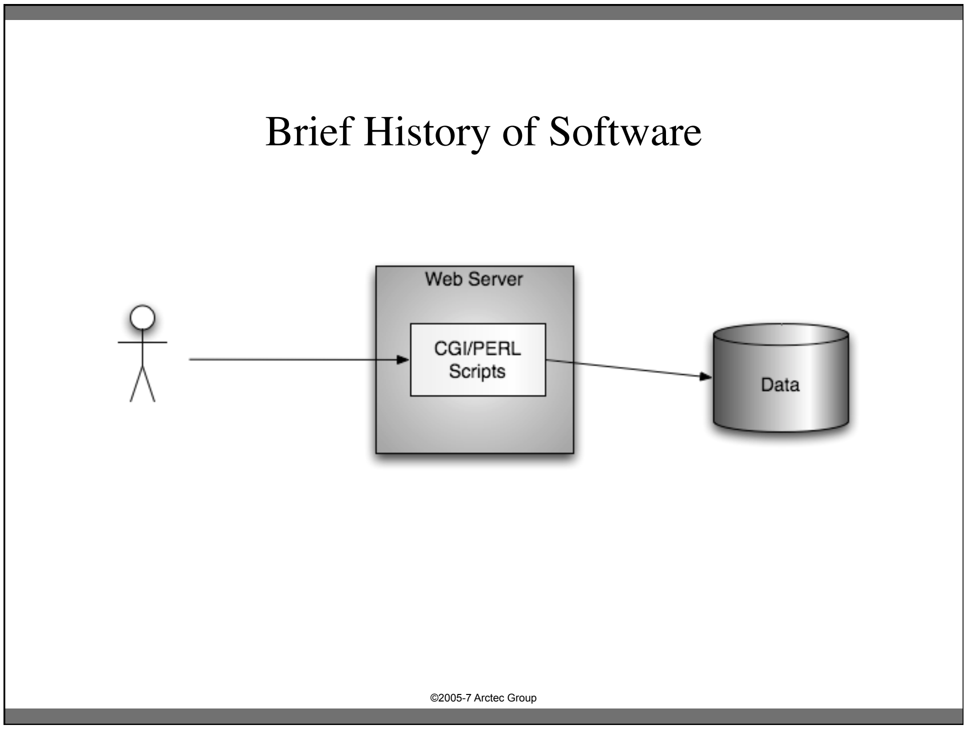
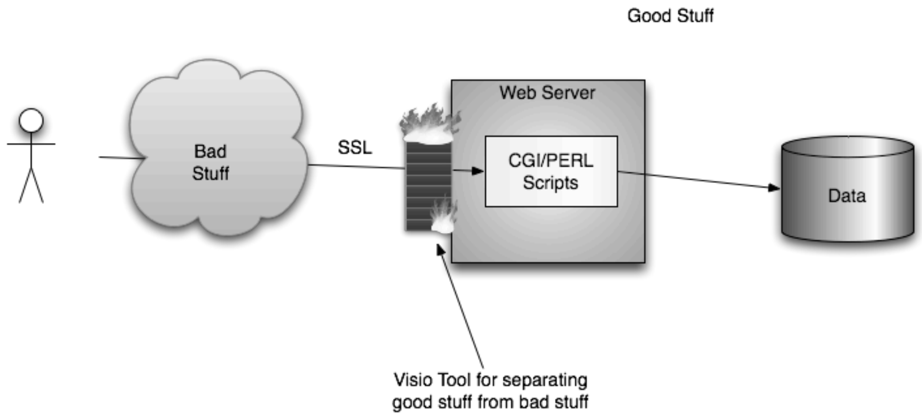
Web Services Security

Presentation by Gunnar Peterson
www.arctecgroup.net

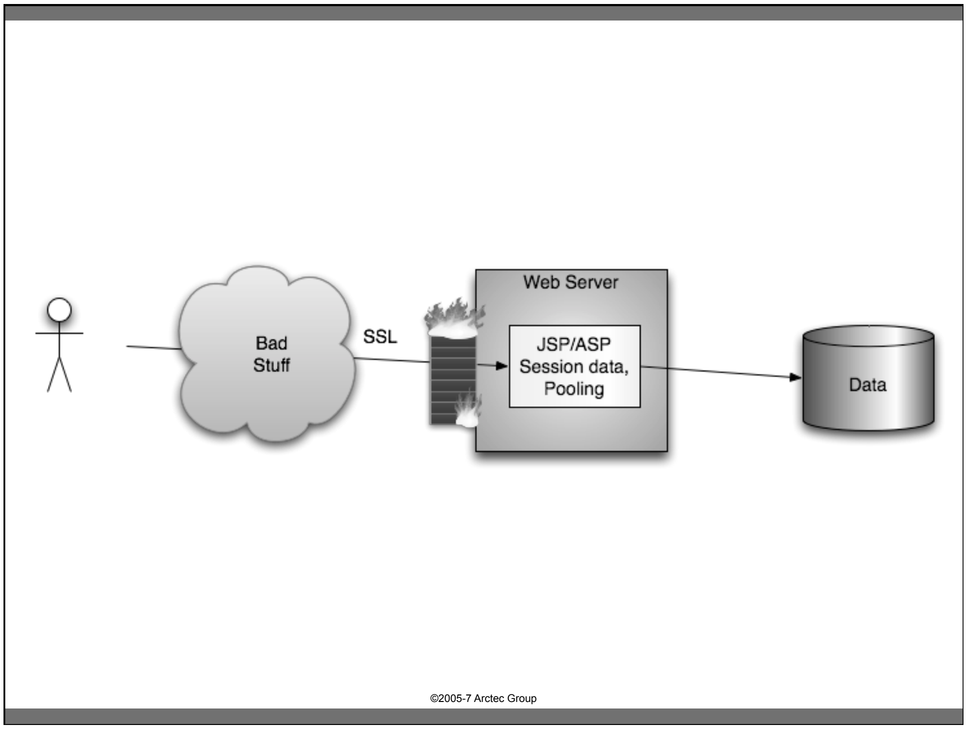


Brief History of Software



Mission Accomplished!

	Software	Security
1995	CGI/PERL	Network firewall & SSL



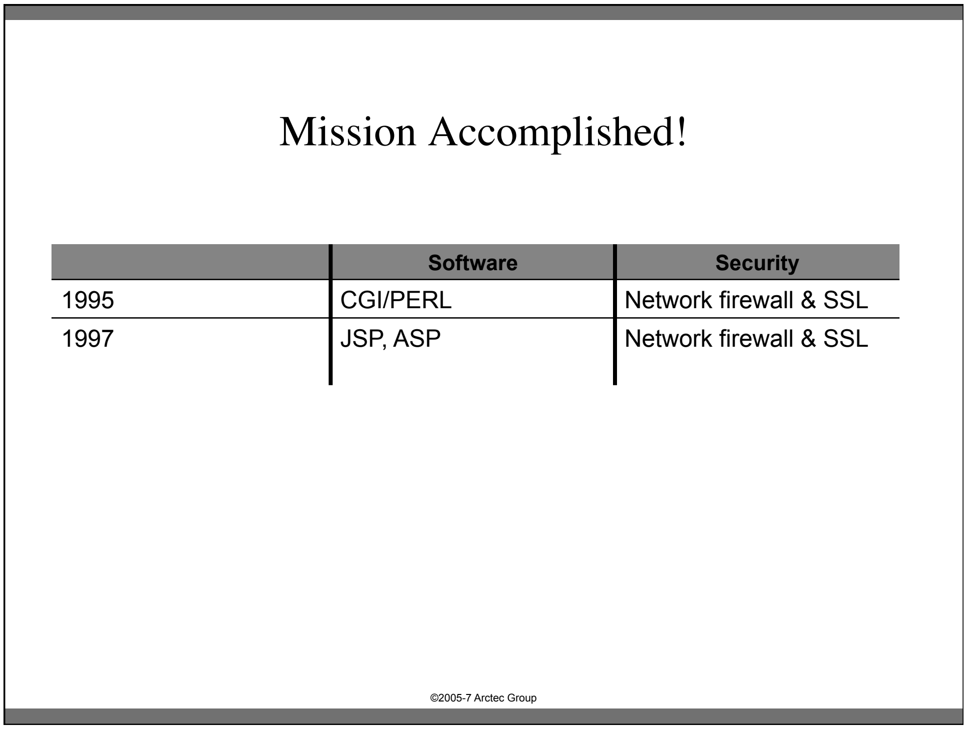
Bad Stuff

SSL

Web Server

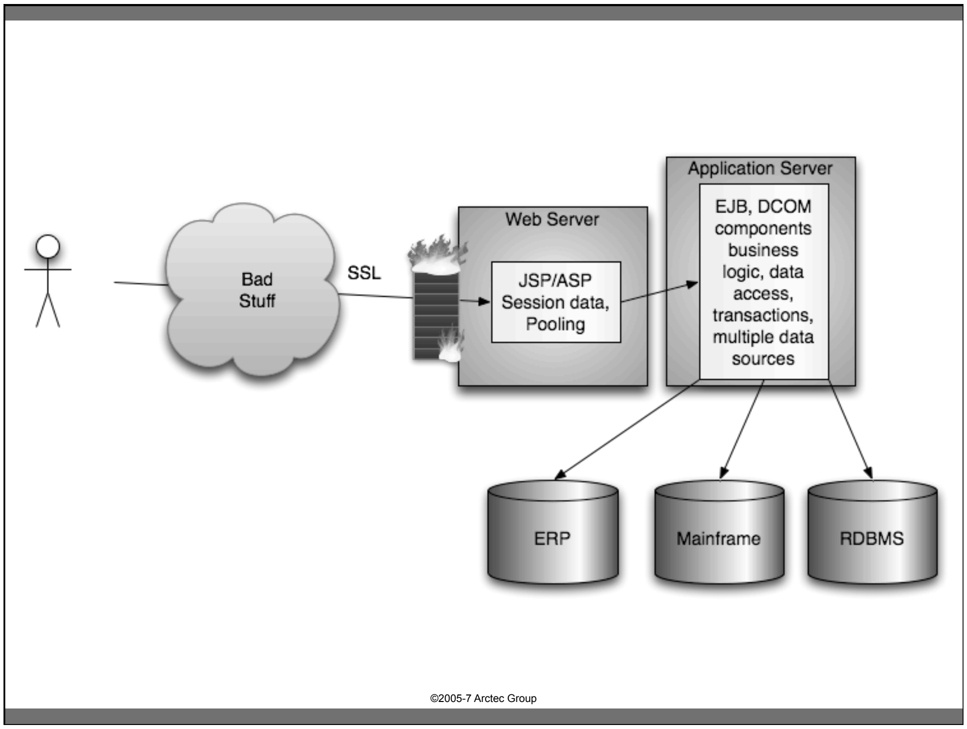
JSP/ASP Session data, Pooling

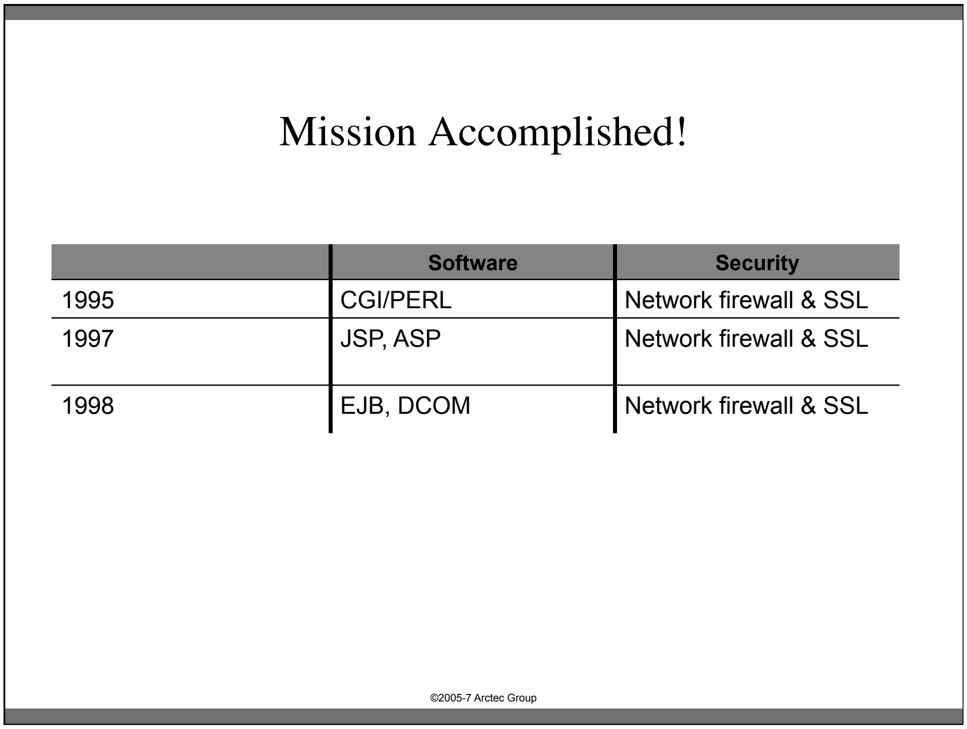
Data



Mission Accomplished!

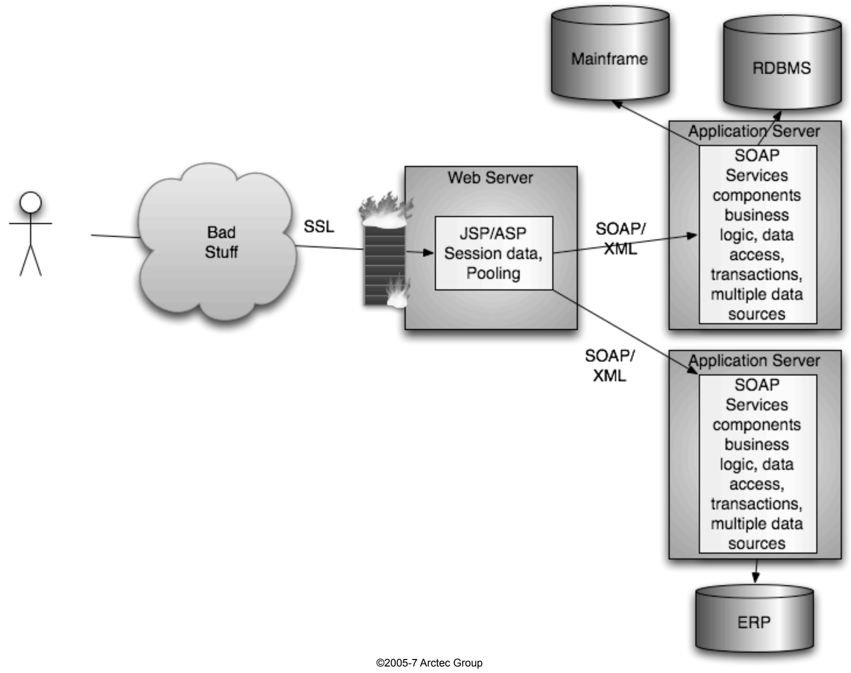
	Software	Security
1995	CGI/PERL	Network firewall & SSL
1997	JSP, ASP	Network firewall & SSL


Bad Stuff
SSL
Web Server
JSP/ASP Session data, Pooling
Application Server
EJB, DCOM components business logic, data access, transactions, multiple data sources
ERP
Mainframe
RDBMS



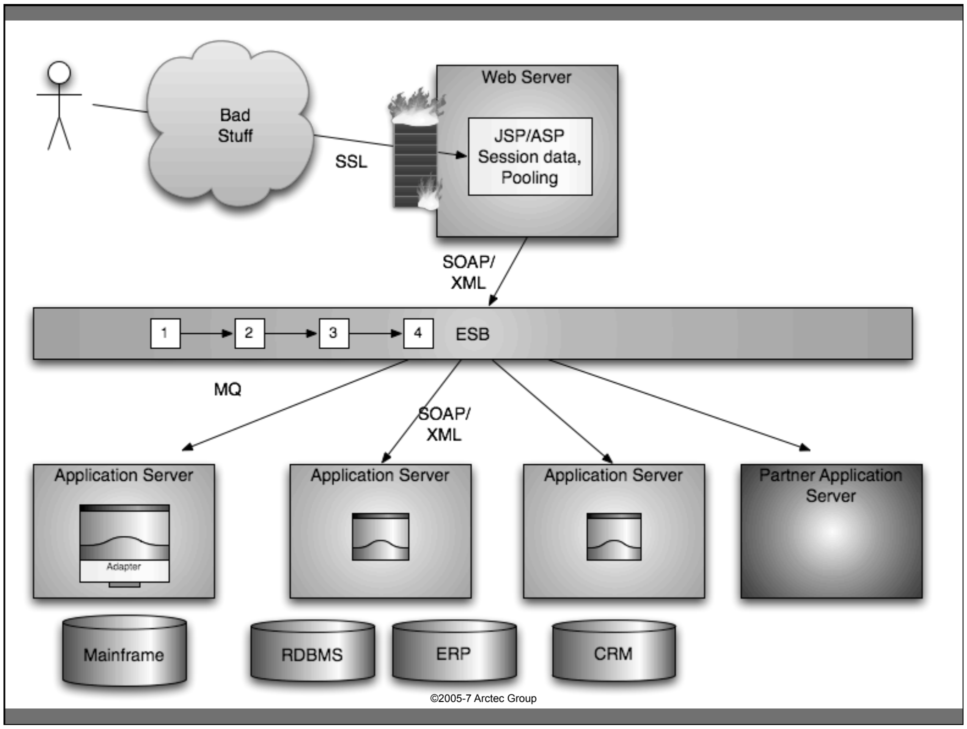
Mission Accomplished!

