

Presented by IBM developerWorks  
[ibm.com/developerWorks/](http://ibm.com/developerWorks/)



# Achieving Enterprise Application Security

IBM developerWorks **Live!**  
Enterprise Application Security

© 2007 IBM Corporation

V2.0.3

## **Achieving Enterprise Application Security**

Brought to you by IBM Corporation 2007

For the latest version of these materials, visit us on the Web at

**[ibm.com/developerworks/offers/techbriefings  
/details/security.html](http://ibm.com/developerworks/offers/techbriefings/details/security.html)**

V2.0.3

---

## Achieving Enterprise Application Security

These handouts are designed to give you copies of today's slides along with other useful materials. The slides have notes that expand upon points raised in the slides.

The latest version of these handouts can be found at

**[ibm.com/developerWorks/offers/techbriefings/  
details/security.html](http://ibm.com/developerWorks/offers/techbriefings/details/security.html)**

### Table of Contents

Achieving Enterprise Application Security .....	3
Welcome .....	4
Basics – Security goals for e-business applications .....	9
Basics – Confidentiality, Integrity, and Non-repudiation .....	16
Basics – Authentication, Authorization, Access Control, Privacy ..	39
Java, Java EE and WebSphere security.....	52
Web services, SOA and AJAX security.....	105
Security products from IBM.....	153
Attacks and malicious code .....	187
Resource and Q&A.....	199

Welcome

## Slide 1



Presented by IBM developerWorks  
ibm.com/developerWorks/


**Achieving  
Enterprise Application Security**

IBM developerWorks **Live!**  
Enterprise Application Security

© 2007 IBM Corporation  
V2.0.1

Application security is a must-have for successful e-commerce, yet it is hard to achieve. The goal is to make the cost of breaking into a system or stealing data more expensive than the value of the system contents or data in question. To accomplish this we use a layered, “defense-in-depth” approach to security. We manage security at multiple points and levels in our system architecture and applications to eliminate single points of security failure. Security must be designed into our systems and applications from the start; it is not an add-on. We need to consider security when defining application requirements, when modeling the application, when coding, when testing, and at application deployment. At runtime we must monitor the application for security events.

## Slide 2

Presented by IBM developerWorks 

## Objectives of this briefing

- Application security is essential for e-commerce. During this briefing you will get an understanding of
  - ▶ Application security
  - ▶ The definitions upon which it is based
  - ▶ The technologies upon which we build a secure stack
  - ▶ And how to build security into your applications
- We will use Java and Java technologies to illustrate many of the concepts and technologies
- You will gain an understanding of how to take a multilayered approach to application security

1-2 Achieving Enterprise Application Security © 2007 IBM Corporation

This briefing introduces application security concepts and standards. It is not intended to be an in-depth study, but to define and position many terms and issues associated with overall application security. We will reveal an application stack with security at every level.

## Slide 3

Presented by IBM developerWorks 

## What we *won't* cover

- Algorithms in depth
  - Encryption, hashing, etc.
- Network and infrastructure security
- Firewalls
- Intrusion detection
- Wireless security
- Risk management
- Forensics



- These are all important, but we don't have time today...

1-3 Achieving Enterprise Application Security © 2007 IBM Corporation

In this briefing we won't examine physical security, network security, disaster recovery or forensics.

## Slide 4

Presented by IBM developerWorks

## Agenda – what we *will* cover

- Application security basics and core technologies
  - ▶ Security goals for e-business applications
  - ▶ Confidentiality, Integrity, and Non-repudiation
  - ▶ Authentication and Authorization/Access Control
  - ▶ Privacy
- Java, Java EE and WebSphere security
- Web services and SOA security
- Security products from IBM
- Attacks and malicious code
- Resources
- Q&A
  
- *We'll take a break about halfway through...*

1-4      Achieving Enterprise Application Security      © 2007 IBM Corporation

But we *will* cover application security end-to-end. We'll start by examining the goals of application security, followed by a review of the basic security concepts. Next we'll see how those basics are applied as we build our security stack, adding additional layers of defense.

We'll proceed through the following topics:

- Introduction and security goals
- Confidentiality, integrity and non-repudiation
- Authentication, authorization and access control
- Java, Java EE and WebSphere security
- Web services, SOA and AJAX security
- A secure infrastructure with IBM security tools
- Attacks and malware

- Resources and wrap up

The security tools and products from IBM we'll discuss include

- Rational RequisitePro, Application Developer and Software Architect
- WebSphere Application Server
- Tivoli Access Manager, Identity Manager and Federated Identity Manager

We'll include demos that illustrate the security capabilities of some of these products.

*Mini glossary:*

**Java EE** – Java Platform Enterprise Edition, a programming platform based on a set of standard specifications and code for building scalable, secure, multi-tier applications in Java. (The former acronym, J2EE was changed to Java EE, or Java Platform Enterprise Edition, because we are way past version 2 of the platform.) Java EE defines a standard set of classes, communications, and facilities available to applications running in a Java-based enterprise application server.

**WebSphere** – the IBM software brand for middleware. In the context here we use the term to primarily represent WebSphere Application Server, the Java-based server that hosts and provides facilities for Java enterprise applications.

**Web services** – a set of communications standards specifying the sending of XML messages from a service requestor (like a client) to a service provider. The standards allow disparate systems to



communicate over the Internet. There are Web services security standards, known collectively as WS-Security, and we'll cover those in this presentation.

**SOA** – Service Oriented Architecture, an architectural approach to designing applications in which smaller functions provided by applications distributed around the network are reused, instead of being rewritten. Web services provide the communications layer in a SOA.

## Basics – Security goals for e-business applications

### Slide 5

Presented by IBM developerWorks 

## Agenda


- Application security basics and core technologies
  - ▶ Security goals for e-business applications
  - ▶ Confidentiality, Integrity, and Non-repudiation
  - ▶ Authentication and Authorization/Access Control
  - ▶ Privacy
- Java, Java EE and WebSphere security
- Web services and SOA security
- Security products from IBM
- Attacks and malicious code
- Resources
- Q&A



1-5 Achieving Enterprise Application Security © 2007 IBM Corporation

Let's get started.

## Slide 6

Presented by IBM developerWorks 

## Application security is just one part of a whole

- Application security is part of overall enterprise security
- Application security is not an add-on, it must be incorporated throughout
  - ▶ At requirements time
  - ▶ At design time
  - ▶ At code time
  - ▶ At test time
  - ▶ At installation time
  - ▶ At runtime

1-6 Achieving Enterprise Application Security © 2007 IBM Corporation

## Slide 7

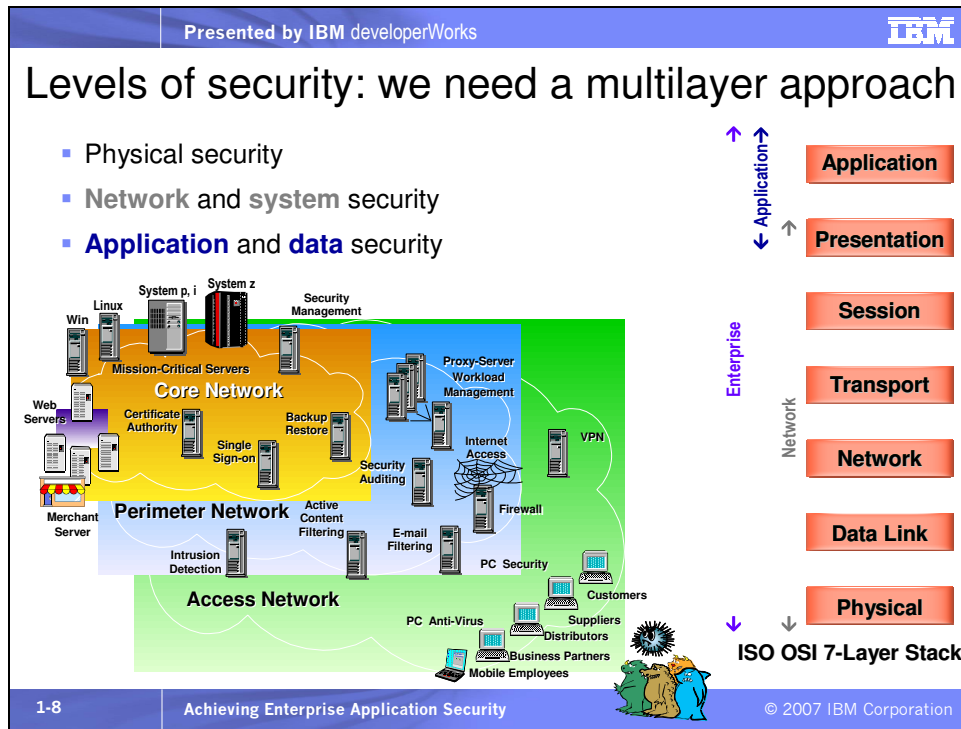
Presented by IBM developerWorks 

## Security goals and requirements

- There is no such thing as absolute security
  - ▶ There are risks and countermeasures to address these risks
- Nothing can ever be proven to be 100% secure
  - ▶ But we can make the cost of breaking into a system more expensive than the value of the information it contains
- Security requirements vary with different applications
  - ▶ There's no universal checklist
  - ▶ There are common requirements, but they may not all apply
- The emphasis must be on letting the "good guys" in as much as keeping the "bad guys" out
- Security must be based on strong, **open standards** to ensure interoperability between platforms

1-7 Achieving Enterprise Application Security © 2007 IBM Corporation

Slide 8



In order to let the good guys in while keeping the bad guys out, security is required at many levels. We need to secure

- Hardware
- Operating System
- Application Software
- Peripheral Devices
- Communication Equipment

Successfully subverting any of these can compromise the system and the information stored on the system. Each point requiring security maps to a level in the OSI stack. The ISO OSI model is the International Standard Organization's Open System Interconnect (ISO/OSI) model. It defines seven network layers starting from the

hardware and wiring and going all the way up to the applications running on the network. Each layer needs security.

The following list summarizes the different layers of the OSI model:

**7 Application** – The highest level of the networking stack, the application level protocol specifies a complete networking application – HTTP, HTTPS, XML, E-mail, directory services

**6 Presentation** – Identifies the messaging and protocol specifications necessary to display and represent information to the user – Encrypted data, compressed POP/SMTP

**5 Session** – Specifies protocols and messaging schemes that will operate together for the fulfillment of a task in a session – POP/25, SSL

**4 Transport** – Provides a means for transporting messages over the network and connecting multiple networks together into an overall network – TCP, UDP

**3 Network** – Identifies a machine as part of an overall network architecture and provides mechanisms for connecting multiple systems together – Packets IP, ARP

**2 Link** – Provides the necessary protocol framing to get two systems to successfully communicate – PPP, 802.11 (wireless)

**1 Physical** – Represents the lowest level of network connectivity – that of the wire or physical connecting that links machines together – ADSL, ATM

*Mini glossary:*

**HTTP** - Hyper Text Transport Protocol (HTTPS is secure HTTP), typically the protocol by which a browser communicates with a server

**XML** - eXtensible Markup Language, a standard based on SGML, or Standard Generalized Markup language. XML is a standard for self-describing data representation in which the name of the data and the value are both contained in a plain-text "document."

Different XML "vocabularies" allow the extensibility of XML itself

**POP3** - Post Office Protocol version 3, the protocol used when retrieving email from an email server

**SMTP** - Simple Mail Transfer Protocol, the protocol used when sending email to an email server

**TCP/IP** - Transmission Control Protocol/Internet Protocol, layers 4 and 3 (transport/network) of the OSI stack. TCP has guaranteed delivery

**UDP** - User Datagram Protocol, a layer 4 protocol in which there is no guaranteed delivery

**ARP** - Address Resolution Protocol, the protocol by which network IP addresses are mapped to hardware MAC addresses. RARP, or Reverse ARP goes in the opposite direction, mapping MAC address to IP address

**MAC** - Media Access Control, in this sense, the physical hardware address of a device on the network

**PPP** - Point-to-Point Protocol, a protocol for router-to-router and host-to-network communications, typically used over lower speed lines, like dial-up connections

**802.11** - a set of wireless networking standards

**ADSL** - Asymmetric Digital Subscriber Line, pertaining to the bit rate of data, a protocol allowing high speed communications over standard phone lines. Related to DSL, commonly provided by phone companies to subscribers for high speed Internet access


**ATM** - Asynchronous Transfer Mode, in this context, a very-high speed low-delay network switching technology

For more information on these networking acronyms, see

<http://www.cisco.com/univercd/cc/td/doc/cisintwk/ita/index.htm>

.

## Slide 9

Presented by IBM developerWorks 

## Seven ISO security requirements

1. **Identification:** who are you?
2. **Authentication:** how do I know your identity is true?
3. **Authorization:** are you allowed to perform this transaction?
4. **Integrity:** is the data you sent the same as the data I received?
5. **Confidentiality:** are we sure that nobody read the data you sent me?
6. **Auditing:** record of all transactions so we can look for security problems after the fact
7. **Non-repudiation:** both sender and receiver can provide legal proof to a third party (e.g. judge) that
  - ▶ the sender did send the message, and
  - ▶ the receiver received the identical message
8. And **Privacy:** addresses the access purpose and data owner choice

1-9 Achieving Enterprise Application Security © 2007 IBM Corporation

1. You usually identify yourself with your username
2. Authentication can be with something you know such as a password, something you have such as a badge or dongle or smartcard, or something you are such as biometry, including fingerprint, retinal scan, etc.
3. Once you are authenticated, the next decision is whether you have permission to do what you are trying to do
4. Integrity is all about making sure data doesn't change in transit. We use hashing and digital signatures to help with this
5. We use encryption to prevent unauthorized access to data
6. Auditing is important, especially time-stamping of activity and transactions
7. Auditing and digital signatures are used for non-repudiation

8. Privacy is a little different. It's a form of access control, but considers the purpose of access against the choices of the data subject (data owner)


## Basics – Confidentiality, Integrity, and Non-repudiation

### Slide 10

Presented by IBM developerWorks

## Agenda

- Application security basics and core technologies
  - ▶ Security goals for e-business applications
  - ▶ Confidentiality, Integrity, and Non-repudiation
  - ▶ Authentication and Authorization/Access Control
  - ▶ Privacy
- Java, Java EE and WebSphere security
- Web services and SOA security
- Security products from IBM
- Attacks and malicious code
- Resources
- Q&A



1-10 Achieving Enterprise Application Security © 2007 IBM Corporation

Now we'll discuss confidentiality, integrity, and non-repudiation.




## Slide 11

Presented by IBM developerWorks

## Confidentiality uses cryptography

- **Encryption** and decryption
  - ▶ The act of "hiding" and "un-hiding" information
- Symmetric key encryption
  - ▶ Uses a shared "secret" key and algorithm to encrypt & decrypt
  - ▶ Advantage: faster than asymmetric
  - ▶ Disadvantage: key must be securely distributed in advance
- Asymmetric key encryption
  - ▶ Uses a complementary pair of keys: either can encrypt but only the other can decrypt
  - ▶ One key is given out to the public, the "public" key (Kpub), while the other is held privately, the "private" key (Kprv)
    - Send a message encrypted w/ Kpub, only Kprv can decrypt
    - Send a message encrypted w/ Kprv, successful decryption with Kpub means sender holds Kprv



1-11 Achieving Enterprise Application Security © 2007 IBM Corporation

You can encrypt/decrypt using a "transformation" algorithm, such as reversing each bit in a message, or you can use a key (a number) with the algorithm to encrypt/decrypt, such as adding one to each byte in a message. Usually we use both, and there are several algorithms available. The key needs to be kept secret between the two parties wishing to securely communicate.

Symmetric encryption is also known as Secret Key encryption because a key is shared between two parties and the key must be kept secret. Some common algorithms include Triple DES (3DES) and AES. Its advantage is that it is fast. Its drawback is the "key distribution problem":

- The key must remain secret, and it must be distributed securely to anyone we want to talk with

- If we want secure conversations with  $n$  partners, we have to manage and distribute  $n$  keys, one for each

Kerberos uses secret key encryption. It is a popular security infrastructure built around a central security token server that allows participants to securely communicate with each other. We'll discuss that shortly.

*Mini glossary:*

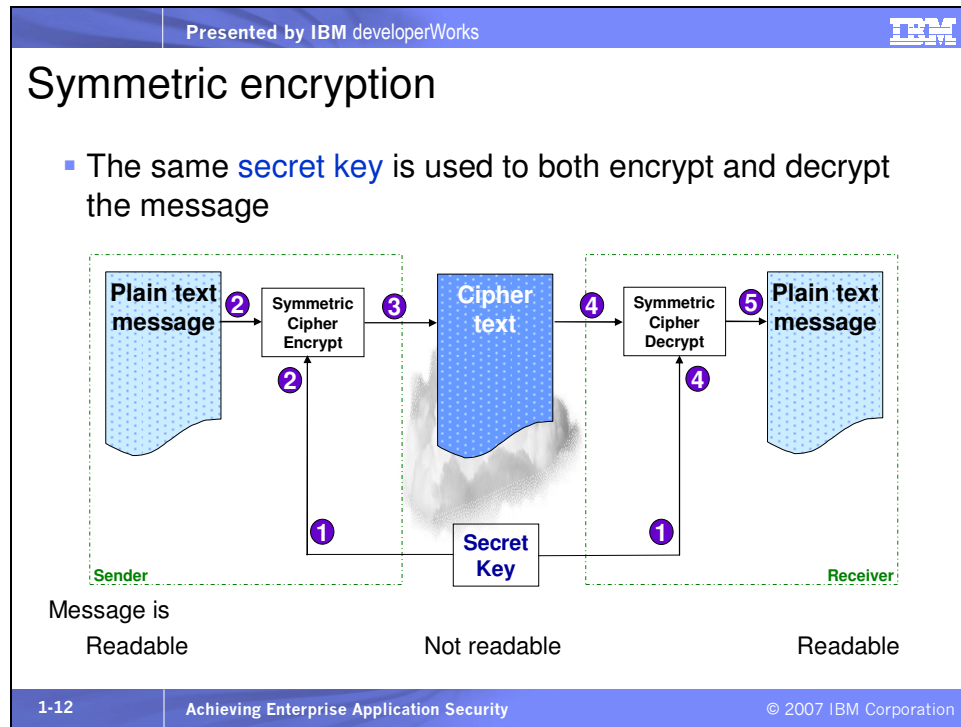
DES is the Data Encryption Standard. The DES algorithm is not secure enough and was cracked long ago. Triple DES significantly enhances security and is widely used. Advanced Encryption Standard is even more secure and is starting to be used more.

Here are some common encryption algorithms:

- Secret Key encryption
  - DES: Data Encryption Standard (56-bit and not particularly secure)
  - DESede: Triple DES (112-bit)
  - Blowfish: by Bruce Schneier
  - RC4 (128-bit)
  - AES: Advanced Encryption Standard (new, 128- & 256-bit)
- Public Key encryption
  - RSA (1024-bit)
  - Elliptic Curve (~170-bit)

XML encryption allows all or parts of an XML document to be encrypted.


## Slide 12



The steps are as follows:

1. Distribute symmetric secret key in advance
2. Use key and symmetric cipher encryption algorithm to encrypt the message
3. Send the message over the wire
4. Use secret key and symmetric cipher decryption algorithm to decrypt the message
5. And voila there is the original message

## Slide 13

Presented by IBM developerWorks 

## Asymmetric encryption...

- Each owner has a pair of complementary, asymmetric keys
  - ▶ They are different from each other
  - ▶ Encrypt with one, decrypt only with the other (in either direction)
  - ▶ We give one away (the Public key) and keep the other secret (the Private key)
  - ▶ If anyone encrypts a message with our public key, only we can decrypt the message (with our private key)
  - ▶ Conversely, if we encrypt a message with our private key, only our public key will decrypt it. So...
    - If a recipient successfully decrypts that message with our public key, they know we sent the message
- Advantage: Key distribution problem gone
- Drawback: asymmetric encryption is slower than symmetric

1-13 Achieving Enterprise Application Security © 2007 IBM Corporation

The keys in a pair only work with each other. You cannot encrypt and then decrypt with the same one of the pair. You need the other to decrypt.

Key point 1: if anyone encrypts a message with our public key, only we can decrypt that message

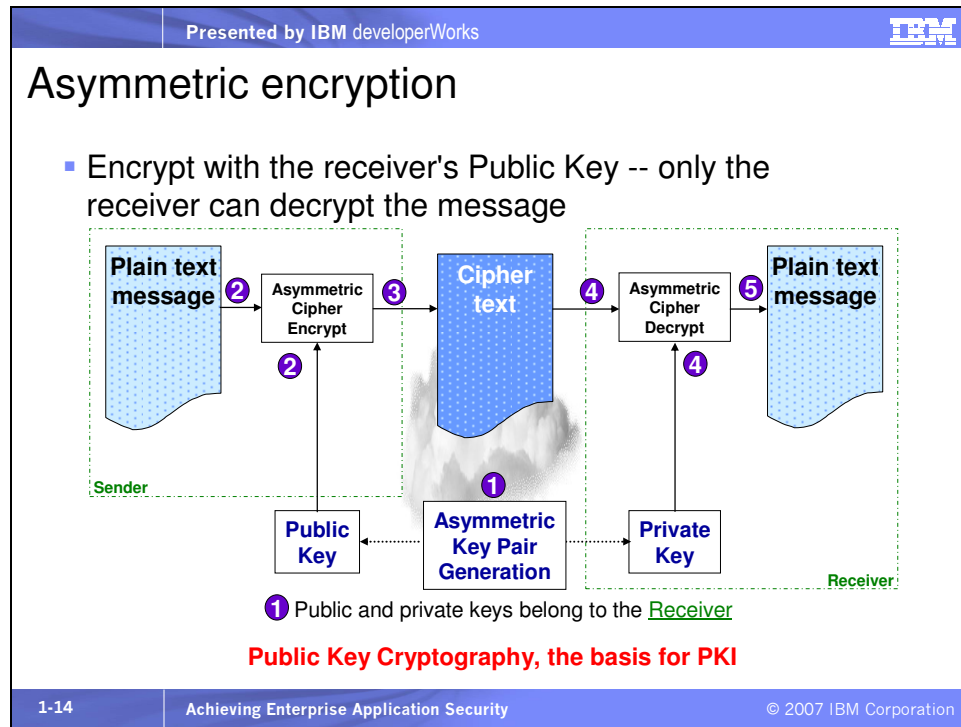
Key point 2: if we send a message we've encrypted with our private key to anyone and they successfully decrypt the message with our public key, they know that we sent (or at least encrypted) the message.

Hashing will help verify that decryption is "successful" by the use of hashing in digital signature -- discussed elsewhere. And key point 2 helps lead towards non-repudiation.

Public Key Cryptography requires a Public Key Infrastructure (PKI). You have to have a trusted authority that issues the keys. IBM developerWorks has a great overview of PKI topics and concepts at [www.ibm.com/developerWorks/security/library/s-pki.html](http://www.ibm.com/developerWorks/security/library/s-pki.html).

How are these key pairs created? Typically administrators (but sometimes users) use a tool to create the pair of keys. The keys have a mathematical relationship with one another. IBM ships a tool called gsk7ikm.exe (Windows version) that is the IBM Key Management tool. This tool can create keys and store them in various types of keyfiles or key database files. The key that will be the private key is stored in a password-protected file. The key that will be the public key can be submitted in a certificate request to a registration authority, which will validate the submitter and have the corresponding certification authority (to be discussed shortly) create a digital certificate containing the public key. The tool also allows a user to sign the certificate themselves, called self-signing. The standard JDK also includes keytool.exe (Windows version) to create keys.


## Slide 14



The steps are as follows:

1. Distribute **receiver's** public key in advance, while receiver keeps own private key
2. Sender uses receiver's public key and asymmetric cipher encryption algorithm to encrypt message
3. Message goes over the wire
4. Receiver uses own private key and asymmetric cipher decryption algorithm to decrypt ciphered message
5. And voila le message originale

## Slide 15

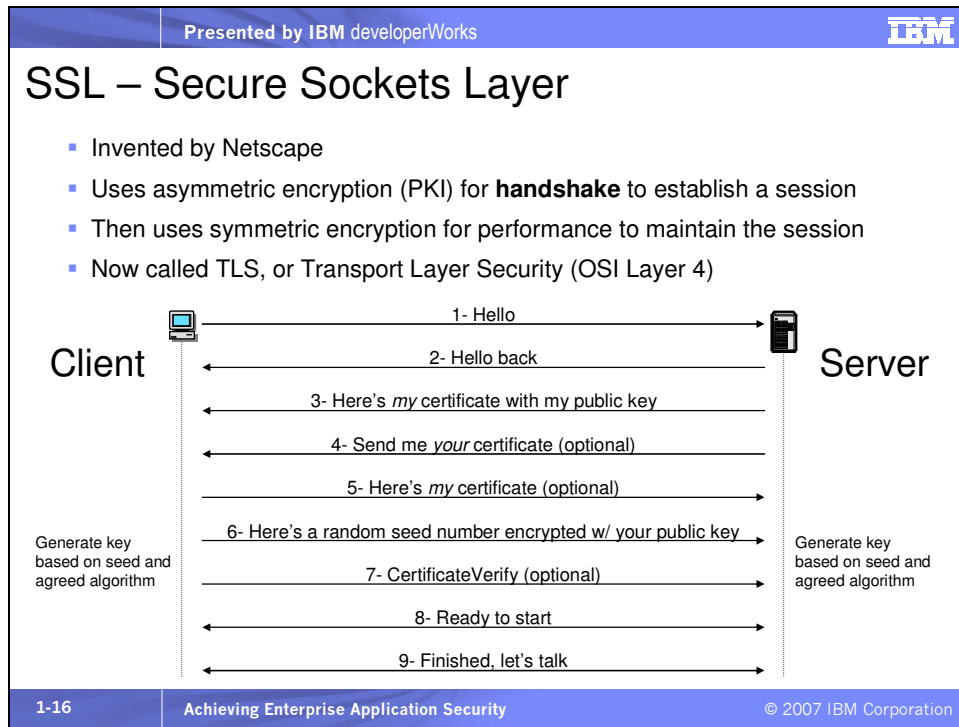
Presented by IBM developerWorks 

## Hybrid approaches

- Cryptography applications frequently combine these approaches
  - ▶ SSL is an example: It uses asymmetric encryption to transmit the secret key, then uses symmetric encryption for the data transmission
- Most of the ISO security issues are solved with protocols based on combinations of symmetric encryption, asymmetric encryption, and hash functions

1-15 Achieving Enterprise Application Security © 2007 IBM Corporation

## Slide 16

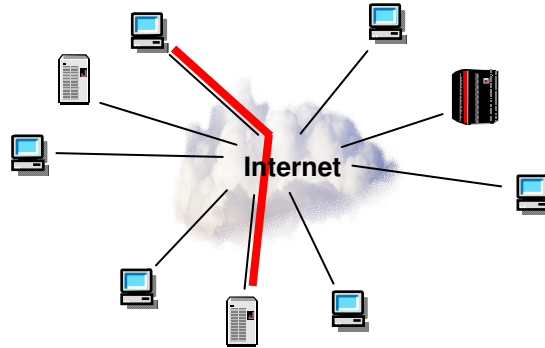


SSL V2 requires a server certificate only. Thus the client definitely knows the server. Usually the client user has to log into the server site so that authentication is in both directions. SSL V3 also requires a client certificate so that authentication is mutual, at the Transport layer (OSI Layer 4). The optional messages in the handshake are sent if client authentication is in use. The CertificateVerify message is used so the client can authenticate; without this message, just sending the client certificate earlier doesn't prove client identity.



## A VPN is a Virtual Private Network

- A VPN is a secure point-to-point connection between two systems, using encryption between the two endpoints
- Acts like a private network, but over the public Internet



- Once the VPN is established, data is "tunneled" across the connection, unavailable to the other systems on the net
- VPNs use SSL, IPSec, PPTP (Windows), or L2TP

VPNs can be established at different OSI layers. IPSec is at the level of OSI layer 3. L2TP and PPTP are layer 2, the datalink layer. L2F or Layer 2 Forwarding is a VPN protocol developed by Cisco. Open source VPN solutions include FreeS/WAN, OpenVPN, CIPE, Poptop, Stunnel, and PPP over SSH. Microsoft also now supports L2TP/IPSec. See <http://buildinglinuxvpns.net>. VPNs are often implemented using dedicated appliances.

