Web Services Security

Topics

Wire level security

User authentication and authorization

WS-Security

Wire Level Security

Transport level security mechanisms generally independent of service type (SOAP, REST, etc) . Most of the focus here is on secure HTTP (HTTPS).

Peer Authentication

Transport level assurance that server and client are communicating with each other and not an imposter.

Confidentiality

Data from one side sent to another is encrypted strong enough so that someone intercepting the message cannot decrypt it.

Integrity

Both server and client need assurance that the messages being received are the same unaltered messages being sent.

Case Study: Amazon S3

- Paying clients are given unique secret keys which can be added in a query string or part of the http header.
- Parts of the message from clients is encoded to form a digital signature which is then used to validate the client and the client's message.
- The entire message is further encrypted over HTTPS.

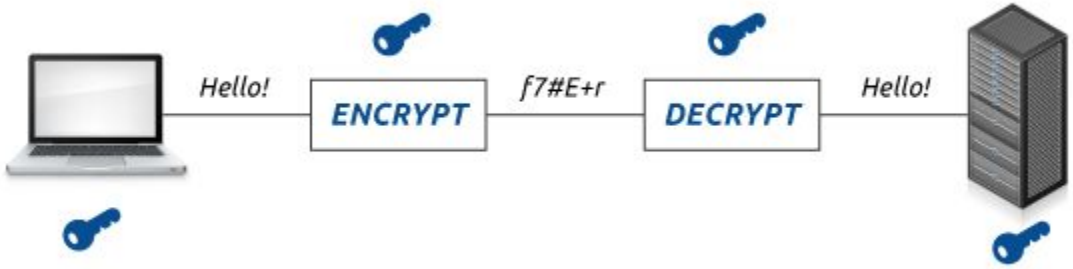
Secure HTTP

Provides the three levels of security previously mentioned : Peer authentication, Integrity, and Confidentiality. All are accomplished via encryption/decryption.

Generally served on port 443.

Symmetric Encryption

Symmetric encryption uses a single key to both encrypt and decrypt a message. It's fast but suffers from the *key distribution problem.*

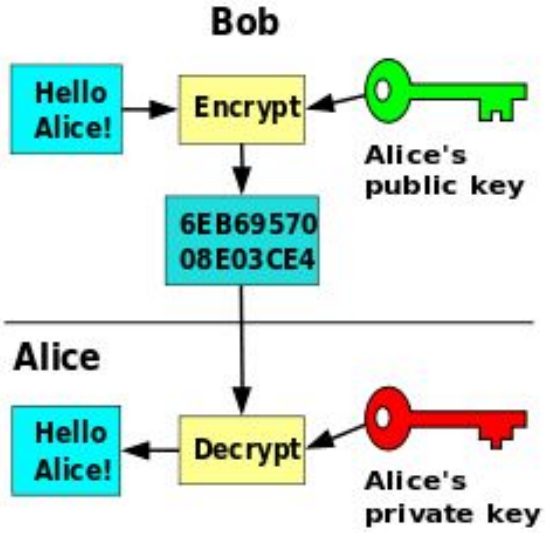


Asymmetric Encryption

The asymmetric approach uses a key pair to perform one way encryption and decryption.

The *private* key is safeguarded while the *public* key is freely distributed.

Roughly 1000 times slower than the symmetric approach.



Peer Authentication using HTTPS

Client challenges a server to authenticate itself. The server responds with its *digital certificate* (a public key) which is checked against a Certificate Authority.

Peer Authentication using HTTPS

The client then uses the server's public key to create a shared master key that is used to securely communicate, thus solving the *key distribution problem.*

User Authentication

Uses wire level security as a foundation.

Can be a user/pw combination or some form of digital certificate.

Authentication and Authorization

Step 1 (reqd): Client provides identification and credential. Success results in an *authenticated subject.*

Step 2 (optional): Role authorization determines access permissions of an authenticated subject.

Container-Managed Security

Managing security roles at the application level can be very cumbersome thus role authorization is generally handled at the container level (ie Tomcat, jetty, etc).

Security Realms

Tomcat uses the notion of *Realms* to specify where user roles and authentication information is stored.

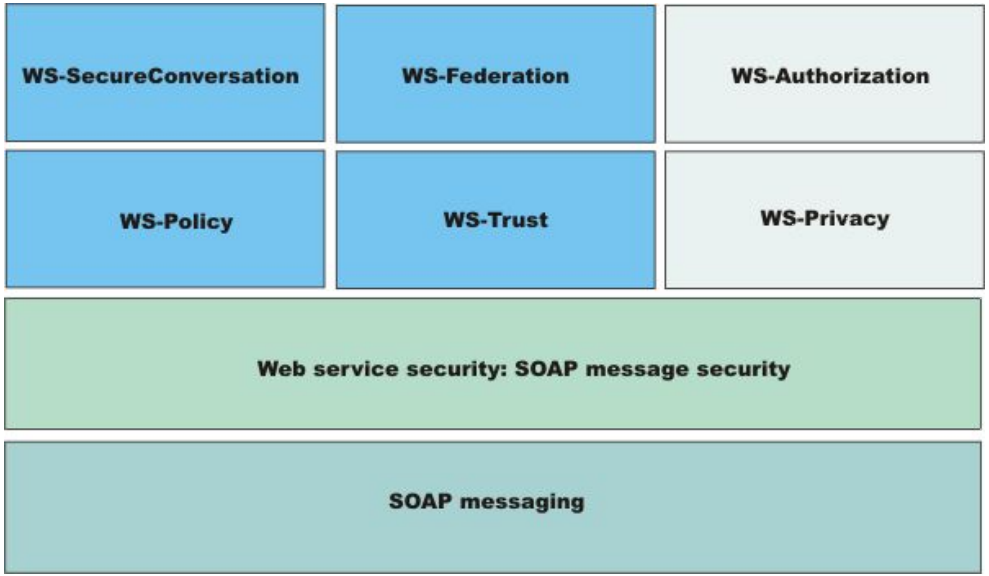
JDBCRealm, JNDIRealm, MemoryRealm, etc..

Password Digests

Instead of storing a password in plain text a proper service should store a password *digest*. This stores a hash of the password. When a client authenticates the sent pw is hashed and tested against the stored digest.

WS-Security

A family of specifications designed to augment both wire level and container-managed security.



WS-Policy

Describes general security capabilities and policies.

Does the service require tokens or a particular encryption algorythm?

WS-Trust

Who does the service trust to validate tokens or digital certificates?

WS-Privacy

Privacy policy enforcement

WS-SecureConversation

Secure web service conversations across different sites and contexts. Focus is on ow a security context is created and how keys are derived and exchanged.

WS-Federation

Addresses the challenge of managing security identities across different platforms and organizations.

WS-Authorization

Management of authorization data such as security tokens and underlying policies for granting access to secured resources.