page, remove the maxlength tag and reload the page in his browser. Other interesting form fields include disabled, readonly and value. As discussed earlier, data (and code) sent to clients must not be relied upon until in responses until it is vetted for sanity and correctness. Code sent to browsers is merely a set of suggestions and has no security value.

Countermeasures

You can use the following countermeasures to prevent parameter manipulation:

- Digital Signatures can be used to verify the users so that parameters are not tempered in transit.
- Encrypt the message payload to provide privacy.
-

III Network Eavesdropping

Network Eavesdropping or network sniffing is a network layer attack consisting of capturing packets from the network transmitted by others' computers and reading the data content in search of sensitive information like passwords, session tokens, or any kind of confidential information. The attack could be done using tools called network sniffers. These tools collect packets on the network and, depending on the quality of the tool, analyze the collected data like protocol decoders or stream reassembling. Depending on the network context, for the sniffing to be the effective, some conditions must be met:

• LAN environment with HUBs

This is the ideal case because the hub is a network repeater that duplicates every network frame received to all ports, so the attack is very simple to implement because no other condition must be met.

• LAN environment with switches

To be effective for eavesdropping, a preliminary condition must be met. Because a switch by default only transmits a frame to the port, a mechanism that will duplicate or will redirect the network packets to an evil system is necessary. For example, to duplicate traffic from one port to another port, a special configuration on the switch is necessary. To redirect the traffic

from one port to another, there must be a preliminary exploitation like the arp spoof attack. In this attack, the evil system acts like a router between the victim's communications, making it possible to sniff the exchanged packets.

• **WAN environment**

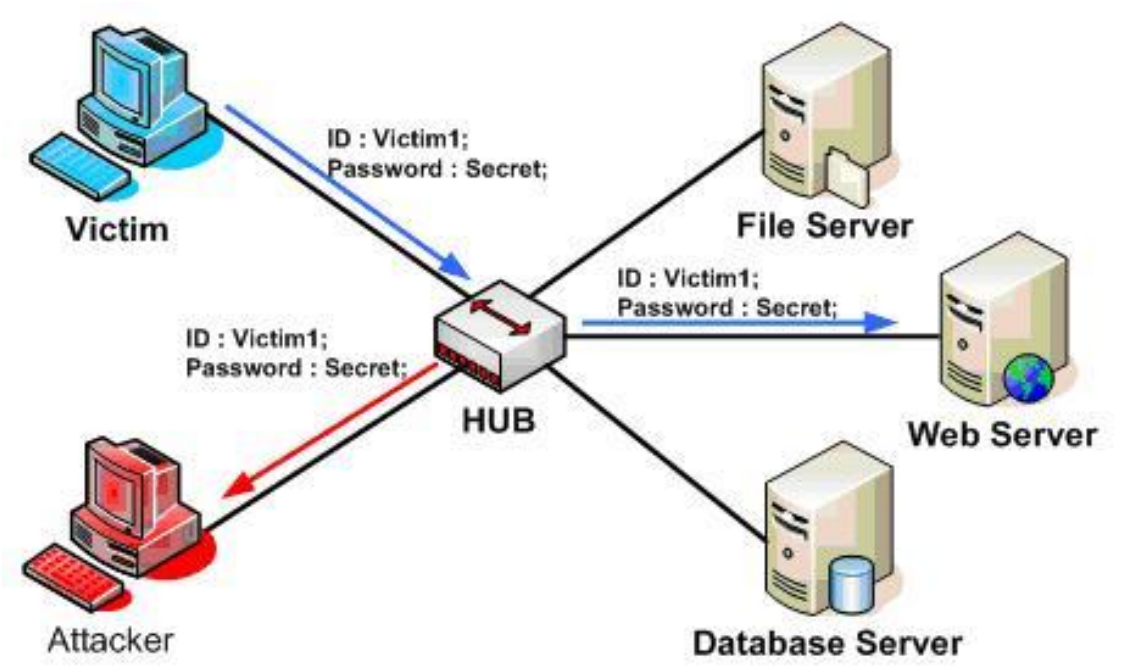
In this case, to make a network sniff it's necessary that the evil system becomes a router between the client server communications. One way to implement this exploit is with a DNS spoof attack to the client system. Network Eavesdropping is a passive attack which is very difficult to discover. It could be identified by the effect of the preliminary condition or, in some cases, by inducing the evil system to respond a fake request directed to the evil system IP but with the MAC address of a different system.

Examples

When a network device called a HUB is used on the Local Area Network topology, the Network Eavesdropping become easier because the device repeats all traffic received on one port to all other ports. Using a protocol analyzer, the attacker can capture all traffic on the LAN discovering sensitive information. With network eavesdropping, an attacker is able to view Web service messages as they flow across the network. For example, an attacker can use network monitoring software to retrieve sensitive data contained in a SOAP message. This might include sensitive application level data or credential information.

Figure 3.2:

Local Eavesdropping attack.



Vulnerabilities

Vulnerabilities that can enable successful network eavesdropping include:

- Credentials passed in plaintext
- No message level encryption used
- No transport level encryption used

Countermeasures

You can use the following countermeasures to protect sensitive messages as they flow across the network:

- Use transport level encryption such as SSL or IPSec. This is applicable only if you control both endpoints.
- Encrypt the message payload to provide privacy. This approach works in scenarios where your message travels through intermediary nodes route to the final destination.

IV Disclosure of Configuration Data [1]

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. This may not be desirable depending on your scenario.

