Web Services

Web Service Security
<table>
<thead>
<tr>
<th>#</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Distributed Information Systems</td>
</tr>
<tr>
<td>2</td>
<td>Middleware</td>
</tr>
<tr>
<td>3</td>
<td>Web Technologies</td>
</tr>
<tr>
<td>4</td>
<td>Web Services</td>
</tr>
<tr>
<td>5</td>
<td>Basic Web Service Technologies</td>
</tr>
<tr>
<td>6</td>
<td>Web 2.0 Services</td>
</tr>
<tr>
<td>7</td>
<td><strong>Web Service Security</strong></td>
</tr>
</tbody>
</table>
Outline

• Motivation

• Technical solution
  – Basic techniques
  – Channel and data security
  – SOAP services security
  – Web 2.0 services security

• Summary

• Resources
Motivation
What is Information Security?

• Definition...
  – “The concepts, techniques, technical measures, and administrative measures used to protect information assets from deliberate or inadvertent unauthorized acquisition, damage, disclosure, manipulation, modification, loss, or use.”
  
    George McDaniel

• ...and “reality”:
  – “The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards – and even then I have my doubts.”

    Gene Spafford
“Security, like correctness, is not an add-on feature.”

Andrew S. Tanenbaum

“Security involves making sure things work, not in the presence of random faults, but in the face of an intelligent and malicious adversary trying to ensure that things fail in the worst possible way at the worst possible time... again and again. It truly is programming Satan’s computer.”

Bruce Schneier
How Much Security?

• There is no 100% security
  – Which does not mean that we shouldn’t bother about it...

• The security engineer needs to evaluate the security requirements of the case, the value of the information to be protected, the costs and drawbacks of security measures... and select adequate security measures
  – What can be adequate for my personal mailbox is probably not adequate for some military secrets...
Basic Security Aspects

- **Confidentiality**
  - Protect information from unauthorised disclosure
- **Integrity**
  - Ensure that information has not been changed by an unauthorised party
- **Authentication**
  - Ensure that the communicating party’s identity is what it is claimed to be (who are you?)
- **Authorisation**
  - Prevent unauthorised usage of resources (what can you do?)
- **Non-repudiation**
  - Ensure that parties cannot deny their previous actions
- **Availability**
  - Ensure that information/service is available when needed
Technical Solution
Basic Techniques
• **Encryption** is used to protect information from disclosure (it guarantees confidentiality)
  – Information is transformed so that unauthorised parties cannot understand it

• It is achieved using encryption algorithms
  – E.g. **AES** (Advanced Encryption Standard), **DES** (Data Encryption Standard – obsolete but still widespread)

• The algorithm uses a parameter called **key**
  – Often the same key is used to encrypt and decrypt, and therefore it must be secret (symmetric cryptography)
    • Public key cryptography can be used to establish a shared secret key between communicating parties (public key cryptography is computationally expensive)
  – The key must be difficult to guess, so it must be long (many bits)
Encryption

plaintext ➔ encryption ➔ ciphertext

encryption key ➔ decryption ➔ ABC

plaintext ➔ encryption ➔ ciphertext

ciphertext ➔ decryption ➔ plaintext

Encryption

plaintext ➔ encryption ➔ ciphertext

encryption key ➔ decryption ➔ ABC

plaintext ➔ encryption ➔ ciphertext

ciphertext ➔ decryption ➔ plaintext
Digital Signatures

- A digital signature guarantees integrity, sender authentication, non-repudiation
  - There are also simpler and more lightweight ways to achieve integrity and authentication, e.g. Message Authentication Codes
- It is generated using a digital signature algorithm
  - E.g. DSA (Digital Signature Algorithm), RSA (Rivest – Shamir – Adleman)
- The algorithm uses a key pair (public key + private key)
  - The private key is known only to its owner, and is used to sign
  - The public key is known to everybody, and is used to verify the signature
  - There is a mathematical relationship between the two keys
Digital Signatures

ABC → signature generation → ABC

document → signature generation → document + signature

sender’s private key

document + signature

sender’s public key

ABC → signature verification

ok

invalid
Certificates

• In order to verify a signature, the receiver needs the sender’s public key, and needs to be sure that that key really belongs to that person/entity
  – Be sure of the association between someone and his/her public key