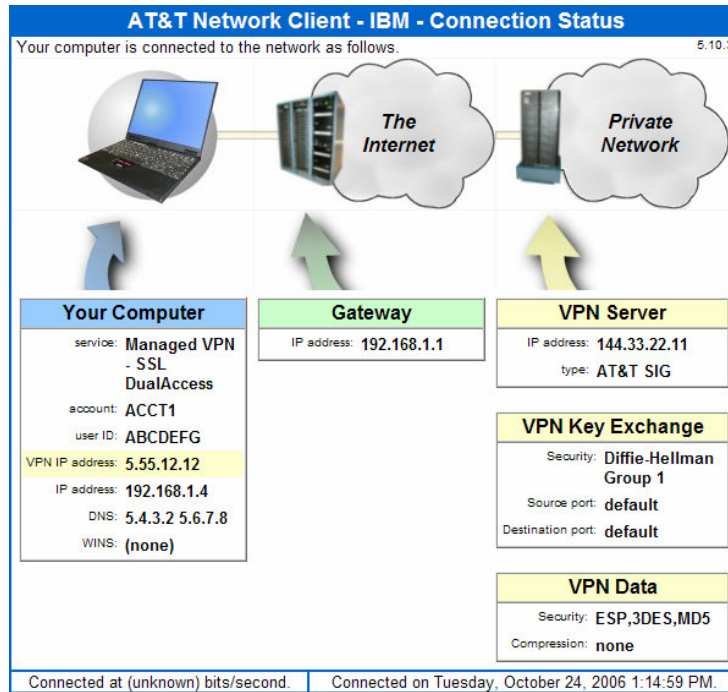
*Mini glossary:*

**IPSec** – Internet Protocol Security, a popular IETF open standard for VPNs that operate at the network layer of the OSI model.

**PPTP** – Point-to-Point Tunneling Protocol, a secure tunneling protocol typically used in Windows that operates at the Data Link layer of the OSI model

**L2TP** – Layer 2 Tunneling Protocol, an IETF standard operating at the Data Link layer for secure VPN connections

## A typical VPN client status screen



What are all these addresses and protocols?

### Your Computer

- Service - allows access to the company intranet and also the Internet
- VPN IP address - IP address of this machine on private network
- IP address - IP address of this machine on the local LAN without the VPN
- DNS - Domain name servers inside private network

### Gateway

IP address - Default gateway on the local LAN

### VPN Server

IP address - address of server at other end of VPN tunnel connection

### VPN Key Exchange

Security - Diffie-Hellman is a method of establishing a shared key over an insecure medium


### VPN Data

Security - ESP is Encapsulating Security Payload, a type of IPSec header that provides encryption and/or integrity protection

### *Mini glossary:*

As mentioned before, 3DES (Triple DES) is an encryption standard based on three successive invocations of DES, the Data Encryption Standard. MD5 is Message Digest 5, a hashing algorithm that creates an irreversible, unique digest of a message.

## Slide 17

Presented by IBM developerWorks 

## XML encryption

- The XML Encryption standard defines ways to encrypt all or parts of an XML document
  - The encrypted information is replaced with a single **<EncryptedData>** element
  - You can encrypt different parts of the same document with different keys
  - You can encrypt the whole document, a single element, or just the text of an element
  - When using XML Encryption with WS-Security for SOAP messages, the information about what was encrypted and how is stored in the **<header>** of the message

1-17 Achieving Enterprise Application Security © 2007 IBM Corporation

Here are the main elements of XML encryption:

**<EncryptedData>** element replaces the content being encrypted.

It contains:

**<EncryptionMethod>** Algorithm used to encrypt the data

**<CipherData>**

**<CipherValue>** Element containing the encrypted data

**<EncryptedKey>** element placed in security header, contains

**<EncryptionMethod>** Alg. used to encrypt symmetric key

**<KeyInfo>** Identifier of key used to encrypt symmetric key

**<CipherData>**

**<CipherValue>** Encrypted symmetric key value

**<ReferenceList>** List of **<DataReference>**s to content

encrypted with this symmetric key

Find the XML Digital Encryption standard at [w3.org/TR/xmlenc-core/](http://w3.org/TR/xmlenc-core/).

XML Canonicalization (C14N) specifies detailed and strict formats for canonical XML content. This is essential when using XML digital signatures. XML Canonicalization provides a normalized form for XML documents. We can't just do a string comparison between two XML documents, because hashing functions don't keep up with semantics. Consider the following two XML snippets:

```
<x y="12" z="13"> and  
<x z='13' y      =      '12'>
```

They are the same to a parser but are obviously different strings. Thus when hashed they will produce different digests in spite of having the same meaning. In order to validate a signature, the hashed content must be identical. So it is put into its canonical form before being hashed. You can find the complete canonicalization spec at [w3.org/TR/xml-c14n](http://w3.org/TR/xml-c14n).

The XML Encryption specification is defined by the W3C Working Group. See <http://www.w3.org/Encryption/>. It was started as joint proposal by IBM, Microsoft, Entrust, with the purpose of encrypting data and representing the result in XML. For example it can encrypt an entire XML document, elements, element content, arbitrary data, or a combination of these. For general information on XML and security, see the Apache XML Security project at <http://xml.apache.org/security/index.html>.


When XML encryption is used with SOAP, the key sent in the message is usually a symmetric key. That key is then encrypted with the public key of the intended receiver. The message is encrypted with (faster) symmetric encryption, but only the intended receiver can decrypt the symmetric key that decrypts the message.

To decrypt an encrypted XML document the appropriate key is needed. If it is a SOAP message the sender (encrypter) could have encrypted the **<KeyInfo>** element with the receiver's (decrypter's) public key. Or the key could simply be a secret key agreed upon by both parties in advance.

*Mini glossary:*

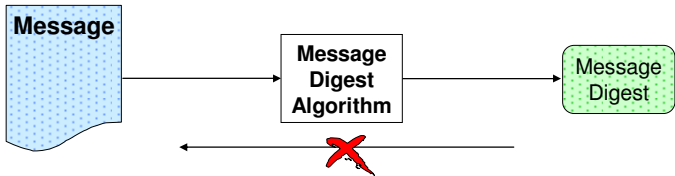
**SOAP** – Simple Object Access Protocol, an XML vocabulary specifying the format of XML messages used in Web services communications

## Slide 18

Presented by IBM developerWorks 

## Integrity needs hash functions

- A *hash* or *message digest* function reduces an arbitrary stream of bytes to a fixed-size number (usually 128 or 160 bits)
- It has two important properties:
  1. A small change to the original input stream produces a huge change in the hash code
  2. You can't go the other way. And given an input stream and its hash code, it's practically impossible to find a second stream with the same hash code



1-18 Achieving Enterprise Application Security © 2007 IBM Corporation

There are two key features of hash algorithms that make them essential:

1. Any change to the input changes the output, so even a single bit difference to a 1MB document as input, will cause the digest to be different.
2. You cannot regenerate same input given output. You can't take a digest and figure out what the original input message or document was.


There are several different hash algorithms available. MD5 (message digest 5) is a 16-byte hash that has been broken (cracked) but is still okay for one-way functions as long as you don't need absolute security. Still, it's best not to use it. SHA-1 (secure hash algorithm-1) is a 20-byte (160 bits) hash that is secure and

good to use. A third type of hash algorithm is called HMAC (hashed method authentication code), which appends a shared secret key to the content and hashes the whole thing.

According to Nelson Christopher, 160 bits is enough to hold the number of atoms in the world, so it is large indeed.

---

## Slide 19


Presented by IBM developerWorks 

### Integrity & non-repudiation with Digital Signature

- Provides proof of **integrity** of data
  - ▶ The signed data has not changed since it was sent
  - ▶ Does NOT provide confidentiality
- Based on hash functions and encryption
  1. Generate a message digest from the data to be signed
  2. Encrypt the digest to create the signature
- To verify the signature
  1. Regenerate a digest of the original data that was signed
  2. Decrypt the first encrypted digest
  3. Compare the two digests; a match verifies the signature
- Along with **Auditing**, Digital Signature gives us **Non-repudiation** when signing messages

**Signing = hash, then encrypt the hash with *sender's* private key**

1-19 Achieving Enterprise Application Security © 2007 IBM Corporation



It's important to keep in mind that signature does not provide confidentiality, only integrity. You can find the XML Digital Signatures standard at [w3.org/TR/xmlsig-core/](http://w3.org/TR/xmlsig-core/).

S/MIME is an IETF standard for Secure / Multipurpose Internet Mail Extensions. MIME defines how the body of an e-mail message is structured and permits e-mail to include enhanced text, graphics, audio, etc. S/MIME defines how to add encryption and digital signatures to parts of MIME message bodies. A description



of S/MIME can be found here at  
<http://www.rsasecurity.com/rsalabs/node.asp?id=2292>.

## Slide 20

Presented by IBM developerWorks

### Digital certificates

- For distribution of owner's Public key
- Irrefutably ties public key to owner
- Contains
  - Owner's distinguished name
  - Owner's public key
  - Time stamp and other information
- Issued and signed by a **Certification Authority**
  - Digital signing proves the certificate is valid
- Common type is **X.509** certificate
- PKI depends on Trust
  - Must trust key-issuer's authority
- For an in-depth introduction to PKI, visit:  
<http://ibm.com/developerworks/security/library/s-pki.html>

- 1 Create digest
- 2 Sign with CA's private key
- 3 Add to certificate

**Certificate**

**Owner's Public Key**

Info...

Dates

---

**CA Signature**

1-20
Achieving Enterprise Application Security
© 2007 IBM Corporation

We distribute a public key using a digital certificate. Certificates are created this way:

1. Owner generates a public/private pair of keys
2. Owner submits his DN and the public key to a CA in a Certificate Request, while locking the private key up in a keyfile
3. The CA verifies the owner & signs the certificate with the CA's private key (standard signature procedure)
4. Owner receives the signed certificate containing his public key and DN back from the CA.

The certificate can now be distributed to everyone -- this is how the public key inside is tied to the owner's identity and is made available to the public

Sometimes a Certification Authority is called a Certificate Authority. The CA works in conjunction with a Registration Authority (RA). The RA, usually visible and accessible on the Internet, receives and verifies requests from clients for digital certificates. The RA does due diligence on potential certificate holders. Then a CA creates and issues digital certificates. The CA is not exposed to the Internet. Both are part of a Public Key Infrastructure (PKI).

You can buy certificates or obtain trial certificates from several companies including Verisign, at

<http://www.verisign.com/products/site/index.html?sl=070303>,

and Thawte. Like Verisign, Thawte offers trial certificates. See

<http://www.thawte.com/ucgi/gothawte.cgi?a=w141001582670490>

00. Many organizations provide their own certificates for their users or employees, signing them with the organization's certificate. Those organizations' own certificates are often signed by Verisign or another "higher level" CA.

This hierarchy of certification authorities is called a "trust chain."

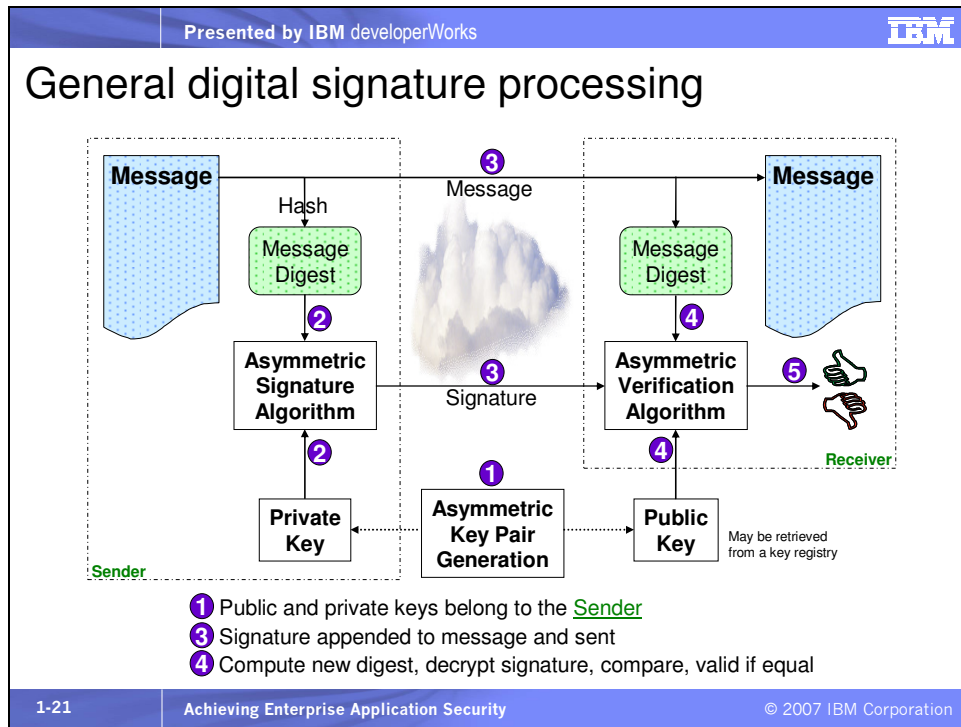
It's also important to keep track of certificates that are no longer valid. A Certificate Revocation List or CRL is a list of certificates that have been revoked for one reason or another.

*Mini glossary:*

**DN** – Distinguished Name, a unique name for an entity, usually as stored in a directory, the uniquely identifies the entity, for example

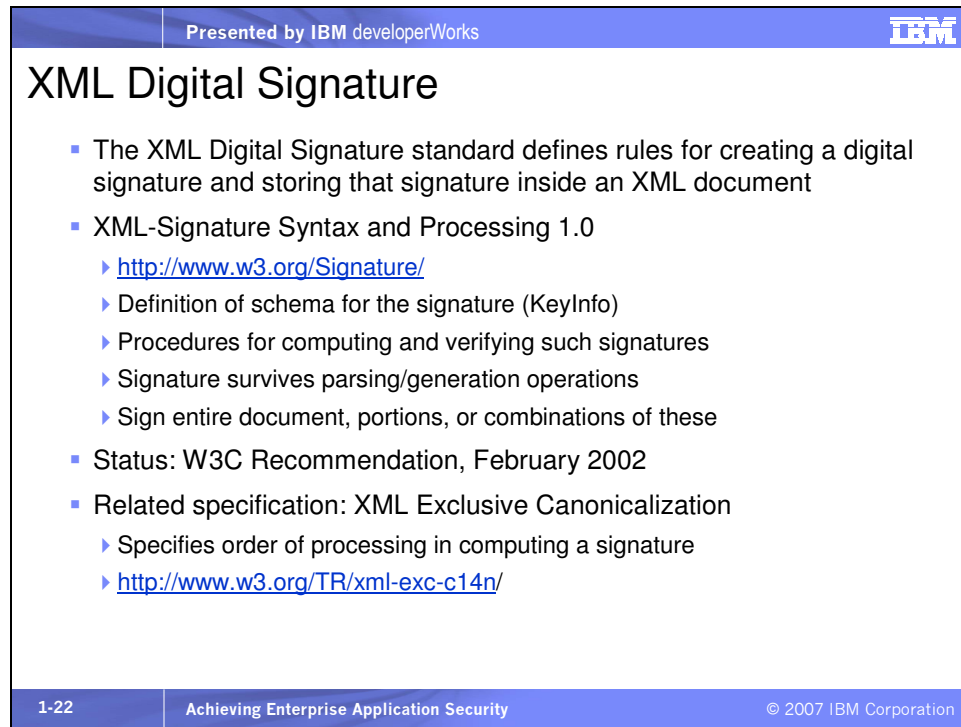
cn=John, ou=Development, o=IBM, c=US, where cn is the common name, ou is the organizational unit, o is the organization, and c is the country. LDAP uses distinguished names.

Slide 21



1. Distribute **sender's** public key in advance
2. Sender creates digest of message, encrypts digest with own private key creating signature
3. Original message and appended signature sent and receiver uses sender's public key and signature as input for verification
4. Receiver computes new digest of original message, decrypts signature to obtain sender's digest, compares the two
5. If they match message integrity is maintained, and sender did send the message

## Slide 22



Presented by IBM developerWorks

## XML Digital Signature

- The XML Digital Signature standard defines rules for creating a digital signature and storing that signature inside an XML document
- XML-Signature Syntax and Processing 1.0
  - ▶ <http://www.w3.org/Signature/>
  - ▶ Definition of schema for the signature (KeyInfo)
  - ▶ Procedures for computing and verifying such signatures
  - ▶ Signature survives parsing/generation operations
  - ▶ Sign entire document, portions, or combinations of these
- Status: W3C Recommendation, February 2002
- Related specification: XML Exclusive Canonicalization
  - ▶ Specifies order of processing in computing a signature
  - ▶ <http://www.w3.org/TR/xml-exc-c14n/>

1-22 Achieving Enterprise Application Security © 2007 IBM Corporation

You can find the XML Digital Signatures standard at [w3.org/TR/xmlldsig-core/](http://w3.org/TR/xmlldsig-core/).

An XML digital signature is stored in a **<Signature>** element that has three main parts:

- **<SignedInfo>** – Information about what is signed
- **<SignatureValue>** – The value of the digital signature itself
- **<KeyInfo>** – The public key used to verify the signature

In order to create an XML digital signature,

- Calculate a **<DigestValue>** and create **<Reference>** elements for data to be signed
- Put **<DigestValue>** and **<Reference>** elements in **<SignedInfo>**

- Sign the entire **<SignedInfo>** element to create a **<SignatureValue>** element
- Add **<SignedInfo>**, **<SignatureValue>**, and **<KeyInfo>** to **<Signature>**

Other things to consider about XML Digital Signatures:

An XML digital signature can apply to more than one thing. If you use a digital signature this way, there will be one **<SignedInfo>** element for each thing you sign.


If a signature applies to multiple items, and the signature doesn't verify, that only tells you that at least one of those items was modified in transit. The signature won't help you figure out which one. (If you're debugging an application, this could be important; if you're getting a message from someone else, you'll most likely just reject the message and move on.)

A digital signature can sign anything, not just XML. A single signature might sign two different parts of an XML message, as well as binary content sent as an attachment to the message.

Each **<Signature>** contains at most one **<KeyInfo>** element, providing information about the key used to sign the message. If a document is signed with different keys, there will be multiple **<Signature>** elements.

The sender's public key is often sent along with the message. It's also possible for the **<KeyInfo>** element to refer to the URL of a public key stored somewhere else.

## Slide 23

Presented by IBM developerWorks 

## Example: XML Signature (no SOAP)

```

<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
    <Reference URI="#wssecurity_body_id_2601212934311668096_1040651106378">
      <Transforms>
        <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
      <DigestValue>AWQKpmksMpzzT4PxcizO980gVHw=</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>bNhT+DsNN9PR [binary data has been truncated]</SignatureValue>
  <KeyInfo>
    <wsse:SecurityTokenReference>
      <wsse:Reference URI="#wssecurity_binary_security_token_id_1603091_4272645" />
    </wsse:SecurityTokenReference>
  </KeyInfo>
</Signature>

```

**What is signed?**

**Signature Value**

**Public Key (optional)**

1-23 **Signature Block** ation

The **<Reference>** in **<SignedInfo>** refers in this case to the content that is hashed (the **<Body>** element not shown here), with the value of that hash in **<DigestValue>**. The **<Reference>** in **<KeyInfo>** refers to the public key required to validate the signature.

To validate the signature:

1. Decrypt **<SignatureValue>** with the sender's (or signer's) public key to get the **sender's** (signer's) digest of **<SignedInfo>**
2. Generate the receiver's digest of **<SignedInfo>** and compare it to the sender's
3. If the digests in steps 1 and 2 match, we know that the signature was calculated with the sender's private key *and* that the message hasn't changed.

**Basics – Authentication, Authorization, Access Control, Privacy**

## Slide 24

Presented by IBM developerWorks 

## Agenda

- Application security basics and core technologies
  - ▶ Security goals for e-business applications
  - ▶ Confidentiality, Integrity, and Non-repudiation
  - ▶ Authentication and Authorization/Access Control
  - ▶ Privacy
- Java, Java EE and WebSphere security
- Web services and SOA security
- Security products from IBM
- Attacks and malicious code
- Resources
- Q&A




1-24

Achieving Enterprise Application Security

© 2007 IBM Corporation

Let's move on to authentication, authorization and access control.

## Slide 25

Presented by IBM developerWorks 

## Authentication

- Authentication is the process of proving your identity
  - ▶ Once authenticated, credentials are typically created for the user so that subsequent authentications are not necessary
- Single sign-on (SSO)
  - ▶ Allows a user to sign on once and then be authenticated for multiple applications
  - ▶ SSO can be implemented with security tokens, such as LTPA, or Lightweight Third Party Authentication
    1. A user of a Web application authenticates to a server
    2. The server creates an LTPA token and stores it in the HTTP response
    3. Subsequent requests from the user contain the token
    4. The server recognizes the token and does not require user reauthentication

1-25      Achieving Enterprise Application Security      © 2007 IBM Corporation

Where do we store the information necessary to authenticate a user? In a directory. And speaking of directory protocols, what is the difference between X.500 and LDAP? X.500 is the ISO standard for directory services. It organizes entries in a hierarchical namespace. An example entry might be

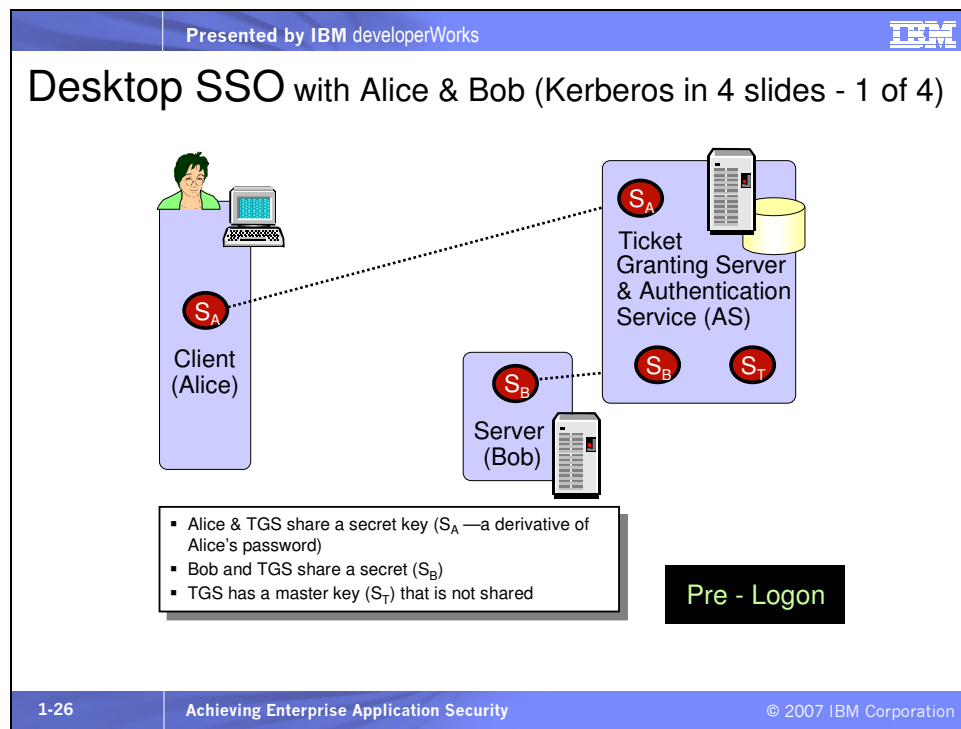
cn=jeff, o=ibm, c=us

This specifies a common name, an organization, and a country. And there might also be a hashed password stored. X.500 Directory Access Protocol (DAP) is a mechanism for accessing X.500 directory services, but DAP is complex to implement. Lightweight Directory Access Protocol, or LDAP, is a streamlined and simplified networking version of DAP for accessing a directory over TCP/IP. LDAP is typically used as a user registry in an



enterprise. Its structure is optimized for read access, slower on writes. The default port to access an LDAP directory is 389, but LDAP access can be more secure by tunneling it over SSL, defaulting to port 636. IBM Tivoli Directory Server (TDS) is an example of an LDAP directory. For more information on LDAP see [http://en.wikipedia.org/wiki/Lightweight\\_Directory\\_Access\\_Protocol](http://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol).

---

 Slide 26


Kerberos is a secret key-based service for providing authentication in a network. The network is assumed to be security hostile, and so there must be a way of ensuring that people logging in really are who they say they are. As we consider how Kerberos accomplishes this, let's consider three basic phases:

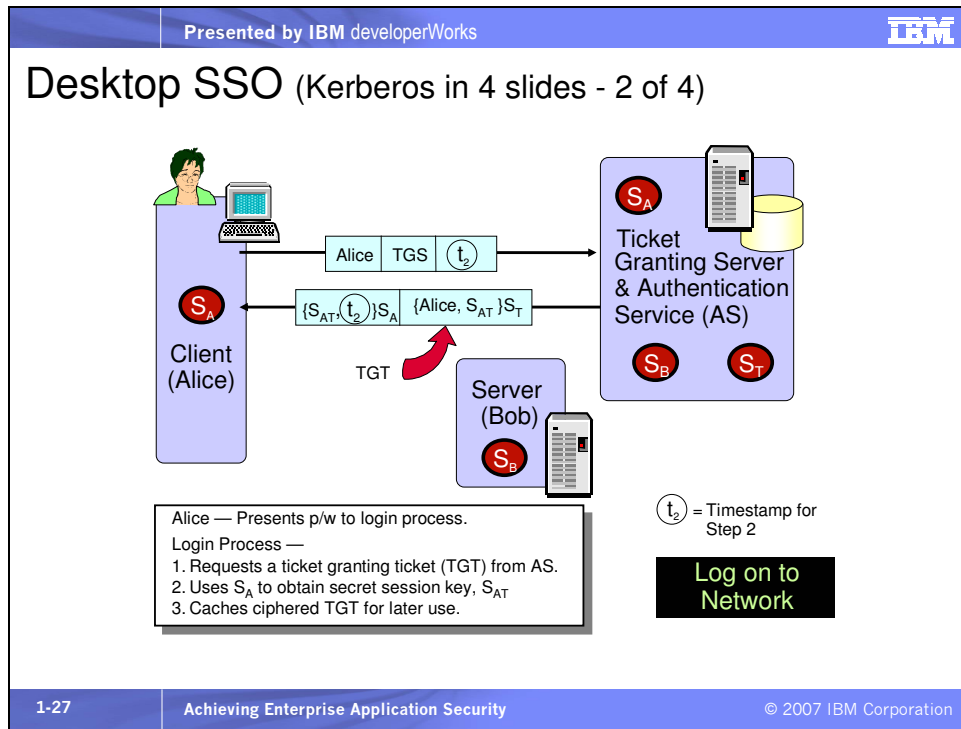
- Pre-login: Before users log in, some set-up must occur.

- Login: This is the login session, the user's initial interaction with the network.
- Accessing remote nodes: In the course of the login session, the user may access remote resources (such as hosts on which the user has accounts, or file servers on which the user has files).

This slide deals with the first phase--pre-login. Here, the Ticket Granting Server shares a secret key with each user and each server that will be using Kerberos. For users, the secret key is derived from the user's password. So, when the user's password changes, so too does the user's master key.

You might ask how the password is shared between the Ticket Granting Server and the user. Essentially, the initial distribution is kept secret through some "out of band" process, such as secure mail, telephone, in person, and so on. When users change their passwords, the change is made via entry of the password during a protected session between the user and the Ticket Granting Server.

Slide 27

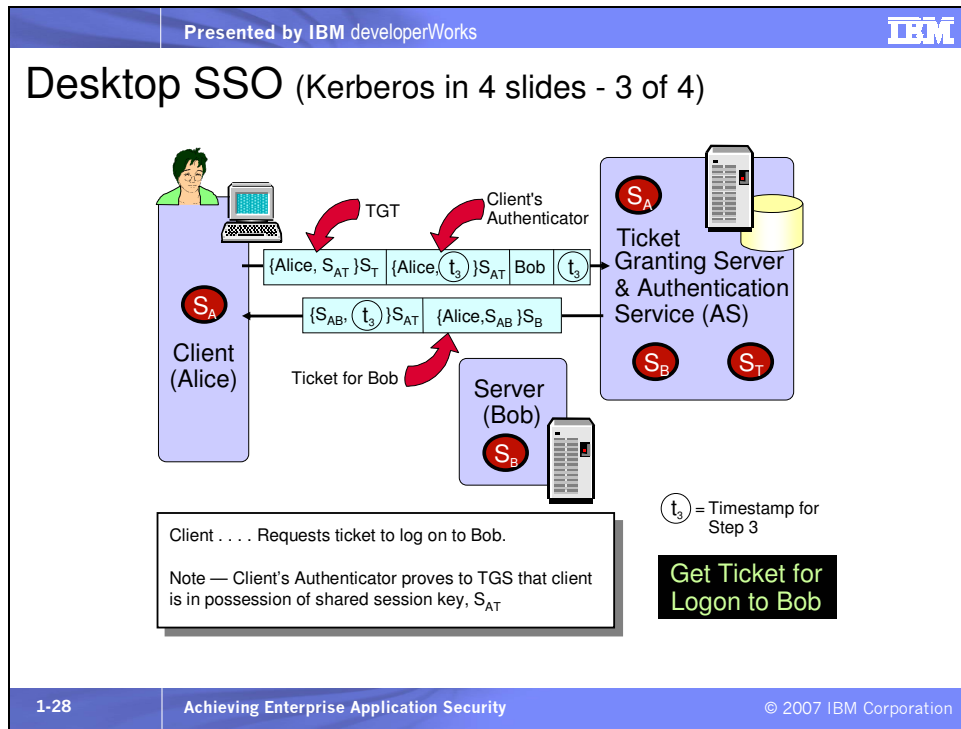


Alice logs into a workstation by supplying her name and password. Alice's master key is derived from her password. The workstation, which performs the authentication protocol on behalf of Alice, could remember her password during the whole login session and use that when proof of her identity was required. But that is not good security practice for various reasons. One worry is that Alice might, during the login session, run untrustworthy software that might steal her password. To minimize harm, the first thing that happens is that the workstation asks the KDC for a session key  $S_{AT}$  for Alice to use for just this one session. The workstation makes this request by sending a message to the KDC, in the clear (unencrypted), providing Alice's account name.