Vulnerabilities

Vulnerabilities that can lead to the disclosure of configuration data include: Unrestricted WSDL files available for download from the Web server

A restricted Web service supports the dynamic generation of WSDL and allows unauthorized consumers to obtain Web service characteristics Weak exception handling.

Countermeasures

You can use the following countermeasures to prevent the unwanted disclosure of configuration data:

Authorize access to WSDL files using NTFS permissions.

- Remove WSDL files from Web server.
- Disable the documentation protocols to prevent the dynamic generation of WSDL.

- Capture exceptions and throw a **Soap Exception** or **SoapHeaderExc**eption that returns only minimal and harmless information — back to the client.

V Message Replay

Replay attack is a common kind of attack; the hackers are using to break the security of a web service. Web service messages can potentially travel through multiple intermediate servers. With a 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.

Vulnerabilities

Vulnerabilities that can enable message replay include:

- Messages are not encrypted
- Messages are not digitally signed to prevent tampering
- Duplicate messages are not detected because no unique message ID is used 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 and impersonates the client. This replay attack does not require the malicious user to know the contents of the message.

Man in the middle attack: The attacker captures the message and then changes some of its contents, for example, a shipping address, and then replays it to the Web service.

Countermeasures

In Web Sphere Application Server Versions 6 and later, when you enable integrity, confidentiality, and the associated tokens within a SOAP message, security is not guaranteed.

This list of security concerns is not complete. You must conduct your own security analysis for your environment.

Ensuring the message freshness

Message freshness involves protecting resources from a replay attack in which a message is captured and resent. Digital signatures, by themselves, cannot prevent a replay attack because a signed message can be captured and resent. It is recommended that you allow message recipients to detect message replay attacks when messages are exchanged through an open network.

You can use the following elements, which are described in the Web services security specifications, for this purpose:

- Timestamp
- Using XML digital signature and XML encryption properly to avoid a potential security hole
- Protecting the integrity of security tokens
- Verifying the certificate to leverage the certificate path verification and the certificate revocation list
- Protecting the username token with a password

VI Tampering [2]

The highest risk for tampering exists at the client side. An attacker can tamper with all assets residing on the client machine or traveling over the HTTP channel. This leads to the following threats that are considered most relevant in this category.

- A SOAP (Simple Object Access Protocol**)** message is replayed, leading to the unintended duplication of a server action or to inconsistencies on the server.
- A SOAP message is tampered with or maliciously constructed, leading to a whole variety of problems on the server side, such as information

VII Denial of service

In addition, sending a client a malicious assembly in a rich client scenario could do denial of service on that client. Also communication overload could be a threat. DoS attacks have been used as tools to make political statements [7] and extortions [8]. The latest high-profile DoS attacks against MasterCard, Visa, and other organizations linked to the late-2010 Wiki Leaks incident [9] only highlight the vulnerability and susceptibility of many organizations to DoS attacks. The increased use of web services technologies to deliver major governmental services (such as the Australian Standard Business Reporting (SBR) system1) and to enable cloud computing (including Amazon clouds2) only highlights the urgency of addressing the DoS problem in web services. Recent work [6] shows that flooding attacks are still an effective way to exhaust a web service provider's CPU resources. Most existing work has not addressed the resource imbalance issue that is the key to successful flooding-based DoS attacks.

DoS attacks on web services

A) Flooding Attack: **T**his attack attempts to exhaust a server's resources by sending a large amount of legitimate requests. The request messages in this case are well-formed and valid without any malicious XML structure or content. Consequently, such an attack cannot be detected by relying on a signature-based XML firewall. Normally, such an attack is mitigated through some forms of lower network layer packet analysis, such as IP address analysis.

B) Semantic Attack: Heavy Cryptographic Processing Attack: A well-known type of a web services semantic attack is the heavy cryptographic processing attack in which an attacker sends a payload with an oversized WS-Security header containing many cryptographic elements (such as nested encryption or a large number of digital signatures). The goal is to overload the server's resources, either through parsing a large security header or by forcing the server to process the numerous cryptographic directives.

CONCLUSION

Securing Web Services is a major concern while using the Web applications and Services. To provide security to Web application different Encryption techniques to encrypt the passwords and messages and Digital Signatures to authenticate the users so that unauthorized persons can't access the web services.

REFERENCES:

[1] msdn.microsoft.com/en- us/library/ff649028.aspx

[2]. Microsoft Patterns and Practices: Building Secure ASP. NET Applications, Microsoft Press, January 2003

[3]. W3C Note, Web Services Description Language (WSDL) 1.1, 15 March 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315/

[4]Web services security provides message integrity, confidentiality, and authentication [5]http://www.someplace.org/login.php

[6] S. Seriatim, A. Clark, and D. Schmidt, "Validating denial of service vulnerabilities in web services," in Network and System Security, International Conference on Network and System Security. IEEE Computer Society, 2010, pp. 175–182.

[7] J. Nayarit, "Political DDoS: Estonia and beyond," in USENIX Security '08. USENIX, July 2008,http://streaming.linux- magazin.de/events/usec08/tech/archive/jnazario/.

[8] J. Leyden, "Techwatch weathers DDoS extortion attack," The Register, 2009, http://www.theregister.co.uk/2009/01/30/ techwatch ddos/.

[9] J. Vijayan, "MasterCard Secure Code service impacted in attacks over Wiki Leaks," Computer World, 2010, http: //www.computerworld.com/s/article/9200541/MasterCard Secure Code service impacted in Attacks over Wiki Leaks.