• Therefore, a way to securely distribute and acquire public keys is needed

• Digital certificates are meant to address this problem
Certificates

- A **certificate** states the association between a subject (person or other entity) and a public key
  - The key and the identity of the subject are stated in the certificate
  - The most common standard for certificates is **X.509**
- A certificate is issued and signed by a **certificate authority (CA)**
  - The infrastructure needed to create, distribute, manage, revoke certificates is called **PKI** (Public Key Infrastructure)
- If the receiver trusts the CA that issued a certificate, he can believe that key really belongs to that subject
Certificates

This certificate has been verified for the following uses:

- SSL Client Certificate
- SSL Server Certificate
- Email Signer Certificate
- Email Recipient Certificate

**Issued To**
- Common Name (CN): Davide Cerri
- Organization (O): <Not Part Of Certificate>
- Organizational Unit (OU): <Not Part Of Certificate>
- Serial Number: 08:76:20

**Issued By**
- Common Name (CN): CA Cert Signing Authority
- Organization (O): Root CA
- Organizational Unit (OU): http://www.cacert.org

**Validity**
- Issued On: 06/04/10
- Expires On: 05/04/12

**Fingerprints**
- MD5 Fingerprint: C3:92:0A:E0:37:B5:84:02:1A:1D:F1:C9:3B:45:87:D4

**Certificate Hierarchy**
- CA Cert Signing Authority
  - Davide Cerri

**Certificate Fields**
- Certificate
  - Version
  - Serial Number
  - Certificate Signature Algorithm
  - Issuer
  - Validity
  - Subject
  - Subject Public Key Info
  - Subject Public Key Algorithm
  - Subject's Public Key

**Field Value**
- Modulus (2848 bits):
  - bf 5d b5 4d 64 14 56 f4 3d 65 5d b5 db ef 8d 5f 1c a0 93 01 5f
  - 64 73 9a da be 8a ff 7e 8d 08 7b 89 65 69 f5 44
  - e8 17 4b c6 32 40 af 3e a8 9a d5 a1 f2 f9 c9 ce
  - c9 6c 48 a5 a2 80 7f 80 7e 87 6f 97 98 84 52 9a
  - 76 a8 82 5e ec 9a 92 61 62 12 4d ea 86 a7 2a 85
  - bf 3c 2c b5 68 1e ed e4 59 3e c4 d1 ce 7c ef 9b
  - b7 ff e8 72 2f e5 f1 9c 12 30 71 3e 1f 5f 43 86 73
  - 45 49 b1 8a 38 85 00 22 7c 63 93 73 ea 7a 11 35

Export...
Technical Solution
Channel and XML Data Security
The **TLS** protocol (Transport Layer Security – IETF RFC 5246), together with its predecessor **SSL** (Secure Socket Layer – originally developed by Netscape), is the most common security solution to protect Web communication. It is still common to say “SSL” even if what is used today is mostly **TLS**.

**TLS/SSL** stays **between TCP and the application-layer protocol** (usually HTTP), providing:

- data flow confidentiality through encryption
- data flow integrity through message authentication codes
- endpoint authentication through public keys and certificates

**HTTPS** means HTTP over TLS/SSL.
Near the Wire: TLS/SSL

• TLS/SSL is well established and quite lightweight, but what it provides is basically a **secure TCP connection**

• This has important consequences:
  - TLS/SSL **protects “the wire”**, not the message
    • Once the data reach the endpoint of the connection, no security guarantee applies anymore
    • It is not possible to selectively protect only part of the message
  - TLS/SSL only **protects point-to-point (at TCP level) communications**
    • Endpoints of “the wire” (at TCP level), not of the message
    • If there are higher-level intermediaries, they can read and modify all data
    • Endpoint authentication is between pairs of TCP endpoints, which can differ from message-level endpoints (because of intermediaries)
Near the Wire: TLS/SSL

sender

intermediary

ultimate receiver
XML Data Security

• There are two complementary standards to protect XML data (W3C Recommendations):
  – **XML Signature**
    • Allows signing XML data (or any kind of data) and representing the signature in XML
    • Guarantees message integrity, authentication, non-repudiation
  – **XML Encryption**
    • Allows encrypting XML data (or any kind of data) and representing the result in XML
    • Guarantees confidentiality