The session key  $S_{AT}$  is then used on Alice's behalf when asking for tickets to resources on the network. The key  $S_{AT}$  is only valid for some small amount of time, generally a few hours. If anyone steals  $S_{AT}$ , then they can impersonate Alice, but only until  $S_{AT}$  expires. During the initial login, the workstation, on Alice's behalf, asks the KDC for a session key for Alice. The TGS generates a session key  $S_{AT}$ , and transmits  $S_{AT}$  (encrypted with Alice's master key,  $S_A$ ) to the workstation. The TGS also sends a ticket-granting ticket (TGT), which contains  $S_{AT}$  (and other information such as Alice's name and the TGT's expiration time) encrypted with the TGS's master key ( $S_T$ ).

The workstation uses Alice's master key  $S_A$  to decrypt the encrypted session key,  $S_{AT}$ . Then the workstation forgets Alice's password and only remembers  $S_{AT}$  and the TGT, for later use.

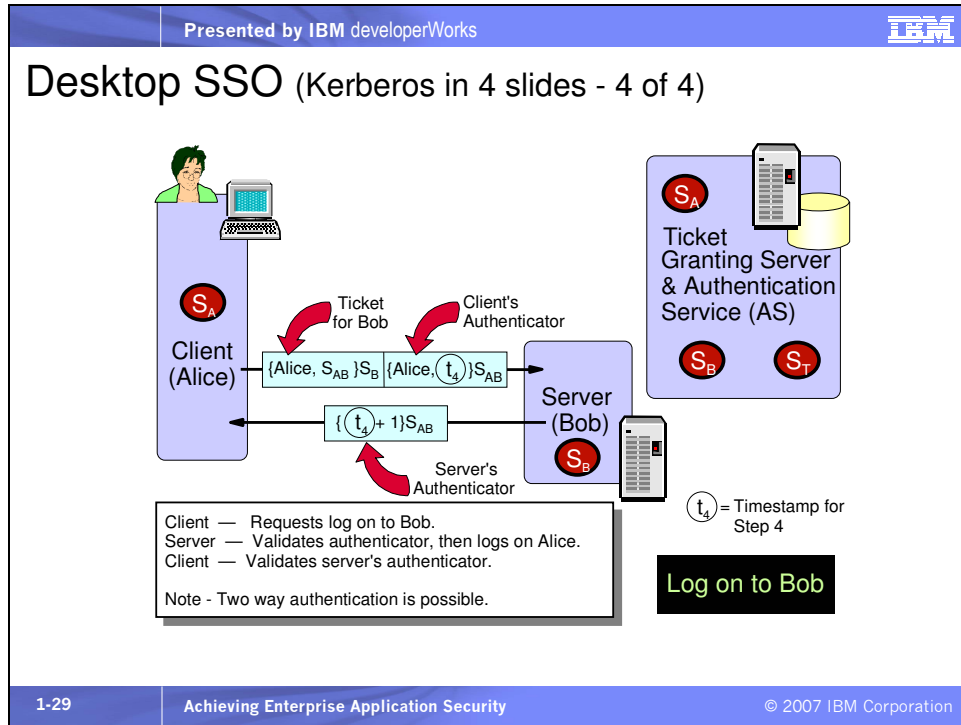
Slide 28



When Alice needs to access a remote resource, her workstation transmits the TGT to the KDC, along with the name of the resource to which Alice needs a ticket (in this case, Bob). (Note that there is other authentication information sent, but for sake of simplicity, we're only covering the main highlights here. If you are interested in a full description, you might try the very readable Kerberos write-up in *Network Security--Private Communication in a Public World*, by Charlie Kaufman, Radia Perlman, and Mike Speciner.) The KDC decrypts the TGT to discover S<sub>AT</sub>, and uses S<sub>AT</sub> to encrypt the Alice-Bob session key for Alice. Essentially, the TGT informs the KDC to use S<sub>AT</sub> instead of Alice's master key. Also included is the ticket for Bob, which consists of the name Alice, the Alice-Bob session key (S<sub>AB</sub>) and some additional information, all encrypted using Bob's

master key,  $S_B$ . Remember, the KDC is aware of all the master keys of all users and servers in its realm, or sphere of control.


Slide 29



The workstation now sends a request to Bob, consisting of the ticket for Bob and the client's authenticator. Bob decrypts the ticket and discovers the key  $S_{AB}$  and the name Alice. Bob now assumes that anyone with knowledge of  $S_{AB}$  is acting on Alice's behalf.

The client's authenticator is used to provide replay detection. Although Kerberos is not yet supported by WebSphere Application Server, Tivoli Access Manager has Kerberos capabilities and supports the above scenarios. (Thanks to Mike Campbell of IBM Tivoli for these four Kerberos slides.)

## Slide 30

Presented by IBM developerWorks 

## Authorization and Access Control

- Pre-requisites:
  - ▶ **Identification**
    - Usually with a username
  - ▶ **Authentication**
    - Proof of who you are, with something you know, have, or are
    - Most common is username / password combination
- Access control is a key aspect of information security
  - ▶ Only **authorized** people have access to information
  - ▶ Trust that there is no way that a user can or will misuse their privileges
- Access control systems **enforce** rules of “authorized access”

1-30 Achieving Enterprise Application Security © 2007 IBM Corporation

Access Control has several requirements:

- Low probability of *rejecting* an *authorized* user
- Low probability of *accepting* an *unauthorized* user
- Ease of administration
- Minimal effect on throughput
- Reliability
- Non-circumvent-able


Yet access control has “induced” vulnerabilities that stem from improper use of access control mechanisms. Some of these vulnerabilities are based on

- Avoiding access control mechanisms
- Misusing privileges
- In the name of increasing

- Performance
- Functionality
- Buzz-word compliance

---

## Slide 31

Presented by IBM developerWorks 

### Access Control and Access Control Lists (ACLs)

- Each protected **object** has an “owner”
- Owner grants to a **subject** the **ability** (permission) to access that **object**
- Access granted to **Subject** = {**users** and **groups**}
- Access permissions stored in terms of
  - ▶ {**subject**, **object**, **permissions**}
  - ▶ {**Fred**, **FileA**, **Read**}
  - ▶ {**Accountants**, **TaxFileB**, **Write**}
- Default is no access unless explicitly allowed

1-31 Achieving Enterprise Application Security © 2007 IBM Corporation

Access control can be discretionary or mandatory. With Discretionary Access Control (DAC) the owner of a resource determines the resource access policy. Two examples of DAC, discussed here, are ACLs and role-based access control (RBAC). In Mandatory Access Control (MAC) the access policies are determined by the system. MAC is typically used in systems that process highly sensitive data, systems that handle multiple classification levels between subjects and resources. Label-Based Access Control (LBAC) is an example of MAC. With LBAC, sensitivity labels are associated with *all* resources, as well as *all*



subjects that might want to access those resources. In order for a subject (user) to access a resource, the subject must have a label equal to or higher than the requested resource. For example, a subject with a Top Secret clearance is permitted to access a resource with a Secret classification.

DB2 9 supports LBAC. For more information on that see <http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.admin.doc/doc/c0021487.htm>.

We'll discuss RBAC next. Access Control Lists (ACLs) are a common way to maintain who has what kind of access to which resources.

## Slide 32

Presented by IBM developerWorks


## Role-based Access Control (RBAC)

- What you have access to is based on your “role”
  - ▶ {Accountant, TaxFileA, Write}
- Note that “role” is very similar to group
  - ▶ In practice, almost all “role” based rules can be implemented using groups
- Requires some means of identifying which role user is “using”
- In Java EE, a security role is associated with an application
  - ▶ Different from a group, associated with an organization

1-32 Achieving Enterprise Application Security © 2007 IBM Corporation

Role-based access control, such as that used by Java EE, provides a level of indirection that allows access control specifications to be independent of the location where the application is installed. The role, instead of a user or group, is specified in the ACL. The independence is due to the further mapping of users and groups to roles. We'll describe this shortly.

## Slide 33

Presented by IBM developerWorks 

## Privacy

- Different from authorization
- Pertains to “*data subject*” or “data owner” vs. “data custodian”
- Accommodates
  - ▶ Data subject choices (opt in/out) for access to their data
  - ▶ Purpose of access to that data
- E.g. data subject’s choice is that a doctor may access their medical records when the doctor’s acting as a physician, but not when acting as a consultant to an insurance company’s marketing department
- Policy-based, policies written with
  - ▶ P3P: Platform for Privacy Preferences
  - ▶ EPAL: Enterprise Privacy Authorization Language

1-33 Achieving Enterprise Application Security © 2007 IBM Corporation

By "data subject" we mean the person whom the data is about. We make a distinction between data that does not uniquely identify the person and data that does, called "personally identifiable information," or PII. It is the PII that we must keep private, according to the preferences of the data subject. These preferences are often expressed as opt-in/opt-out choices.

Java, Java EE and WebSphere security

Slide 34


Presented by IBM developerWorks 

## Agenda

- Application security basics and core technologies
  - ▶ Security goals for e-business applications
  - ▶ Confidentiality, Integrity, and Non-repudiation
  - ▶ Authentication and Authorization/Access Control
  - ▶ Privacy
- **Java, Java EE and WebSphere security**
- Web services and SOA security
- Security products from IBM
- Attacks and malicious code
- Resources
- Q&A




## Slide 35

Presented by IBM developerWorks 

## Security layer comparison

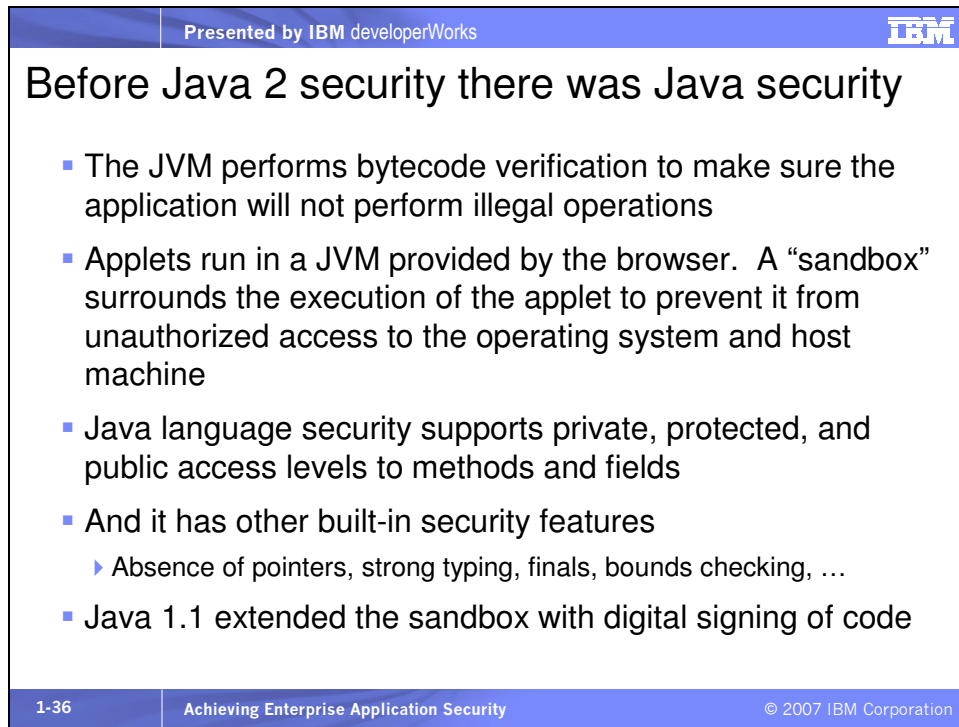
- Java 2 Security
  - ▶ Code-based security
  - ▶ Defined in Policy files
  - ▶ Enforced at runtime
- JAAS Security
  - ▶ Principle- and Subject-based security
  - ▶ Defined in application code
  - ▶ Enforced programmatically
- Java EE Security
  - ▶ Role-based security
  - ▶ Defined in configuration settings or within application code
  - ▶ Enforced by runtime and/or programmatically
- WS-Security
  - ▶ End-to-end propagation of credentials inside XML Web services messages




1-35 Achieving Enterprise Application Security © 2007 IBM Corporation

We can add to this very high level outline that in a Service Oriented Architecture (SOA), cross-domain trust or "federation" and the flexible management and propagation of security credentials and identities are imperative for successful security. We'll discuss that soon.

## Slide 36



Presented by IBM developerWorks 

## Before Java 2 security there was Java security

- The JVM performs bytecode verification to make sure the application will not perform illegal operations
- Applets run in a JVM provided by the browser. A “sandbox” surrounds the execution of the applet to prevent it from unauthorized access to the operating system and host machine
- Java language security supports private, protected, and public access levels to methods and fields
- And it has other built-in security features
  - Absence of pointers, strong typing, finals, bounds checking, ...
- Java 1.1 extended the sandbox with digital signing of code

1-36 Achieving Enterprise Application Security © 2007 IBM Corporation

The original sandbox model prevented any code not on the CLASSPATH from accessing resources outside of the sandbox. This code, loaded by the ClassLoader, was untrusted. JDK 1.1 changed the sandbox model to allow signed code to access resources outside of the sandbox. This allowed signed code downloaded from Web sites to execute with trust as long as the signature could be validated.

The Java language has several security features built-in. Private access means that a data primitive, object or method can only be accessed by code within the same class defining the primitive, object or method. Protected access means access from any class in the same package as that defining the target. Public access means that access is allowed from anywhere in the code.


Other built-in security features include the absence of pointers (preventing unauthorized memory access), strong typing, final primitives, objects and methods that can't be changed at runtime, disallowal of operations on uninitialized variables, and array bounds checking (prevents buffer overruns).

*Mini glossary:*

**JVM** – the Java Virtual Machine creates a layer of independence between the execution of the Java application and the underlying operating system. The Java application sees the same JVM regardless of which platform the JVM runs on. This is why Java applications are portable. The JVM has been ported to many platforms but to the Java application there appears to be no difference between the same JVM running on different platforms.

**JDK** – Java Development Kit

## Slide 37

Presented by IBM developerWorks 

## Java 2 security is policy-based

- Security is extended to local code besides applets
- *Policy* is declarative, stored in a file or other repository
- Adds finer-grained access control to resources
  - ▶ *Permission* class
    - Represents what the program is allowed to do
    - Can be granted to a code class or group of classes -- the *CodeSource* class associates the codeBase URL (its location) with signers (certificates)
- Each class belongs to one *ProtectionDomain*
  - ▶ Relates a *CodeSource* object to a set of Permissions
- The *AccessController* class is an evolution of the Java 1.1 *SecurityManager* class
  - ▶ Used from the code to check or control permissions and access

1-37      Achieving Enterprise Application Security      © 2007 IBM Corporation

With Java 2 security access control can be specified at the class level for particular resources and for particular types of access rather than all or nothing. Standalone applications can now be subject to security checking. The security policy is easy to configure and is stored external to the code, usually indicated by a URL. Thus the policy file can be local or remote.

The *Permission* class must implement an *implies()* method, which actually provides access control and allows flexibility. For example, if a *FilePermission* is granted to allow access to `file:/c:/a/b/*`, then permission to access `file:/c:/a/b/c` is implied. A *CodeSource* object may or may not have any signers. If not, the codebase specified in the *CodeSource* is unsigned. The URL in a *CodeSource* object may be local or remote. Either way the




execution of the code is under control of the permissions granted for the codeBase in the policy file. Note that the URL of a codeBase may also specify a WAR or JAR file.

A class loader loads a class from a particular URL. Based on the ProtectionDomain containing the CodeSource with the URL, the corresponding set of permissions dictate what that class will be allowed to do. If a protected resource is to be accessed, all classes on the stack must have adequate permissions. For example, if class A is loaded from one CodeSource (thus is in one ProtectionDomain) and it calls Class B (from another CodeSource), if class B tries to access class C, both class A and B must have appropriate permission to access class C. In fact every non-Java core class on the stack must have permission to execute the action. To override this we use AccessController.doPrivileged(), which stops security checks walking back up the stack.

The AccessControlContext class is related to the AccessController class and allows a program to examine its current calling context, the state of the AccessController. The class AccessControlContext is also useful when an access control decision has to be made in a different context, since it allows a context to be passed from one thread to another.

## Slide 38

Presented by IBM developerWorks

## Java 2 security – page 2

- Java.security.Policy
  - ▶ Embodies a policy file (by default)
  - ▶ Policy file contains a set of Permissions for a set of CodeSource objects

```
grant codeBase "http://www.ibm.com", signedBy "jeff"
{
    permission java.io.FilePermission "/tmp/abc", "read";
};
```
  - Grants read permission to code from ibm.com, signed by Jeff, on the /tmp/abc directory
  - ▶ Can programmatically create permissions
    - `perm = new java.io.FilePermission("/tmp/abc", "read");`
- Policies are typically stored in `.../jre/lib/security/java.policy`
  - ▶ Check it out! While you're there, also check the file `java.security`


1-38Achieving Enterprise Application Security© 2007 IBM Corporation

For more information on Permissions, see

<http://java.sun.com/javase/6/docs/technotes/guides/security/permissions.html>.


In the same directory as `java.policy` you will find a file named `java.security`. This is the master security properties file. The header of the file explains itself best, "In this file, various security properties are set for use by `java.security` classes. This is where users can statically register Cryptography Package Providers ('providers' for short). The term 'provider' refers to a package or set of packages that supply a concrete implementation of a subset of the cryptography aspects of the Java Security API. A provider may, for example, implement one or more digital signature algorithms or message digest algorithms."

## Slide 39

Presented by IBM developerWorks 

## All that JAAS

- Java Authentication and Authorization Service
- Java implementation of the Pluggable Authentication Module (PAM) framework
- Allows applications to authenticate and authorize independently from the underlying technology
  - E.g. userID/pwd, smartcard, biometric, etc.
- Before JAAS
  - Java authentication was based only on digitally signed methods and classes executing in the Java runtime
  - Could not authorize based on the principal making a request to access a resource, only on the code being used or where it came from
- After JAAS we can authenticate based on *who* is executing the code



1-39 Achieving Enterprise Application Security © 2007 IBM Corporation

The Pluggable Authentication Module framework was originally developed for the Solaris operating system by Sun. JAAS provides this capability in a Java implementation for any platform that has Java SE implemented. JAAS service providers implement technology-specific authentication modules that are loaded based on a configuration. The modules provide the same API to the application regardless of the underlying technology. JAAS is part of Java SE, the Java Platform Standard Edition.

The main JAAS classes are contained in the following packages:

- `javax.security.auth.*` - base classes and interfaces for auth\* mechanisms
- `javax.security.auth.callback.*` - classes and interfaces for defining authentication credentials of the application

- `javax.security.auth.login.*` - classes for logging in and out
- `javax.security.auth.spi.*` - interfaces that JAAS providers implement per their specific technologies

---

## Slide 40

Presented by IBM developerWorks

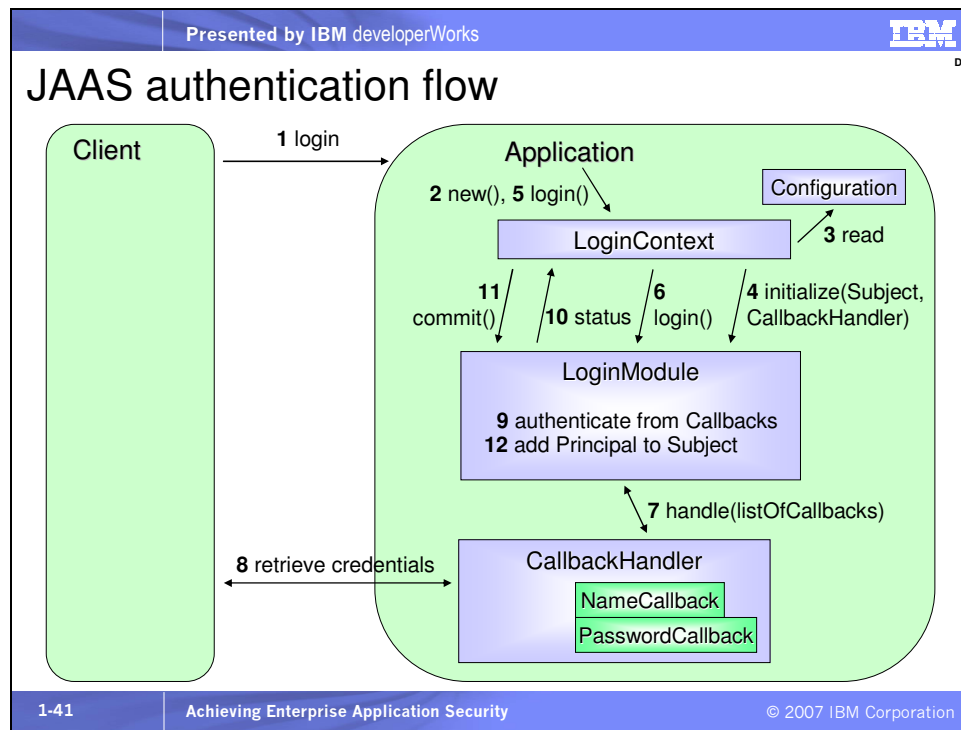
### JAAS – page 2

- Two JAAS classes that are common to both authentication and authorization are
  - The `javax.security.auth.Subject` class
    - ▶ A group of related entities around a user, a group, an app.
    - ▶ Associated with the source of a request
  - When authenticated, the Subject is populated with one or more `java.security.Principal` objects
    - ▶ Identities associated with credentials of various types
    - ▶ E.g. a person's social security number, their name, their employee number, a digital certificate, each distinguishing the user

1-40 Achieving Enterprise Application Security © 2007 IBM Corporation

A Subject represents a grouping of related information for a single entity, such as a person. Such information includes the Subject's identities, or "Principals," as well as its security-related attributes (passwords and cryptographic keys, for example), called "credentials." Besides user-identifying data, a credential can also contain information that allows a subject to perform specific activities. This type of credential might be a cryptographic key that allows the Subject to sign or encrypt data, for example. In fact, any class can be a credential, thus there is no specific credential class in JAAS.

## Slide 41



Some of the main JAAS authentication classes are

### **javax.security.auth.login.LoginContext**

- instantiated by the application
- uses LoginModule for basic methods to authenticate Subjects
- reads the Configuration and loads the appropriate LoginModules

### **javax.security.auth.login.Configuration**

- specifies LoginModules for application login
- read by LoginContext
- usually a separate file but can be from a database
- allows different LoginModules to be plugged in without requiring any modifications to the code itself

### **javax.security.spi.LoginModule**

- implemented by JAAS providers

- initialized with a Subject, a Configuration, shared state, and options
- uses the CallbackHandler to communicate with the user

### **javax.security.auth.callback.CallbackHandler**

- interacts with a user identity based on a particular authentication technology
- implemented by the application or application server, passed to the LoginContext (along with the Configuration), then forwarded to the LoginModule
- may be part of client application, in which case application may implement CallbackHandler.handle()

### **javax.security.auth.callback.Callback**

- stores authentication data
- populated by CallbackHandler for use by LoginModule
- Examples of Callback classes are
  - NameCallback -- request the user's name
  - PasswordCallback -- request the user's password
  - ChoiceCallback -- display a list of choices
  - ConfirmationCallback -- ask for YES/NO, OK/CANCEL
  - LanguageCallback -- the Locale used for localizing text
  - TextInputCallback -- retrieve generic text information
  - TextOutputCallback -- display information, warning, and error messages

Here's the flow of the interactions of the JAAS classes. This example shows the client separate from the application, suggesting

the client is a browser and the application is a Web application. But they could be one and the same.

The LoginContext class provides the layer of independence between the application and the underlying authentication mechanism. The application [2] instantiates and interacts with the LoginContext and the LoginContext loads one or more LoginModules based on the [3] Configuration the LoginContext reads. The LoginContext then [4] instantiates and initializes the LoginModule (or modules), passing it a Subject and a CallbackHandler. Next the application [5] calls LoginModule.login() and the LoginContext calls LoginModule.login [6].

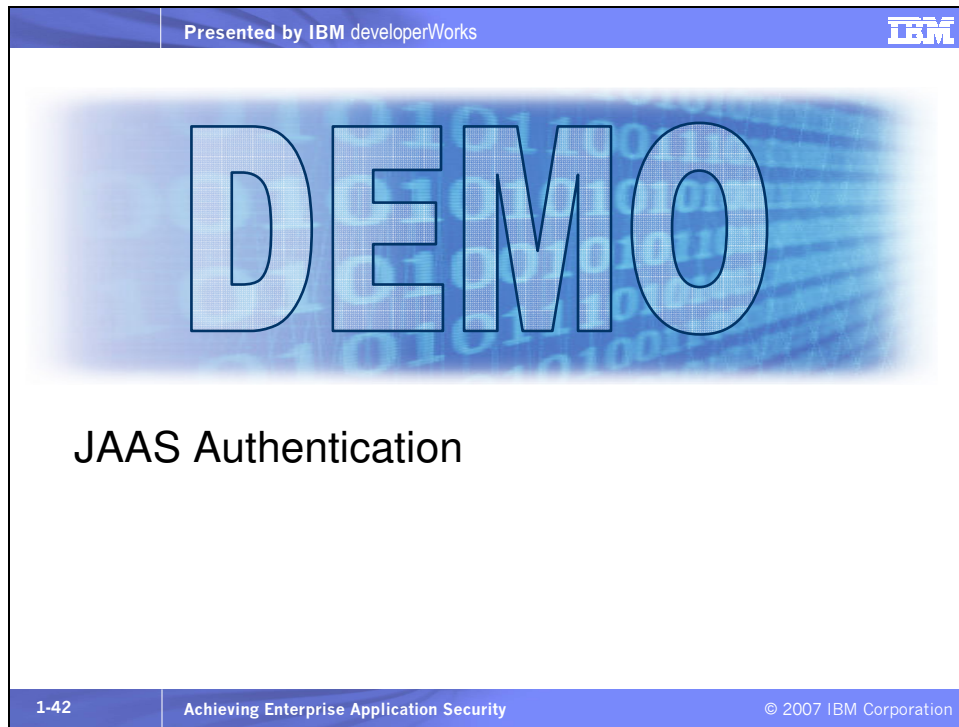
LoginModule.login() [7] passes an array of Callbacks to the CallbackHandler's handle() method. Think of a Callback as a data object. Based on its type and the type of Callback, the CallbackHandler [8] retrieves credentials from the client application, and stores the values in the Callbacks. These are [7] passed back to the LoginModule, which uses them to [9] authenticate the client. The type of authentication used will vary based on the type and vendor of the LoginModule. If authentication succeeds, LoginModule.login() creates a new Principal and [10] returns true to the LoginContext(). The LoginContext then [11] calls LoginModule.commit(), during which the LoginModule associates the Principal with the Subject [12]. The user is now authenticated.

Sometimes the exact flow shown here can vary a bit, such as when the client creates a request context containing user credentials and passes that request context to the application in advance. Then the CallbackHandler can use the passed request context without going back to the client application.

JAAS can also be used for single sign-on by sharing the security credentials or the JAAS shared state. This allows SSO access to multiple applications by sharing the state among the several LoginModules for the applications. These LoginModules must all be defined in a Configuration file.



## Slide 42



Presented by IBM developerWorks

IBM

# DEMO

## JAAS Authentication

1-42 Achieving Enterprise Application Security © 2007 IBM Corporation

Now we'll show you a short demonstration of JAAS authentication. If you'd like to try to run this code yourself, there is a Web article on which this demo is based, at <http://java.sun.com/developer/technicalArticles/Security/jaasv2/>, by Qusay H. Mahmoud. The article leads you through an introduction to JAAS and the code we use in our demo is based on the article's code.

In our version of this simple JAAS authentication demo there is no password necessary and as long as you enter "Cat" to the prompt, you will be authenticated. To run this demo you must have a path and classpath for your Java SE 1.4 or 5.0 runtime. The demo is run from the command prompt.

### *Description of files*

- MyClient.java
    - 1) Creates a login context with
      - Configuration: AnimalLogin
      - CallbackHandler: DemoCallbackHandler
    - 2) Attempts to login with the context
    - 3) Prints if authentication succeeds or fails
  - demo.conf
    - 1) Specifies the configuration named AnimalLogin and the AnimalLoginModule for it
  - AnimalLoginModule.java
    - 1) Uses the CallbackHandler to get the user input for their favorite animal type
    - 2) If answer is "Cat" then creates a Principal with name of "Zoe the Cat"
    - 3) Else returns false
  - DemoCallbackHandler.java
    - 1) Used by AnimalLoginModule, prompts the user with the question, returns answer
  - policy.txt
    - 1) Grants CreateLoginContext and modifyPrincipals permissions
- Commands for demo (and expected results)

With this first command Java security is not enabled so there should be no problem creating a LoginContext or creating a Principal while authenticating the user. Remember, you're looking for "Cat." When you answer incorrectly with "Dog" you should see

the authentication fail. Then run it again and answer correctly with "Cat."