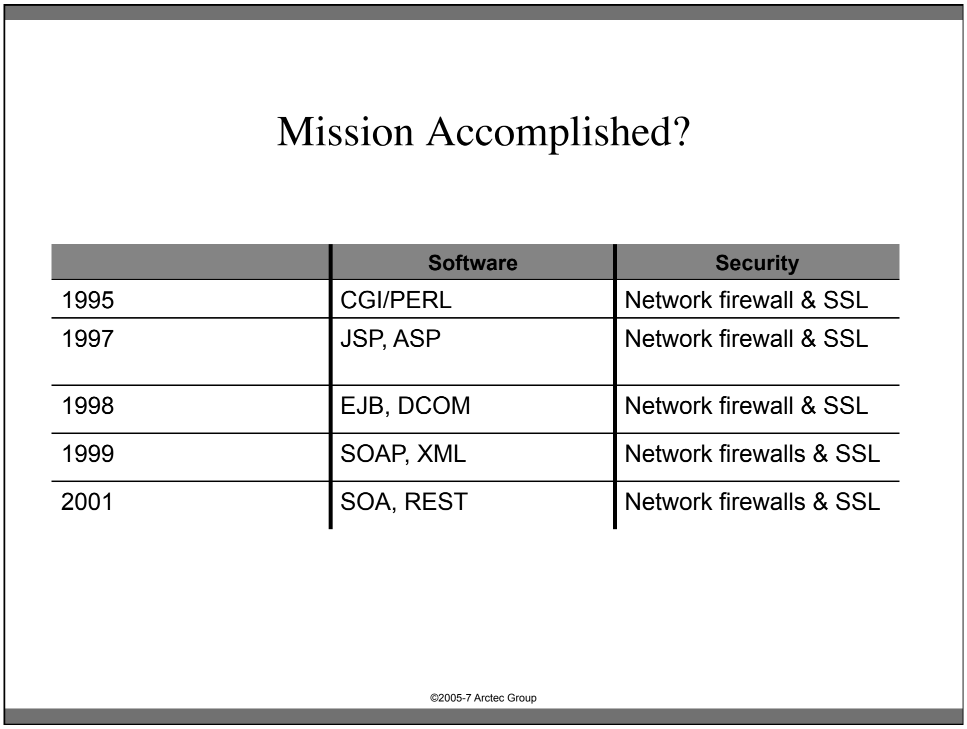
	Software	Security
1995	CGI/PERL	Network firewall & SSL
1997	JSP, ASP	Network firewall & SSL
1998	EJB, DCOM	Network firewall & SSL


Mainframe
RDBMS
Application Server
SOAP Services components business logic, data access, transactions, multiple data sources
Web Server
JSP/ASP Session data, Pooling
Bad Stuff
SSL
SOAP/ XML
SOAP/ XML
Application Server
SOAP Services components business logic, data access, transactions, multiple data sources
ERP

Mission Accomplished?

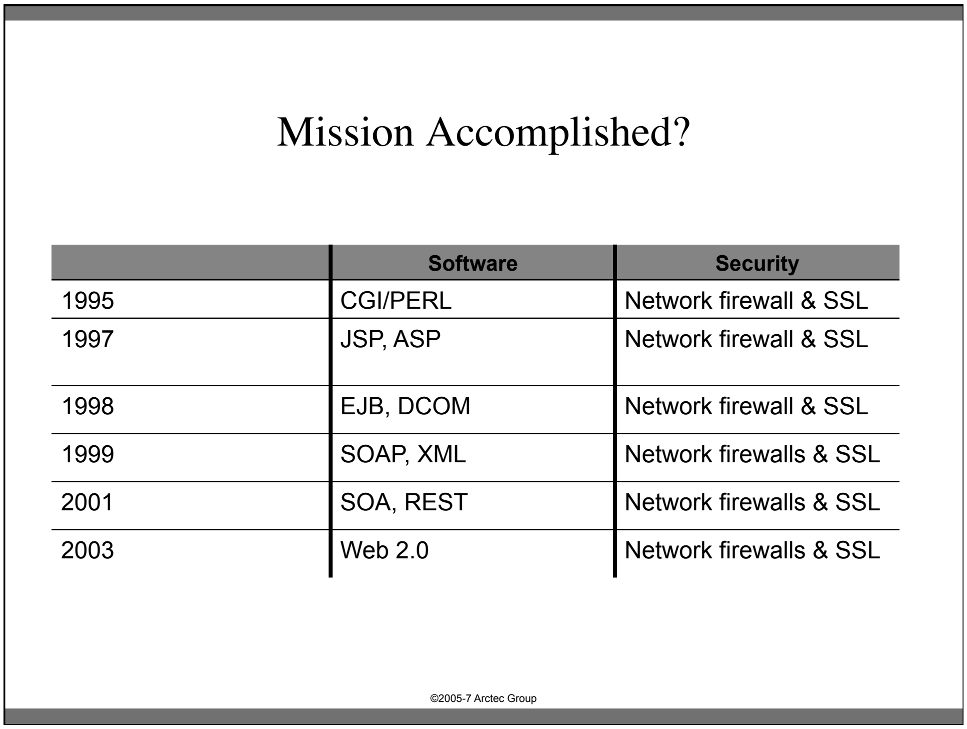
	Software	Security
1995	CGI/PERL	Network firewall & SSL
1997	JSP, ASP	Network firewall & SSL
1998	EJB, DCOM	Network firewall & SSL
1999	SOAP, XML	Network firewalls & SSL





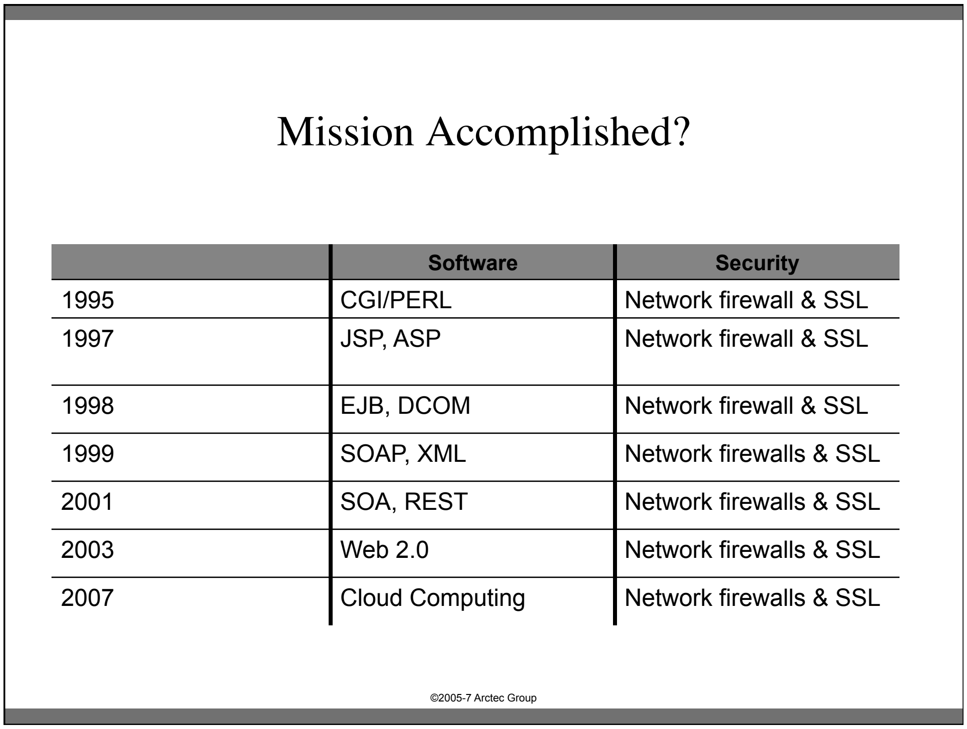
Mission Accomplished?

	Software	Security
1995	CGI/PERL	Network firewall & SSL
1997	JSP, ASP	Network firewall & SSL
1998	EJB, DCOM	Network firewall & SSL
1999	SOAP, XML	Network firewalls & SSL
2001	SOA, REST	Network firewalls & SSL



Mission Accomplished?

	Software	Security
1995	CGI/PERL	Network firewall & SSL
1997	JSP, ASP	Network firewall & SSL
1998	EJB, DCOM	Network firewall & SSL
1999	SOAP, XML	Network firewalls & SSL
2001	SOA, REST	Network firewalls & SSL
2003	Web 2.0	Network firewalls & SSL



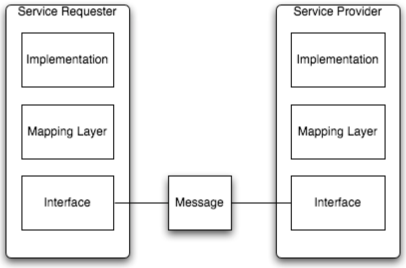
Mission Accomplished?

	Software	Security
1995	CGI/PERL	Network firewall & SSL
1997	JSP, ASP	Network firewall & SSL
1998	EJB, DCOM	Network firewall & SSL
1999	SOAP, XML	Network firewalls & SSL
2001	SOA, REST	Network firewalls & SSL
2003	Web 2.0	Network firewalls & SSL
2007	Cloud Computing	Network firewalls & SSL

Why Services?

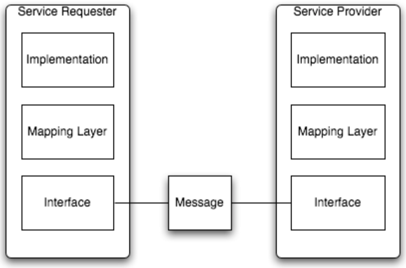
- Service Oriented Architecture goals
  - Virtualization - connect Bangalore, Beijing, and Bloomington
  - Interoperability - get Java, .Net working together
  - Reusability - how many claims/pricing/order mgmt systems does one company need?

High level view of services



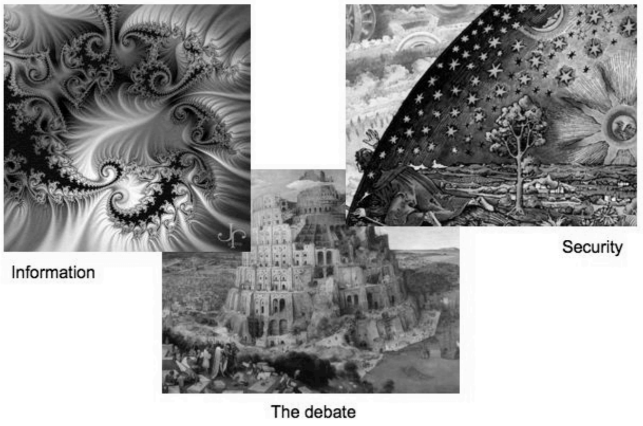
- Service interface
  - Method signature, types, values
- Mapping layer
  - Mapping message to runtime implementation types and values
- Implementation
  - Application logic
- Message
  - Data payload

Lions, Tigers, and Port 80, Oh My!



- First came SOAP - invented as a firewall friendly protocol
- Bruce Schneier: “calling SOAP firewall friendly is like skull friendly bullet”

Information Security: A new oxymoron

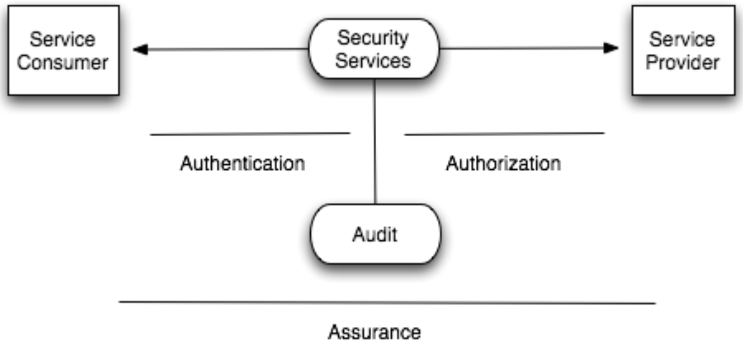


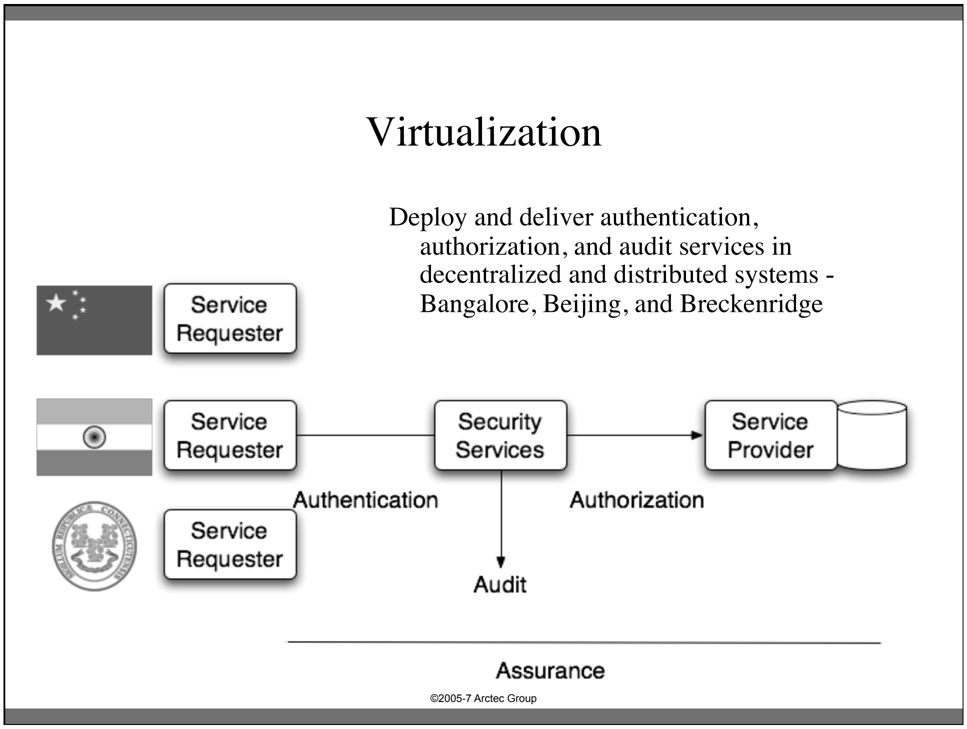
Source: Robert Garigue http://1raindrop.typepad.com/1_raindrop/2007/02/thinking_about_.html

Security Goals

- Security as a Service
  - Virtualization
  - Interoperability
  - Reusability

Security Mechanisms





Virtualization

Deploy and deliver authentication, authorization, and audit services in decentralized and distributed systems - Bangalore, Beijing, and Breckenridge

Service Requester

Service Requester

Service Requester

Security Services

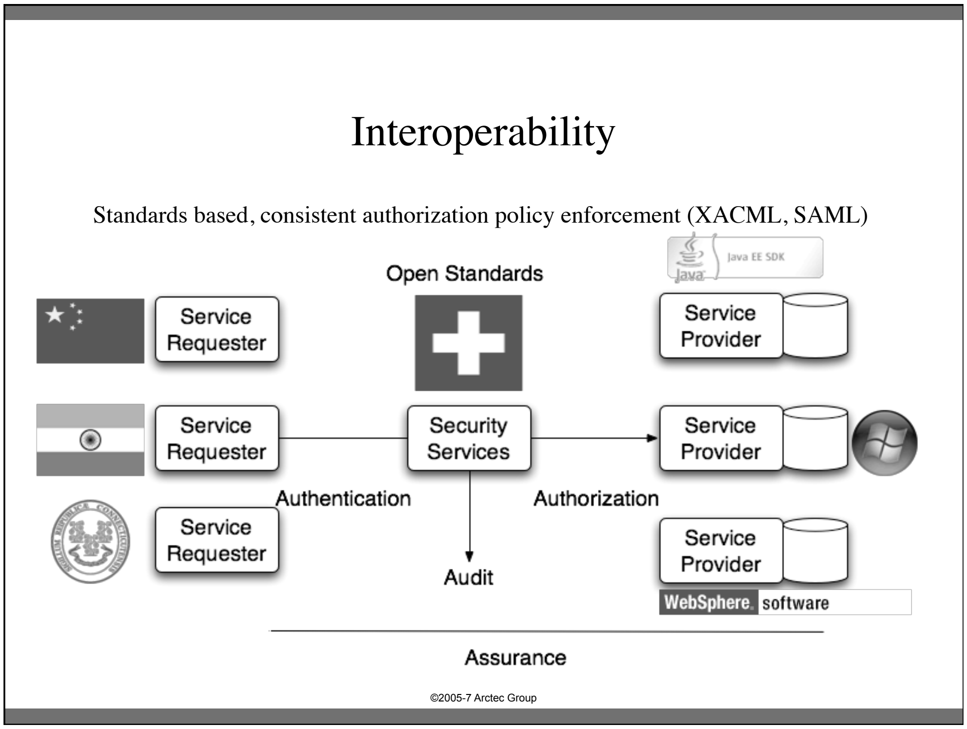
Service Provider

Authentication

Authorization

Audit

Assurance



Interoperability

Standards based, consistent authorization policy enforcement (XACML, SAML)