• They operate on the **XML document content**
  – Protect the data, not the connection

• They can be **selectively applied** to part of the document
• With XML Signature, the signature and the signed data can be in the same document (enveloped or enveloping signature) or in separate documents (detached signature)
XML Signature example (detached signature)

```xml
<Signature Id="MyFirstSignature" xmlns=http://www.w3.org/2000/09/xmldsig#>
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http://www.w3.org/2006/12/xml-c14n11"/>
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
    <Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126/">
      <Transforms>
        <Transform Algorithm="http://www.w3.org/2006/12/xml-c14n11"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>dGhpcyBpcyBub3QgYSBz aWduYXR1cmUK.../DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>...</SignatureValue>
  <KeyInfo>
    <KeyValue>
      <DSAKeyValue>
        <P>...</P><Q>...</Q><G>...</G><Y>...</Y>
      </DSAKeyValue>
    </KeyValue>
  </KeyInfo>
</Signature>
```
XML Data Security

- XML Encryption example
  - Data to be protected are replaced by the `<EncryptedData>` element, which contains encrypted version of the data (plus some other information)

original

```xml
<PaymentInfo>
  <Name>John Smith</Name>
  <CreditCard>
    <Number>1234567890123456</Number>
    <Issuer>MyBank</Issuer>
    <Expiration>12/11</Expiration>
  </CreditCard>
</PaymentInfo>
```

with XML Encryption

```xml
<PaymentInfo>
  <Name>John Smith</Name>
  <EncryptedData>
    ...
  </EncryptedData>
</PaymentInfo>
```
Technical Solution
SOAP Services Security
• Information about the **identity** of the user/client is needed for authentication and authorisation

• X.509 certificates (or even username-password pairs) can be used to securely convey identity information, but this may be not enough
  – We may want to avoid the need for every service involved in the process to (be able to) directly check this information
  – Identity alone may not be relevant or enough for authorisation purposes, and other information (**user attributes**) may be needed
    • Attribute-Based Access Control (**ABAC**)
Identity: SAML

- **SAML** (Security Assertion Markup Language – OASIS standard) allows generating and exchanging *security assertions* about subjects between trusting parties
  - **Authentication statements**
    - State that the subject has been successfully authenticated at the specified time and using the specified method
    - Used for single sign-on
  - **Attribute statements**
    - State that the subject is associated with the specified attributes (expressed as name-value pairs)
    - Used for ABAC
  - **Authorisation decision statements**
    - State that the subject has (or has not) been granted access to the specified resource
    - Intentionally limited – XACML provides much more
Example structure of a SAML assertion

```xml
<Assertion>
  <Issuer> ... </Issuer>
  <ds:Signature> ... </ds:Signature>
  <Subject> ... </Subject>
  <Conditions NotBefore="2010-06-01" NotOnOrAfter="..."> ... </Conditions>
  <AuthnStatement> ... </AuthnStatement>
  <AttributeStatement>
    <Attribute Name="...">
      <AttributeValue> ... </AttributeValue>
    </Attribute>
    <Attribute> ... </Attribute>
    ...
  </AttributeStatement>
</Assertion>
```
Example: SAML Web single sign-on profile

Client ➔ Service ➔ SAML Authority

- Service request
- Authentication needed
- Authentication request
- Authentication assertion
- Authentication assertion
- Service response
• **WS-Security** (OASIS Standard) specifies how to **secure** a SOAP message
  – It leverages on other security standards as building blocks

• **WS-Security** allows the inclusion in a SOAP message of:
  – **Digital signatures**
    • Guarantee message integrity and origin
    • XML Signature is used

  – **Encrypted parts** (body blocks or header blocks)
    • Guarantee message confidentiality
    • XML Encryption is used

  – **Security tokens**
    • Assert “claims”, e.g. about identity or attributes of the client
• A **security token** contains a set of “claims” about some properties of an entity which are relevant for security purposes
  – E.g. identity or other attributes

• WS-Security supports **different types** of security tokens
  – E.g. username and password, X.509 certificates, SAML assertions

• **Signed tokens** are tokens “endorsed” by a third party acting as an authority
  – E.g. X.509 certificates and SAML assertions
WS-Security defines the `<wsse:Security>` header, which contains signatures and security tokens.