- Command 1:

```
java -Djava.security.auth.login.config=demo.conf DemoClient
```

```
Is your favorite animal a dog or a cat? Dog
```

```
Authentication failed. Login Failure: all modules ignored
```

```
Is your favorite animal a dog or a cat? Cat
```

```
Authentication succeeded
```

Now turn on security by running the DemoClient using the –

*Djava.security.manager* command line option when you run the application again. The execution will fail because with security enabled you do not have permission to create a LoginContext or create and modify a Principal.

- Command 2:

```
java -Djava.security.manager -
```

```
Djava.security.auth.login.config=demo.conf DemoClient
```

```
LoginContext cannot be created. Access denied
```

```
(javax.security.auth.AuthPermission  
createLoginContext.AnimalLogin)
```

```
Exception in thread "main" java.lang.NullPointerException  
at DemoClient.main(DemoClient.java:23)
```

To gain permission you need to specify a policy file granting the two permissions needed to execute the code.

*Policy.txt:*

```
grant codebase "file:./*" 
```

```
{  
    permission javax.security.auth.AuthPermission  
        "createLoginContext";  
    permission javax.security.auth.AuthPermission  
        "modifyPrincipals";  
};
```


The JAAS policy file is an adaptation of the Java 2 policy file that grants permissions for code to execute based on the origin of the code. The JAAS policy file adds the "who" to the permission. This demo's policy file is at the Java 2 security level. It authorizes the code to create a LoginContext and the right to modify Principals. Now run the application again with security enabled, but this time specifying the policy file to use. This time you should have permission to execute the application, and if you answer the question correctly, authentication should succeed.

- Command 3:

```
java -Djava.security.manager -Djava.security.policy==policy.txt -  
    Djava.security.auth.login.config=demo.conf DemoClient  
Is your favorite animal a dog or a cat? Cat  
Authentication succeeded
```

The double == for the policy file are used to override any default security policy.

## Slide 43

Presented by IBM developerWorks

## JAAS authorization...

- An extension of the existing authorization schemes in Java SE policy files
  - ▶ The caller must already be authenticated with a Subject and be associated with the current access control object
  - ▶ Call the `doAs()` or `doAsPrivileged()` method of the Subject class, which in turn invokes the `run()` method containing the code to be executed as the specified subject
- Retrieve the Subject from the LoginContext after a successful login
  - ▶ Then call `Subject.doAs()` or `Subject.doAsPrivileged()` with the action to perform

```
Subject mySubject = ctx.getSubject();
PrivilegedAction myAction = new MyAction();
Subject.doAsPrivileged(mySubject, myAction, null);
...
try {
    ctx.logout(); }
catch(LoginException le) {
    System.out.println("Logout: " + le.getMessage()); }
```
- The `myAction` argument has the method executed via `doAsPrivileged()`
  - ▶ Contains all code requiring authorization

1-43Achieving Enterprise Application Security© 2007 IBM Corporation

The second "A" in "JAAS" stands for "authorization." These are the main JAAS authorization classes:

### **java.security.Policy**

- systemwide access control policy for authorization based on an authenticated Subject

### **javax.security.auth.AuthPermission**

- encapsulates permissions required for JAAS authentication and guards access to Policy, Subject, LoginContext and Configuration objects

### **javax.security.auth.PrivateCredentialsPermission**

- encapsulates permissions for accessing the private credentials of a Subject

The Subject.doAs() method associates the current Subject with the current access control context and invokes the run() method on the action, which contains all the necessary code to be executed.


Subject.doAsPrivileged() can be called instead of Subject.doAs(), with an AccessControlContext as the third parameter. This enables the Subject to associate only with the passed AccessControlContext.

A sample Action class might look something like this:

```
public class MyAction implements PrivilegedAction {
    public Object run() {
        File f = new File("Text.txt");
        if (f.exists())
            System.out.println("File exists in the current
                               working directory");
    }
}
```

The anonymous inner subclass of MyAction is where the privileged code is called. Since the inner class is run on a separate thread, it is possible to associate the current or a different security context with it.

## Slide 44

Presented by IBM developerWorks		
<h2 style="margin: 0;">JAAS authorization</h2> <ul style="list-style-type: none"> <li>▪ The corresponding required policy.txt file shows that a user named TylerD is allowed to read a file named Test.txt           <pre style="margin: 5px 0 0 20px;">grant codebase "file:./*" {     permission javax.security.auth.AuthPermission       "createLoginContext";     permission javax.security.auth.AuthPermission       "modifyPrincipals";     permission javax.security.auth.AuthPermission       "doAsPrivileged"; }; grant codebase "file:./*", <i>Principal MyPrincipal</i> "TylerD" {     permission java.io.FilePermission "MyDoc.txt", "read"; };</pre> </li> <li>▶ The portion in <i>italics</i> is JAAS-specific, adding the "who" factor</li> </ul>		
1-44	Achieving Enterprise Application Security	© 2007 IBM Corporation

For JAAS the JSE java.security.Policy API is upgraded to support Principal-based queries and Principal-based grant entries in policy files.

The policy file grant statement has this format:

```
Grant CodeBase ["URL"],
    Signedby ["signers"],
    Principal [Principal_Class] "Principal_Name",
    Principal ... {
    permission Permission_Class
        ["Target_Name"]
        [, "Permission_Actions"]
        [, signedBy "SignerName"];
};
```

JAAS is the strategic programming model for WebSphere Application Server security and is fully supported. Login Configurations can be configured through the WAS Web Administrative Console. To do so select Security Center > JAAS Configuration > Application Login Configuration.

The configuration repository is either an ASCII file or WCCM (WebSphere Common Configuration Model). There is a default login configuration for authentication to the WebSphere security runtime.

For a Client Container application, the CallbackHandler specified in the deployment descriptor is used. WLogin is the default generic Configuration and for J2C (as described later) there is a default mapping module named DefaultPrincipalMapping, used to map a user to a Principal. WLoginModuleImpl is the WebSphere default LoginModule. If authentication is successful, using defaults, it returns a WSPincipal and a WSCredential associated with the authenticated WSSubject.

WebSphere also provides a default CallbackHandler named WSCallbackHandlerImpl. It can be used programmatically to deliver data to a LoginModule without prompting the user. If user prompting is desired, WebSphere provides WSGUICallbackHandlerImpl and WSSTDINCallbackHandlerImpl. In addition to the standard Callbacks, WebSphere provides WSRealmNameCallbackImpl for passing the realm name




authentication data, and WSCredTokenCallbackImpl for passing token byte data.

Again, a very good article on JAAS that has influenced some of this section on JAAS is entitled “Java Authentication and Authorization Service (JAAS) in Java 2, Standard Edition (J2SE) 1.4,” by Qusay H. Mahmoud, available at


<http://java.sun.com/developer/technicalArticles/Security/jaasv2/>.

## Slide 45

Presented by IBM developerWorks 

### Other Java security technologies

- Java Cryptography Architecture -- JCA
  - ▶ A set of APIs spanning major security areas, including cryptography, public key infrastructure, authentication, secure communication, and access control
- Java Cryptography Extension – JCE
  - ▶ Based on JCA and provides the framework and implementations for encryption, key generation and key agreement, and Message Authentication Code (MAC) algorithms
- XML Digital Signature (Java 6, JSR 105)
  - ▶ Now incorporated in Java 6
  - ▶ Provides a standard Java API for generating and validating XML signatures



1-45 Achieving Enterprise Application Security © 2007 IBM Corporation

In this presentation we only have time to scratch the surface on this subject. The best place to learn about Java security is here: <http://java.sun.com/javase/6/docs/technotes/guides/security/>.


Also see

<http://java.sun.com/javase/6/docs/technotes/guides/security/overview/jsoverview.html>.

Java supports the idea of the security provider. The security provider provides some security-related functionality, such as the capability to create message digests, or provide cryptography, or digital signatures, and so on. Security providers are configurable and several come with the platform. The `java.security.Provider` class encapsulates the notion of a security provider in the Java platform. The `Provider` class is part of the overall Cryptography Architecture. See

<http://java.sun.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html> for more information on JCA and JCE. See <http://java.sun.com/javase/6/docs/technotes/guides/security/jsse/JSSERefGuide.html> for more about JSSE. And see <http://java.sun.com/javase/6/docs/technotes/guides/security/xmlsig/overview.html> for more information on the new XML Digital Signature API.

## Slide 46

Presented by IBM developerWorks 

## Other Java security technologies - comms

- Java Secure Socket Extension – JSSE
  - ▶ Enables secure Internet communications
  - ▶ Provides a framework and implementation for SSL & TLS protocols with Java
  - ▶ Includes functionality for data encryption, server authentication, message integrity, and optional client authentication
  - ▶ Can use for the secure passage of data between a client and a server running any application protocol
- Java Generic Security Service Application Programming Interface -- GSS-API
  - Provides uniform generic access to pluggable security services for authentication and secure messaging
- Java Simple Authentication and Security Layer (SASL)
  - ▶ Defines a protocol for authentication and optional establishment of a security layer between client and server applications
  - ▶ Defines how authentication data is to be exchanged but not the data contents
  - ▶ Used by protocols like LDAP and IMAP to enable pluggable authentication

1-46 Achieving Enterprise Application Security © 2007 IBM Corporation

Java application can securely communicate since Java provides APIs and frameworks supporting this. See <http://java.sun.com/javase/6/docs/technotes/guides/security/jsse/JSSERefGuide.html> for more about JSSE.

One of the strengths of the GSS-API is its support for Kerberos as the key authentication mechanism and in particular single sign-on support with it. Java GSS is a token-based API and it can rely on the application to handle communications. Thus an application developer can choose from a variety of transport protocols. And because it is token-based, the GSS-API supports a variety of encryption mechanisms. In Java 6 Java GSS supports additional encryption algorithms and supports SPNEGO, or Simple and Protected GSS-API Negotiation mechanism, which is a pseudo-

security mechanism that enables GSS-API peers to securely negotiate a common security mechanism to be used. For more on Java GSS see

<http://java.sun.com/javase/6/docs/technotes/guides/security/jgs/tutorials/index.html>. For more information on SPNEGO see <http://en.wikipedia.org/wiki/SPNEGO>. For more information on SASL see <http://docs.sun.com/source/817-6707/sasl.html>.

---

## Slide 47

Presented by IBM developerWorks

### Java EE security is *declarative*...

- Authentication will be discussed shortly
- Authorization is based on the granting of permissions to *security roles*
- Security roles represent *types* of users, specific to an application or a group of applications
  - ▶ E.g. teller, manager, supervisor, janitor
- A permission represents a set of one or more operations on some set of one or more resources, e.g.
  - ▶ Operation: http\_POST      Resource: http://myco.com/login.html
  - ▶ Operation: execute (implied)      Resource: getBalance(acctNumber)
- Configuring security means defining security roles and assigning them to operations on resources


1-47      Achieving Enterprise Application Security      © 2007 IBM Corporation

It's important to distinguish roles from users and groups in Java EE. Security roles are associated with and pertinent to an application. Regardless of where an application is deployed, the security roles remain the same. For example, a banking application will probably have roles such as teller, branch manager, customer, etc. regardless of at which bank the application is

installed. The same can't be said for the users of the application. At bank A there will be one set of users and groups of those users and at bank B the population will be different. By groups we mean groupings of the local population according to the local company, such as Joe's Department, Personnel, etc., independent of applications used by the company. Users and groups are usually stored in an LDAP directory.


Thus when we deploy a banking application at a bank, we take the local population of users and groups and map those to the security roles they have. This level of indirection allows an application to be installed at multiple sites and only requires the mapping of users and groups to security roles to configure declarative security for the local population.

## Slide 48

Presented by IBM developerWorks 

## Java EE security is *declarative*

- This configuration is stored in
  - ▶ *security-constraints* within Web.xml for Web resources
  - ▶ *method-permissions* within the EJB module's deployment descriptor for ejb methods
- Later, at deployment time we map users and groups to security roles
  - ▶ Because it may involve many mappings for each individual user, it's more common to first gather users into a smaller set of groups, and then map those groups to security roles



1-48 Achieving Enterprise Application Security © 2007 IBM Corporation

Here's what a method-permission and a security-constraint look like. Any user that has the employee role will be able to access these resources.

### In EJB JAR

```
<method-permission>
```

```
  <role-name>employee</role-name>
```

```
  <method>
```

```
    <ejb-name>AccountBean</ejb-name>
```

```
    <method-name>getBalance</method-name>
```

```
  </method>
```

```
</method-permission>
```

### In WAR

```
<web-app>
```

```
<display-name>Banker2005</display-name>
<security-role>
  <role-name>employee</role-name>
</security-role>
<security-constraint>
  <web-resource-collection>
    <web-resource-name>AccountInfo
    </web-resource-name>
    <url-pattern>/account/*</url-pattern>
    <http-method>GET</http-method>
    <http-method>POST</http-method>
  </web-resource-collection>
  <auth-constraint>
    <role-name>employee</role-name>
  </auth-constraint>
</security-constraint>
</web-app>
```

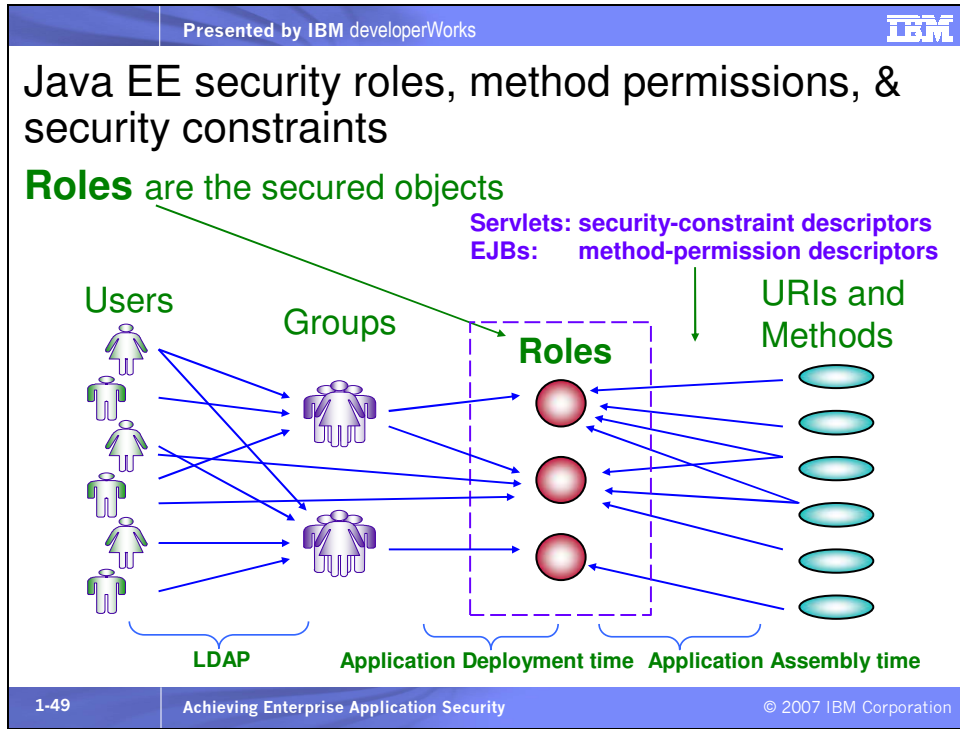
“Role references” provide further flexibility. Sometimes applications are developed by several separate teams. Each team creates some of the modules of the application and finally an application assembler combines them to create the application. Some of the development teams may have used some of the Java EE security calls within their code. There could be a problem with this if one team has called a role “supervisor,” while another team, intending the same role, has named the role “boss” or

“department\_head.” Each role name is intended to represent the same type of user but the role names are literally different. In this case, the application assembler can create a role reference that maps department\_head, boss, and supervisor to “manager.” Then at deployment time users and groups are mapped to the manager role. Each team’s code will still work even though the role name literal happens to be different.

Run-as role maps allow even more flexibility. We can optionally delegate execution of an EJB method to a specified userID/password identity. Normally, when EJB1 calls EJB2, the identity in the current security context is that of the initial caller of EJB1. This is called “Run-As Caller.” But we can specify a run-as role for EJB1 and at deployment time associate an identity with the role. Then, when EJB1 calls EJB2, the new identity will be checked for authorization to access EJB2.method(). We can also specify that an EJB should run with the identity of the application server. This typically provides the highest level of permission.




Slide 49



Sometimes it's easier to see these relationships in a picture.

## Slide 50

Presented by IBM developerWorks 

## Java EE security is *programmatic*

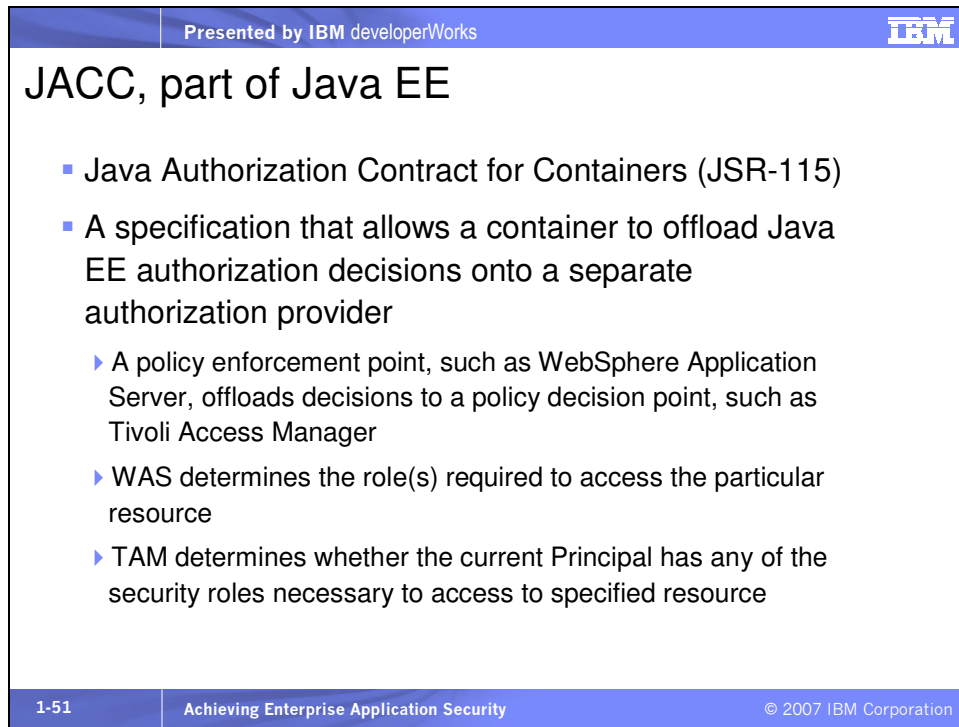
- Within an application we can obtain and use security information for further security control
- In a Web application the methods available are
  - ▶ **getRemoteUser ()**
    - Returns the user name with which the client authenticated
  - ▶ **isUserInRole (String roleName)**
    - Determines whether the user is in a particular role
  - ▶ **getUserPrincipal ()**
    - Returns a `java.security.Principal` object
- In an enterprise application the methods available are
  - ▶ **boolean isCallerInRole (String roleName)**
    - Make additional authorization checks on the current user
  - ▶ **java.security.Principal getCallerPrincipal ()**
    - Obtain the name of the current caller from `principal.getName ()`

1-50 Achieving Enterprise Application Security © 2007 IBM Corporation

The `isCallerInRole (String roleName)` method returns true if the user is in the specified role, and false if it is not. The role name specified in the method is really a security role reference, not a role. If the security role reference is not defined for the EJB, the method returns null.

The Web application methods are implemented on the `HttpServletRequest`. The enterprise information comes from the current caller's `javax.ejb.EJBContext` security context.

## Slide 51



Presented by IBM developerWorks

## JACC, part of Java EE

- Java Authorization Contract for Containers (JSR-115)
- A specification that allows a container to offload Java EE authorization decisions onto a separate authorization provider
  - ▶ A policy enforcement point, such as WebSphere Application Server, offloads decisions to a policy decision point, such as Tivoli Access Manager
  - ▶ WAS determines the role(s) required to access the particular resource
  - ▶ TAM determines whether the current Principal has any of the security roles necessary to access to specified resource

1-51 Achieving Enterprise Application Security © 2007 IBM Corporation

From the JACC 1.0 specification: "This specification defines new `java.security.Permission` classes to satisfy the Java EE authorization model. The specification defines the binding of container access decisions to operations on instances of these permission classes. The specification defines the semantics of policy providers that employ the new permission classes to address the authorization requirements of Java EE, including the following:

- The definition of roles as named collections of permissions
- The granting to principals of permissions corresponding to roles
- The determination of whether a principal has been granted the permissions of a role (e.g. `isCallerInRole()`)

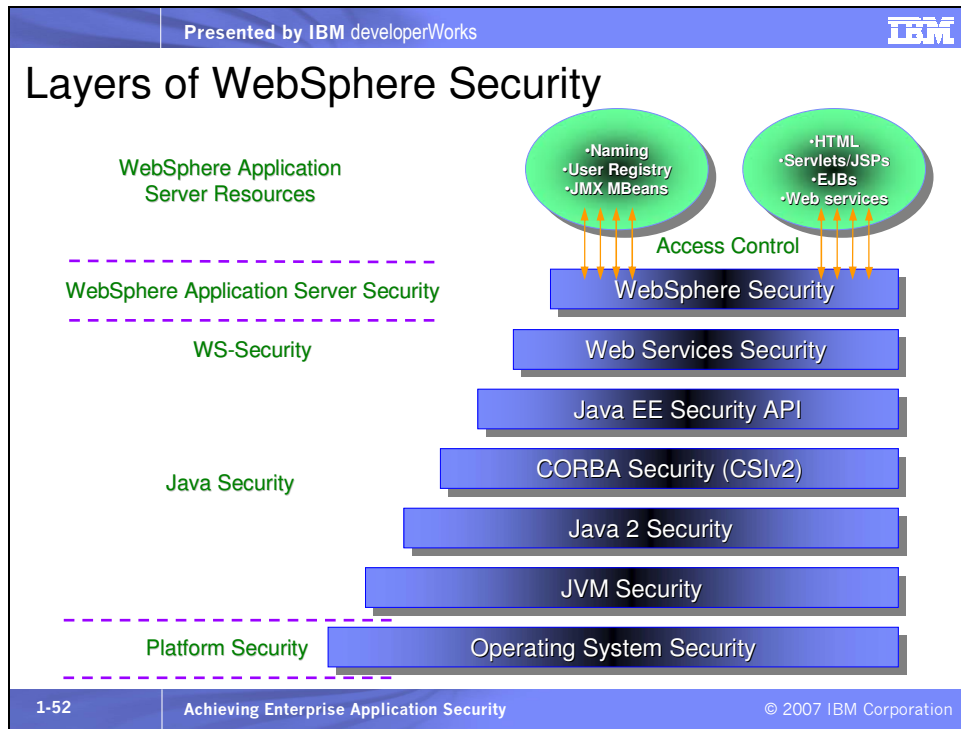
- The definition of identifier to role mappings that bind application embedded identifiers to application scoped role names.

The specification defines the installation and configuration of authorization providers for use by containers. The specification defines the interfaces that a provider must make available to allow container deployment tools to create and manage permission collections corresponding to roles."

A permission represents a set of activities (a set of one or more operations on some set of one or more resources) that is the target of an authorization decision. A Principal is defined as both a security attribute acquired as a result of authentication by entities that perform activities, and as an entity that performs activities. In this context a Role is a named set of permissions that may be granted to principals.

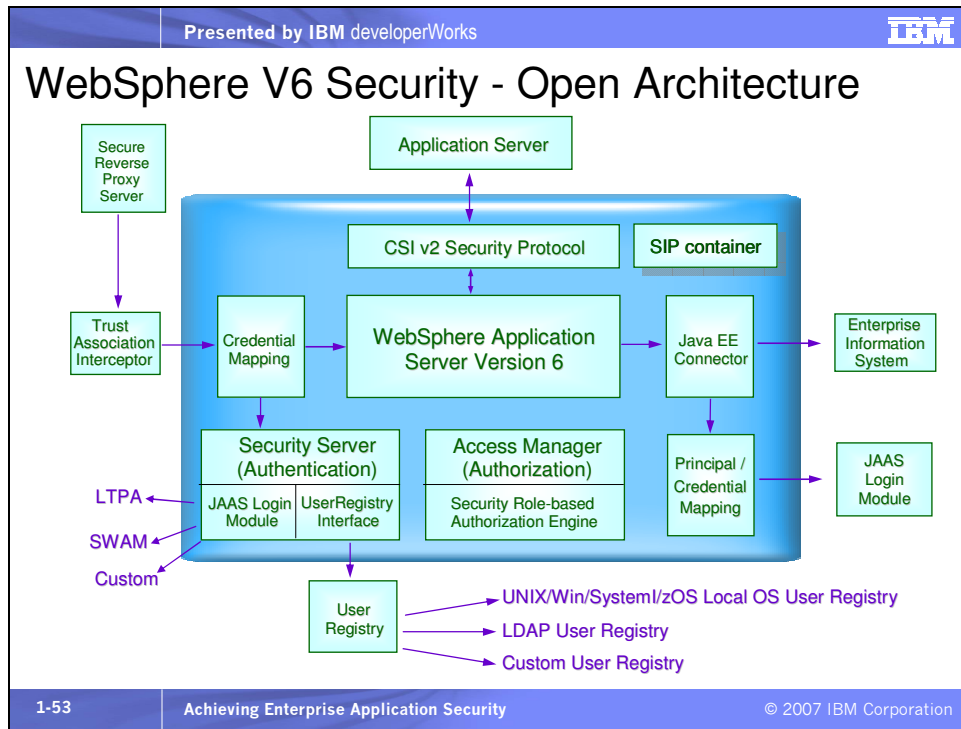
JACC classes are in the `javax.security.jacc` package.

Slide 52



WebSphere Application Server V6.1 supports all of Java security, Java EE security, Web service security, and SOA security.

Slide 53



The WAS V6 security architecture, based on open standards such as Java EE, provides flexibility in the mixing or addition of security components.

This table shows the various authentication options provided by WebSphere Application Server. The Java client refers to a standalone Java client application. Servlet 2.2+ refers to *j\_security\_check* as the **<form>** action.

<u>Challenge Type</u>	<u>Authentication Mechanism</u>	<u>User Registry</u>	<u>Client Support</u>
None	None	None	Web, Java
Custom	Custom form	Application	Web, Java
Form	Servlet 2.2+	LDAP, Custom	Web
Basic	LTPA	LDAP, Custom	Web, Java
	SWAM	LDAP, LocalOS	Web, Java
	Access Manager	LDAP	Web, Java
Digest	Available for SIP container and LDAP server, and Web services		

Certificate

LTPA

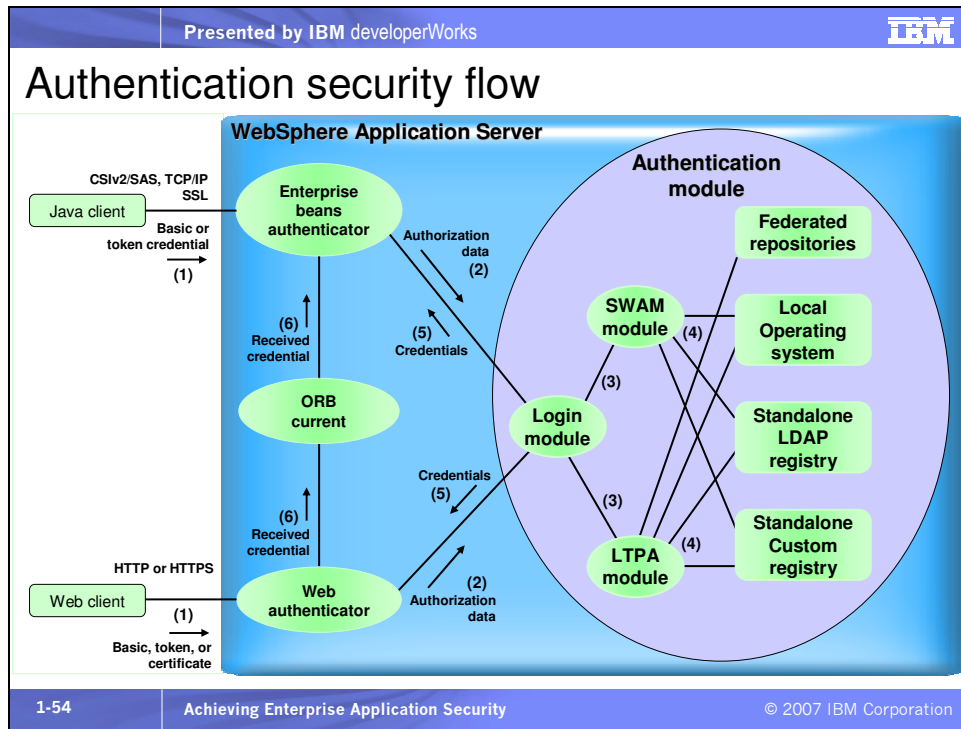
LDAP, Custom Web, Java

LTPA, or Lightweight Third Party Authentication, is a token-based authentication technology used throughout IBM middleware. In addition to LDAP, Local OS, and Custom user registries, WebSphere supports a federated repository. Recently the Simple WebSphere Authentication Mechanism (SWAM) authentication mechanism has been deprecated.