Open Standards

Java EE SDK

Service Requester

Service Requester

Service Requester

Security Services

Service Provider

Service Provider

Service Provider

WebSphere. software

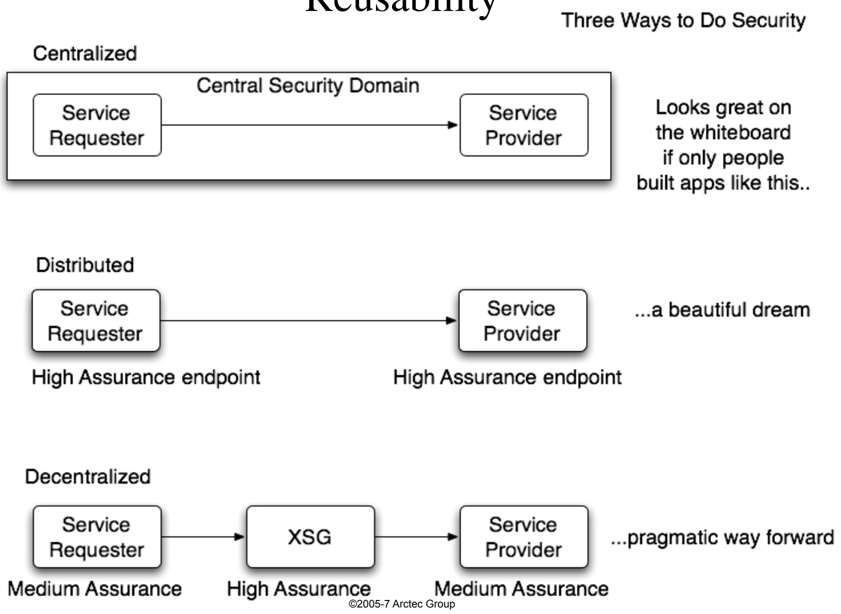
Authentication

Authorization

Audit

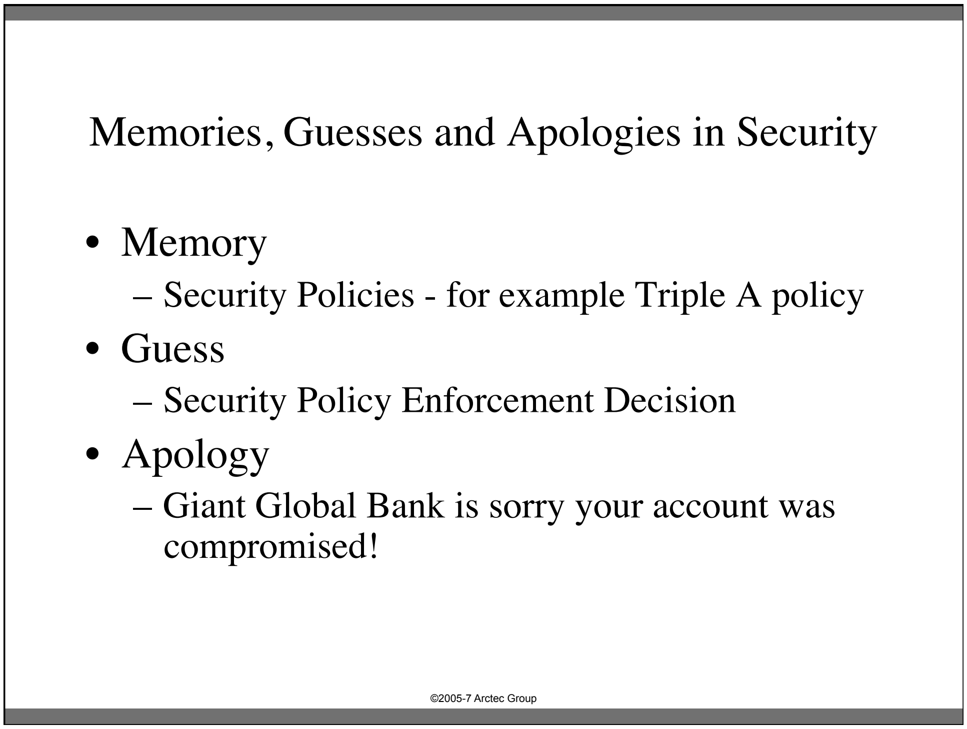
Assurance

Reusability



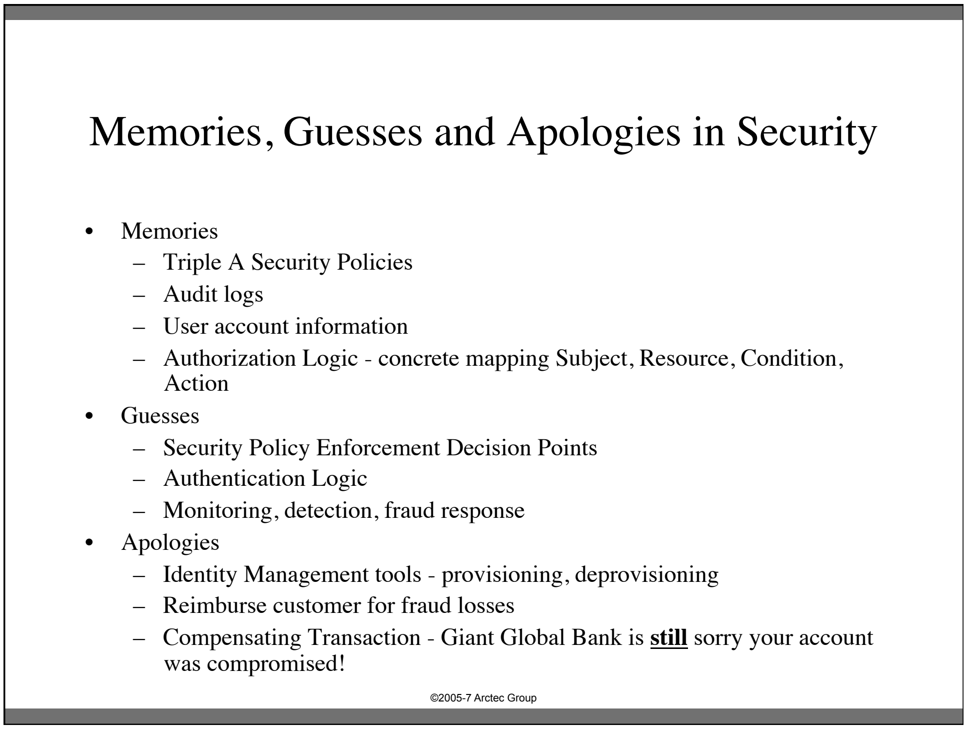
Service Oriented Security in Einstein's Universe

- Mainframes are Newton's world
  - The computer
  - The price
  - The record
- Distributed computing is Einstein's world
  - Pat Helland: Computers don't make decisions, computers *try* to make decisions.
  - Its all about Memories, Guesses and Apologies
  - Security mechanisms don't make policy-based decisions, security mechanisms *try* to make policy-based decisions



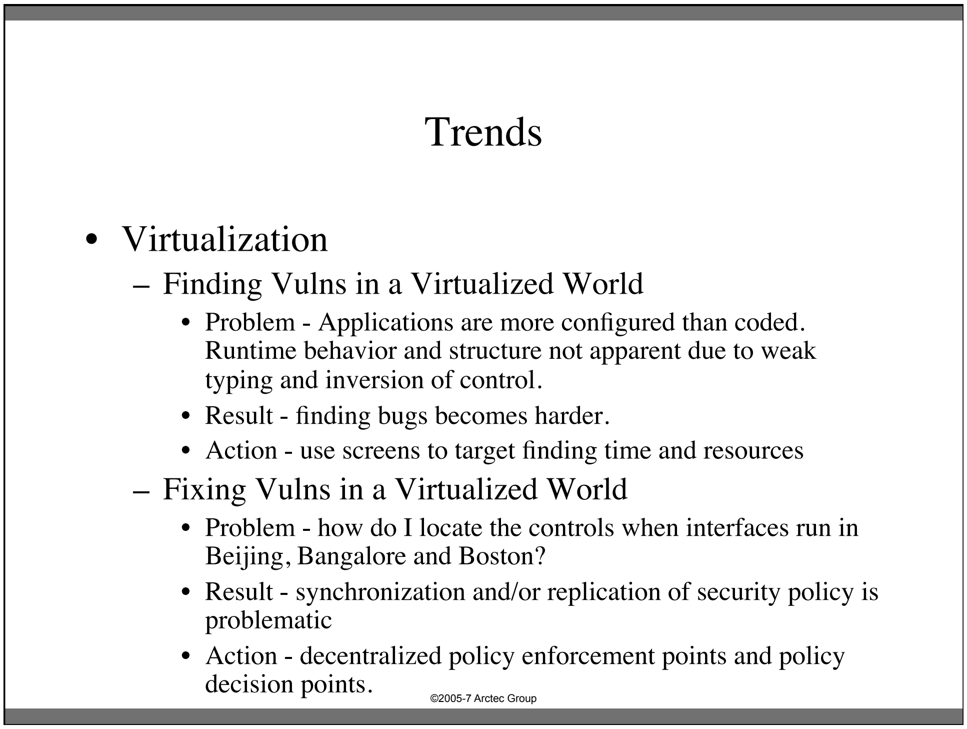
Memories, Guesses and Apologies in Security

- Memory
  - Security Policies - for example Triple A policy
- Guess
  - Security Policy Enforcement Decision
- Apology
  - Giant Global Bank is sorry your account was compromised!



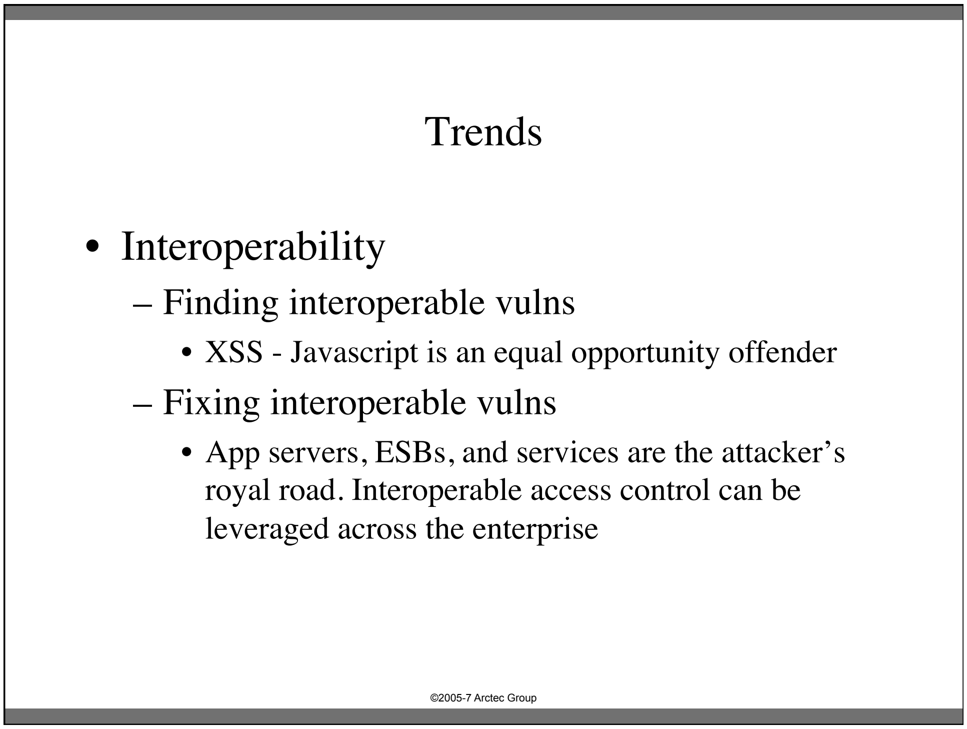
Memories, Guesses and Apologies in Security

- Memories
  - Triple A Security Policies
  - Audit logs
  - User account information
  - Authorization Logic - concrete mapping Subject, Resource, Condition, Action
- Guesses
  - Security Policy Enforcement Decision Points
  - Authentication Logic
  - Monitoring, detection, fraud response
- Apologies
  - Identity Management tools - provisioning, deprovisioning
  - Reimburse customer for fraud losses
  - Compensating Transaction - Giant Global Bank is **still** sorry your account was compromised!



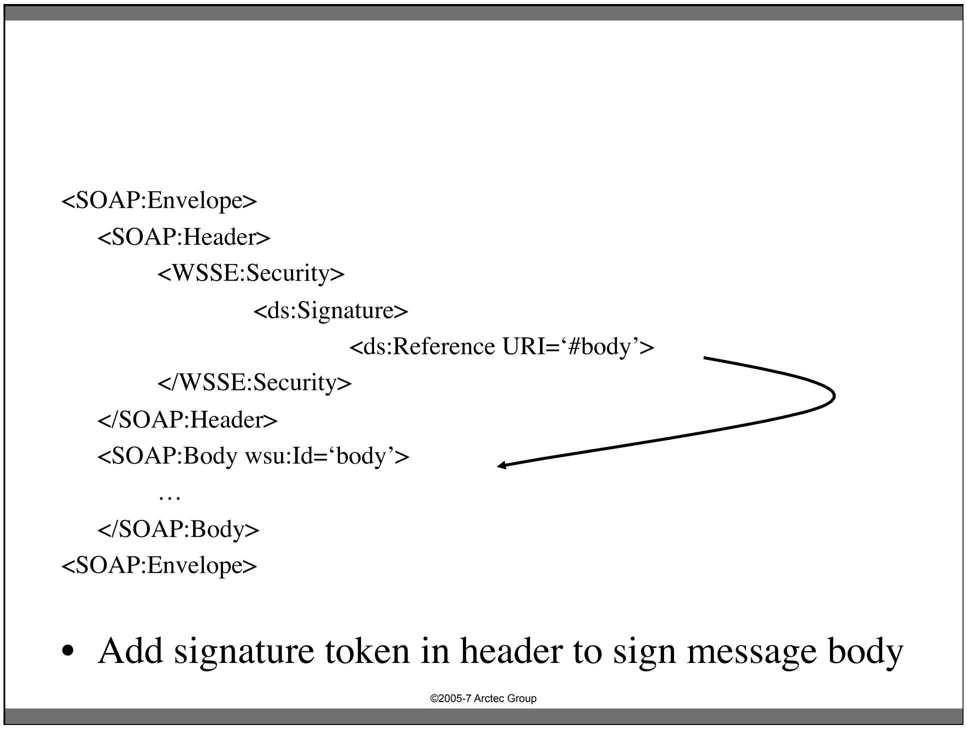
Trends

- Virtualization
  - Finding Vulns in a Virtualized World
    - Problem - Applications are more configured than coded. Runtime behavior and structure not apparent due to weak typing and inversion of control.
    - Result - finding bugs becomes harder.
    - Action - use screens to target finding time and resources
  - Fixing Vulns in a Virtualized World
    - Problem - how do I locate the controls when interfaces run in Beijing, Bangalore and Boston?
    - Result - synchronization and/or replication of security policy is problematic
    - Action - decentralized policy enforcement points and policy decision points.



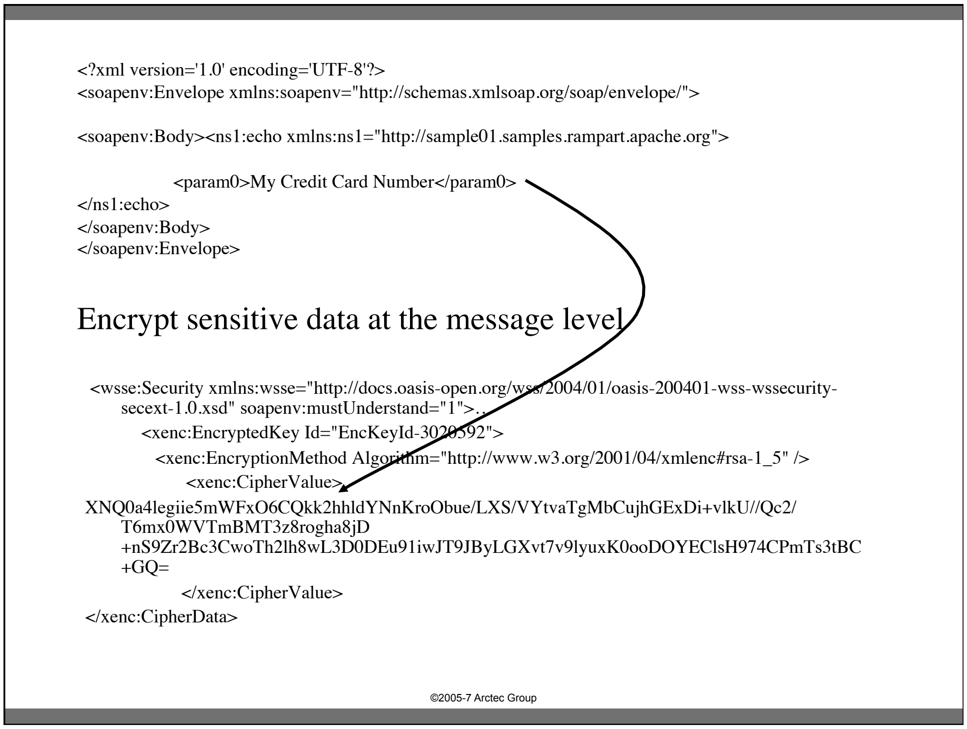
Trends

- Interoperability
  - Finding interoperable vulns
    - XSS - Javascript is an equal opportunity offender
  - Fixing interoperable vulns
    - App servers, ESBs, and services are the attacker’s royal road. Interoperable access control can be leveraged across the enterprise



```
<SOAP:Envelope>
    <SOAP:Header>
          <WSSE:Security>
                    <ds:Signature>
                              <ds:Reference URI=‘#body’>
          </WSSE:Security>
    </SOAP:Header>
    <SOAP:Body wsu:Id=‘body’>
          …
    </SOAP:Body>
<SOAP:Envelope>
```