- A single SOAP message can contain multiple `<wsse:Security>` headers, addressed to different entities.
SOAP Message Security: Other Standards

• **WS-Trust**
  – Defines the concept of a *Security Token Service* (STS), and provides methods for issuing, renewing, and validating security tokens
  – OASIS Standard

• **WS-SecureConversation**
  – Allows to establish a *security context* that can be reused across multiple SOAP messages (for performance reasons), rather than securing single messages (the security context is a security token)
  – OASIS Standard
SOAP Message Security: Other Standards

• **WS-Policy**
  - Allows entities to advertise their **policy requirements or capabilities** through a set of policy assertions
  - Provides a **general framework** for any kind of policy (security, QoS, etc.)
  - W3C Recommendation

• **WS-SecurityPolicy**
  - Provides a **standardised set of policy assertions** for use with the WS-Policy framework, with respect to security features provided by WS-Security, WS-Trust, WS-SecureConversation
  - OASIS Standard
Authorisation: XACML

- **XACML** (eXtensible Access Control Markup Language) is an OASIS standard to represent **access control policies in XML**
- Policies consist of **rules** about subjects, resources, actions and environment
  - Authorisation is the permission given to a subject to perform an action on a resource
    - **Subject** is the entity that requests access
    - **Resource** is the entity to be accessed (data, service,...)
    - **Action** defines the type of access (e.g., read, write...)
    - **Environment** can optionally provide additional information
  - Subjects, resources, actions and environments are **described using attributes**
XACML data flow

PAP: Policy Administration Point
PDP: Policy Decision Point
PEP: Policy Enforcement Point
PIP: Policy Information Point

Figure from XACML specification
Technical Solution
Web 2.0 Services Security
REST Services and Security

• **REST lacks a well-articulated security model**
  – Moreover, the “idea” of being quick and easy does not help

• Basic techniques such as TLS/SSL can of course be applied, but they only address part of the problem

• Simple, ad-hoc solutions are often used

• Some “grassroot” efforts towards standards exist, to address some specific problems:
  – OpenID, to have a unified identity
  – OAuth, to delegate access rights (e.g. in mashups)
Identity: OpenID

- **OpenID** allows users to use a **unified identity on the Web**, rather than a separate identity for each service
  - The user id (“the OpenID”) is a URI
  - The user identity is managed by an **OpenID provider**, which can authenticate the user using some credential
    - Service providers such as Google, Yahoo, Flickr, Wordpress.com are OpenID providers
    - Standalone OpenID providers also exist
- The user **provides only his OpenID** to other services, which **rely on the OpenID provider** for authentication
  - The flow is very similar to SAML Web single sign-on profile
    - SAML has however a broader scope, and is designed to be extensible and used in different contexts
In Web 2.0 it is often the case that a service A needs to access another service/resource B on behalf of the user.

- If service B requires authentication, the user can be requested to give to service A his credentials to access service B, which is not a good practice.

**OAuth** (IETF Informational RFC 5849) gives the user the possibility to authorise service A to access service B on his behalf, without sharing his credentials to access service B with service A.

OAuth is currently supported, among others, by Twitter, Yahoo, Facebook.
Authorisation & Delegation: OAuth

- OAuth identifies three roles:
  - resource owner (user)
  - client (consumer)
  - server (service provider)

- The main steps are:
  - The client redirects the user to the server
  - The server asks the user if he wants to grant the client access to his data
  - If the user agrees, the client obtains a token that it can use to access the server
Summary
Summary

• Basic “building blocks”, not specific to Web services, can be used:
  – Protecting the communication channel: TLS/SSL
  – Protecting the data: XML Encryption & XML Signature

• For SOAP-based services there is a complex and comprehensive stack of specifications and standards dealing with security
  – WS-Security and related specifications, SAML, XACML

• For REST services there is no such framework
  – Some specifications for specific problems: OpenID, OAuth
Where is the Weakest Link?

From xkcd (http://xkcd.com/538/)
References
References

• Reference reading

• Wiki and Web references
    http://www.w3.org/TR/xmlsig-core/
    http://www.w3.org/TR/xmlenc-core/
    http://www.oasis-open.org/committees/security
    http://www.oasis-open.org/committees/wss
    http://www.oasis-open.org/committees/xacml/
    http://openid.net/
    http://oauth.net/
Questions?