Authorization providers include

- Internal
- External – based on JACC

Slide 54



Authentication is the responsibility of the application server, but it can also be offloaded to an external security server, such as Tivoli Access Manager WebSEAL.

Access to anything running in the EJB container is via RMI/IIOP. WAS provides a security service that is compliant with Common Security Interoperability version 2, the CSIV2 protocol. There is another service called Security Authentication Service (SAS) that has been used in previous versions before CSIV2. SAS (IBM) is deprecated and it is only kept to provide interoperability with WAS versions older than V5.0. It does not appear in the administrative console unless a version V6.0 or older server is federated into the cell. For CSIV2 WAS provides

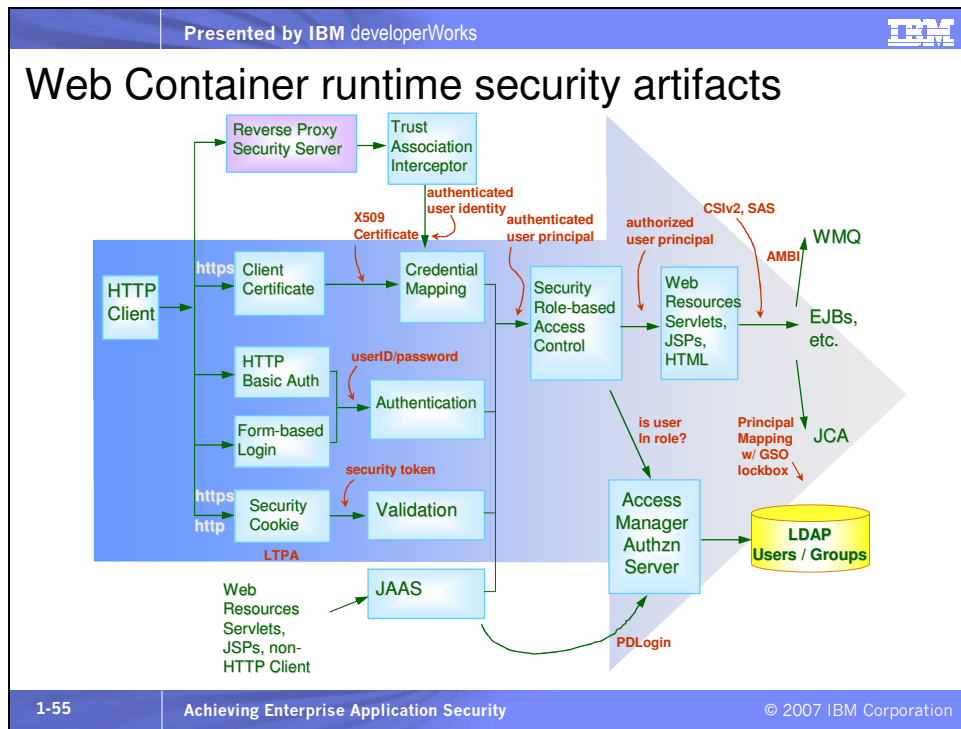
- *Authentication* capabilities on the CORBA level



- *Transport channel encryption*; WebSphere provides IOP transport channel protection using the SSL protocol.

This picture illustrates the four registries or repositories WAS can use for user and group information. When a user registry such as LDAP is not configured in WAS, local OS is used by default.

Slide 55



WAS provides container-based, provider-based, and JAAS-based authentication.

Container-based authentication occurs when a user or other application, such as any HTTP client, accesses an application running in WAS, and WAS is responsible for authentication. In this case (as shown) HTTP authentication can be

- Basic, in which the userID and password are transported in the header of an HTTP request

- Forms-based, where an HTML form requests a username and password and the action of the Submit button is to access a special WAS servlet called *j\_security\_check*, which then performs authentication based on the corresponding passed *j\_username* and *j\_password* parameters,
- Digest-based, where a userID and digest of the password are sent to WAS and WAS compares the pair to the expected pair stored in a registry, such as an LDAP directory, and
- Client certificate-based, where a client-side certificate is sent to the server in the request and WAS obtains the user identity from within the certificate.
- Token-based, such as an LTPA token that is sent with the request

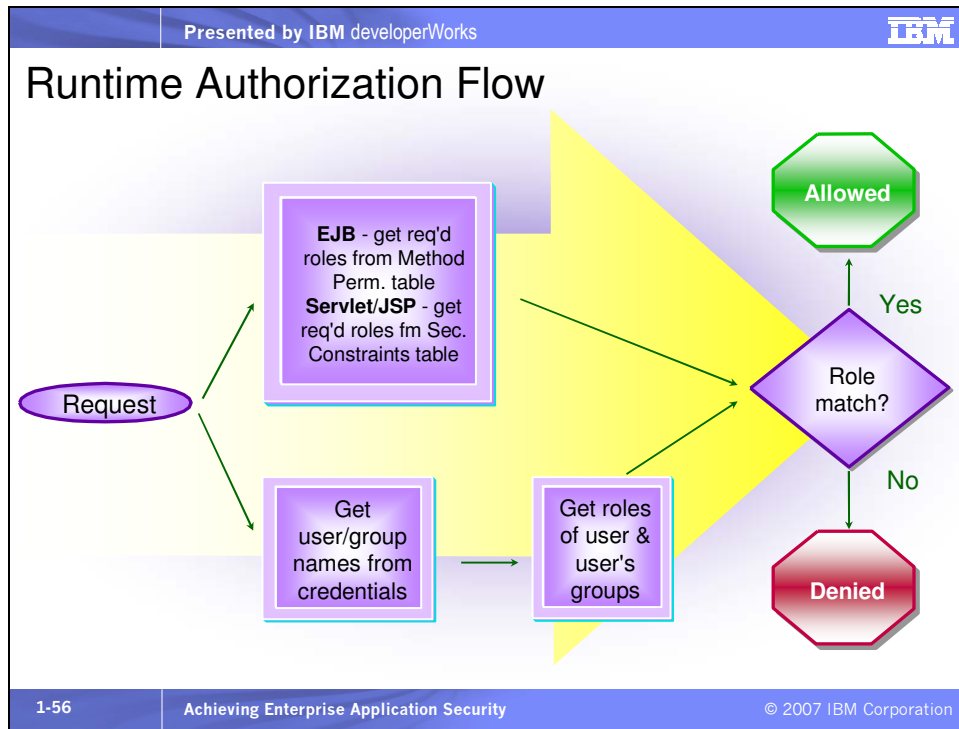
Authentication provider-based authentication involves a separate provider responsible for the authentication. An example of this type of authentication is the user of Tivoli Access Manager WebSEAL as a reverse proxy security server in front of WAS. Using a Trust Association Interceptor (TAI), WAS can trust that the request has been forwarded by WebSEAL and that WebSEAL has authenticated the user.

JAAS-based authentication is the third type of authentication supported. In this case, the application itself, via a JAAS login module, authenticates the user. The big advantage of JAAS is that it separates the application and application server from the

underlying authentication mechanism, thereby providing the most flexibility.

On this slide we see the flexibility of authentication within the Web container of WAS and the security artifacts that are managed within WAS. What's out of the ordinary is the unique authentication capability we get by the combination of WAS and TAM. The TAM custom JCA principal mapping module, discussed later, can select a userID/password set from the GSO lockbox based on the type of JCA connection, as shown here. And of course EJBs can be accessed directly from a remote client using the RMI/IIOP protocol, which is capable of carrying a security context from which WAS will authenticate and/or authorize a user for access to specific methods on the target EJB.

## Slide 56



The authorization flow consists of obtaining all the roles of the user and comparing those to the roles required to access the resource, whether that be a Web resource like an HTML page, a servlet or JSP, or an enterprise resource like an EJB. Roles can be associated with a user directly or by association with a group to which the user belongs.

## What was added with WAS V6.0?

- Things got easier
  - ▶ Enhanced TAI (since V5.1.1)
  - ▶ AMPrincipalMapper still in TAM JAR and included with WAS (since V5.x)
  - ▶ TAM Authorization Server included with WAS (since V5.x)
  - ▶ TAM Policy Server included with WAS V6
    - Supports GSO
    - Requires pdadmin command prompt TAM admin (included)
      - No TAM Web Portal Manager browser-based admin console

The inclusion of the Tivoli Access Manager Policy Server with WAS V6 allows the creation of Global Sign-on (GSO) resources and associated usernames and passwords. The TAM command prompt "pdadmin" application must be used to administer this because the browser-based TAM Web Portal Manager application is not included with WAS V6.

The Trust Association Interceptor (TAI) has additional functionality in WAS V5.1.1 and beyond that enables it to return a Subject with any information you want. That Subject will be passed into WEB\_INBOUND as the initial information and will stay in there throughout the login. Thus, if a custom TAI were written, it is possible that the TAI could populate the Subject with the appropriate userID/password for a JCA call to come. Then another custom principal mapping module would be able to use that set of credentials.

## Security enhancements in WAS V6.1



- Installation as non-root/non-Administrator user
  - ▶ IHS still requires root install for SSL
- Default user registry out-of-the-box
- Administrative Security enabled out of the box (no config)
- Highly simplified communications security
  - ▶ Centralized SSL management, integrated certificate /key mgmt.
- Significantly simpler post-install security hardening (LTPA preconfigured)
- Single sign-on for Windows desktops (SPNEGO)
- Integrated capability for federating LDAP repositories (basic identity mngmt)
- Simplified security configuration user interface
  - ▶ Including addition of a configuration wizard
- Security configuration report
  - ▶ Shows all of your security settings in one consolidated report
- Simplified Certificate/Key Management
- SIP security

You can apply digest authentication and use a Trust Association Interceptor (TAI) for a SIP application by applying Lightweight Directory Access Protocol (LDAP) security to the application. Tivoli Directory Server must be installed and LTPA must be enabled.

*Mini glossary:*

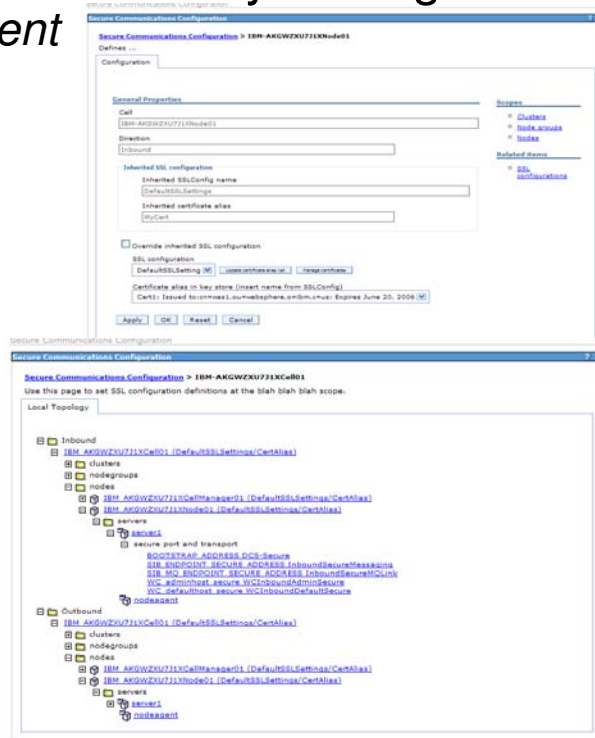
**SIP** – Session Initiation Protocol, an OSI application layer protocol for managing peer-to-peer sessions

**SPNEGO** – Simple Protected Negotiation, single sign-on used in Windows

# WAS V6.1 simplified certificate/key management

## *Major usability improvement*

- Integration of key management tools into the console
- Easier to understand and use the configuration of SSL attributes
- Management of Web server and plug-in certificates built into the console
- Use of the TrustManager to automatically trust hosts or signers
- Easier to refresh an expiring certificate



Certificate management is among the many improvements to security in WAS V6.1. Ikeyman is a standalone application that was shipped with several IBM products for some time. The integration of ikeyman functionality in base WAS administration allows the management of IBM HTTP Server Plug-in certificates and application server certificates in one place. There are also major usability improvements.

## Slide 57

Presented by IBM developerWorks

## J2EE Connector Architecture security

- Designed to extend the end-to-end security model for Java EE-based apps to include integration with EISs
- An app server and EIS collaborate to ensure the proper authentication of a resource principal, establishing a connection to an underlying EIS
- Two authentication mechanisms are identified
  - ▶ Basic Password: basic userID/password-based authentication mechanism that is specific to an EIS
  - ▶ Kerb V5: Kerberos version 5-based authentication mechanism
- The TAM GSO Principal Mapper JAAS login module for J2C (JCA) resources can be used to select a resource-specific userID and password for the current Subject

1-57© 2007 IBM Corporation

Achieving Enterprise Application Security

From the WebSphere Application Server V6.1 Security Handbook:  
“Applications define whether to use application-managed sign-on or container-managed sign-on in the resource-ref elements in the deployment descriptor. Each resource-ref element describes a single connection factory reference binding. The res-auth element in a resource-ref element, whose value is either Application or Container, indicates whether the enterprise bean code perform sign-on or WebSphere Application Server can sign-on to the resource manager using the principal mapping configuration. The resource-ref element is typically defined at application assembly time with an assembly tool. The resource-ref can also be defined, or redefined, at deployment time.”



Container-managed sign-on removes the work from an application having to manage its own secure J2C (JCA)\* connection. But the userID and password for container-managed J2C security are specified on the connection factory as a JAAS authentication alias. This means that with the default mapping module, every connection created by the factory will send the same userID/password combination along with the J2C request, regardless of the currently authenticated Subject making the request. To overcome this limitation, the TAM GSO Principal Mapper JAAS login module for J2C resources can be configured for the connection factory, rather than the default mapping module. Based on the currently authenticated Subject and the resource to be accessed, the TAM principal mapping module will query the TAM GSO lockbox to obtain a userID/password pair configured for the resource and the Subject. Then this credential set will be passed with the J2C request. This has great value when a user is known to an EIS but in a different form or with different credentials. Thus we achieve single sign-on across a J2C connection.

See the handout after the section on Tivoli Access Manager for a visual description of how the TAM GSO Principal Mapper JAAS login module works, and what's inside the TAM GSO (Global Sign-On) lockbox.

Rational Application Developer can configure container-managed authentication for a JCA connector, including the TAM principal mapping module.

\*Instead of JCA, J2C is the acronym used here by IBM for J2EE Connector Architecture to avoid confusion with the use of JCA, which represents Java Cryptography Architecture.

---

## Slide 58

Presented by IBM developerWorks

### What's wrong with this servlet code?

```
public void doLogin(HttpServletRequest request,
    HttpServletResponse response) throws ... {
    String name    = req.getParameter("lastname");
    String passwd  = req.getParameter("password");
    bAuthenticated = authenticate(name, passwd);
}

boolean authenticate(String username, String password) {
    Statement stmt = conn.createStatement();
    ResultSet rs   = stmt.executeQuery("SELECT Lname FROM Users
        WHERE Lname = '" + username + "' AND
        Password = '" + password + "'");

    rs.next();
    String user = rs.getString(1);
    if (user == "") return(false) else return(true);
}
```

1-58Achieving Enterprise Application Security© 2007 IBM Corporation

This code is an example of a servlet that performs authentication. The servlet receives a user's last name and a password as parameters and calls the `authenticate()` method. This method builds an SQL statement and executes it. If the result contains a name, then the user is assumed to be in the database with the same last name and password and is therefore authenticated. Let's


say *lastname* and *password* are passed in the HTTP request for authentication and used directly.

*Mini glossary:*

**Servlet** – a type of Java class that runs in an application server. The servlet has a specific API that is called by the application server when an HTTP request is sent to the application. So the servlet is often the entry point for HTTP requests coming from a browser or other application.

---

## Slide 59

Presented by IBM developerWorks 

### Without parameter validation this is what's wrong

- For the last name and password suppose the user types in  
`Last name: ' OR ''='`  
`Password: ' OR ''='`
- Now the query will be  

```
SELECT Lname FROM Users WHERE Lname = '' OR ''=''  
                                AND Password = '' OR ''=''
```
- The query will compare '' to '' and this will always be true
- The query will return the first Lname in the table
- The user will be authenticated as the first user in the table
- This is an *SQL injection attack*

1-59 Achieving Enterprise Application Security © 2007 IBM Corporation

Suppose instead of typing their real last name and password, the user typed what's shown. The user will be authenticated not only incorrectly, but as a different user, whichever user happens to be at the top of the table.

This is just one example of an SQL injection attack. There are many types of these and it is very important to validate strings used in database operations before using those strings. Make sure you validate all parameters in an HttpServletRequest if those parameters can in any way be vulnerable to what a user enters in a Web form. This holds true for any application, Web or rich client or mobile or otherwise.

---

## Slide 60

Presented by IBM developerWorks

### Some rules to keep in mind

- ALWAYS validate user input both in the browser and in the servlet
- Never use directly passed parameters in a SQL query without validation
- Use prepared statements instead of simple SELECTs
- Use JAAS authentication or let the application server handle authentication – don't authenticate in your code

1-60 Achieving Enterprise Application Security © 2007 IBM Corporation

If your code needs to be secure, never use request parameters directly without validating them first. This is especially true when performing database operations based on parameters retrieved from an HTTP request. The parameters could contain SQL code that performs some illegal database operation. This is called a SQL injection attack. Using prepared statements makes the code

cleaner, reusable, faster, and easier to see whether you are secure. Above all, use JAAS authentication or let the application server handle authentication with built in mechanisms.

---

## Slide 61

Presented by IBM developerWorks

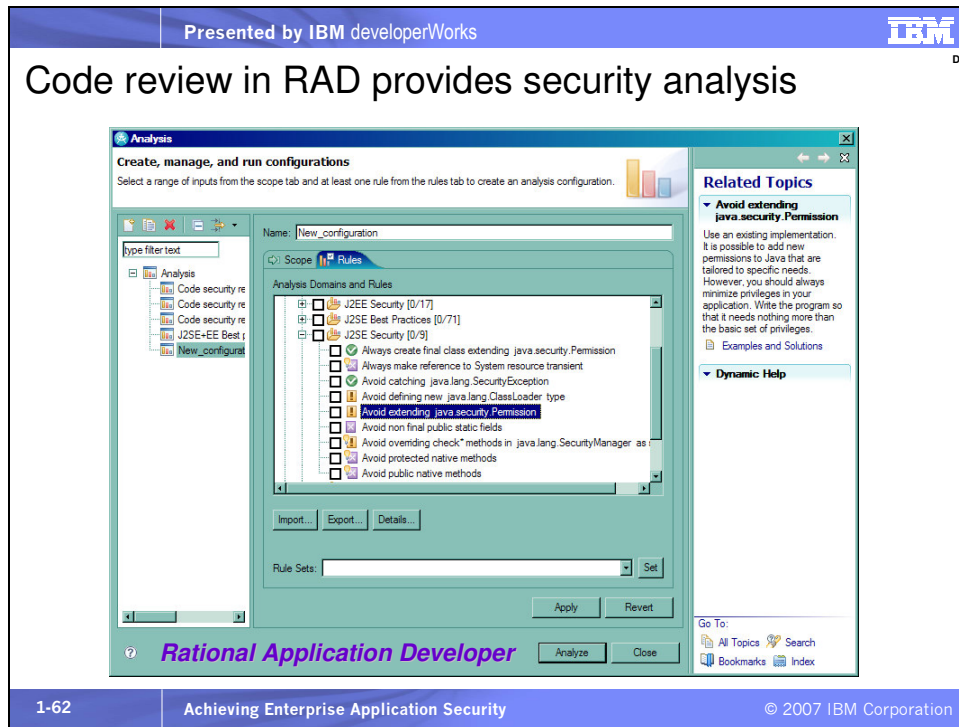
### Rational Application Developer & Java security

- Automated *code review* against security rules
  - ▶ Performs *static analysis* on code
  - ▶ Finds violations of security rules in our Java code
- Rule sets include
  - ▶ Design Principles
  - ▶ Globalization
  - ▶ J2EE Best Practices
  - ▶ **J2EE Security**
  - ▶ J2SE Best Practices
  - ▶ **J2SE Security**
  - ▶ Naming
  - ▶ Performance
  - ▶ Private API

1-61 Achieving Enterprise Application Security © 2007 IBM Corporation

Besides the code review rule sets, there are Architectural Discovery and UML Model analysis rules that you can configure.

## Slide 62

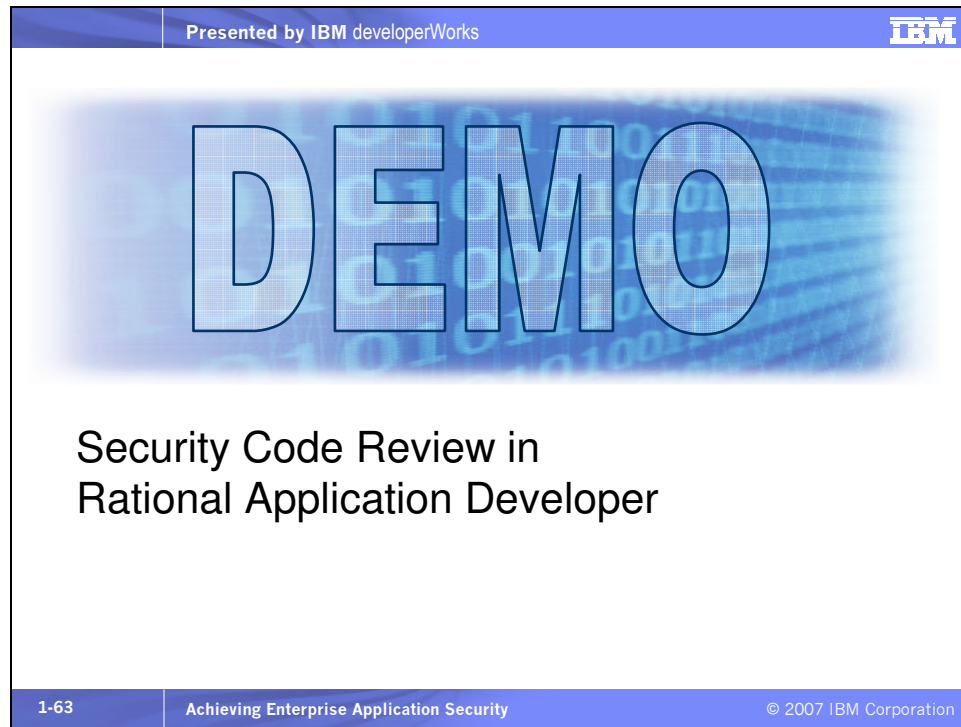


Here we see some of the Java SE rules we can select. There is extensive help available for each rule. The Help portion of the dialog expands out from the right, as shown.

*Mini glossary:*

**J2SE** – Java 2 Standard Edition, the standard set of Java classes and APIs (application programming interfaces). J2SE was renamed Java Platform, Standard Edition, or **Java SE**.

## Slide 63



Presented by IBM developerWorks

# DEMO

## Security Code Review in Rational Application Developer

1-63 Achieving Enterprise Application Security © 2007 IBM Corporation

Let's take a look at how Rational Application Developer (and Rational Software Architect) can help us find security flaws in our code. This demo assumes experience with one of these products, and we'll use RSA V7 for now.

We'll make the instructions generic enough that you can use any Java code for this demo. In the Java perspective, pick a project, package, or class on which you'd like to run a Code Review. Right-click the project (or package/class) and select Analysis > Analysis.... Click on New\_configuration under Analysis on the left, and at the top right enter a name for your new Analysis configuration, such as "security code review 1." You can set the Scope radio button to analyze the entire workspace or a working

set. Click on Analyze a resource working set and select the projects you'd like to analyze.

Now select the Rules tab towards the top and here is where you pick which sets of rules you want to run. Expand Code Review for Java and check J2EE Security (if your project is a Web or enterprise Java project) and J2SE security. Then click Analyze at the bottom. The analysis will start and your configuration will be saved for reuse.


If you have any rule violations in your projects the Analysis Results view will open. You can expand any results down to the file and line where the problem occurs, double-click on the file, and sometimes even fix the problem automatically with QuickFix in the editor.

Look for a detailed on demand demo video of this functionality at <http://www.devx.com/OnDemandDemos/Door/30314> or <http://www-128.ibm.com/developerworks/offers/lp/demos/>.



## Web services, SOA and AJAX security

### Slide 64

Presented by IBM developerWorks 


### Agenda

- Application security basics and core technologies
  - ▶ Security goals for e-business applications
  - ▶ Confidentiality, Integrity, and Non-repudiation
  - ▶ Authentication and Authorization/Access Control
  - ▶ Privacy
- Java, Java EE and WebSphere security
- **Web services and SOA security**
- Security products from IBM
- Attacks and malicious code
- Resources
- Q&A

1-64 Achieving Enterprise Application Security © 2007 IBM Corporation

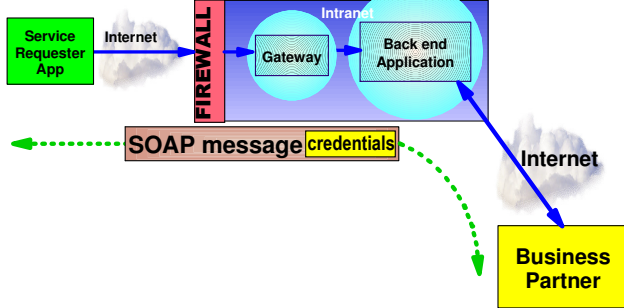
Now let's talk about security for Web services and Service-Oriented Architecture.

## Slide 65

Presented by IBM developerWorks 

## Message-level Security for SOAP messages

- By placing security information **in the message itself**, we can overcome the point-to-point limitations of SSL and achieve an end-to-end solution
- To ensure interoperability, we need a standard to define new security elements to extend the SOAP message and allow the use of both proven and emerging security technologies



1-65 Achieving Enterprise Application Security © 2007 IBM Corporation

We can use SSL to provide confidentiality and some integrity to messages we send over the Internet. The problem with SSL, though, is that it is point-to-point. If we want to send a message that may take several hops to reach its destination, each pair of hops will need to establish their own SSL session. Thus any security information about the original sender will be lost before reaching that destination using SSL.


*Mini glossary:*

**SOAP** – Simple Object Access Protocol, an XML vocabulary that specifies the format of XML messages sent between a service requestor and a service provider. The requestor is the client and the provider is the server. In the process of handling a request, a

service can call another service, thus making the service a client or requestor as well.

---

## Slide 66

Presented by IBM developerWorks 

### The Web services security model

- Capabilities-Based
  - ▶ WS-Security provides mechanisms to associate security tokens with messages
  - ▶ Identity mapping and user attributes support ACL-based models
- Policy-Driven
  - ▶ WS-Policy enables services to describe required claims (and more - described next)
- Decentralized
  - ▶ WS-Trust enables anyone to become a trust broker; e.g. provide identity or group membership
  - ▶ End-to-end security, don't assume point-to-point, don't assume back-channels

1-66 Achieving Enterprise Application Security © 2007 IBM Corporation

## Slide 67

Presented by IBM developerWorks

## WS-Security

- A foundational set of SOAP message extensions for building secure Web services
  - ▶ Defines elements to be used in the SOAP envelope for message-level security
- Defines how to identify the creator of the message
  - ▶ Carries multiple credential types
- Provides message integrity
  - ▶ Integrity of all or parts of a message
  - ▶ Builds on XML-Signature
  - ▶ Supports multiple and overlapping signatures
- Provides message confidentiality
  - ▶ Confidentiality of all or part of a message
  - ▶ Builds on XML Encryption

1-67 Achieving Enterprise Application Security © 2007 IBM Corporation

WS-Security defines a set of enhancements to the SOAP specification of messaging to enable protection of the message through authentication, confidentiality, and assurance of integrity.

The specification also describes

- How to encode binary security tokens,
- A framework for XML-based tokens, and
- How to include opaque encrypted keys

It also includes extensibility mechanisms that can be used to further describe the characteristics of the tokens that are included with a message. WS-Security was first defined by IBM and Microsoft and is now standardized at version 1.1, managed by OASIS ([www.oasis-open.org](http://www.oasis-open.org)).

## Slide 68

Presented by IBM developerWorks

## WS-Security: SOAP Message Security

- Defines the use of formerly incompatible proven and emerging security technologies:
  - ▶ Kerberos, PKI, HTTPS, IPSEC, XrML
  - ▶ XML Signature, XML Encryption, XKMS from W3C
  - ▶ SAML, XACML, SwA from OASIS
- Original WS-Security specification proposal and roadmap:
  - ▶ <http://www.ibm.com/developerworks/webservices/library/ws-secure/>
  - ▶ <http://www.ibm.com/developerworks/webservices/library/ws-secmap/>
- OASIS Web Services Security: SOAP Message Security specification
  - ▶ [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wss](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss)


1-68 Achieving Enterprise Application Security © 2007 IBM Corporation

Specifically, the 1.1 standard adds the following profiles:

- WS-Security Core Specification 1.1
- Username Token Profile 1.1
- X.509 Token Profile 1.1
- SAML Token Profile 1.1
- Kerberos Token Profile 1.1
- Rights Expression Language (REL) Token Profile 1.1
- SOAP with Attachments (SWA) Profile 1.1

See [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wss](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss) for details.