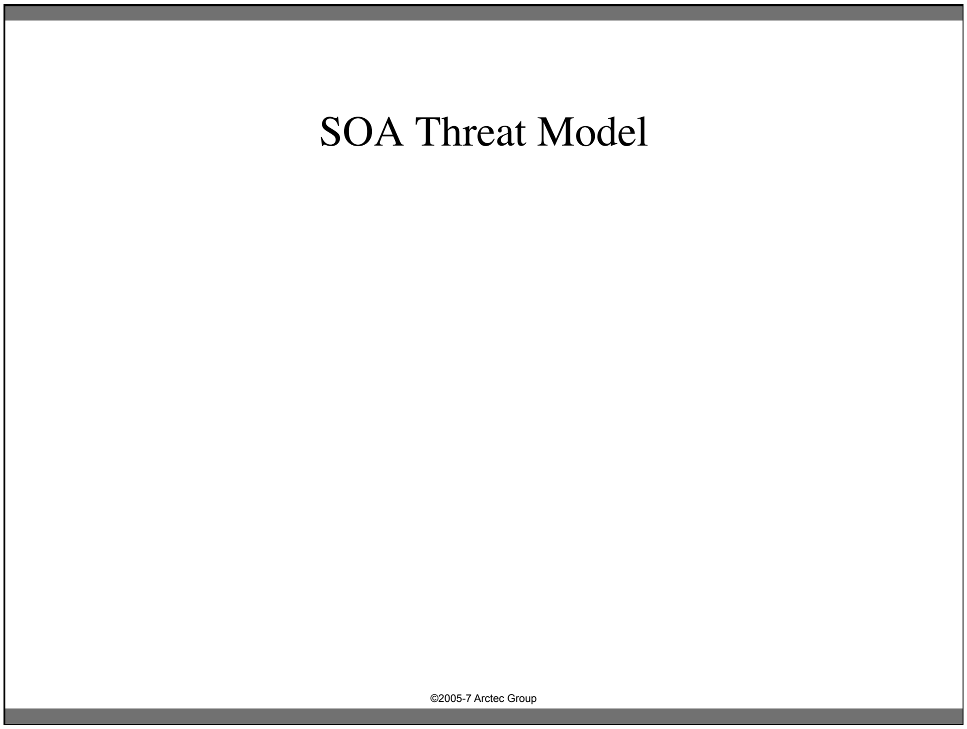
- Add signature token in header to sign message body



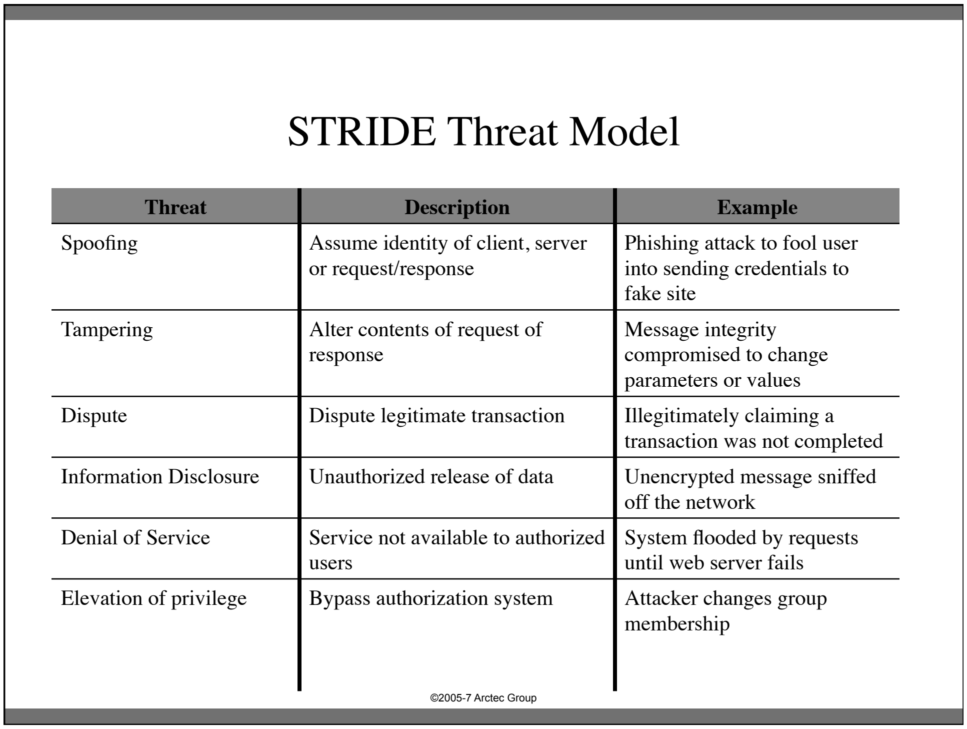
```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

<soapenv:Body><ns1:echo xmlns:ns1="http://sample01.samples.rampart.apache.org">

          <param0>My Credit Card Number</param0>
</ns1:echo>
</soapenv:Body>
</soapenv:Envelope>
```

Encrypt sensitive data at the message level

```
 <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
    secext-1.0.xsd" soapenv:mustUnderstand="1">..
      <xenc:EncryptedKey Id="EncKeyId-3020592">
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
           <xenc:CipherValue>
XNQ0a4legiie5mWFxO6CQkk2hhldYNnKroObue/LXS/VYtvaTgMbCujhGExDi+vlkU//Qc2/
    T6mx0WVTmBMT3z8rogha8jD
    +nS9Zr2Bc3CwoTh2lh8wL3D0DEu91iwJT9JByLGXvt7v9lyuxK0ooDOYEClsH974CPmTs3tBC
    +GQ=
           </xenc:CipherValue>
 </xenc:CipherData>
```

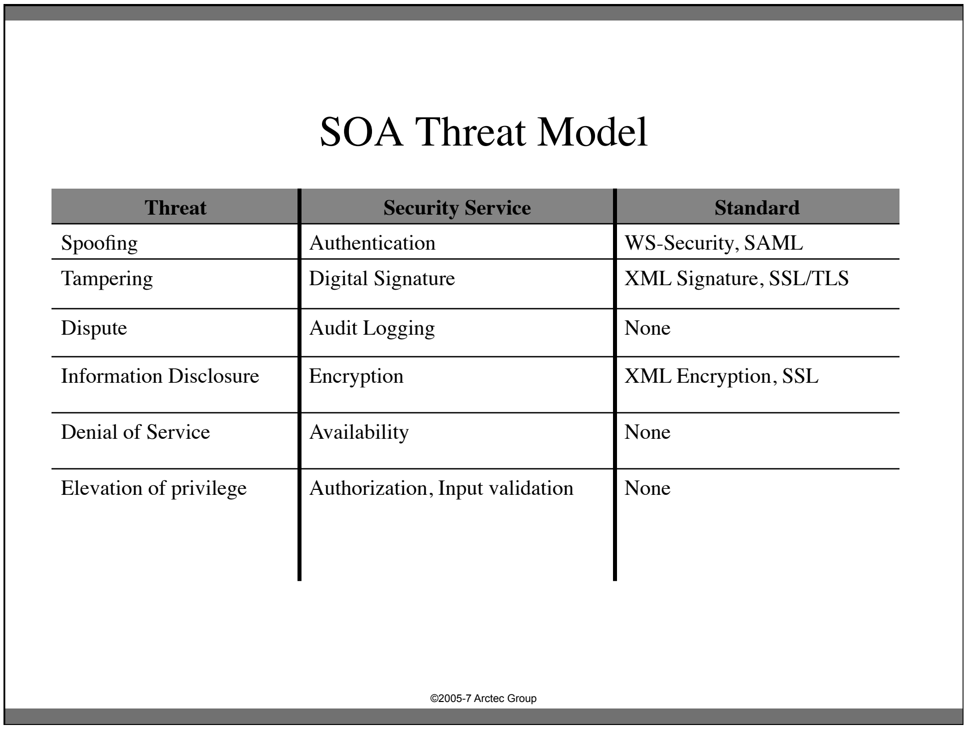


SOA Threat Model



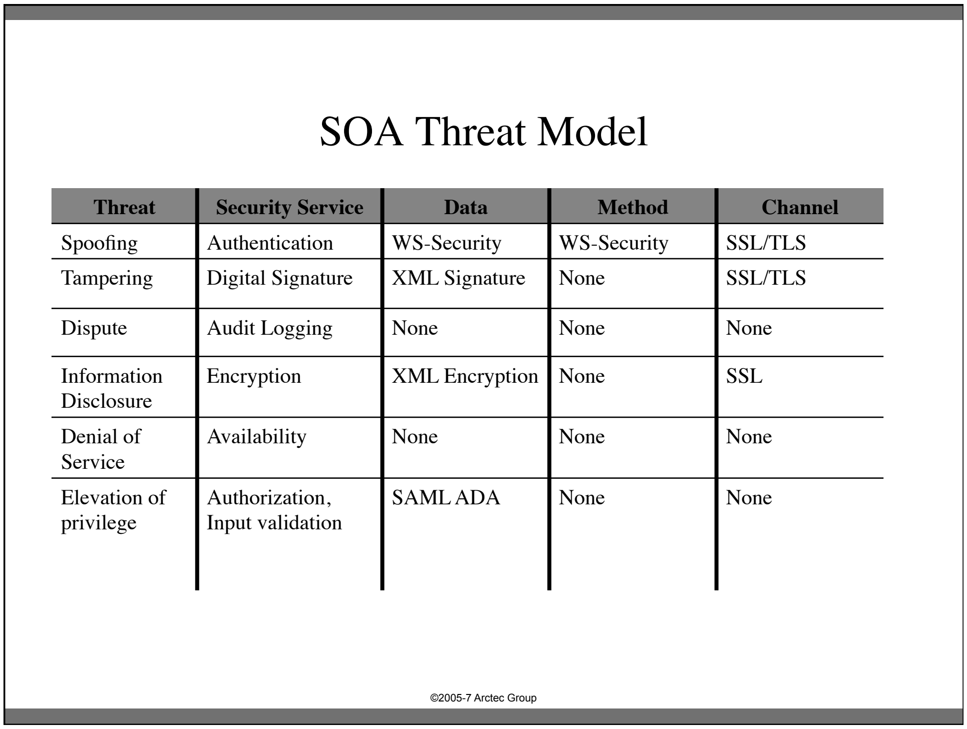
STRIDE Threat Model

Threat	Description	Example
Spoofing	Assume identity of client, server or request/response	Phishing attack to fool user into sending credentials to fake site
Tampering	Alter contents of request of response	Message integrity compromised to change parameters or values
Dispute	Dispute legitimate transaction	Illegitimately claiming a transaction was not completed
Information Disclosure	Unauthorized release of data	Unencrypted message sniffed off the network
Denial of Service	Service not available to authorized users	System flooded by requests until web server fails
Elevation of privilege	Bypass authorization system	Attacker changes group membership



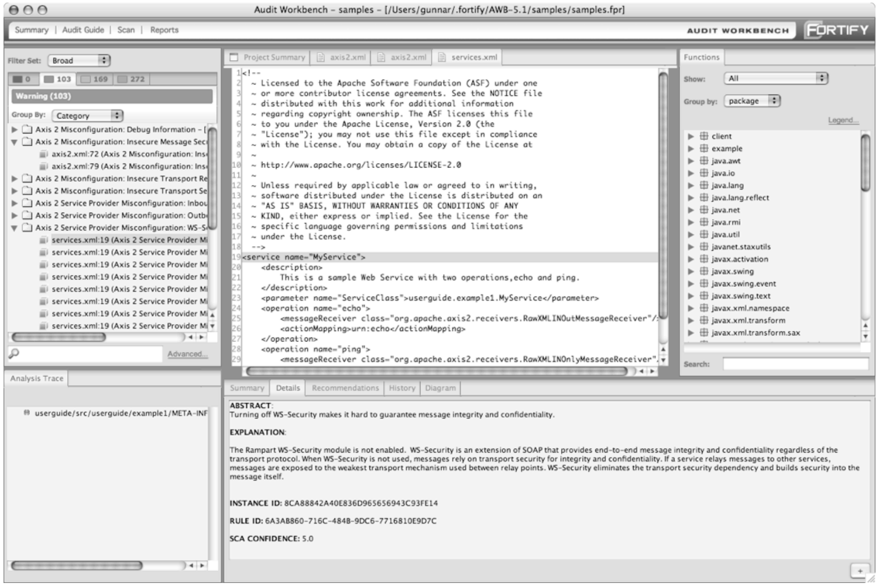
SOA Threat Model

Threat	Security Service	Standard
Spoofing	Authentication	WS-Security, SAML
Tampering	Digital Signature	XML Signature, SSL/TLS
Dispute	Audit Logging	None
Information Disclosure	Encryption	XML Encryption, SSL
Denial of Service	Availability	None
Elevation of privilege	Authorization, Input validation	None



SOA Threat Model

Threat	Security Service	Data	Method	Channel
Spoofing	Authentication	WS-Security	WS-Security	SSL/TLS
Tampering	Digital Signature	XML Signature	None	SSL/TLS
Dispute	Audit Logging	None	None	None
Information Disclosure	Encryption	XML Encryption	None	SSL
Denial of Service	Availability	None	None	None
Elevation of privilege	Authorization, Input validation	SAML ADA	None	None


Audit Workbench – samples – [/Users/gunnar/.fortify/AWB-5.1/samples/samples.fpr]
Summary | Audit Guide | Scan | Reports
AUDIT WORKBENCH
FORTIFY
Filter Set: Broad
0
103
169
272
Warning (103)
Group By: Category
Axis 2 Misconfiguration: Debug Information – [
Axis 2 Misconfiguration: Insecure Message Sec
axis2.xml:72 (Axis 2 Misconfiguration: Ins
axis2.xml:79 (Axis 2 Misconfiguration: Ins
Axis 2 Misconfiguration: Insecure Transport Re
Axis 2 Misconfiguration: Insecure Transport Se
Axis 2 Service Provider Misconfiguration: Inbou
Axis 2 Service Provider Misconfiguration: Outb
Axis 2 Service Provider Misconfiguration: WS-S
services.xml:19 (Axis 2 Service Provider Mi
Advanced...
Analysis Trace
userguide/src/userguide/example1/META-INF
Project Summary
axis2.xml
axis2.xml
services.xml
Functions
Show: All
Group by: package
Legend...
client
example
java.awt
java.io
java.lang
java.lang.reflect
java.net
java.rmi
java.util
javanet.staxutils
javax.activation
javax.swing
javax.swing.event
javax.swing.text
javax.xml.namespace
javax.xml.transform
javax.xml.transform.sax
Search:
Summary
Details
Recommendations
History
Diagram

```
<!--
  ~ Licensed to the Apache Software Foundation (ASF) under one
  ~ or more contributor license agreements. See the NOTICE file
  ~ distributed with this work for additional information
  ~ regarding copyright ownership. The ASF licenses this file
  ~ to you under the Apache License, Version 2.0 (the
  ~ "License"); you may not use this file except in compliance
  ~ with the License. You may obtain a copy of the License at
  ~
  ~ http://www.apache.org/licenses/LICENSE-2.0
  ~
  ~ Unless required by applicable law or agreed to in writing,
  ~ software distributed under the License is distributed on an
  ~ "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
  ~ KIND, either express or implied. See the License for the
  ~ specific language governing permissions and limitations
  ~ under the License.
  -->
<service name="MyService">
    <description>
        This is a sample Web Service with two operations,echo and ping.
    </description>
    <parameter name="ServiceClass">userguide.example1.MyService</parameter>
    <operation name="echo">
        <messageReceiver class="org.apache.axis2.receivers.RawXMLINOutMessageReceiver"/
        <actionMapping>urn:echo</actionMapping>
    </operation>
    <operation name="ping">
        <messageReceiver class="org.apache.axis2.receivers.RawXMLINOnlyMessageReceiver".
```

ABSTRACT:
Turning off WS-Security makes it hard to guarantee message integrity and confidentiality.

EXPLANATION:

The Rampart WS-Security module is not enabled. WS-Security is an extension of SOAP that provides end-to-end message integrity and confidentiality regardless of the transport protocol. When WS-Security is not used, messages rely on transport security for integrity and confidentiality. If a service relays messages to other services, messages are exposed to the weakest transport mechanism used between relay points. WS-Security eliminates the transport security dependency and builds security into the message itself.

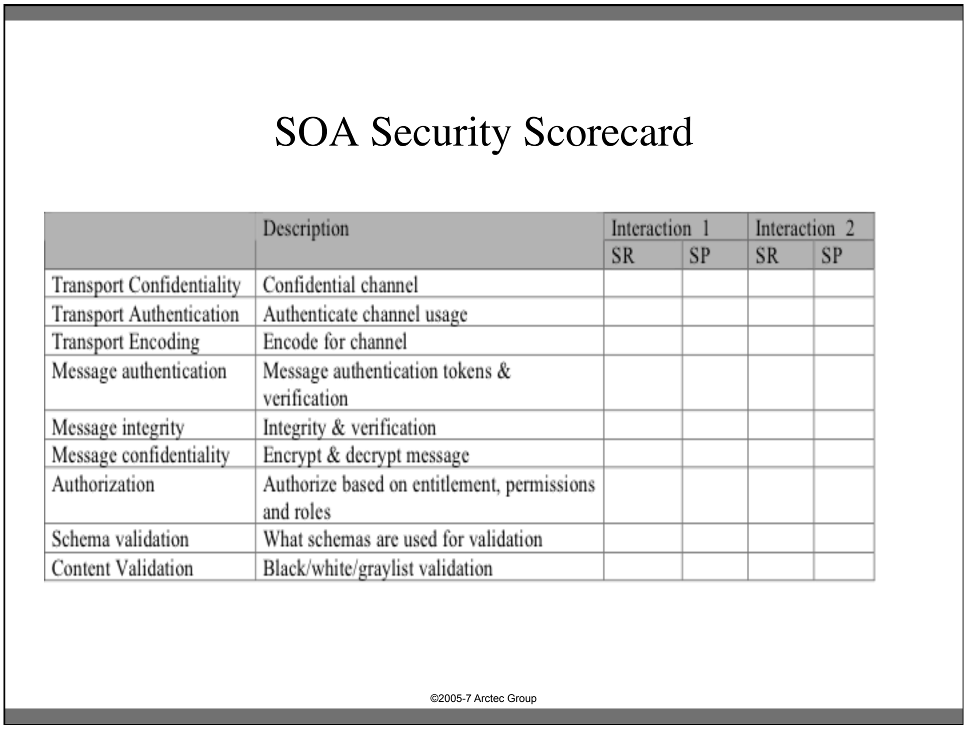
INSTANCE ID: 8CA88842A40E836D965656943C93FE14

RULE ID: 6A3AB860-716C-484B-9DC6-7716810E9D7C

SCA CONFIDENCE: 5.0

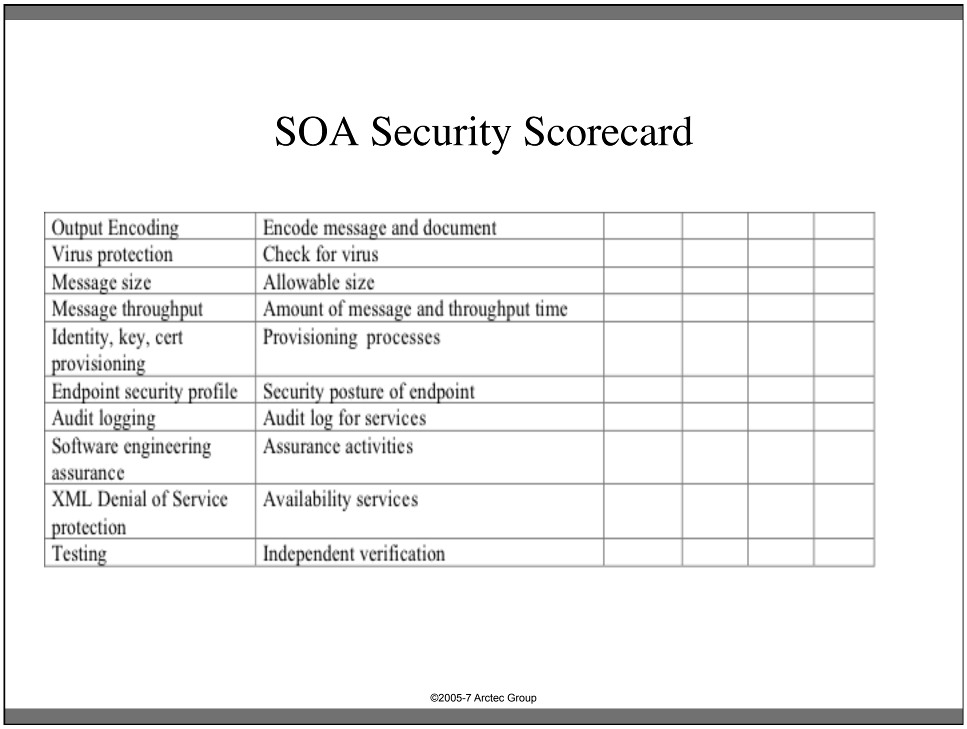
Trends

- Reusability
  - Reusable Findings & Fixes
    - Consider two bug findings
      - Session management bug: session state is passed around to every component, service and user. Makes for many high priority findings in audit report, also the fix is required on virtually every program
      - Data validation bug: Data access object (DAO) has a SQL injection hole. One major high priority finding in report. DAO used by many business logic classes, one fix location serves many classes



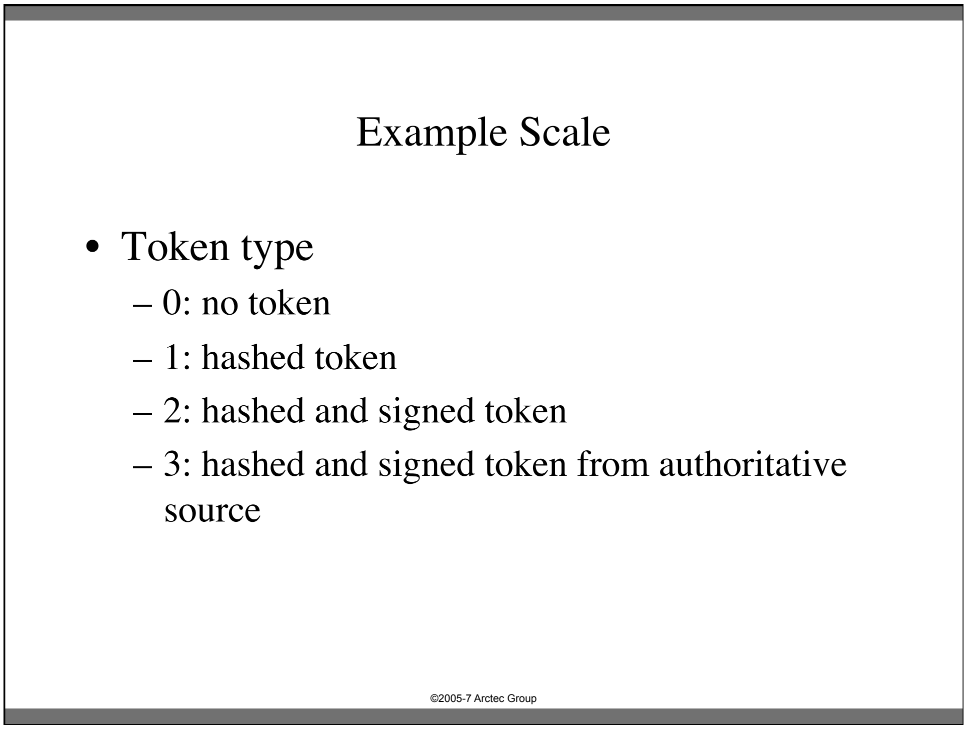
SOA Security Scorecard

	Description	Interaction 1		Interaction 2	
		SR	SP	SR	SP
Transport Confidentiality	Confidential channel				
Transport Authentication	Authenticate channel usage				
Transport Encoding	Encode for channel				
Message authentication	Message authentication tokens & verification				
Message integrity	Integrity & verification				
Message confidentiality	Encrypt & decrypt message				
Authorization	Authorize based on entitlement, permissions and roles				
Schema validation	What schemas are used for validation				
Content Validation	Black/white/graylist validation				



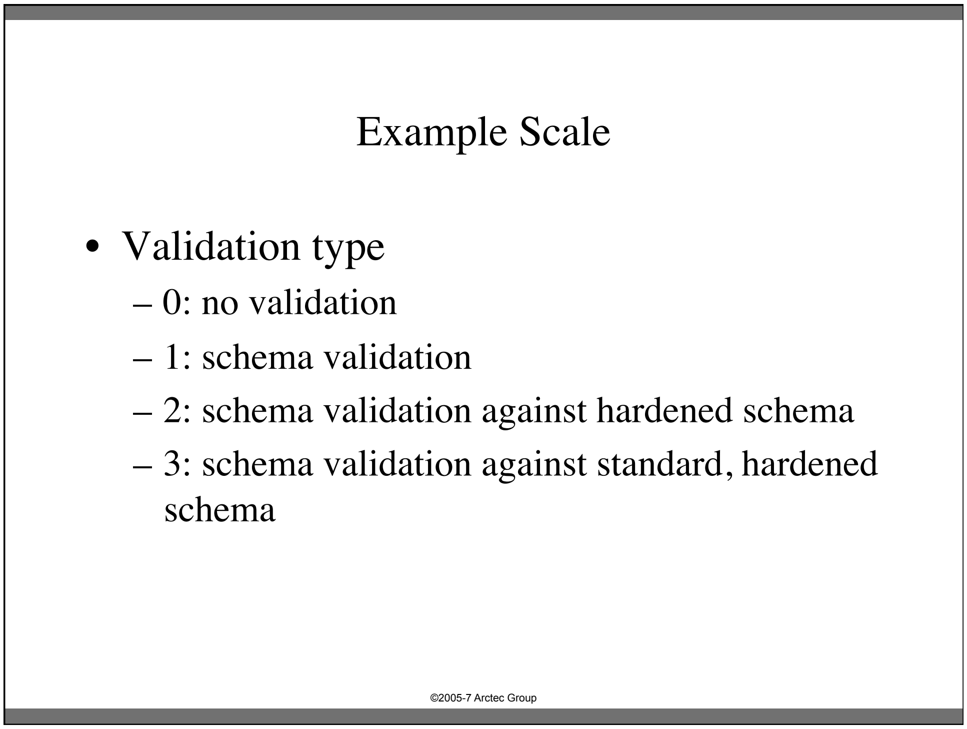
SOA Security Scorecard

Output Encoding	Encode message and document				
Virus protection	Check for virus				
Message size	Allowable size				
Message throughput	Amount of message and throughput time				
Identity, key, cert provisioning	Provisioning processes				
Endpoint security profile	Security posture of endpoint				
Audit logging	Audit log for services				
Software engineering assurance	Assurance activities				
XML Denial of Service protection	Availability services				
Testing	Independent verification				



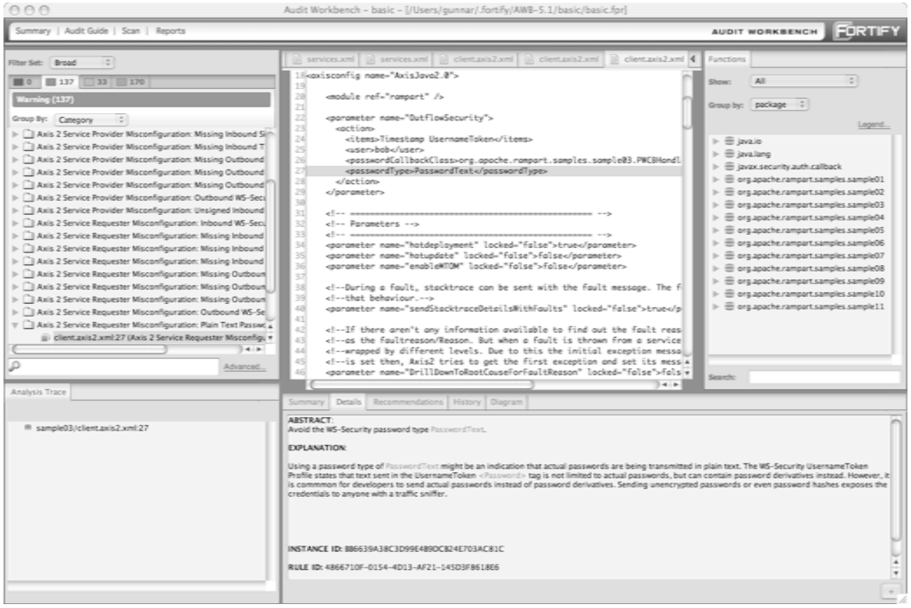
Example Scale

- Token type
  - – 0: no token
  - – 1: hashed token
  - – 2: hashed and signed token
  - – 3: hashed and signed token from authoritative source



Example Scale

- Validation type
  - – 0: no validation
  - – 1: schema validation
  - – 2: schema validation against hardened schema
  - – 3: schema validation against standard, hardened schema


Audit Workbench – basic – [/Users/gunnar/.fortify/AWB-5.1/basic/basic.fpr]
Summary | Audit Guide | Scan | Reports
AUDIT WORKBENCH
FORTIFY
Filter Set: Broad
0
137
33
170
Warning (137)
Group By: Category
Advanced...
Analysis Trace
services.xml
services.xml
client.axis2.xml
client.axis2.xml
client.axis2.xml
Functions
Show: All
Group by: package
Legend...
java.io
java.lang
javax.security.auth.callback
org.apache.rampart.samples.sample01
org.apache.rampart.samples.sample02
org.apache.rampart.samples.sample03
org.apache.rampart.samples.sample04
org.apache.rampart.samples.sample05
org.apache.rampart.samples.sample06
org.apache.rampart.samples.sample07
org.apache.rampart.samples.sample08
org.apache.rampart.samples.sample09
org.apache.rampart.samples.sample10
org.apache.rampart.samples.sample11
Search:
Summary
Details
Recommendations
History
Diagram

- Axis 2 Service Provider Misconfiguration: Missing Inbound Si
- Axis 2 Service Provider Misconfiguration: Missing Inbound T
- Axis 2 Service Provider Misconfiguration: Missing Outbound
- Axis 2 Service Provider Misconfiguration: Missing Outbound
- Axis 2 Service Provider Misconfiguration: Missing Outbound
- Axis 2 Service Provider Misconfiguration: Outbound WS-Secu
- Axis 2 Service Provider Misconfiguration: Unsigned Inbound
- Axis 2 Service Requester Misconfiguration: Inbound WS-Secu
- Axis 2 Service Requester Misconfiguration: Missing Inbound
- Axis 2 Service Requester Misconfiguration: Missing Inbound
- Axis 2 Service Requester Misconfiguration: Missing Inbound
- Axis 2 Service Requester Misconfiguration: Missing Outboun
- Axis 2 Service Requester Misconfiguration: Missing Outboun
- Axis 2 Service Requester Misconfiguration: Missing Outboun
- Axis 2 Service Requester Misconfiguration: Outbound WS-Se
- Axis 2 Service Requester Misconfiguration: Plain Text Passwo
  - client.axis2.xml:27 (Axis 2 Service Requester Misconfigu

sample03/client.axis2.xml:27

```
<axisconfig name="AxisJava2.0">

    <module ref="rampart" />

    <parameter name="OutflowSecurity">
      <action>
        <items>Timestamp UsernameToken</items>
        <user>bob</user>
        <passwordCallbackClass>org.apache.rampart.samples.sample03.PWCBHandl
        <passwordType>PasswordText</passwordType>
      </action>
    </parameter>

    <!-- ================================================ -->
    <!-- Parameters -->
    <!-- ================================================ -->
    <parameter name="hotdeployment" locked="false">true</parameter>
    <parameter name="hotupdate" locked="false">false</parameter>
    <parameter name="enableMTOM" locked="false">false</parameter>

    <!--During a fault, stacktrace can be sent with the fault message. The f
    <!--that behaviour.-->
    <parameter name="sendStacktraceDetailsWithFaults" locked="false">true</p

    <!--If there aren't any information available to find out the fault reas
    <!--as the faultreason/Reason. But when a fault is thrown from a service
    <!--wrapped by different levels. Due to this the initial exception messa
    <!--is set then, Axis2 tries to get the first exception and set its mess
    <parameter name="DrillDownToRootCauseForFaultReason" locked="false">fals
```

ABSTRACT:
Avoid the WS-Security password type PasswordText.

EXPLANATION:

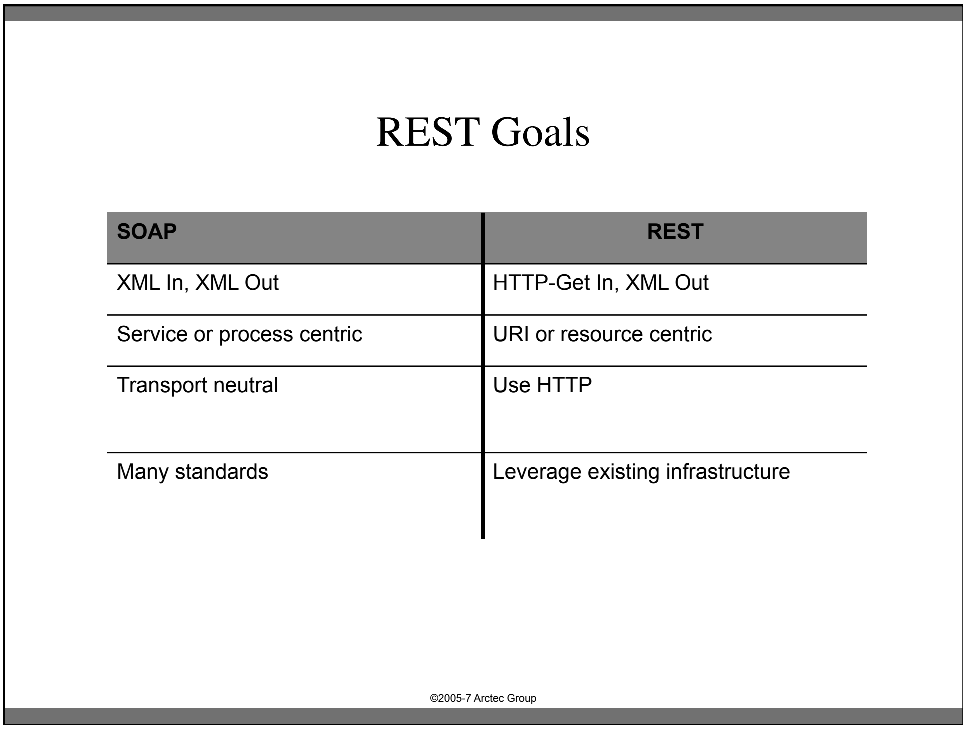
Using a password type of PasswordText might be an indication that actual passwords are being transmitted in plain text. The WS-Security UsernameToken Profile states that text sent in the UsernameToken <Password> tag is not limited to actual passwords, but can contain password derivatives instead. However, it is commmon for developers to send actual passwords instead of password derivatives. Sending unencrypted passwords or even password hashes exposes the credentials to anyone with a traffic sniffer.