## Slide 69

Presented by IBM developerWorks 

## Security information is in the SOAP header


```
<soapenv:Envelope . . . >
  <soapenv:Header>
    <wsse:Security />
  </soapenv:Header>
  <soapenv:Body>
    . . .
  </soapenv:Body>
</soapenv:Envelope>
```

1-69 Achieving Enterprise Application Security © 2007 IBM Corporation

Security elements may include information about XML encryption, digital signatures, and security tokens, as well as a number of other types of security information. There are several types of security tokens:

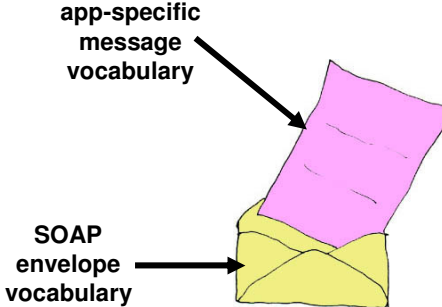
- Username/password
- Encryption details
- XML Signature
- x.509 certificate
- Kerberos ticket
- XrML
- SAML

Slide 70

Presented by IBM developerWorks 


## SOAP message structure...

- The SOAP specification defines the "envelope" vocabulary
  - ▶ The "envelope" wraps the message itself
  - ▶ The message is a different vocabulary
  - ▶ A namespace prefix is used to distinguish the two parts



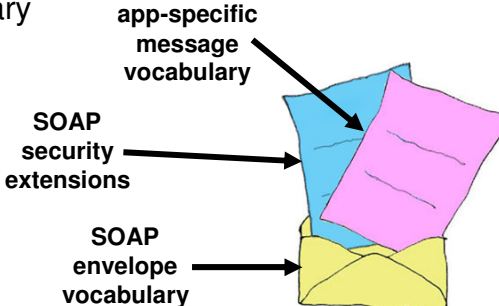
1-70 Achieving Enterprise Application Security © 2007 IBM Corporation

Slide 71

Presented by IBM developerWorks 


## SOAP message structure

- The SOAP specification defines the "envelope" vocabulary
  - ▶ The "envelope" wraps the message itself
  - ▶ The message is a different vocabulary
  - ▶ A namespace prefix is used to distinguish the two parts
- WS-Security defines extensions to the "envelope" vocabulary
  - ▶ Container for security tokens
    - Username
    - x.509 certificate
    - Kerberos ticket
    - XrML
    - XML Signature
    - SAML
  - ▶ Encryption details



1-71 Achieving Enterprise Application Security © 2007 IBM Corporation

Slide 72

Presented by IBM developerWorks 

## The WS-Security **Security** element

The WS-Security specification defines a vocabulary that can be used inside the SOAP envelope. `<wsse:Security>` is the “container” for security-related information.

```

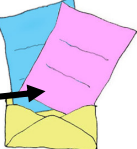
<S:Envelope
  xmlns:S="http://www.w3.org/2002/06/soap-envelope">
  <S:Header>
    <wsse:Security
      xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/07/secext">
      Security information
    </wsse:Security>
  </S:Header>
  <S:Body> App-specific content </S:Body>
</S:Envelope>
    
```

Define and use WS-Security namespace

Security information

App-specific content

SOAP Envelope




1-72 Achieving Enterprise Application Security © 2007 IBM Corporation

**wsse** stands for “Web Services Security Extensions.”



## Slide 73

Presented by IBM developerWorks 


## Security Tokens for the security element

- A *Security Token* is a collection of “claims”
  - ▶ A claim is a declaration made by some entity, such as name, identity, key, group, privilege, capability, etc
  - ▶ “username” is an example of an unsigned security token
- A *Signed Security Token* is one that is cryptographically signed by a specific authority
  - ▶ An X.509 certificate is a signed security token
  - ▶ A Kerberos ticket is also a signed security token
- An *XML Security Token* is one that is defined with a separate XML schema, rather than simple or encrypted text
  - ▶ SAML and XrML are examples
  - ▶ Can be included directly in <wsse:Security> container

1-73 Achieving Enterprise Application Security © 2007 IBM Corporation

We'll discuss SAML and define XrML in a few minutes.

## Slide 74

Presented by IBM developerWorks 

## The WS-Security UsernameToken element


- This element can be used to provide a user name within a `<wsse:Security>` element, for *Basic Authentication*

```
<S:Envelope
  xmlns:S="http://www.w3.org/2002/06/soap-envelope">
  <S:Header>
    <wsse:Security
      xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/07/secext">
      <wsse:UsernameToken wsu:ID="myToken">
        <wsse:Username>tiberphaid</wsse:Username>
        <wsse:Password>passsw0rd</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </S:Header>
  <S:Body>
    App-specific content
  </S:Body>
</S:Envelope>
```

1-74 Achieving Enterprise Application Security © 2007 IBM Corporation

Use the WSSLogin JAAS configuration to have WebSphere Application Server recognize this token as the authenticated user.

## Slide 75

Presented by IBM developerWorks 

## The WS-Security BinarySecurityToken element

- Signed security tokens, such as a Kerberos ticket or x.509 certificate, are binary content. They must be encoded for inclusion in the wsse:Security container

```

<S:Envelope
  xmlns:S="http://www.w3.org/2002/06/soap-envelope">
  <S:Header>
    <wsse:Security
      xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/07/secext">
      <wsse:BinarySecurityToken wsu:ID="myToken"
        ValueType="wsse:Kerberosv5ST"
        EncodingType="wsse:Base64Binary">
        XIFNWZz99UUbAlqIEmJZc0
      </wsse:BinarySecurityToken>
    </wsse:Security>
  </S:Header>
  <S:Body> App-specific content </S:Body>
</S:Envelope>

```


Security Info

1-75 Achieving Enterprise Application Security © 2007 IBM Corporation

*Mini glossary:*

**Base64 encoding** – base64 is an encoding format that takes every 3 bytes of any data and converts them to 4 bytes or characters, taken from a set of 64 possible characters: A-Z, a-z, 0-9, and sometimes / and +. Thus any data -- binary, textual or otherwise, becomes textual or ASCII, albeit with 1/3 larger size. The benefit is that now the ASCII data can be stored in an XML message. Base64 decoding at the receiving end returns the data to its original form.

## Slide 76

Presented by IBM developerWorks 

## Using XML Signature with SOAP


- As we have seen, XML Digital Signatures tells us how to sign arbitrary XML content.
- How do we use XML Signatures with SOAP messages?
  - ▶ WS-Security defines new elements in the SOAP header and body to contain an XML Signature
  - ▶ Standardization of these elements means that implementations from different vendors can interoperate with signatures

```
<Header>                                Signature information is stored in the SOAP header
  <Security>
    <Signature>
      signature info
    </Signature>
  </Security>
</Header>
```

1-76      Achieving Enterprise Application Security      © 2007 IBM Corporation

The major change when XML Signature is used in a SOAP message is that the **<Signature>** element is placed within a **<Security>** element inside the SOAP **<Header>** element.

## Slide 77

Presented by IBM developerWorks 

## Example : SOAP with XML Signature

```
<S:Envelope>
  <S:Header>
    <wsse:Security S:mustUnderstand="1"
      xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/07/secext">
      <wsse:BinarySecurityToken EncodingType="wsse:Base64Binary">
        MIIDQTC4ZzO7tIberPhaidlq ... [truncated]
      </wsse:BinarySecurityToken>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        ...see XML Signature example for full content...
      </ds:Signature>
    </wsse:Security>
  </S:Header>

  <S:Body>
    <m:OrderAircraft quantity="1" type="777" config="Atlantic"
      xmlns:m="http://www.boeing.com/AircraftOrderSubmission"/>
  </S:Body>
</S:Envelope>
```

1-77 Achieving Enterprise Application Security © 2007 IBM Corporation

Here's a question: do you think signing a SOAP message using WS-Security is secure? That it proves who the message sender is? That it proves the message isn't tampered with? Well the answer to all three questions is "maybe."


Using a digital signature to sign the body of a message and then validating the signature at the receiver is how we should be able to know for sure the body has not changed. In the procedure, there is a signature element in the header and one thing it contains is a reference to the body (typically) of the message, which has been signed. And validating that signature proves the body hasn't change.

Actually, it proves that what has been signed has not changed. It does not prove that what has been signed has not been moved

intact, somewhere else within the message, and the content at the original body location replaced? That's what a "brown bag" attack does to a signed SOAP message. The body content is moved up into a header. That content is unchanged. Its ID remains the same. The reference in the signature reference element "points" to that ID because it specifies the ID. It does not "point" to a physical location within the message. So the (previous) body content can still be the target of the signature and the signature can be verified. The brown bag attacker intercepts the message, moves the body content into a header, and replaces the body content with its own devious or malicious content. The message receiver continues to validate the original signature without knowing the message **<body>** has been compromised. The service implementation is called with the bogus business data.

One solution for this is to send signed messages over SSL. SSL encrypts at the transport level and prevents manipulation of the SOAP message. But a better solution is to always make sure that the target of a signed SOAP message has a security policy about what is required to be signed within the message, the LOCATION of that signed content, and other information that prevents the **<body>** content from being replaced. WebSphere Application Server supports policies of this type. These policies can be configured for Web services in Rational Application Developer.

## Slide 78

Presented by IBM developerWorks 

## WS-Security changes the XML Encryption format

**<EncryptedData>** element still replaces the body content being encrypted. It still contains

- ▶ **<EncryptionMethod>** – Algorithm used to encrypt the data
- ▶ **<CipherData>**
  - **<CipherValue>** - Element containing the encrypted data


**<EncryptedKey>** element moved to security header contains

- ▶ **<EncryptionMethod>** - Algorithm used to encrypt key
- ▶ **<KeyInfo>** - Key used to encrypt content data
- ▶ **<CipherData>**
  - **<CipherValue>** - Encrypted symmetric key
- ▶ **<ReferenceList>** - List of references to encrypted content

1-78 Achieving Enterprise Application Security © 2007 IBM Corporation

The **<EncryptedData>** element replaces the data that was encrypted, right in place and usually inside the body of the message. The **<EncryptedKey>** element is placed inside a **<Security>** header.

## Slide 79


Presented by IBM developerWorks 

## Steps to encrypt SOAP XML

1. Create an `<EncryptedKey>` element
2. Add the appropriate `<KeyInfo>` element inside the `<EncryptedKey>`
3. Add a `<ReferenceList>` inside the `<EncryptedKey>` as well
4. Encrypt the data in the message, replacing it with an `<EncryptedData>` element
5. Create a `<DataReference>` element that points to the encrypted data, then add it to the `<ReferenceList>`
6. [Repeat steps 4 & 5 as needed]

1-79 Achieving Enterprise Application Security © 2007 IBM Corporation

## Slide 80

Presented by IBM developerWorks 

## Example: entire document encrypted

```

<PayBalanceDue xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>4019 2445 0277 5567</Number>
    <Issuer>Bank of the Internet</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PayBalanceDue>

```

Purple text is data we want to encrypt  
Green text is left unencrypted

```

<EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
  Type='http://www.isi.edu/in-notes/iana/assignments/media-types/text/xml'>
  <CipherData><CipherValue>A23B4C6</CipherValue></CipherData>
</EncryptedData>

```

“PayBalanceDue” element identity is hidden in encrypted form, can't even tell what kind of transaction it is!

The real cipher would be longer than this


1-80 Achieving Enterprise Application Security © 2007 IBM Corporation



These examples illustrate the flexibility of XML encryption. We can encrypt an entire document replacing all of it with an **<EncryptedData>** element, as shown here. (The XML content to be encrypted is *italicized* in these examples.)

---

## Slide 81

Presented by IBM developerWorks 

### Example: one element & subelements encrypted

```
<PayBalanceDue xmlns='http://example.org/paymentv2'>
  <Name>John Smith<Name/>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>4019 2445 0277 5567</Number>
    <Issuer>Bank of the Internet</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PayBalanceDue>
```


Purple text is data we want to encrypt  
Green text is left unencrypted

```
<PayBalanceDue xmlns='http://example.org/paymentv2'>
  <Name>John Smith<Name/>
  <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
    Type='http://www.isi.edu/in-notes/iana/assignments/media-types/text/xml'>
    <CipherData><CipherValue>A23B4C6</CipherValue></CipherData>
  </EncryptedData>
</PayBalanceDue>
```

1-81Achieving Enterprise Application Security© 2007 IBM Corporation

Or we can encrypt an element and its subelements.

Slide 82

Presented by IBM developerWorks 

### Example: three sibling elements encrypted

```

<PayBalanceDue xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>4019 2445 0277 5567</Number>
    <Issuer>Bank of the Internet</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PayBalanceDue>
    
```

Purple text is data we want to encrypt  
Green text is left unencrypted

```


<PayBalanceDue xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <CreditCard Limit='5,000' Currency='USD'>
    <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
      Type='http://www.isi.edu/in-notes/iana/assignments/media-types/text/xml' >
    <CipherData><CipherValue>A23B4C6</CipherValue></CipherData>
    </EncryptedData>
  </CreditCard>
</PayBalanceDue>
    
```

Three sibling elements get replaced by ONE EncryptedData element

1-82 Achieving Enterprise Application Security © 2007 IBM Corporation

Or we can encrypt subelements only.

Slide 83

Presented by IBM developerWorks 

### Example: element text (only) encrypted

```

<PayBalanceDue xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>4019 2445 0277 5567</Number>
    <Issuer>Bank of the Internet</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PayBalanceDue>
    
```

Purple text is data we want to encrypt  
Green text is left unencrypted

```

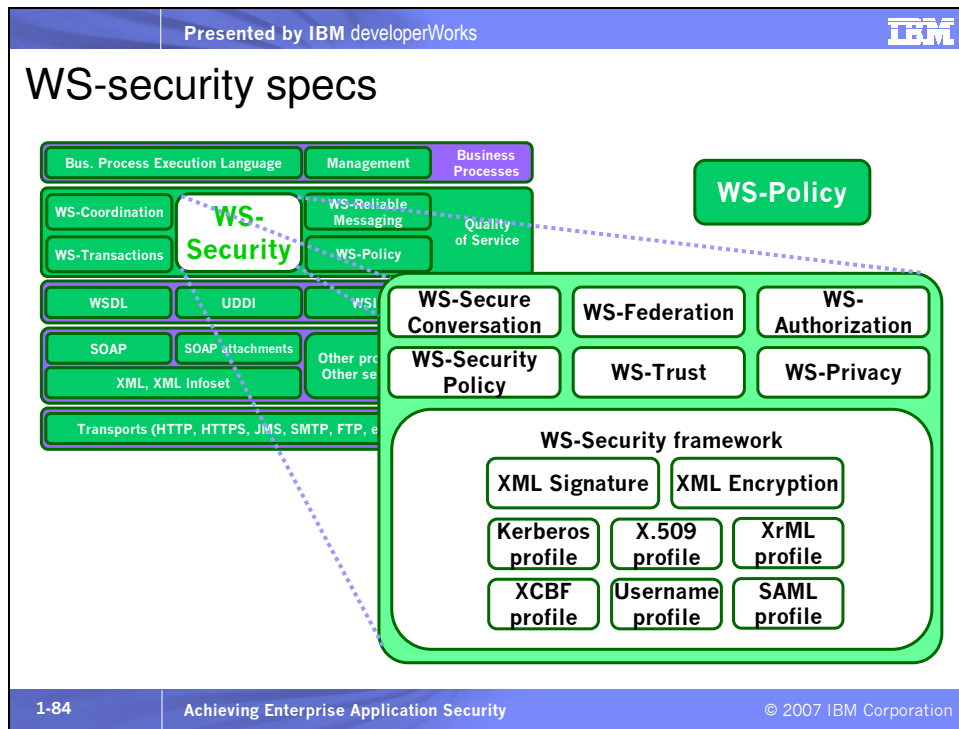
<PayBalanceDue xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>
      <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
        Type='http://www.isi.edu/in-notes/iana/assignments/media-types/text/xml' >
      <CipherData><CipherValue>A23B4C6</CipherValue></CipherData>
      </EncryptedData>
    </Number>
    <Issuer>Bank of the Internet</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PayBalanceDue>
    
```

Text got replaced by an EncryptedData element

1-83 Achieving Enterprise Application Security © 2007 IBM Corporation

Or we can encrypt just the data value of a single field.  
(Thanks to Mark Colan for several of these examples.)

## Slide 84



WS-Security does not want to reinvent the wheel. So it makes use of as many existing security specifications as it can. XCBF is the XML Common Biometric Format, used for biometric authentication, such as fingerprint, retina, breath, and so on. It also uses or works with


- XML Signature
- XML Encryption
- Security Assertion
- SAML -- Security Assertion Markup Language
- XrML -- XML Rights Markup Language
- XACML -- XML Access Control Markup Language

- XKMS -- XML Key Management Specification

We'll cover most of these WS-\* specifications in the next few slides.

---

## Slide 85


Presented by IBM developerWorks 

### What is a policy?

- A policy is a set of capabilities, requirements, preferences, and general characteristics about entities in a system
- The elements of a policy (policy assertions) can express:
  - ▶ Security requirements or capabilities
  - ▶ Various Quality of Service (QoS) characteristics
  - ▶ Any other kinds of policies that are required
- WS-Policy defines a general purpose, extensible model and grammar (“framework”) for describing policies in a Web services system
  - ▶ Simple, declarative policies
  - ▶ More complex, conditional policies

1-85 Achieving Enterprise Application Security © 2007 IBM Corporation

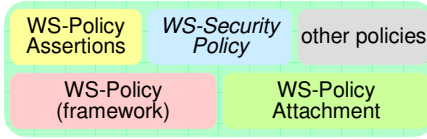
## Slide 86

Presented by IBM developerWorks 

## WS-Policy

<http://ibm.com/developerworks/webservices/library/ws-polfram>

- **WS-Policy** defines the
- framework for policy definition
  - ▶ the container element <Policy>
  - ▶ the organizing operator elements
  - ▶ the "Preference" and "Usage" concepts / attributes
  - ▶ an inclusion / reuse mechanism
- Other policy-related specifications::
  - ▶ **WS-PolicyAssertions** – character encoding, national language
  - ▶ **WS-SecurityPolicy** – security-related policy assertions
  - ▶ **WS-PolicyAttachment** - how to attach a policy to WSDL or UDDI
  - ▶ Policies for other technologies – e.g. WS-Privacy, WS-Authorization, WS-ReliableMessaging




The diagram consists of five colored boxes arranged in two rows. The top row contains three boxes: a yellow box labeled 'WS-Policy Assertions', a light blue box labeled 'WS-Security Policy', and a light green box labeled 'other policies'. The bottom row contains two boxes: a pink box labeled 'WS-Policy (framework)' and a light green box labeled 'WS-Policy Attachment'. The boxes are interconnected by thin lines, suggesting a conceptual framework or relationship between these components.

1-86 Achieving Enterprise Application Security © 2007 IBM Corporation

WS-PolicyAttachment addresses a **significant gap in the standards**. If a service's policy says all requests to this service must be encrypted, the client has no way of finding that out without WS-PolicyAttachment. The specification is in draft form, here: <http://www.w3.org/Submission/WS-PolicyAttachment/>. From the site, "This specification, Web Services Policy Attachment (WS-PolicyAttachment) defines two general-purpose mechanisms for associating such policies with the subjects to which they apply. This specification also defines how these general-purpose mechanisms may be used to associate WS-Policy with WSDL and UDDI descriptions."

Slide 87

Presented by IBM developerWorks 

### Example policy

- Policy Expression
  - XML InfoSet
  - Describes combinations of assertions
- Operators
  - Describes semantics of combinations of assertions
- Assertions
  - The leaf nodes in the policy expressions
  - No core assertions defined in WS-Policy
  - New assertions can be defined using namespace mechanisms


```

<Policy>
  <ExactlyOne>
    <All>
      <SecurityToken ...STA.../>
      <Algorithm ...AA.../>
    </All>
    <All>
      <SecurityToken ...STB.../>
      <Algorithm ...AB.../>
    </All>
  </ExactlyOne>
</Policy>
    
```

All – every contained elements must be applied  
 ExactlyOne – one of the contained elements must be applied  
 OneOrMore – one or more of the elements must be applied

1-87      Achieving Enterprise Application Security      © 2007 IBM Corporation

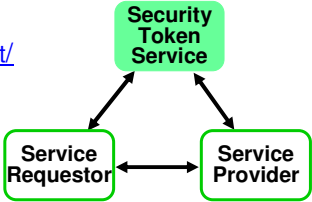
Slide 88

Presented by IBM developerWorks 

### WS-Trust

<http://ibm.com/developerworks/webservices/library/ws-trust/>

- A model for direct and brokered trust relationships
  - Third parties and intermediaries
  - Manage credentials across different trust domains
- Defines:
  - The “security token service”
    - A trusted authority for security tokens implemented as a Web service
  - SOAP messages sent to this service for security token issuance, validation and exchange



```

graph TD
    STS[Security Token Service]
    SR[Service Requestor]
    SP[Service Provider]
    STS --> SR
    STS --> SP
    SR <--> SP
    
```

1-88      Achieving Enterprise Application Security      © 2007 IBM Corporation

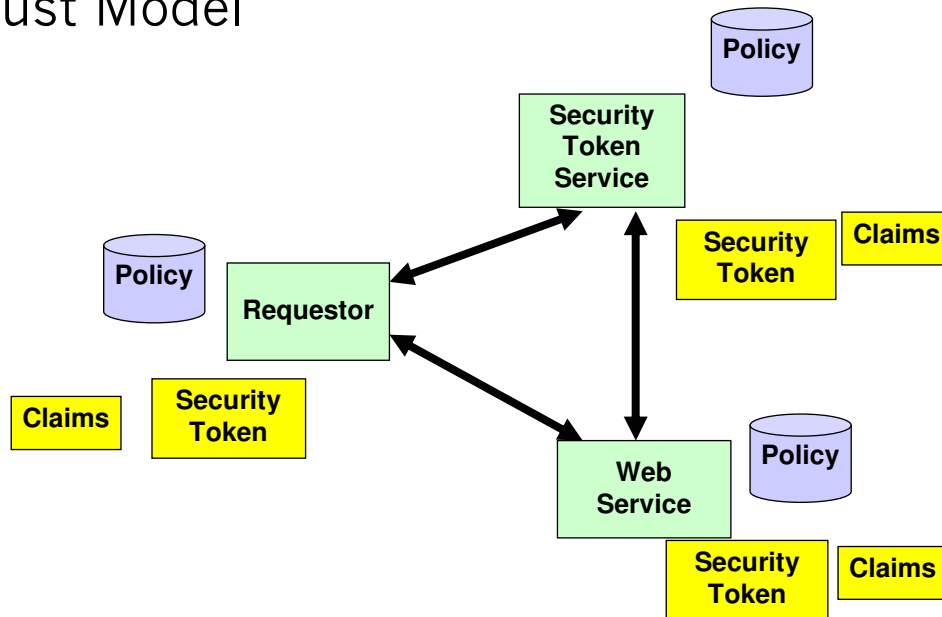
The Security Token Service (STS) is an Access Control Service that provides Permission Tokens. The STS is trusted by the service requestor and the service provider.

WS-Trust defines two "profiles" of the model

- Smart/Active clients (SOAP)
- Passive clients (Browser – HTTP/S)

Active clients such as applications can access a STS with SOAP message requests. Passive clients such as browsers can access the STS using HTTP or HTTPS.

## Securing e-commerce sites - Web Services Trust Model



In this picture a separate Security Token Service (STS), trusted by clients and application servers, provides security tokens containing user credentials. The tokens are created by the STS and passed with messages client-to-service and service-to-service.

## Slide 89

Presented by IBM developerWorks

## WS-Federation

<http://ibm.com/developerworks>

- A *federation* is a collection of security realms (e.g. partner organizations) that have established trust to share security information about users belonging to the realms:
  - identification, authentication
  - attributes, authorization
- WS-Federation
  - builds on the WS-Trust model
  - can share this data using different or like mechanisms
  - defines mechanisms for the brokering of trust and for security token exchange between trust domains
  - does not require local identities at target services
  - optionally allows hiding of identity info and other attributes

The diagram illustrates the WS-Federation process between two organizations:

- Requester's organization:** Contains a **Security Token Service (STS)** and a **Service Requestor**. Both have associated **Security Token(s)**.
- Provider's organization:** Contains a **Security Token Service (STS)** and a **Service Provider**. Both have associated **Security Token(s)**. The Service Provider also has a **Policy**.
- Trust:** A green double-headed arrow labeled **TRUST** connects the STS of both organizations.
- Process Flow:**
  1. A vertical double-headed arrow between the Requester's STS and Requester's Service Requestor.
  2. An arrow from the Requester's STS to the Provider's STS.
  3. An arrow from the Provider's STS to the Provider's Service Provider.

1-89      Achieving Enterprise Application Security      © 2007 IBM Corporation

WS-Federation describes how to share identities and attributes across multiple trust domains. It is layered on WS-Trust. Tokens issued by one domain's STS are used to request a new security token from the STS of another domain. WS-Federation also describes and manages a way to allow the same user to be recognized with different identities in different domains.

WS-Federation was originally announced by IBM, BEA, Microsoft, RSA, and VeriSign. It has several characteristics. It

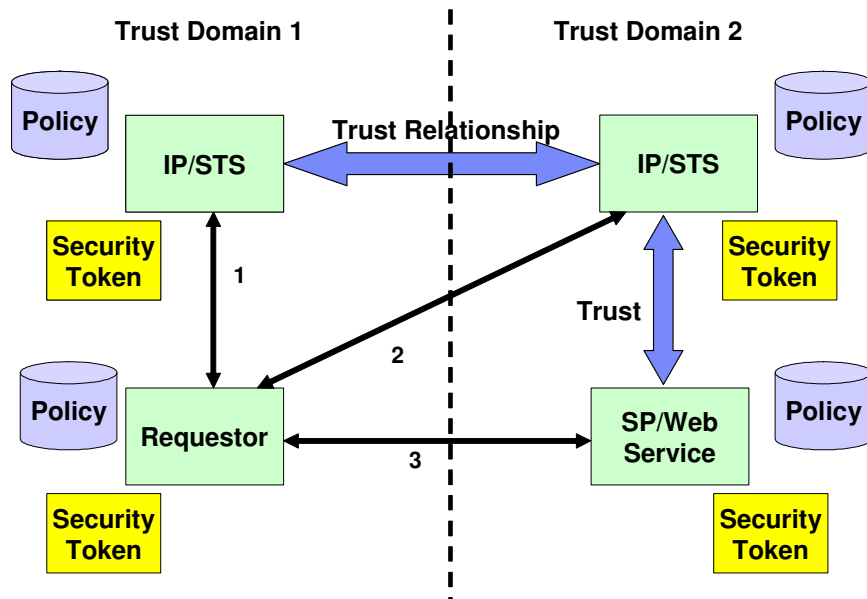
- Enables security realms to federate
- Enhances policy to enable federation of related services
- Describes federation messages
- Describes federated Attribute and Pseudonym service relationships

WS-Federation supports two profiles:



- WS-Federation: Passive Requestor Profile, which uses the cross trust realm identity, authentication and authorization federation mechanisms in WS-Federation to support passive requestors, such as Web browsers
- WS-Federation: Active (Smart) Requestor Profile, which uses the cross trust realm identity, authentication and authorization federation mechanisms in WS-Federation to support active requestors, such as SOAP-enabled applications

## Federation Token Exchanges




IP – Identity Provider STS – Security Token Service

In

this picture a security token is passed from the requestor's trust realm and is used to acquire authentication on the respondent's trust realm, in order to access the resource. The Requestor authenticates to its own identity provider (IP) and receives an identity token (1). It then tries to access the Web service and is redirected to the IP in Trust Domain 2 (2). From there it receives

an access token and accesses the resource (3). The service provider (SP), running the resource, uses its own identity provider (IP) to validate the token received in the request or simply trusts the token because it trusts its own IP (vertical trust arrow). The Trust Domain 2 IP can validate the initial credentials and issue an access token because of its trust relationship with Trust Domain 1's IP. So tokens from one domain are used in another, allowing for federation across the domains. The specification defines a schema for the various messages that would be used in this scenario. The SP in Trust Domain 2 providing the Web service can get further authentication and authorization information from its STS, including the user identity as it is known in Trust Domain 2. This scenario is known as Direct token exchange. The specification also supports indirect token exchange and delegated token exchange.