INSTANCE ID: 8B6639A38C3D99E489DCB24E703AC81C

RULE ID: 4866710F-0154-4D13-AF21-145D3FB618E6

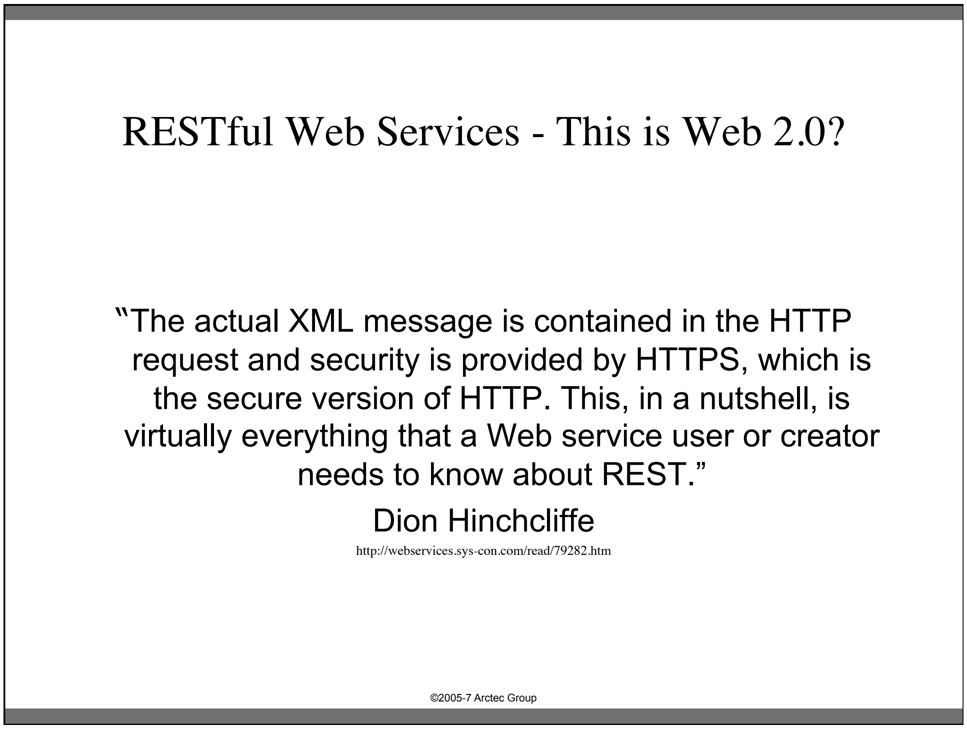
Putting it all together

- Use value assessment to focus time and effort
- Use scoring index to improve quality



REST Goals

SOAP	REST
XML In, XML Out	HTTP-Get In, XML Out
Service or process centric	URI or resource centric
Transport neutral	Use HTTP
Many standards	Leverage existing infrastructure



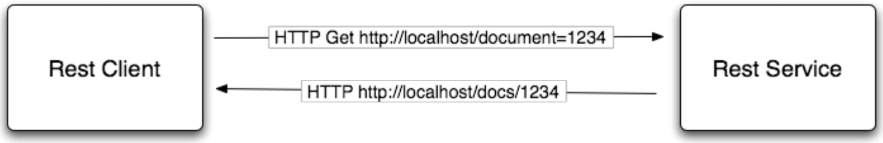
RESTful Web Services - This is Web 2.0?

"The actual XML message is contained in the HTTP request and security is provided by HTTPS, which is the secure version of HTTP. This, in a nutshell, is virtually everything that a Web service user or creator needs to know about REST."

Dion Hinchcliffe

http://webservices.sys-con.com/read/79282.htm

Rest


Rest Client
HTTP Get http://localhost/document=1234
HTTP http://localhost/docs/1234
Rest Service

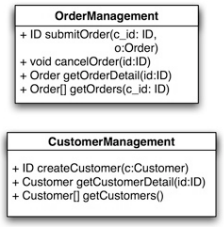
Input URI: http://localhost
Resource Request: 1234
Resource return: http://localhost/docs/1234

REST Server -- look Ma, no WSDL

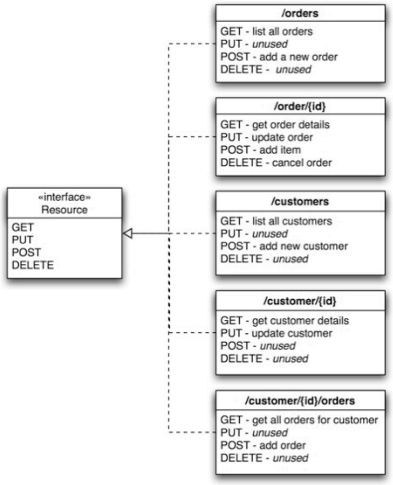
```
@WebServiceProvider()
@ServiceMode(value = Service.Mode.PAYLOAD)
public class RestSourcePayloadProvider implements Provider<DOMSource> {

    public DOMSource invoke(DOMSource request) {
        MessageContext mc = wsContext.getMessageContext();
        String path = (String)mc.get(Message.PATH_INFO);
        String query = (String)mc.get(Message.QUERY_STRING);
        String httpMethod = (String)mc.get(Message.HTTP_REQUEST_METHOD);

         if (httpMethod.equalsIgnoreCase("POST")) {
                return updateCustomer(request);
        } else if (httpMethod.equalsIgnoreCase("GET")) {
            if (path.equals("/customerservice/customer") && query == null) {
                return getAllCustomers();
}
```

Rest Client

```
public static void main(String args[]) throws Exception {
         QName serviceName = new Qname
         ("http://apache.org/hello_world_xml_http/wrapped", "cutomerservice");
        QName portName = new Qname
         ("http://apache.org/hello_world_xml_http/wrapped","RestProviderPort");
        String endpointAddress = "http://localhost:9000/customerservice/
customer";
         url = new URL(endpointAddress + "?id=1234");
         in = url.openStream();
        source = new StreamSource(in);
        printSource(source);

         Service service = Service.create(serviceName);
        service.addPort(portName, HTTPBinding.HTTP_BINDING,  endpointAddress);
        Dispatch<DOMSource> dispatcher = service.createDispatch(portName,
DOMSource.class, Service.Mode.PAYLOAD);
        Map<String, Object> requestContext = dispatcher.getRequestContext();
requestContext.put(MessageContext.HTTP_REQUEST_METHOD, new String("GET"));
        requestContext.put(MessageContext.QUERY_STRING, "id=1234");
        requestContext.put(MessageContext.PATH_INFO, path);
         DOMSource returnSource = dispatcher.invoke(null);
        printSource(returnSource);
```


OrderManagement
+ ID submitOrder(c_id: ID, o:Order)
+ void cancelOrder(id:ID)
+ Order getOrderDetail(id:ID)
+ Order[] getOrders(c_id: ID)
CustomerManagement
+ ID createCustomer(c:Customer)
+ Customer getCustomerDetail(id:ID)
+ Customer[] getCustomers()

Service oriented: Every service has its own interface and operations, but does not represent an entity



Resource oriented: entities or collections represented by a URI

Source (http://www.innoq.com/blog/st/2006/06/30/rest_vs_soap_oh_no_not_again.html)

REST Request Authentication

Summary of HMAC-SHA1 Request Authentication

1. You construct a request to AWS.

2. You use your Secret Access Key to calculate the request signature, a Keyed-Hashing for Message Authentication code (HMAC) with an SHA1 hash function, as defined in the next section of this topic.

3. You send the request data, the signature, and your Access Key ID to AWS.

4. AWS uses the Access Key ID to look up the Secret Access Key.

5. AWS generates a signature from the request data and the Secret Access Key using the same algorithm you used to calculate the signature in the request.

6. If the signature generated by AWS matches the one you sent in the request, the request is considered to be authentic. If the comparison fails, the request is discarded, and AWS returns an error response. error response.

(note: append timestamp to request to limit replays to 15 minute window)

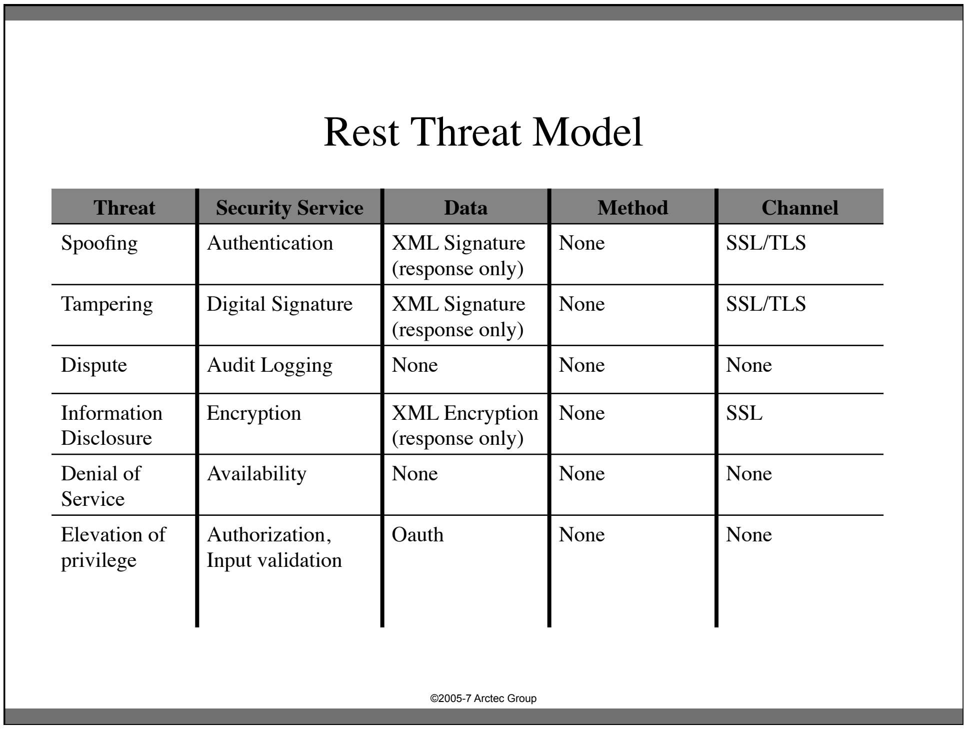
http://docs.amazonwebservices.com/AWSSimpleQueueService/2006-04-01/RequestAuthenticationArticle.html

Rest Request Authentication

```
"Authorization: AWS " + AWSAccessKeyId + ":"
        + base64(hmac-sha1(VERB + "\n"
        + CONTENT-MD5 + "\n"
        + CONTENT-TYPE + "\n"
        + DATE + "\n"
        + CanonicalizedAmzHeaders + "\n"
        + CanonicalizedResource))
```

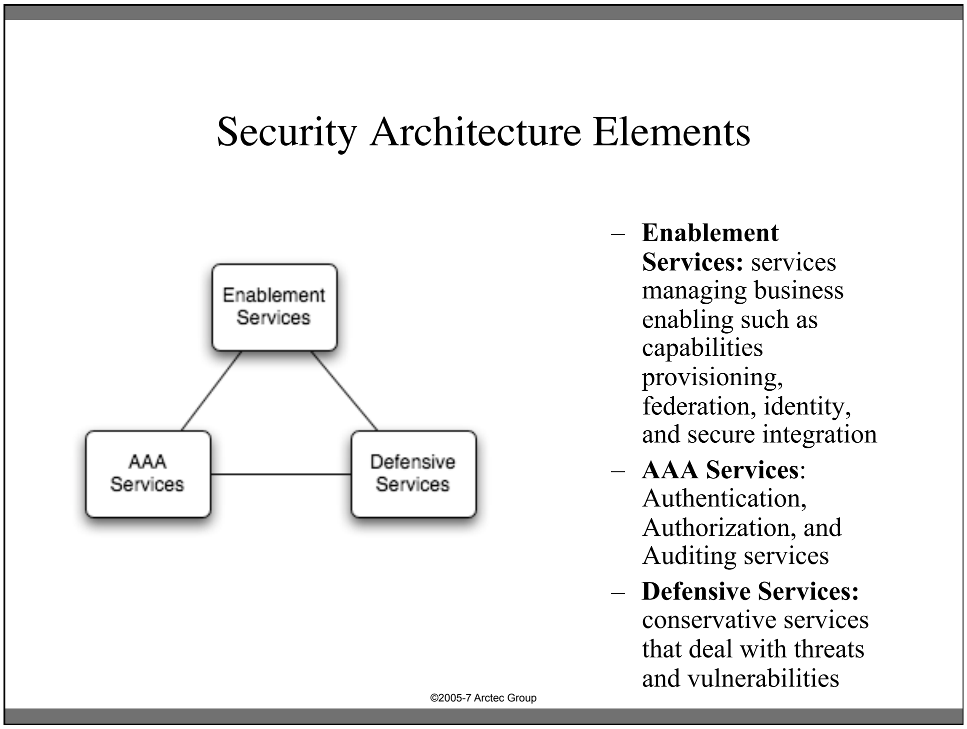
Example:

```
PUT /quotes/nelson HTTP/1.0
Authorization: AWS 44CF9590006BF252F707:jZNOcbfWmD/A/f3hSvVzXZjM2HU=
Content-Md5: c8fdb181845a4ca6b8fec737b3581d76
Content-Type: text/html
Date: Thu, 17 Nov 2005 18:49:58 GMT
X-Amz-Meta-Author: foo@bar.com
X-Amz-Magic: abracadabra
```



Rest Threat Model

Threat	Security Service	Data	Method	Channel
Spoofing	Authentication	XML Signature (response only)	None	SSL/TLS
Tampering	Digital Signature	XML Signature (response only)	None	SSL/TLS
Dispute	Audit Logging	None	None	None
Information Disclosure	Encryption	XML Encryption (response only)	None	SSL
Denial of Service	Availability	None	None	None
Elevation of privilege	Authorization, Input validation	Oauth	None	None

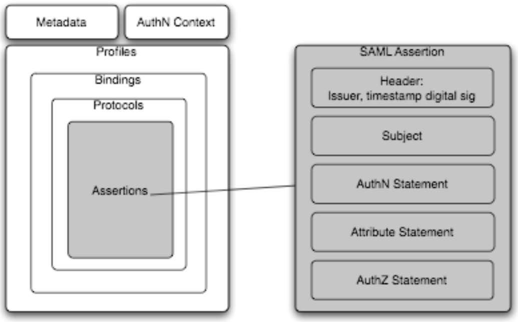


Security Architecture Elements

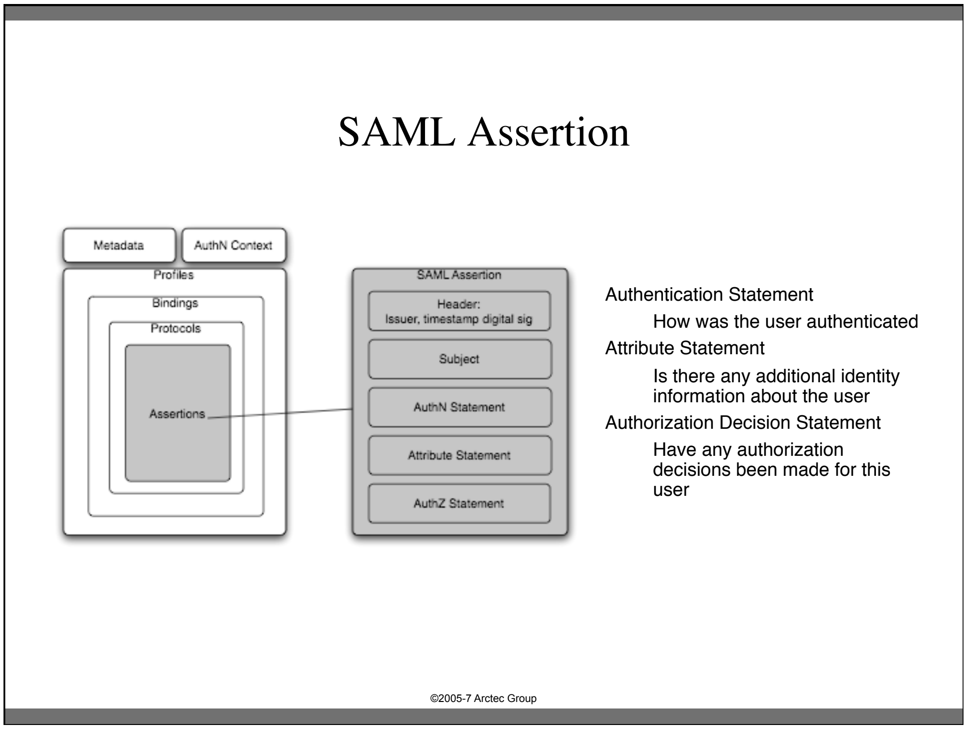
- **Enablement Services:** services managing business enabling such as capabilities provisioning, federation, identity, and secure integration
- **AAA Services**: Authentication, Authorization, and Auditing services
- **Defensive Services:** conservative services that deal with threats and vulnerabilities

AAA Services: SAML

SAML Assertion



- Headers & Control Information
  - SAML Issuer
  - Timers
  - XML Encryption spec supports:
    - Block Encryption: TRIPLE DES, AES-128, AES-256
    - Key Transport: RSA-v1.5, RSA-OAEP
  - Digital Signature spec supports:
    - Digest: SHA1
    - MAC: HMAC-SHA1
    - XML Canonicalization: CanonicalXML (Without comments)
    - Transform: Enveloped Signature
    - Signature: RSAwithSHA1 (recommended in XML Signature but needed for interoperability)