## Slide 90

Presented by IBM developerWorks 

## WS-SecureConversation

<http://ibm.com/developerworks/webservices/library/ws-secon/>

- Establish a secure, shared security context in which to exchange multiple messages
- Defines the mechanisms for
  - Establishing and sharing security contexts
  - Deriving session keys from security contexts
- Defines 3 ways of establishing a security context
  - Security context token created by a security token service
  - Security context token created by one of the communicating parties and propagated with a message
  - Security context token created through negotiation

1-90 Achieving Enterprise Application Security © 2007 IBM Corporation

Together, WS-SecureConversation, WS-Trust, and WS-SecurityPolicy are coordinated through another specification called WS-SX, or WS-Secure Exchange. The purpose of the OASIS WS-SX Technical Committee, headed by IBM, Microsoft, and Nortel and including many participant companies, is to define extensions to OASIS Web Services Security to enable trusted SOAP message exchanges involving multiple message exchanges and to define security policies that govern the formats and tokens of such messages.

Slide 91

Presented by IBM developerWorks

## SecurityContextToken Example

- SecurityContext Header
- Element identifies the security context using a URI
- Indicates the creation time of the security context
- Indicates the expiration time of the security context
- Holds the shared secrets of the security context
- References the shared secret of the security context

```

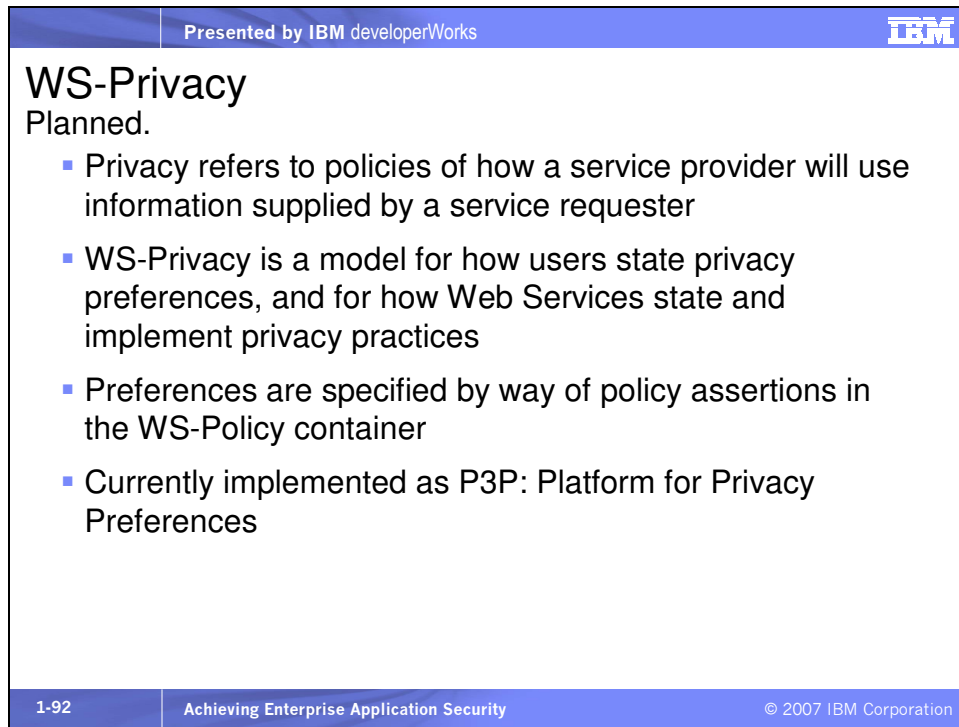
<wsse:SecurityContextToken wsu:Id="...">
  <wsu:Identifier>...</wsu:Identifier>
  <wsu:Created>...</wsu:Created>
  <wsu:Expires>...</wsu:Expires>
  <wsse:Keys>
    <xenc:EncryptedKey Id="...">...
  </xenc:EncryptedKey>
  <wsse:SecurityTokenReference>...
</wsse:SecurityTokenReference>
  ...
</wsse:Keys>
</wsse:SecurityContextToken>

```

1-91
Achieving Enterprise Application Security
© 2007 IBM Corporation

Tivoli Federated Identity Manager implements WS-Trust, WS-Federation and WS-SecureConversation.

## Slide 92



Presented by IBM developerWorks

## WS-Privacy

Planned.

- Privacy refers to policies of how a service provider will use information supplied by a service requester
- WS-Privacy is a model for how users state privacy preferences, and for how Web Services state and implement privacy practices
- Preferences are specified by way of policy assertions in the WS-Policy container
- Currently implemented as P3P: Platform for Privacy Preferences

1-92 Achieving Enterprise Application Security © 2007 IBM Corporation

From <http://www.serviceoriented.org/ws-privacy.html> – "By using a combination of WS-Policy, WS-Security, and WS-Trust, organizations can state and indicate conformance to stated privacy policies. This specification will describe a model for how a privacy language may be embedded into WS-Policy descriptions and how WS-Security may be used to associate privacy claims with a message. Finally, this specification will describe how WS-Trust mechanisms can be used to evaluate these privacy claims for both user preferences and organizational practice claims."

IBM has released a specification called Enterprise Privacy Authorization Language, or EPAL. The specification defines "an interoperability language for exchanging privacy policy in a structured format between applications or enterprises." From

<http://xml.coverpages.org/ni2003-07-09-a.html> -- "EPAL is designed as 'a formal language to specify fine-grained enterprise privacy policies. It concentrates on the core privacy authorization while abstracting from all deployment details such as data model or user-authentication. The Platform for Privacy Preferences (P3P) specification released by the World Wide Web Consortium in April 2002 supports the communication of privacy policies from business applications to consumer applications. EPAL goes one step further, providing an XML language that enables organizations to enforce P3P policies behind the Web, among applications and databases.'"

## Slide 93

Presented by IBM developerWorks		IBM
<h2>WS-Authorization</h2> <p>Planned.</p> <ul style="list-style-type: none"><li>▪ Will describe how access policies for a Web service are specified and managed</li><li>▪ Will describe how claims may be specified within security tokens, and how these claims will be interpreted at the endpoint</li><li>▪ Will provide a schema to uniformly describe a user's authorization profile, which defines how the user is able to access different sets of Web methods on a service</li><li>▪ More information: see Web Services Security Roadmap<ul style="list-style-type: none"><li>▶ <a href="http://ibm.com/developerworks/webservices/library/ws-secmapi/">http://ibm.com/developerworks/webservices/library/ws-secmapi/</a></li></ul></li></ul>		
1-93	Achieving Enterprise Application Security	© 2007 IBM Corporation

## Slide 94

Presented by IBM developerWorks		IBM
<h2>Other related WS-* specifications</h2> <ul style="list-style-type: none"><li>▪ WS-Reliability<ul style="list-style-type: none"><li>▶ A standard specification for a SOAP-based protocol for exchanging SOAP messages with guaranteed delivery, no duplicates, and guaranteed message ordering</li><li>▶ Defined as SOAP header extensions independent of the underlying protocol (this specification contains a binding to HTTP)</li></ul></li><li>▪ WS-Transaction<ul style="list-style-type: none"><li>▶ Defines a set of protocols to coordinate the outcomes of distributed application actions. Includes<ul style="list-style-type: none"><li>▶ WS-Coordination</li><li>▶ WS-AtomicTransaction</li><li>▶ WS-BusinessActivity</li></ul></li></ul></li></ul>		
1-94	Achieving Enterprise Application Security	© 2007 IBM Corporation

See the entire list at [http://www.oasis-open.org/committees/tc\\_cat.php?cat=ws](http://www.oasis-open.org/committees/tc_cat.php?cat=ws). WS-Notification (including WS-Base Notification, WS-Brokered Notification, WS-Topics, and Pub-Sub for WS) provides an event mechanism for Web services. WebSphere Application Server supports WS-Reliability, WS-Coordination and WS-AtomicTransaction.

## Slide 95

Presented by IBM developerWorks

## Composable Headers

Addressing

Security

Reliability

```

<S:Envelope ... >
  <S:Header>
    <wsa:ReplyTo>
      <wsa:Address>http://business456.com/User12</wsa:Address>
    </wsa:ReplyTo>
    <wsa:To>http://fabrikam123.com/Traffic</wsa:To>
    <wsa:Action>http://fabrikam123.com/Traffic/Status</wsa:Action>
  </S:Header>
  <wsse:Security>
    <wsse:BinarySecurityToken
      ValueType="wssec:X509v3"
      EncodingType="wssec:Base64Binary">
      dWJzY3JpYmVyLVBic.....eFw0wMTEwMTAwMD
    </wsse:BinarySecurityToken>
  </wsse:Security>
  <wsrm:Sequence>
    <wsu:Identifier>http://fabrikam123.com/seq1234</wsu:Identifier>
    <wsrm:MessageNumber>10</wsrm:MessageNumber>
  </wsrm:Sequence>
  </S:Header>
  <S:Body>
    <app:TrafficStatus
      xmlns:app="http://highwaymon.org/payloads">
      <road>520W</road><speed>3MPH</speed>
    </app:TrafficStatus>
  </S:Body>
</S:Envelope>

```

1-95
Achieving Enterprise Application Security
© 2007 IBM Corporation

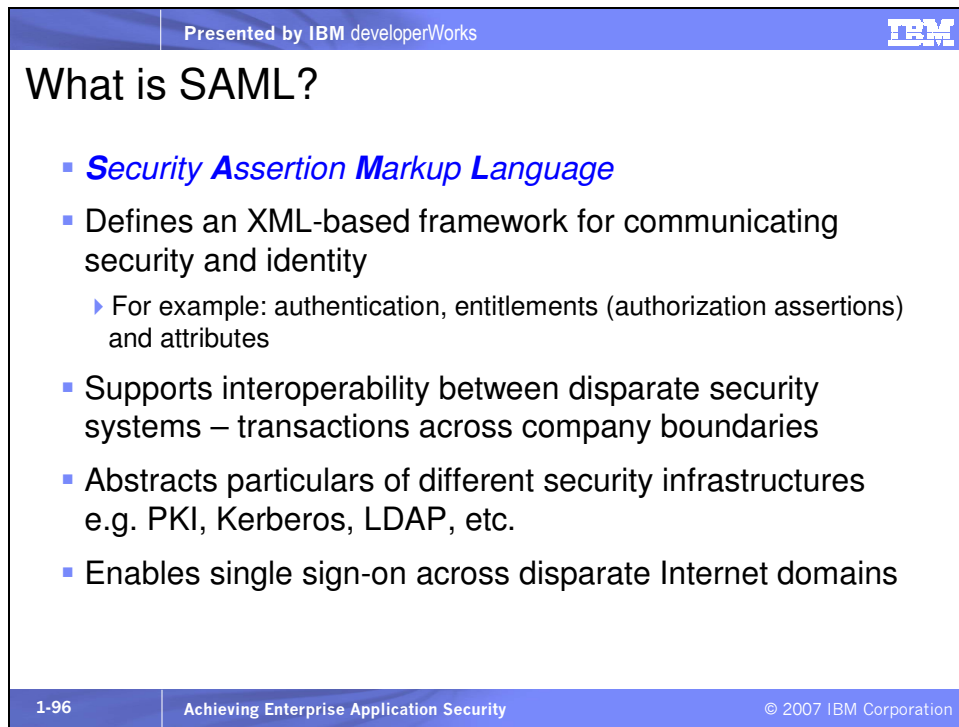
In this example we see how we can have multiple elements in a SOAP header, of different types including security. The first is a WS-Addressing element that provides a callback address (URL) in the **<ReplyTo>** element. The second is a WS-Security element containing a binary security token, actually an X.509v3 digital certificate, encoded as Base64. The third element contains a WS-Reliability element with a sequence number of the message.



Another example might have a security token element, a digital signature element, and an encryption security element. Notice that the **<Body>** element contains the application-specific elements. WS-Addressing is used to address messages to appropriate recipients and provide reply semantics, such as this callback URL.

---

## Slide 96



Presented by IBM developerWorks

### What is SAML?

- **Security Assertion Markup Language**
- Defines an XML-based framework for communicating security and identity
  - For example: authentication, entitlements (authorization assertions) and attributes
- Supports interoperability between disparate security systems – transactions across company boundaries
- Abstracts particulars of different security infrastructures e.g. PKI, Kerberos, LDAP, etc.
- Enables single sign-on across disparate Internet domains

1-96 Achieving Enterprise Application Security © 2007 IBM Corporation

SAML assertions are carried in and referenced from **<wsse:security>** Headers. SAML assertions are used with XML Signature to bind the statements of the assertions (i.e. the claims) to the SOAP message. The following example illustrates a SOAP message containing a SAML assertion in a security header:

```
<S12:Envelope xmlns:S12="...">
  <S12:Header>
    <wsse:Security xmlns:wsse="...">
      <saml:Assertion
        AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
        IssueInstant="2003-04-17T00:46:02Z"
        Issuer="www.opensaml.org"
        MajorVersion="1"
        MinorVersion="1"
        xmlns="urn:oasis:names:tc:SAML:1.0:assertion">
        <saml:AuthenticationStatement
          AuthenticationMethod=
            "urn:oasis:names:tc:SAML:1.0:am:password"
          AuthenticationInstant="2002-06-19T16:57:30.000Z">
        <saml:Subject>
          <saml:NameIdentifier
            NameQualifier="www.example.com"
            Format="">
            uid=zoë,ou=people,ou=saml-demo,o=example.com
          </saml:NameIdentifier>
          <saml:SubjectConfirmation>
            <saml:ConfirmationMethod>
              urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
            </saml:ConfirmationMethod>
          </saml:SubjectConfirmation>
        </saml:Subject>
        </saml:AuthenticationStatement>
      </saml:Assertion>
      . . .
    </wsse:Security>
  </S12:Header>
  <S12:Body>
    . . .
  </S12:Body>
</S12:Envelope>
```

## Slide 97

Presented by IBM developerWorks

## SAML

- Framework for exchanging security assertions
  - ▶ Profiles map assertion use to messaging frameworks
- Use Cases
  - ▶ Single Sign-On
    - Web user authenticates at a Web site. Web user then accesses another Web site without re-authenticating
  - ▶ Authorization Service
    - User attempts to access a resource or service. The access controller for that resource (policy enforcement point) checks the user's rights with a policy decision point
  - ▶ Attribute Service
    - User moves from one Web site to another – customer loyalty information or context is passed to simplify the user's experience as part of a federated information services

1-97      Achieving Enterprise Application Security      © 2007 IBM Corporation

SAML *profiles* describe the HTTP exchanges required to ultimately transfer assertions from an identity provider to a service provider. SAML 1.1 specifies three single sign-on profiles. The profiles encompass how SAML protocols, bindings and assertions combine to support a defined use case:

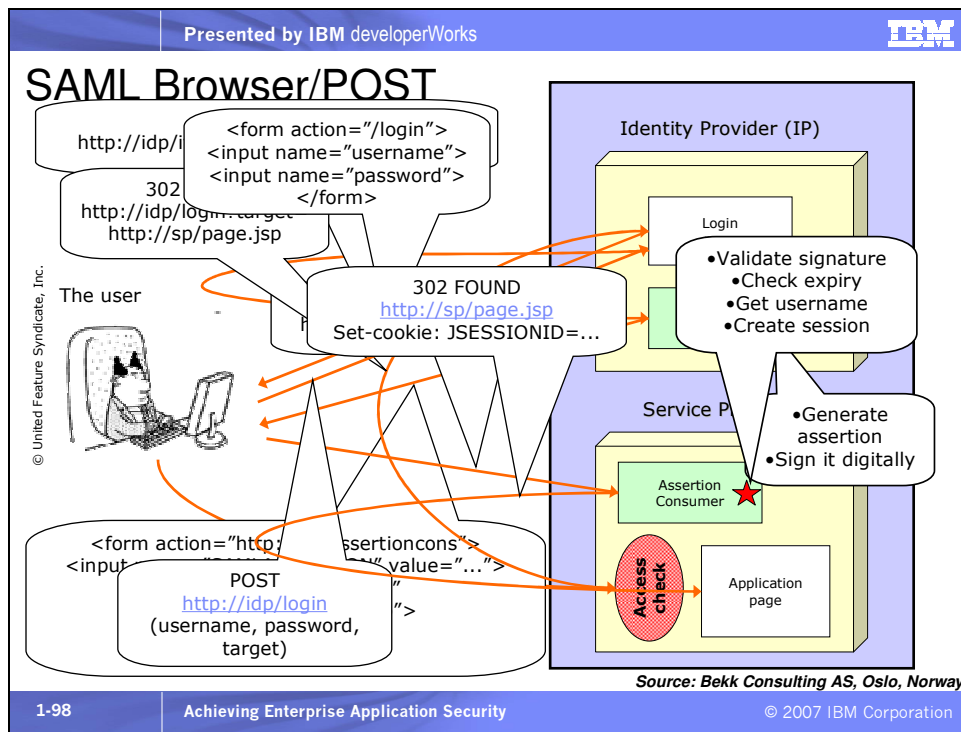
- WS-Security SAMLToken
- Browser/Artifact Profile
- Browser/POST Profile

The latter two profiles define the flow of messages and support cross-domain single sign-on.

The Browser/Artifact Profile employs a "pull" mechanism. The profile essentially passes an *SSO assertion* from the identity provider to the service provider *by reference*, which is subsequently

dereferenced via a back-channel exchange (i.e., the service provider "pulls" the assertion from the identity provider). The Browser/POST Profile relies on a "push" operation. In contrast to the Browser/Artifact Profile, the Browser/POST Profile passes an SSO assertion *by value*. No back-channel communication is needed in this case. In effect, the identity provider "pushes" the assertion to the service provider. (Thanks to wikipedia for this information.)

Slide 98



SAML can appear complex when it is really straightforward; it just has many steps. In browser/POST shown here, the process goes as follows:

1. The user/browser attempts to access a resource from the service provider (SP) with an HTTP GET

2. The SP refers the request (or the browser could be redirected) with a code of temporarily moved to login page of the Identity Provider (IP). The original requested address becomes the TARGET parameter in the request
3. The IP returns the login page to the browser with a form requesting the username and password
4. The browser sends an HTTP POST with the form data back the IP
5. The IP authenticates the user and redirects or refers the return to its Inter-site Transfer Service (IsTS), which generates an assertion and signs it
6. The IP's IsTS returns an HTML document to the browser containing a FORM element whose ACTION attribute is the URL of the SP's Assertion Consumer service (ACS). The originally requested page is stored in the form in a hidden field name of TARGET. A second hidden field in the form, named SAMLASSERTION (in this picture) contains the base-64 encoded SAML response from the IP
7. The browser POSTs the form to the SP's ACS. Sometimes this POST can be automated with JavaScript so no activity is required by the user. The ACS validates the signature on the assertion, checks the expiration, retrieves the username and creates a session for the user
8. The ACS refers or redirects the request, now containing the session ID of the authenticated user, to the original resource at the SP

Thanks to Bekk Consulting AS, Oslo, Norway for this slide.

---

Slide 99

Presented by IBM developerWorks

## Rational Application Developer & WS-Security

- Rational Application Developer provides a great deal of support for secure Web services
  - ▶ Easily sign and encrypt SOAP messages
  - ▶ Add security tokens to SOAP messages
    - See the tutorial at
      - <http://www.ibm.com/developerworks/rational/education/dw-rt-secureservices/index.html>
      - And the self-running demos at
        - <http://www.ibm.com/developerworks/offers/lp/demos/>
    - ▶ Apply security policies to message senders & receivers
    - ▶ Monitor the transmission of secure SOAP messages
    - ▶ More....

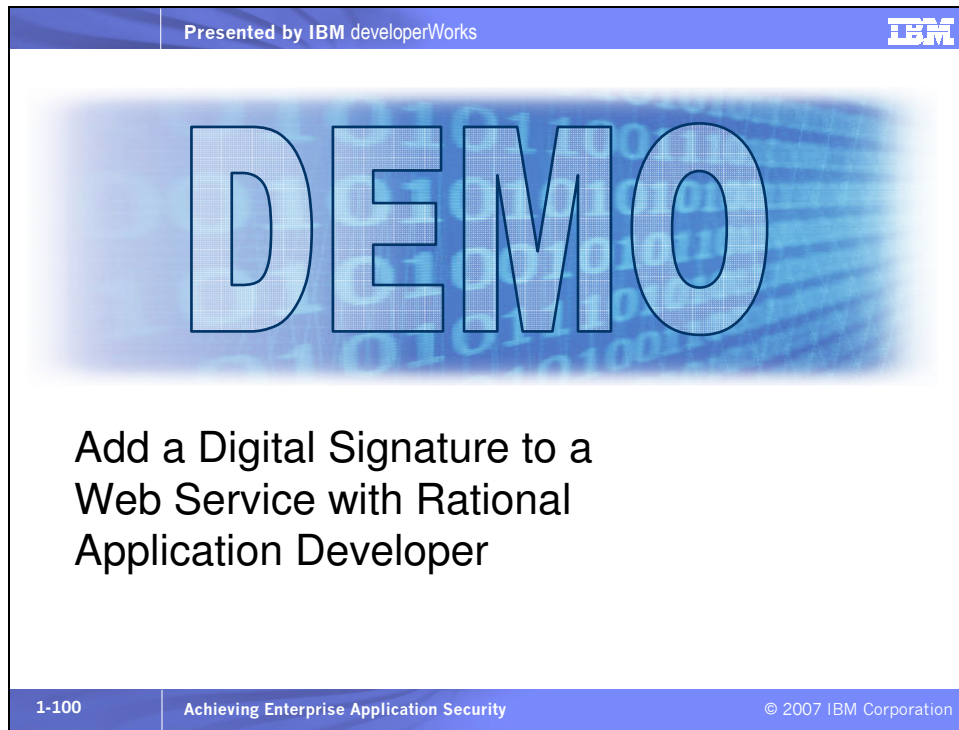
1-99 Achieving Enterprise Application Security © 2007 IBM Corporation


Besides what is shown in the slide, there are links to self-running demos at <http://www.devx.com/OnDemandDemos/Door/30314> that shows how to create Web services using Digital Signatures, Basic Authentication, and other secure techniques. This site and the self-running demos site shown in the slide are mirror sites of each other. To watch how to use Digital Signatures with Web services using Rational Application Developer V6, click on the link entitled "Adding Security to your Web Services Digital Signatures." You will need a free IBM developerWorks user ID and password to view this demo. And besides the URL in the slide, there's a version of the tutorial with an older version of RAD at

<http://www6.software.ibm.com/developerworks/education/r-secureradws/index.html>

---

Slide 100



Presented by IBM developerWorks 

# DEMO

Add a Digital Signature to a  
Web Service with Rational  
Application Developer

1-100 Achieving Enterprise Application Security © 2007 IBM Corporation

This demo shows how easy it is to create Web services that expect a signed request, and how easy it is to sign a Web service request. We'll use Rational Application Developer V7 (RAD) but this can also easily be done with Rational Software Architect or previous versions of RAD, RSA and the WebSphere Studio products. We'll assume you have some experience with these tools, or with Eclipse 3.2, which can also be used.

If you already have a Web project in RAD with some Java code you can use that, or you can use this simple payment calculator class as your service implementation:

```
/**
 * PaymentCalculator.java
 */

package paycalc;

/**
 * @author tylerdurden
 * @input
 * int loan          -- amount of loan;
 * double interest  -- interest rate percent;
 * int term          -- term of loan in years;
 */
public class PaymentCalculator
{
    public int loan;
    public double interest;
    public int term;

    public double calculateMonthlyPayment(int loan,
                                          int percentInterestRate,
                                          int termYears)
    {
        int termMonths      = 12 * termYears;
        float fInterestRate = (float)percentInterestRate / 100;
        float monthlyPrincipal = (float)loan / termMonths;
        float monthlyInterest
            = (float)loan * fInterestRate / termMonths;

        return monthlyPrincipal + monthlyInterest;
    }

    public static void main(String[] args)
    {
        PaymentCalculator pc = new PaymentCalculator();
        System.out.println("Monthly PaymentCalculator Payment is $"
            + pc.calculateMonthlyPayment(50000, 5, 4));
    }
}
```

The `calculateMonthlyPayment()` method will be the single operation on the service. The `main()` method is just for testing the class.

Before you create and test the Web service, you need to know the default address of the TCP/IP Monitor that will display the Web service messages. If you're using RAD or RSA V7, under Window > Preferences expand Run/Debug. Click on TCP/IP Monitor and note



the local port setting for the configured Monitor. You will use this later to set the endpoint for your test client. Select the configured Monitor and click Start. If you're using another version of these products, see the Help for information about how to configure the TCP/IP Monitor.

Start your WebSphere Test Environment (WTE) server. Create a Web project, create a package called `paycalc` in the Java Resources: src folder, and add this class inside the `paycalc` package. Right-click on the class and select New... > Other > Web Service. On the wizard dialog select Start service in the top half and Test client in the bottom half. Also select Monitor the Web service and Overwrite files without warning at the lower left. Click Next twice accepting all the defaults, and on the Web Service Java Bean Identity dialog select a Security configuration of XML digital signature. At this point you can click Finish. Your service will be deployed first, followed by your client. (You may need to add the projects to the WTE server manually.)

The browser should open with the test client displayed. Click on `calculateMonthlyPayment()`, enter some test values and click Invoke to make sure your service and requester are running. Now click `getEndpoint()`, then Invoke, and copy the returned URL to the clipboard. Click `setEndpoint()`, paste in the copied URL, and change the port to that of the TCP/IP Monitor you noted earlier. Click Invoke, and now test your service again with `calculateMonthlyPayment()`. When you click Invoke this

time the TCP/IP Monitor should open displaying the SOAP request on the lower left and the response on the lower right. Double-click its tab to zoom the TCP/IP Monitor view and for each SOAP message change the view type from Byte to XML using the small dropdown control. Now you can see the digital signature information in each message.

Remember, you can watch a video or follow a detailed tutorial at the URLs on the previous slide.

## Slide 101


Presented by IBM developerWorks 

## WS-I.org Basic Security Profile

- In order to ensure that your Web services are securely interoperable with other companies Web services, it's important that you are compliant with the Web Services Interoperability Organization Basic Security Profile
- Find it at
  - ▶ <http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html>

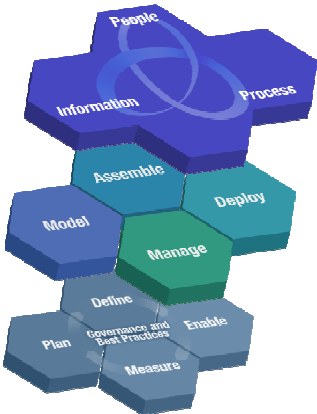
1-101 Achieving Enterprise Application Security © 2007 IBM Corporation

## Slide 102

Presented by IBM developerWorks 

## Service-Oriented Architecture security


- In SOA we need to secure
  - ▶ Every transport, service, and message
- We need to align security with business processes at each step in the lifecycle, with governance
  - ▶ Model: define corporate and business security policies and requirements
  - ▶ Assemble: declare secure application policies and build secure applications
  - ▶ Deploy: into a secure infrastructure for people/process/information
  - ▶ Manage: monitor and manage security events and policies
- Need distributed identity management with single sign-on



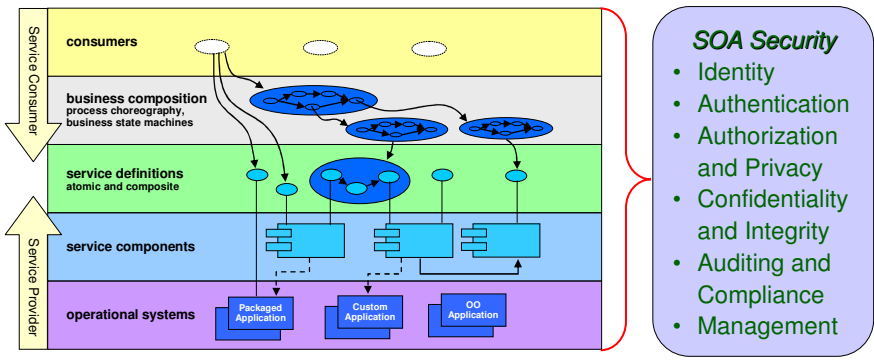
1-102 Achieving Enterprise Application Security © 2007 IBM Corporation

We'll talk about distributed identity management more when we cover Tivoli Federated Identity Manager.

Slide 103

Presented by IBM developerWorks 

## SOA security applies to the entire stack



The diagram illustrates the SOA security stack, divided into two main sections: Service Consumer and Service Provider. The Service Consumer section includes consumers, business composition (process choreography, business state machines), and service definitions (atomic and composite). The Service Provider section includes service components and operational systems (Packaged Application, Custom Application, OO Application). A red bracket on the right groups all these layers under SOA Security, which includes Identity, Authentication, Authorization and Privacy, Confidentiality and Integrity, Auditing and Compliance, and Management.