SAML Assertion

Authentication Statement

- How was the user authenticated

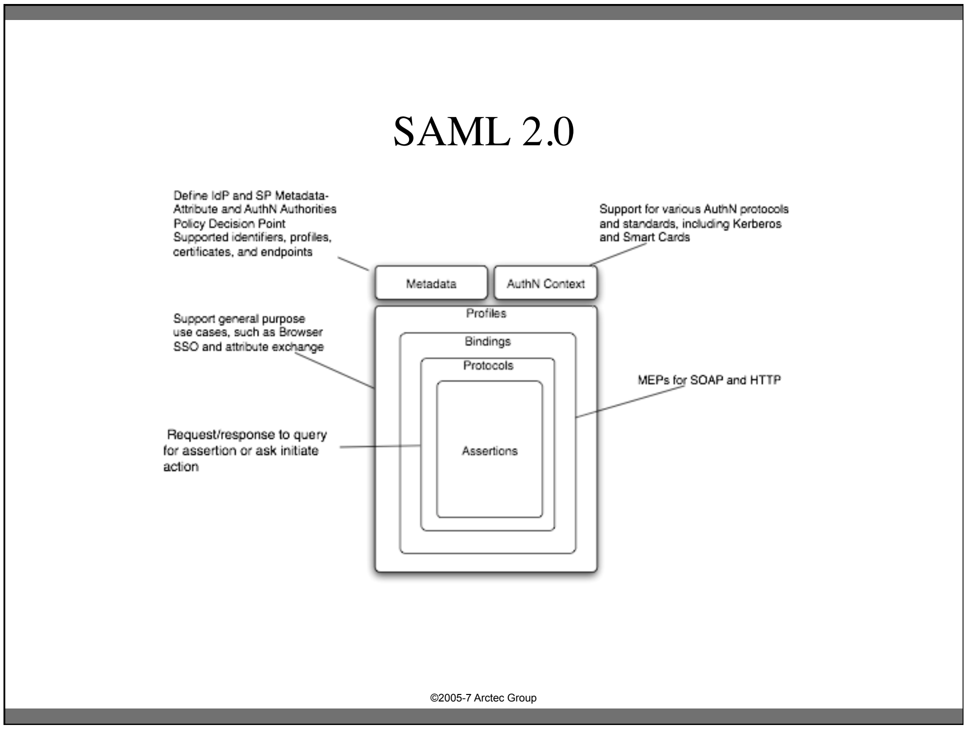
Attribute Statement

- Is there any additional identity information about the user

Authorization Decision Statement

- Have any authorization decisions been made for this user

©2005-7 Arctec Group



SAML 2.0

Define IdP and SP Metadata-
Attribute and AuthN Authorities
Policy Decision Point
Supported identifiers, profiles,
certificates, and endpoints

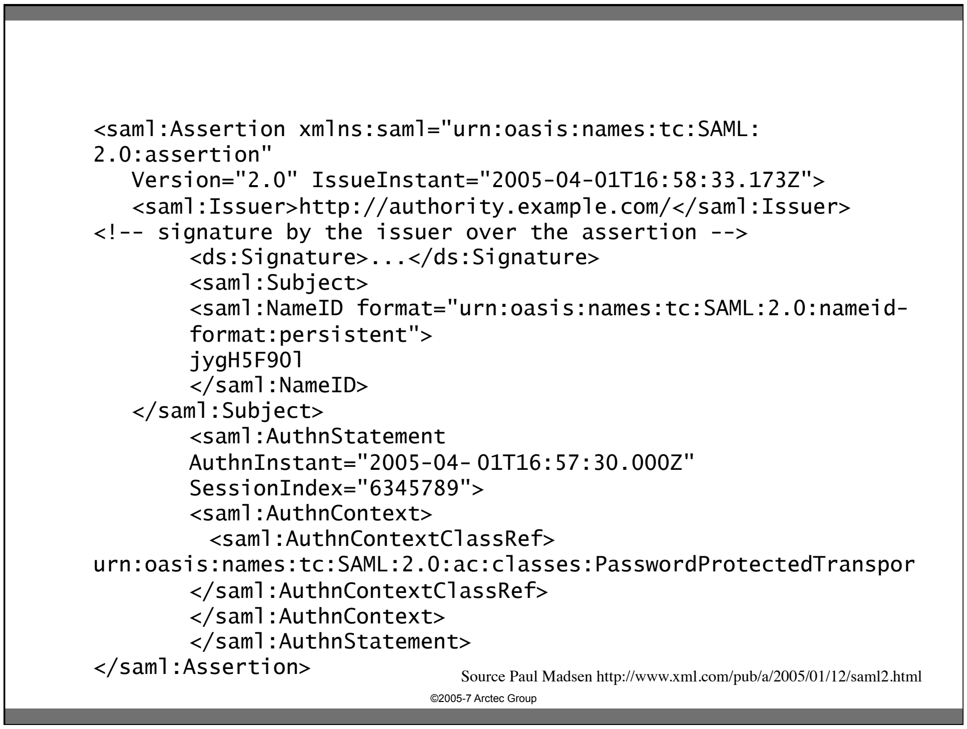
Support for various AuthN protocols and standards, including Kerberos and Smart Cards

Support general purpose use cases, such as Browser SSO and attribute exchange

MEPs for SOAP and HTTP

Request/response to query for assertion or ask initiate action

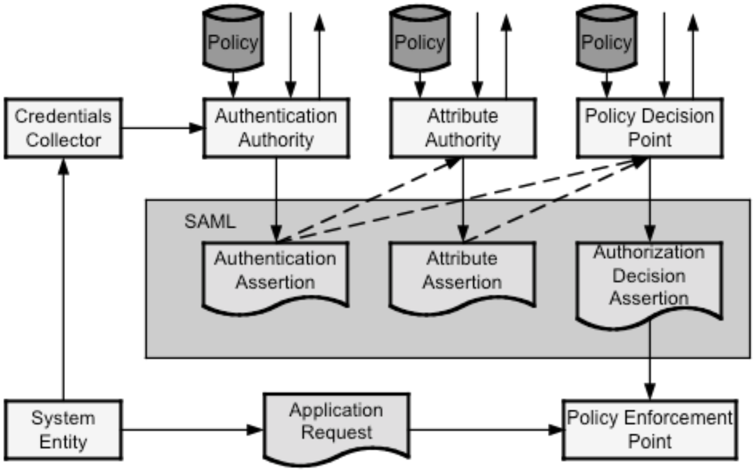
©2005-7 Arctec Group



```
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:
2.0:assertion"
   Version="2.0" IssueInstant="2005-04-01T16:58:33.173Z">
   <saml:Issuer>http://authority.example.com/</saml:Issuer>
<!-- signature by the issuer over the assertion -->
        <ds:Signature>...</ds:Signature>
        <saml:Subject>
        <saml:NameID format="urn:oasis:names:tc:SAML:2.0:nameid-
        format:persistent">
        jygH5F90l
        </saml:NameID>
   </saml:Subject>
        <saml:AuthnStatement
        AuthnInstant="2005-04- 01T16:57:30.000Z"
        SessionIndex="6345789">
        <saml:AuthnContext>
          <saml:AuthnContextClassRef>
urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTranspor
        </saml:AuthnContextClassRef>
        </saml:AuthnContext>
        </saml:AuthnStatement>
</saml:Assertion>
```

Source Paul Madsen http://www.xml.com/pub/a/2005/01/12/saml2.html

SAML Producer Consumer Model



- Source http://lists.oasis-open.org/archives/security-services/200506/msg00031.html

Defensive Architecture: Security Gateway

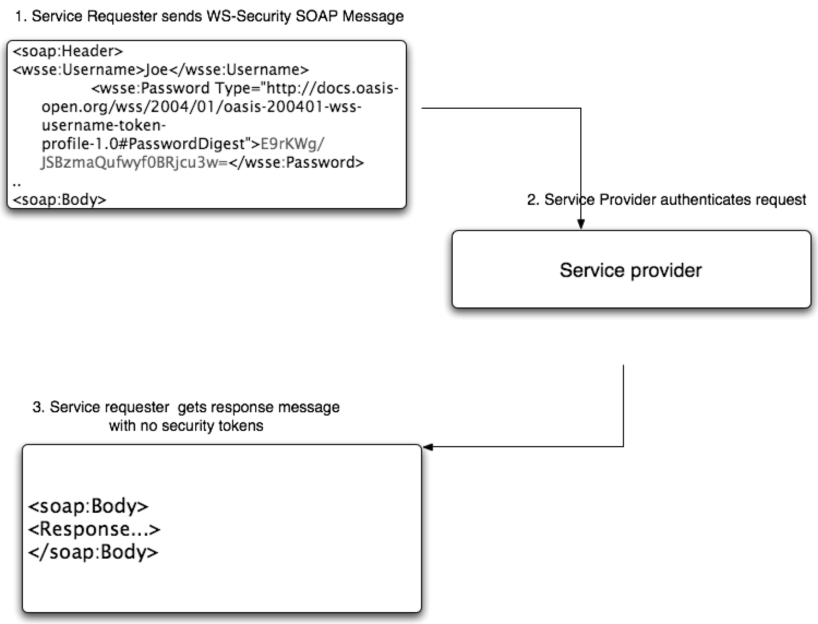
- Vulnerability
  - In SOAP and Rest style Web services there is no default authentication, messages are typically sent in XML over HTTP and contain nothing that can be used to perform authentication.
  - Simply applying general purpose security standards like WS-Security is not adequate, the WS-Security Username token may pass the user's password in plaintext form. For example:

```
<SOAP>
<SOAPHeader>
<wsse:Username>Joe</wsse:Username>
<wsse:Password Type="http://docs.oasis-open.org/wss/
2004/01/oasis-200401-wss-username-token-
profile-1.0#PasswordText">MyPassword</wsse:Password>
```

- Vulnerability
  - The next step beyond Username Token with Password in cleartext is to look at hashing the password

```
                    <wsse:Username>Joe</wsse:Username>
                    <wsse:Password Type="http://docs.oasis-open.org/
wss/2004/01/oasis-200401-wss-username-token-
profile-1.0#PasswordDigest">E9rKWg/JSBzmaQufwyf0BRjcu3w=</
wsse:Password>
```

  - This token is marginally stronger, but also lacks a timestamp and nonce so may be vulnerable to message replay and other attacks. Further, if the password is hashed, its likely there is a cleartext password sitting somewhere in the system that generated it. WS-Security provides a general purpose framework for transmitting claims, but the standard is treated differently in practice in implementation.



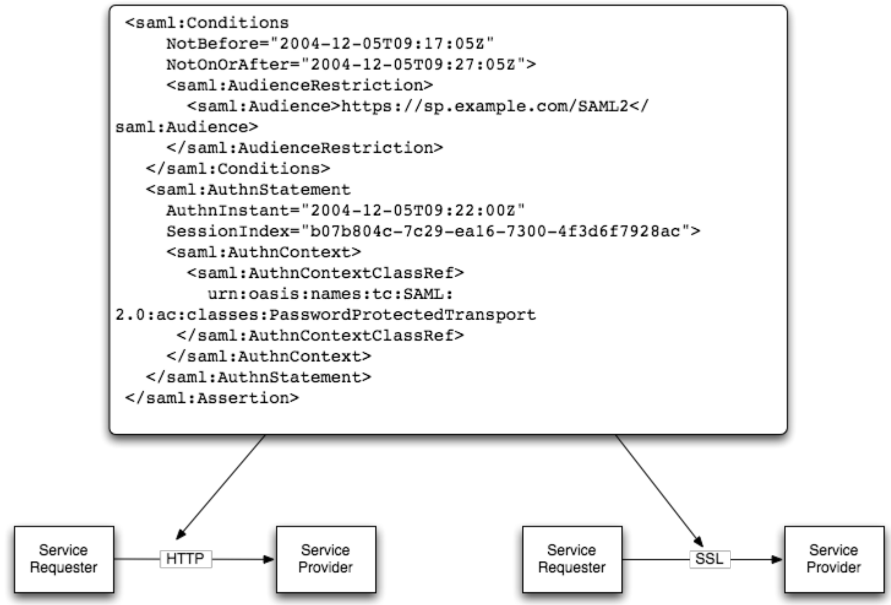
1. Service Requester sends WS-Security SOAP Message

```
<soap:Header>
<wsse:Username>Joe</wsse:Username>
            <wsse:Password Type="http://docs.oasis-
    open.org/wss/2004/01/oasis-200401-wss-
    username-token-
    profile-1.0#PasswordDigest">E9rKWg/
    JSBzmaQufwyf0BRjcu3w=</wsse:Password>
..
<soap:Body>
```

2. Service Provider authenticates request

3. Service requester gets response message with no security tokens

```
<soap:Body>
<Response...>
</soap:Body>
```

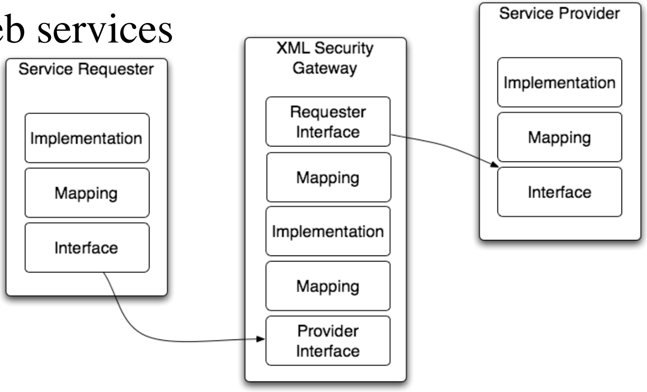


```
<saml:Conditions
    NotBefore="2004-12-05T09:17:05Z"
    NotOnOrAfter="2004-12-05T09:27:05Z">
    <saml:AudienceRestriction>
      <saml:Audience>https://sp.example.com/SAML2</
saml:Audience>
    </saml:AudienceRestriction>
  </saml:Conditions>
  <saml:AuthnStatement
    AuthnInstant="2004-12-05T09:22:00Z"
    SessionIndex="b07b804c-7c29-ea16-7300-4f3d6f7928ac">
    <saml:AuthnContext>
      <saml:AuthnContextClassRef>
        urn:oasis:names:tc:SAML:
2.0:ac:classes:PasswordProtectedTransport
     </saml:AuthnContextClassRef>
    </saml:AuthnContext>
  </saml:AuthnStatement>
</saml:Assertion>
```

XML Security Gateway Pattern

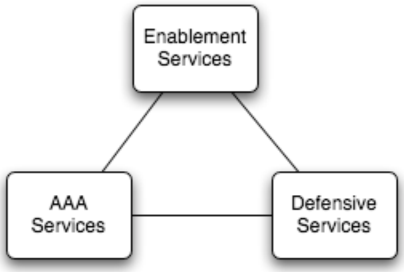
- Context: The primary goal of Web services is to solve interoperability and integration problems. Web services traverse multiple technologies and runtimes.
- Problem: Web service requesters and providers do not agree upon binary runtimes like J2EE, instead they agree upon service contracts, message exchange patterns, and schema. Service and message level authentication, authorization, and auditing services for Web services are not delivered by a single container, rather these services must span technical and organizational boundaries

- Solution: Use a XML Security Gateway to provide decentralized security services for Web services



```
<wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
    <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
AssertionID="Id-000001129354af1c-0000000000000002"
IssueInstant="2007-05-16T05:20:39Z" Issuer="CN=Test,OU=Unknown”
MajorVersion="1" MinorVersion="1">
      <saml:Conditions NotBefore="2007-05-16T04:40:35Z"
NotOnOrAfter="2007-05-16T06:40:35Z"/>
      <saml:AuthorizationDecisionStatement Decision="Permit" Resource="http://
host/service">
        <saml:Subject>
          <saml:NameIdentifier Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:X509SubjectName">Test</saml:NameIdentifier>
        </saml:Subject>
        <saml:Action>getCustomerDetails</saml:Action>
      </saml:AuthorizationDecisionStatement>
<dsig:SignatureValue>V6pRhOSnrvS8xT+WXIbNvlrOhVkAUMVI4YZ27KfG/
jDLMwSbrsD6E3tA4OrI6naL
U+gt2OsYr58rD+AILpxNkOuxZMWdLcj3zr0gljt339DvYL6MRJBZ3KvpDmrw16PM
w8Wo7ac1tGcLFVW5PV5locPs+fOV+rOGHafYTGGlubQ=</dsig:SignatureValue>
        <dsig:KeyInfo Id="Id-000001129354af1d-0000000000000004">
          …
    </saml:Assertion>
  </wsse:Security>
  </soap:Header>
 <soap:Body>
<ns0:getCustomerDetails xmlns:ns0="http://servicehost"/>
<customernumber>1234</customernumber>
```

Enablement



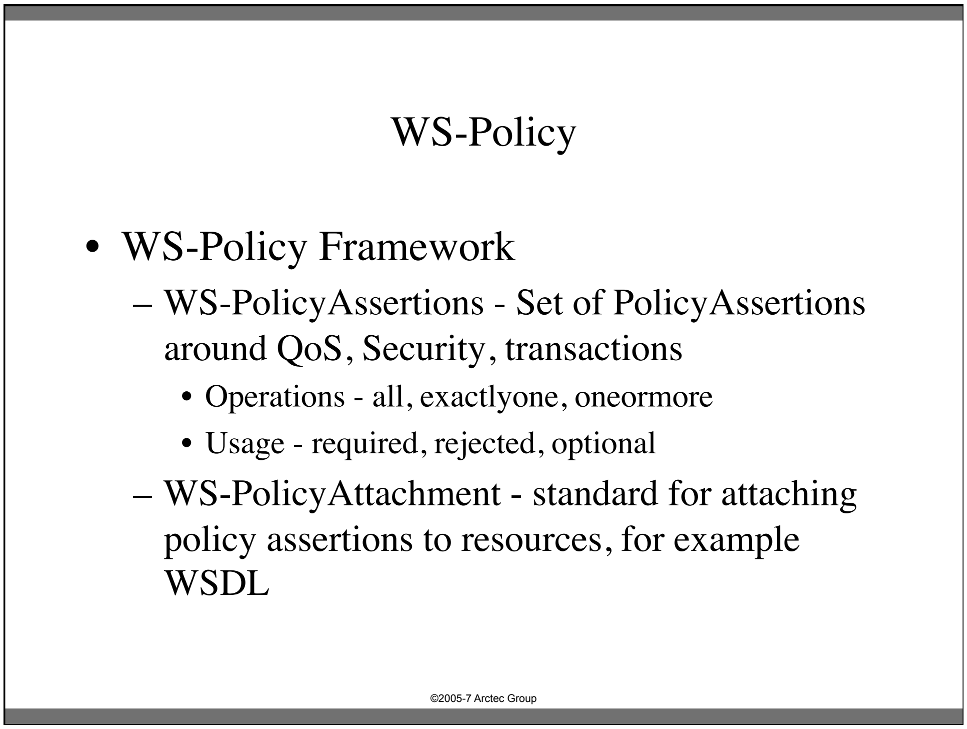
Enablement Services

AAA Services

Defensive Services

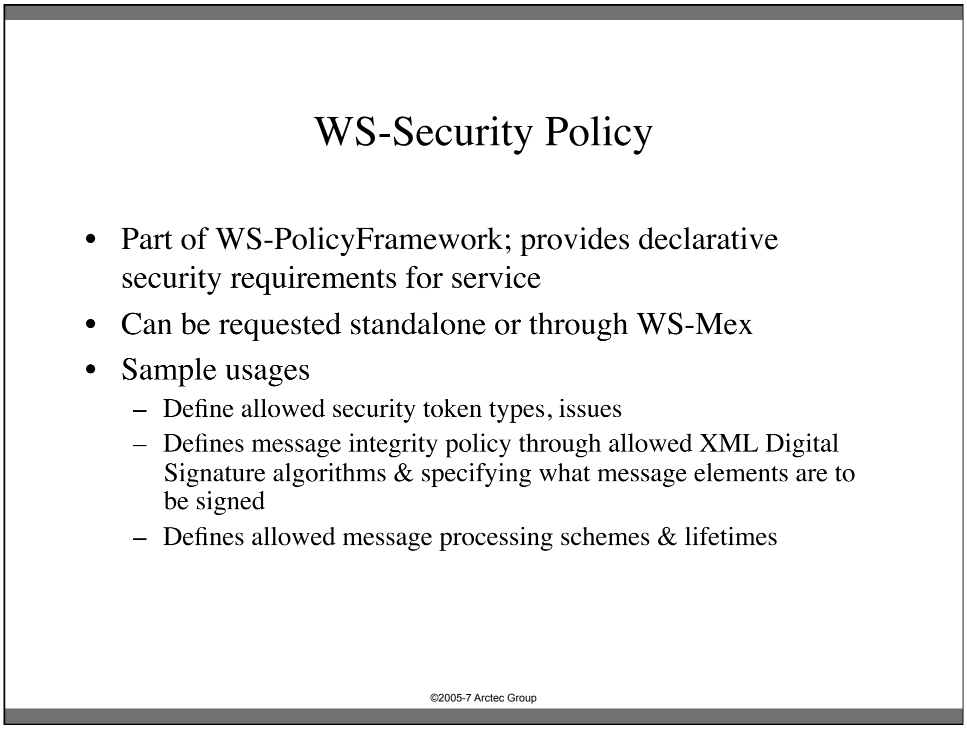
Policy

“Security should depend on policy
not topology.”
-Bill Gates Feb. 6, 2007



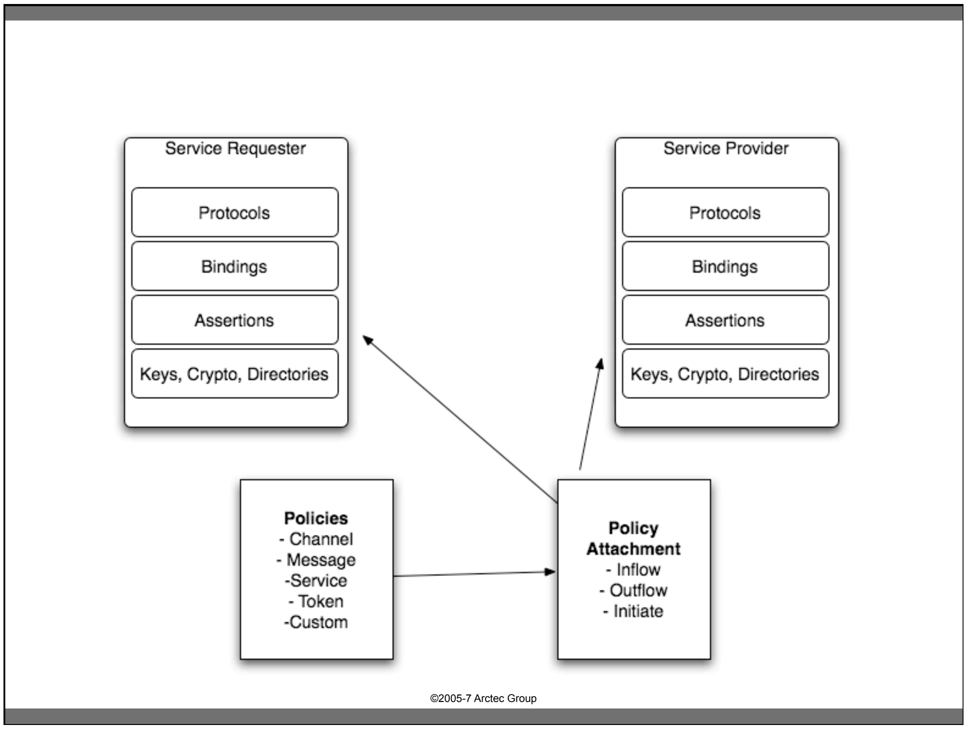
WS-Policy

- WS-Policy Framework
  - WS-PolicyAssertions - Set of PolicyAssertions around QoS, Security, transactions
    - Operations - all, exactlyone, oneormore
    - Usage - required, rejected, optional
  - WS-PolicyAttachment - standard for attaching policy assertions to resources, for example WSDL



WS-Security Policy

- Part of WS-PolicyFramework; provides declarative security requirements for service
- Can be requested standalone or through WS-Mex
- Sample usages
  - Define allowed security token types, issues
  - Defines message integrity policy through allowed XML Digital Signature algorithms & specifying what message elements are to be signed
  - Defines allowed message processing schemes & lifetimes



Service Requester

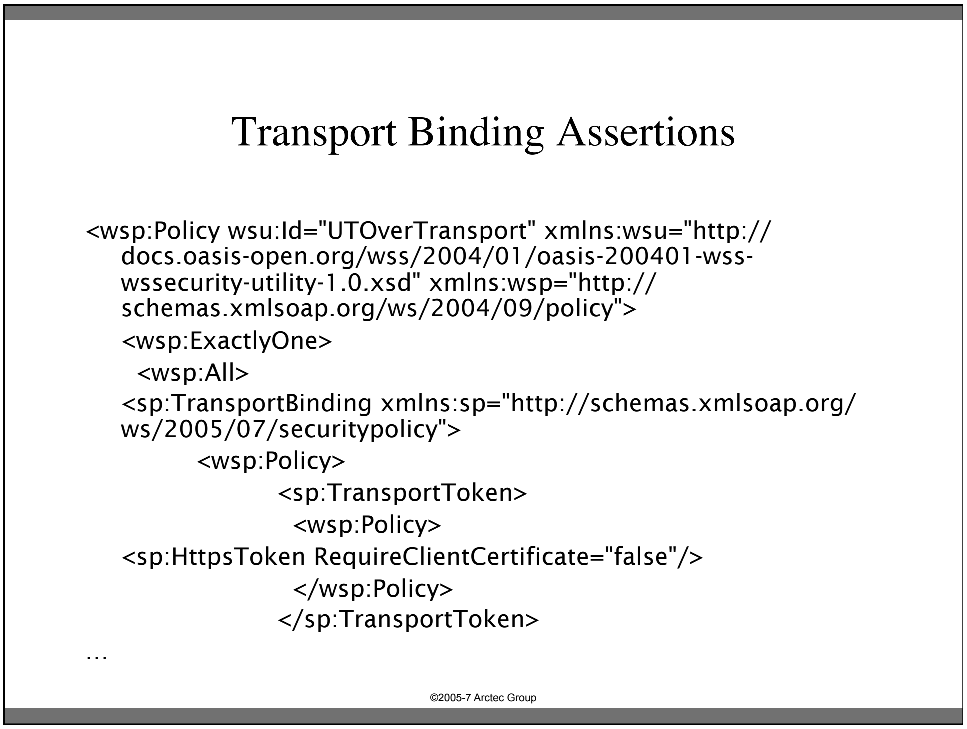
- Protocols
- Bindings
- Assertions
- Keys, Crypto, Directories

Service Provider

- Protocols
- Bindings
- Assertions
- Keys, Crypto, Directories

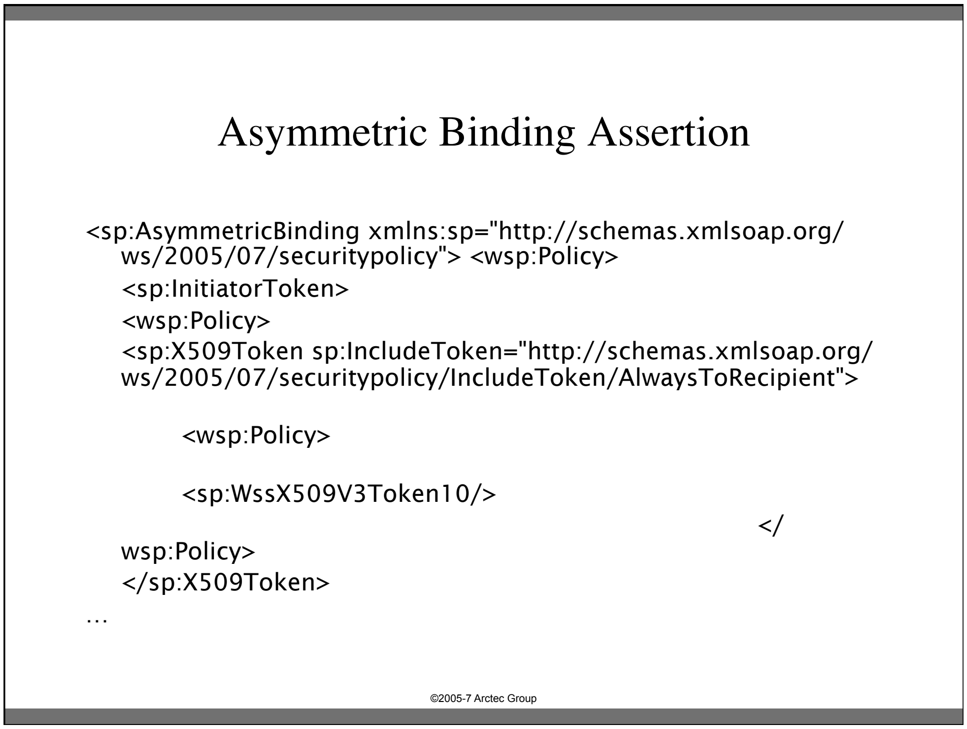
Policies
- Channel
- Message
-Service
- Token
-Custom

Policy Attachment
- Inflow
- Outflow
- Initiate



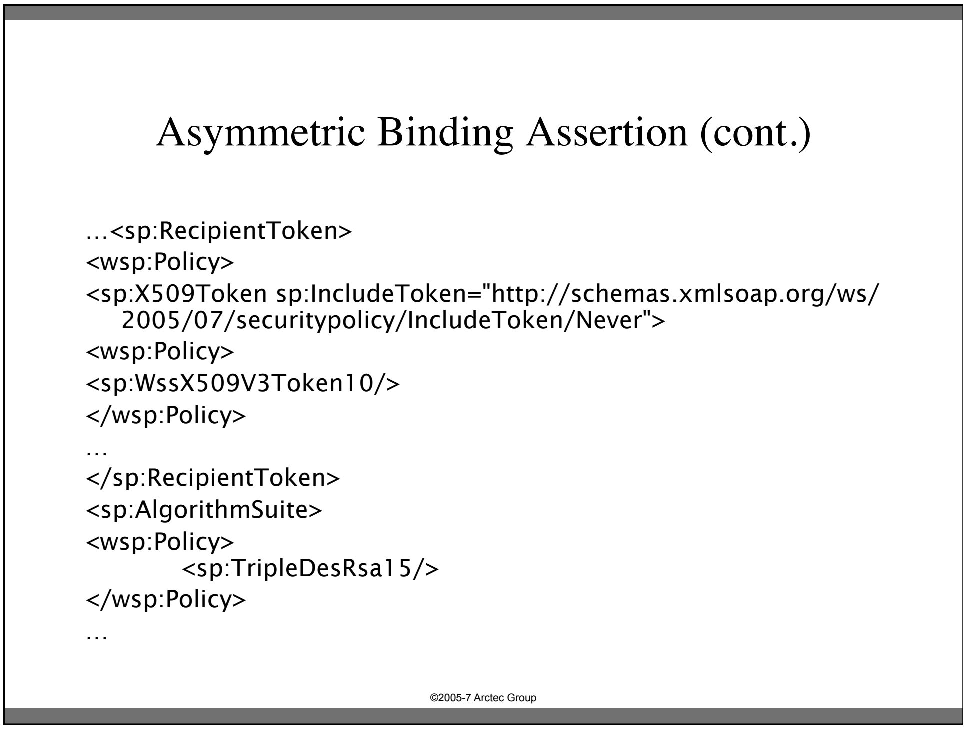
Transport Binding Assertions

```
<wsp:Policy wsu:Id="UTOverTransport" xmlns:wsu="http://
    docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
    wssecurity-utility-1.0.xsd" xmlns:wsp="http://
    schemas.xmlsoap.org/ws/2004/09/policy">
    <wsp:ExactlyOne>
      <wsp:All>
    <sp:TransportBinding xmlns:sp="http://schemas.xmlsoap.org/
    ws/2005/07/securitypolicy">
            <wsp:Policy>
                    <sp:TransportToken>
                      <wsp:Policy>
    <sp:HttpsToken RequireClientCertificate="false"/>
                      </wsp:Policy>
                    </sp:TransportToken>
…
```



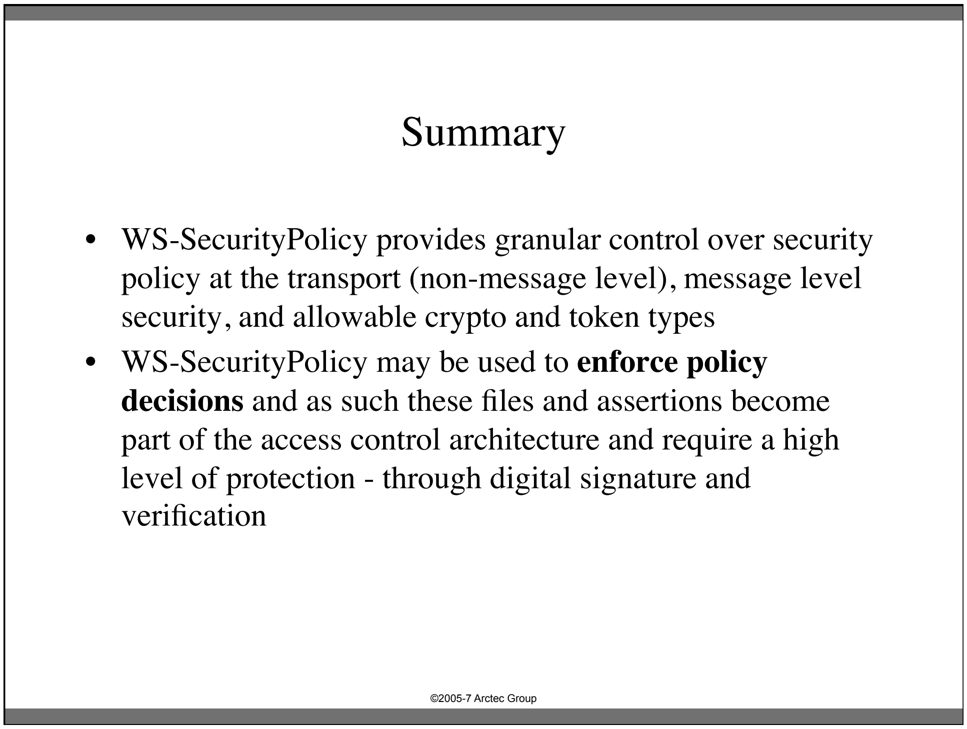
Asymmetric Binding Assertion

```
<sp:AsymmetricBinding xmlns:sp="http://schemas.xmlsoap.org/
   ws/2005/07/securitypolicy"> <wsp:Policy>
   <sp:InitiatorToken>
   <wsp:Policy>
   <sp:X509Token sp:IncludeToken="http://schemas.xmlsoap.org/
   ws/2005/07/securitypolicy/IncludeToken/AlwaysToRecipient">

        <wsp:Policy>

        <sp:WssX509V3Token10/>
                                                      </
   wsp:Policy>
   </sp:X509Token>
…
```



Asymmetric Binding Assertion (cont.)

```
…<sp:RecipientToken>
<wsp:Policy>
<sp:X509Token sp:IncludeToken="http://schemas.xmlsoap.org/ws/
   2005/07/securitypolicy/IncludeToken/Never">
<wsp:Policy>
<sp:WssX509V3Token10/>
</wsp:Policy>
…
</sp:RecipientToken>
<sp:AlgorithmSuite>
<wsp:Policy>
        <sp:TripleDesRsa15/>
</wsp:Policy>
…
```



Summary

- WS-SecurityPolicy provides granular control over security policy at the transport (non-message level), message level security, and allowable crypto and token types
- WS-SecurityPolicy may be used to **enforce policy decisions** and as such these files and assertions become part of the access control architecture and require a high level of protection - through digital signature and verification