- Service Consumer
  - consumers
  - business composition  
process choreography,  
business state machines
  - service definitions  
atomic and composite
- Service Provider
  - service components
  - operational systems
    - Packaged Application
    - Custom Application
    - OO Application

**SOA Security**


- Identity
- Authentication
- Authorization and Privacy
- Confidentiality and Integrity
- Auditing and Compliance
- Management

- We need to implement, manage and monitor security at each level

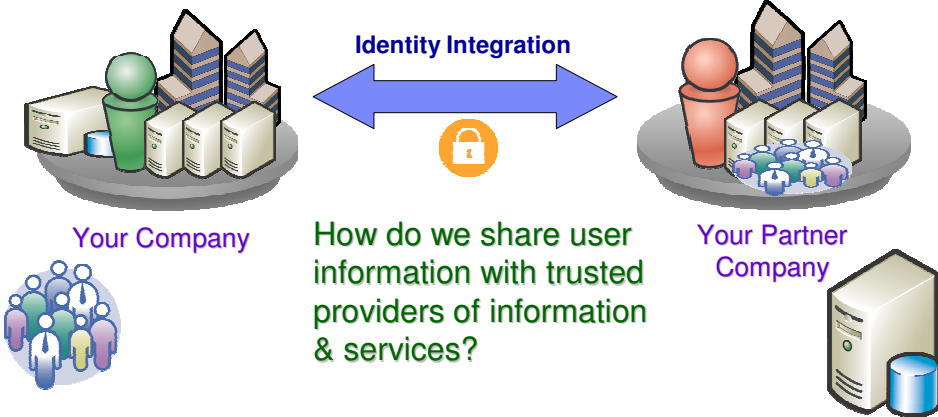
1-103 Achieving Enterprise Application Security © 2007 IBM Corporation

With each service used in our architecture we need full security. We need to propagate identities throughout the architecture, where an individual may have different identities but only log in once. We also need all messages fully secure with all the Web service security we've already discussed, plus management of that across domains.

## Slide 104

Presented by IBM developerWorks 

## Federated Identity is a "Process" within SOA



**Your Company**      **Identity Integration**      **Your Partner Company**


How do we share user information with trusted providers of information & services?

**Identity Management as a business process for cross-enterprise collaboration, with Trust, Integrity & Privacy of Data**

1-104      Achieving Enterprise Application Security      © 2007 IBM Corporation

Tivoli Federated Identity Manager supports this, covered soon.

## Slide 105

Presented by IBM developerWorks 

## AJAX security


- The trend is towards composition of SOA business services into Rich Internet Applications using AJAX
- AJAX security starts with JavaScript security
  - ▶ JavaScript security is based on the Java "sandbox" model
  - ▶ Browsers have security policies to prevent wide JavaScript access to the file system
  - ▶ But...
    - JavaScript should be assumed to be hostile and only deliberately given permission to run
    - AJAX apps run in the untrustworthy browser and run asynchronously, with less user visibility

1-105      Achieving Enterprise Application Security      © 2007 IBM Corporation

AJAX-based applications can be a user entry point into a SOA, and AJAX security is very important. AJAX applications are still open to the same attacks that plague the Web in general. At the very least SSL can help to secure AJAX communication from a browser to a server. AJAX calls are the same as normal HTTP calls, so you can run them using HTTPS, use sessions, etc. just like you typically would.

---

## Slide 106

Presented by IBM developerWorks 

### AJAX security – the same-origin policy

- The *Same-origin* policy says a JavaScript script only has access to same site from which it originated
  - ▶ E.g. suppose AJAX code is executed from a page from `http://www.somesite.com`
  - ▶ The code sends an AJAX request, using an XMLHttpRequest object

```
http_request.open('GET', 'http://www.anothersite.com/apage', true);
http_request.send(null);
```
  - ▶ This will fail with a 'permission denied' error! The second parameter is the URL of the page you're requesting. You cannot call pages on 3rd-party domains
  - ▶ The domains (& protocol & port) must match for the `open()` call
- There is also a *signed-script* policy
  - ▶ Mozilla browsers allow signed script execution outside the sandbox

1-106      Achieving Enterprise Application Security      © 2007 IBM Corporation

If you need to access multiple domains from an AJAX application, write an application proxy that runs on your server and forwards requests to another domain on your behalf, "bridging" access from the browser to the foreign server. Then the browser will see all communication with the same domain. This is a workaround to the

same-origin policy. But it can be a security hole. Let's look at an example of two sites:

`www.myISP.com/siteOne` and `www.myISP.com/siteTwo`

In this example only the context root (the start of the path after the domain) is different. If an ISP named myISP, serves pages from siteOne and siteTwo and those are actually unrelated sites, they will all appear to have the same origin. One of them could be malicious and would be able to access data from the other. To avoid this problem, if your site must be secure, make sure you have a unique domain name for your site, such as `www.siteOne.com` or `www.siteTwo.com`. Do not share an ISP's domain name with another site. Bridging, described above, allows you to control which foreign domains are accessed.

A further security capability of Mozilla-based browsers is called "object signing." This technology was originally developed by Netscape to allow Java applet controls or code downloaded from a trusted Web site to be executed in the browser and access resources outside the "sandbox." If the user trusts the site and the browser can verify that the code is truly from the site, then the code can execute outside of the sandbox. Mozilla-based browsers support this for JavaScript and thus for AJAX. Microsoft-based browsers have a limited version of this for (Windows-only) ActiveX controls, called "Authenticode," which does not yet support JavaScript.

## Slide 107

Presented by IBM developerWorks 

## Use security tokens in AJAX requests

- In the HTTP request header, e.g. for Basic Authentication

```
<html>
  <head>
    <script language="JavaScript">
      // basicAuthToken is the base-64 encoded
      concatenation of the current userID & password
      var basicAuthToken = "dG90b2p3b29m";
      // Include the token in the request
      var req = new XMLHttpRequest();
      req.setRequestHeader("authorization", basicAuthToken);
      . . .
    </script>
  </head>
  . . .
</html>
```

- Or use secure Web service technologies

1-107 Achieving Enterprise Application Security © 2007 IBM Corporation

You can use the HTTP session to store authentication and authorization data for multiple uses. You might use JSON (JavaScript Object Notation) because it has some security features. As a general rule, always validate user input before sending it in an AJAX request. And careful server-side validation is necessary for any application responding to AJAX requests. AJAX applications can make Web service requests just like any other application. Signing, encryption, basic authentication and other token technologies are all possible if the AJAX application is accessing a Web service.

As an additional constraint due to the asynchronous nature of AJAX, it is wise to implement sequence control of some kind as part of the AJAX requests sent into a SOA. Responses may arrive



at different times so sequence management can help restore order. This can also help avoid replay attacks.

The final tip for AJAX security: make sure your users are using up-to-date browsers!


Some content for this section comes from

- [http://developer.mozilla.org/en/docs/AJAX:Getting\\_Started](http://developer.mozilla.org/en/docs/AJAX:Getting_Started),
- <http://www.devarticles.com/c/a/JavaScript/JavaScript-Security/>, and
- [http://search400.techtarget.com/qna/0,289202,sid3\\_gci1198365,00.html](http://search400.techtarget.com/qna/0,289202,sid3_gci1198365,00.html).

These are three good sources of more information.

## Security products from IBM

### Slide 108

Presented by IBM developerWorks 

## Agenda

- Application security basics and core technologies
  - ▶ Security goals for e-business applications
  - ▶ Confidentiality, Integrity, and Non-repudiation
  - ▶ Authentication and Authorization/Access Control
  - ▶ Privacy
- Java, Java EE and WebSphere security
- Web services and SOA security
- **Security products from IBM**
- Attacks and malicious code
- Resources
- Q&A

1-108 Achieving Enterprise Application Security © 2007 IBM Corporation

Continuing on to security products from IBM....

## Slide 109

Presented by IBM developerWorks

## Define security requirements with RequisitePro

The screenshot shows the Rational RequisitePro interface for the 'Banker2007 Project'. The left pane displays a project tree with folders for Coverage Analysis, Features and Vision, Security Features, Functional, Nonfunctional, Vision Document, All Features, Features to Stakeholder Requests, Glossary, Impact Analysis, Software Requirements, Stakeholder Requests, Use Cases, and Requirements Management Plan. The 'Nonfunctional' folder is expanded, showing several requirements including FEAT2, FEAT5, FEAT6, FEAT7, and FEAT8. The 'Requirements' table in the center lists these features with their priority, status, and other attributes.

Requirements:	Priority	Status	Pla	Act	Diff	Stability
FEAT2: Double firewall...	High	Proposed			Me	Medium
FEAT5: All executable...	High	Proposed			Me	Medium
FEAT6: User must...	High	Proposed			Me	Medium
FEAT7: Authentication...	Medium	Proposed			Me	Medium
FEAT8: Authorization...	Medium	Proposed			Me	Medium
* <Click here to create a...	Medium	Proposed			Me	Medium

For the functions available only to bank management there will be a challenge-response system that reduces the possibility of unauthorized access to these features. Here are the Use Case steps:

- 1) authenticated user attempts to access a restricted function
- 2) System responds with predefined challenge for that user
- 3) User answers the challenge with the proper response
- 4) System provides the restricted function

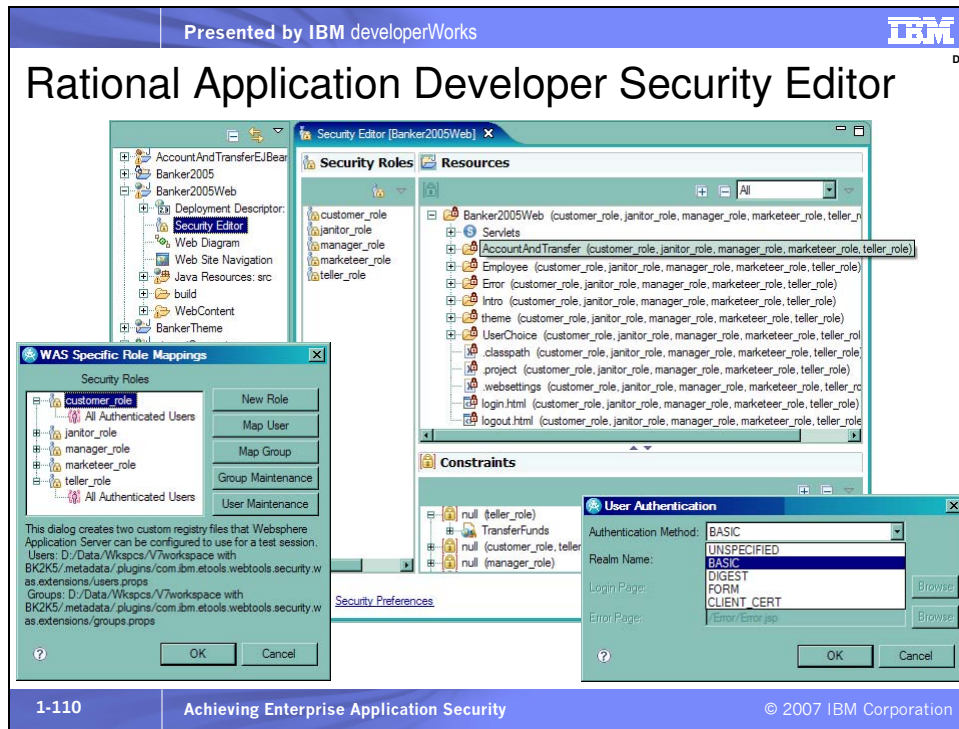
FEAT7: Authentication system  
Tivoli Access Manager WebSEAL will be the reverse proxy security server used in the DMZ to authenticate users. WebSEAL will use Tivoli Directory Server as the LDAP server for the user registry.

1-109 View saved as All Features 5 requirements

Rational RequisitePro is the tool of choice for defining project requirements. In this picture RequisitePro is being used to define functional and nonfunctional security requirements for the new Banker2007 application.

Rational ClearQuest is a tool that provides end-to-end application development lifecycle management, with full traceability and integration with all the other Rational Software Delivery Platform products, including RequisitePro. ClearQuest also supports test management with security. In order to make a change to a test plan, test suite, or other project artifact, ClearQuest can be configured to require a digital signature from the user to ensure compliance with government, corporate or industry regulations for IT governance.

## Slide 110



The Security Editor in Rational Application Developer and Rational Software Architect lets the developer configure declarative security for an application. The developer can associate security roles, create security constraints for servlets and JSPs, and create method-permissions for EJBs.

Clicking on the Role Mapping button at the lower right opens the WAS Specific Role Mappings dialog allowing a developer to work with security roles, map users and groups to those roles, and even maintain those users and groups. Clicking on the Authentication button offers the selection of a type of authentication for the specified realm name, which, by default, is the name of the project. Rational Software Architect also allows the application of patterns, such as design patterns, to models. There is a set of applicable

security patterns that can be downloaded from the developerWorks RAS assets at

<http://www.ibm.com/developerworks/rational/products/patternsolutions/assets.html>. Here's some information from the site:

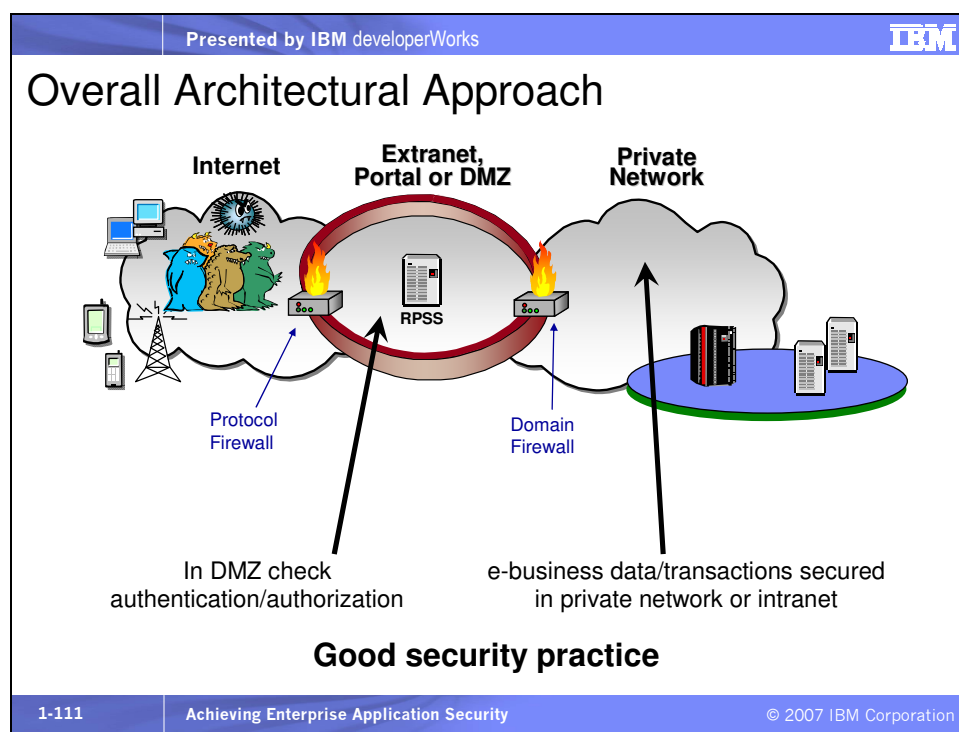
### **"Security patterns**

The security patterns asset is a RAS asset that extends the enterprise patterns asset. It enables you to incorporate security policies when you model your applications. The asset contains two EJB security configuration patterns and an application pattern. You can apply the security configuration patterns to update your EJB deployment descriptor with method level authorization and identity delegation policies. When you apply the application pattern to a class, it generates code which submits Common Base Events using the Common Event Infrastructure."

An article here explains the procedure:

<http://www.ibm.com/developerworks/ibm/library/i-odoebp21/>

## Slide 111



A De-Militarized Zone, or DMZ, is a "gap" between two firewalls, separating two networks as shown. It is wise to put user access control inside the DMZ, because access to DMZ does not imply access to trusted network. The DMZ can be the primary tool to protect a proprietary system connected to Internet because it controls establishment of connections between the two networks.

It provides access control between the two networks such that

- Only authorized traffic, as specified by security policy, is allowed to pass
- It controls packets of traffic in both directions

The firewalls at the edge of the DMZ may provide traffic filtering to individual endpoints & ports (OSI Layer 3), or may provide packet filtering based on message content (higher OSI layers). The

firewall is tamper-resistant and hardened against penetration. It includes audit and monitoring facilities. The DataPower XS40 Security Gateway, which we'll discuss shortly, is an example of a firewall or gateway that provides higher level filtering and validation of message content.

Some vendors are now building firewall products that can filter traffic that has been sent over SSL. These firewalls act as an SSL endpoint, decrypt the data, inspect it, and then send it along over a second SSL session to the original target endpoint. This is a bad industry development, because it breaks the entire SSL "contract" between sender and receiver. Instead of having a single SSL session from sender endpoint to receiver endpoint, there is a period within the firewall appliance or application, albeit brief, where the data is no longer encrypted. At this point an unscrupulous administrator or other insider could steal the data. Not good. Firewalls should pass through untouched SSL-encrypted data that otherwise conforms to firewall rules.

One way to determine the overall security of an architecture is to use "penetration testing." This involves running an application that attempts to break through all defenses using a variety of protocols, ports, and even message content. There are several tools on the market that do this.

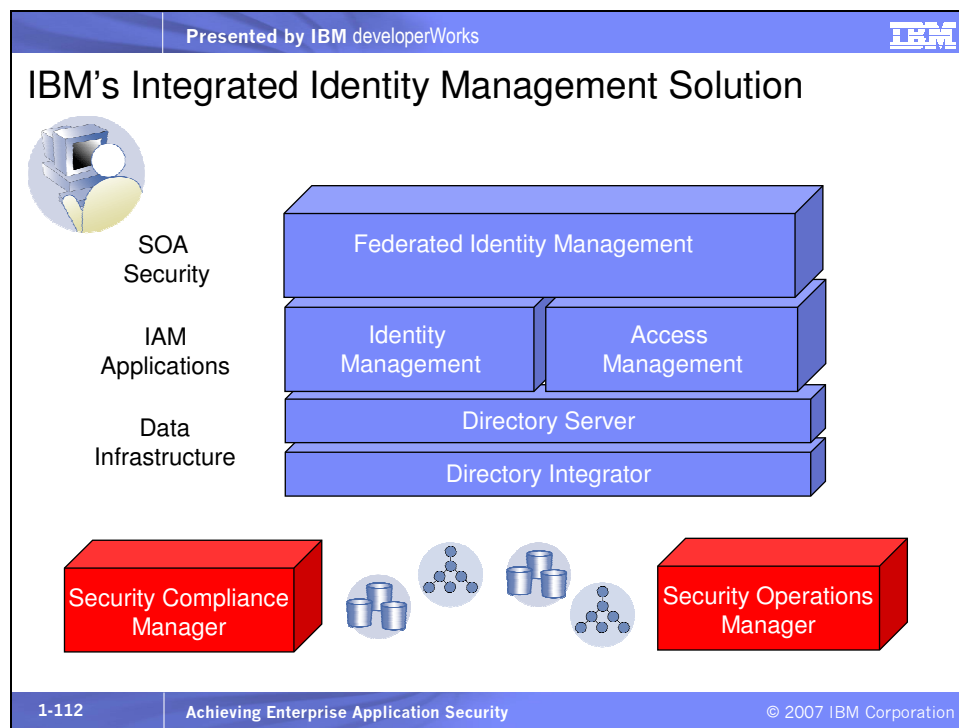
*Mini glossary:*

**RPSS** – reverse proxy security server, such as Tivoli Access Manager WebSEAL, receives requests for resources on the network,

performs authentication and some degree of authorization, and passes the request on to the resource often with additional security credentials attached to the request.

---

Slide 112



Tivoli has the industry-leading portfolio for addressing inefficient business processes.

These security products are under the Tivoli brand:

- **Identity Manager** delivers provisioning, de-provisioning, and self-service. Tivoli Identity Manager is often abbreviated **TIM**.
- **Access Manager** delivers controlled access to applications. Tivoli Access Manager is often abbreviated **TAM**.
- **Directory Server** provides the most powerful and scalable LDAP-based directory in the industry. Tivoli Directory Server is often abbreviated **TDS**.

- **Directory Integrator** addresses the underlying issue that the average large firm stores user information in 181 places (Forrester, 2000). Since this identity management software relies on having a clean, valid list of current users, how do we gather this pragmatically? Well, Directory Integrator allows us to synchronize all of these user information sources together to accomplish this goal. Tivoli Directory Integrator is often abbreviated **TDI**.
- **Security Compliance Manager** is a security policy compliance product that checks systems and applications for vulnerabilities and identifies violations against security policies.
- **Security Operations Manager** is a Security Information & Event Management (SIEM) solution to improve the effectiveness, efficiency, and visibility of security operations, succeeding Tivoli Risk Manager as security monitoring solution.

Finally, most firms are looking at delivering their next-generation applications through Web services, and security is the #1 issue there. Most folks call the technology for addressing this "federated identity management," and IBM Tivoli is one of the leaders, with **Tivoli Federated Identity Manager**, or **TFIM**.



## Slide 113

Presented by IBM developerWorks

## IBM Tivoli Security Products

- **Tivoli Access Manager**
  - ▶ Policy-driven Access Management
  - ▶ Improved user experience through application **single sign-on**
  - ▶ Easier deployment of applications through central policy definition
- **Tivoli Identity Manager**
  - ▶ User Provisioning Management
  - ▶ Efficient management of users through automation and self-care
  - ▶ Access provisioning of resources
  - ▶ Workflow integration with business systems such as HR, approvals
- **Tivoli Federated Identity Manager**
  - ▶ Federated Identity Management
  - ▶ Identity provider for cross-domain trust
  - ▶ Supports WS-Federation, SAML, and Liberty

1-113      Achieving Enterprise Application Security      © 2007 IBM Corporation

Tivoli Access Manager is an *Integrated Security Platform for e-business* that delivers single sign-on to Web-based applications and much more. It lets the right people in, to access the right applications and data by providing integrated, policy-based security management of users, access control, portals, Web applications, messaging applications, custom applications and more. It extends to address the security of the WebSphere MQ family (with Access Manager for Business Integration working with the base Access Manager's services), z/OS or OS390 and UNIX/Linux environments (with Access Manager for Operating Systems working with the base Access Manager's services) and ensures availability in multi-enterprise deployments. Key services that Access Manager provides are single sign-on to Web-based

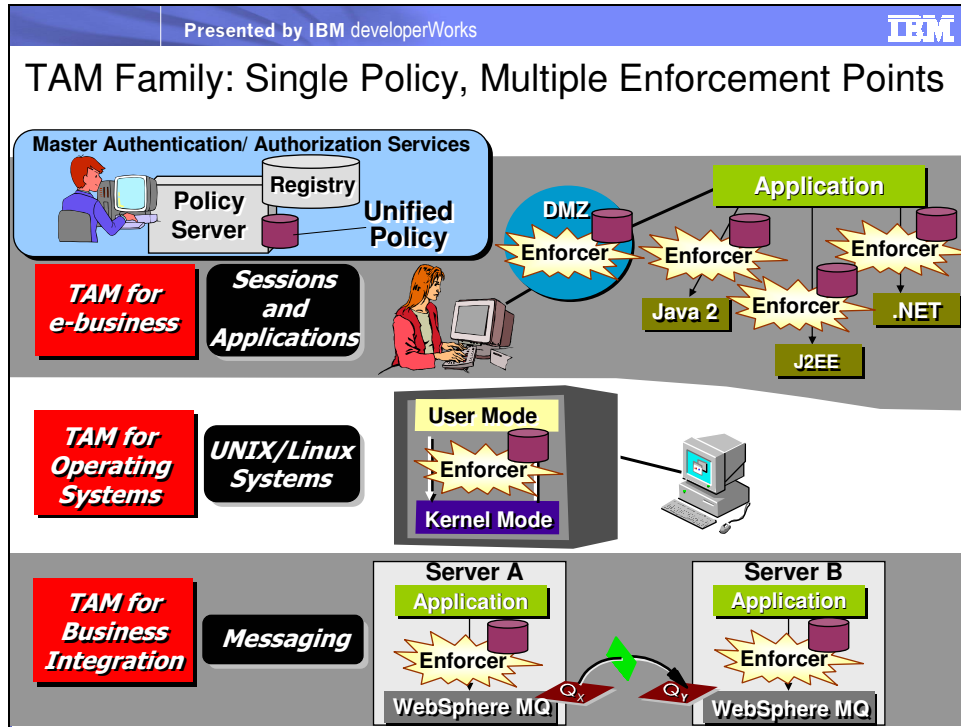
applications and (the heart of Access Manager) policy-based access to applications and data. In addition, the platform addresses distributed, delegatable management of users and access rights. The access rights or permissions address Web applications, including portals, and custom applications.

Tivoli Access Manager for Enterprise Single Sign-on, or E-SSO is a single sign-on solution that eliminates the need for users to remember user names or passwords. It offers logon and password change support for almost any Windows, Web, Java and Host-based application. The client-based software provides automatic password generation and policy support. It leverages any enterprise directory or database as a central repository and provides complete API sets to all integration points.

*Mini glossary:*


**WebSphere MQ** – IBM's ubiquitous message queuing technology that supports guaranteed delivery of messages.

Slide 114

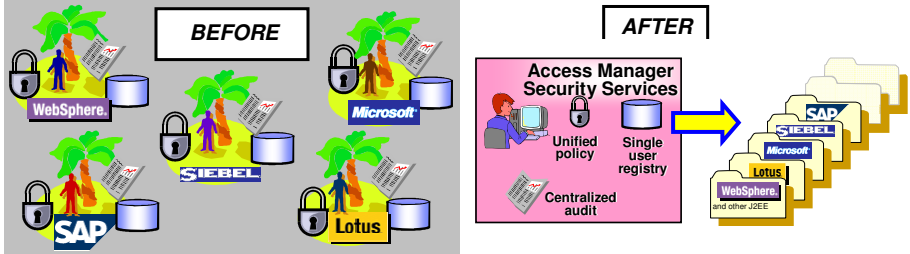


Tivoli Access Manager is actually a family of products that share centralized management and security infrastructure. TAM begins with the Policy Server, central to its architecture. TAM also manages a user registry, usually an LDAP directory. The Policy Server contains access control information, among other things. The user registry contains information about users and groups.

Slide 115

Presented by IBM developerWorks 

## Tivoli Access Manager for e-business






**BEFORE**

- Too many passwords to remember
- Multiple admins with multiple access control tools
- User and access control information everywhere
- Compliance? To what?

**AFTER**

- Web single sign-on
- Single admin or delegated admins with a single tool
- User and security info centralized/understandable
- Policy + audit = compliance

 = Security policy    
  = User & group info    
  = Audit

1-115     Achieving Enterprise Application Security     © 2007 IBM Corporation

Access Manager brings a directory-based, integrated approach to user management and security services for e-business.

The left-hand side of the picture depicts the “islands of security” approach ... wherein customers implement security within applications or more typically using unique security mechanisms provided by a particular application environment. There is nothing nefarious about this; it’s a natural thing to occur that companies implement security and develop specialized skills for each of the application areas they are investing in. But other than perhaps some significant expertise on the part of the security teams that have to stay on top of each application environment, what does the company gain from this? This kind of evolution is actually problematic, in that these security solutions can never be

effectively cobbled together, and the result is that the company that is set up this way has no real idea what their e-business or Web security policy is.

The “after” picture is one involving partners, customers, employees, suppliers, and any other e-business participant attempting to access some resources in the secure intranet. By placing a security layer (HTTP reverse proxy called WebSEAL) in the demilitarized zone, we can accomplish secure, policy-based, and highly available transactions to take place.

The integration that Access Manager affords enables

- Sharing of user and group information in a common directory
- Single sign-on to the target applications
- Efficient exchange of personalization data to maximize the users’ satisfaction with the transaction
- Defense in depth – the capability of layering access points, in order to prevent unauthenticated or unauthorized users from entering the secure intranet


Plug-and-play security for application providers such as Siebel, SAP, Plumtree, BroadVision and others

Note that Tivoli Access Manager for e-business also offers the option of using a Plug-In along with (or instead of) the Proxy (WebSEAL).

The reference to securing SOAP transactions is a way of introducing how Web Services Security Management (WSSM) is rolling out in Tivoli products already. For example, one large Tivoli

customer is using TAMEb to secure SOAP transactions from a parts # application, where the application has an identity and SOAP flows over HTTP.

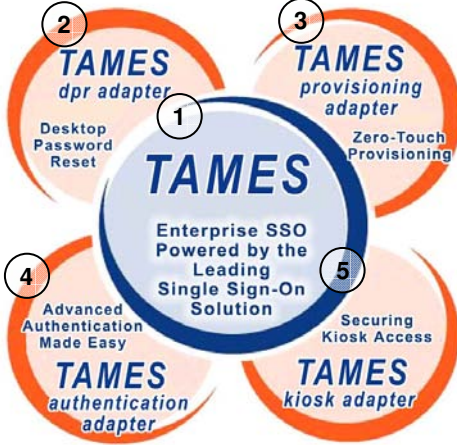
Slide 116

Presented by IBM developerWorks 

## TAM E-SSO – A Family of Products

**TAMES Products:**

- ① **TAMES – Base SSO Product**
- ② **DPR Adapter** - Windows-based Self-Service Password Reset
- ③ **Provisioning Adapter** - Integration with TIM
- ④ **Authentication Adapter** – Flexible, Strong Authentication
- ⑤ **Kiosk Adapter** – Secure Access to Multiple User Access Systems



1-116 Achieving Enterprise Application Security © 2007 IBM Corporation

TAM for E-SSO (TAMES) is the **enterprise single sign-on** solution. TAMES's patented architecture eliminates the burden of integration so customers can start reaping the benefits of single sign-on in days - not months.

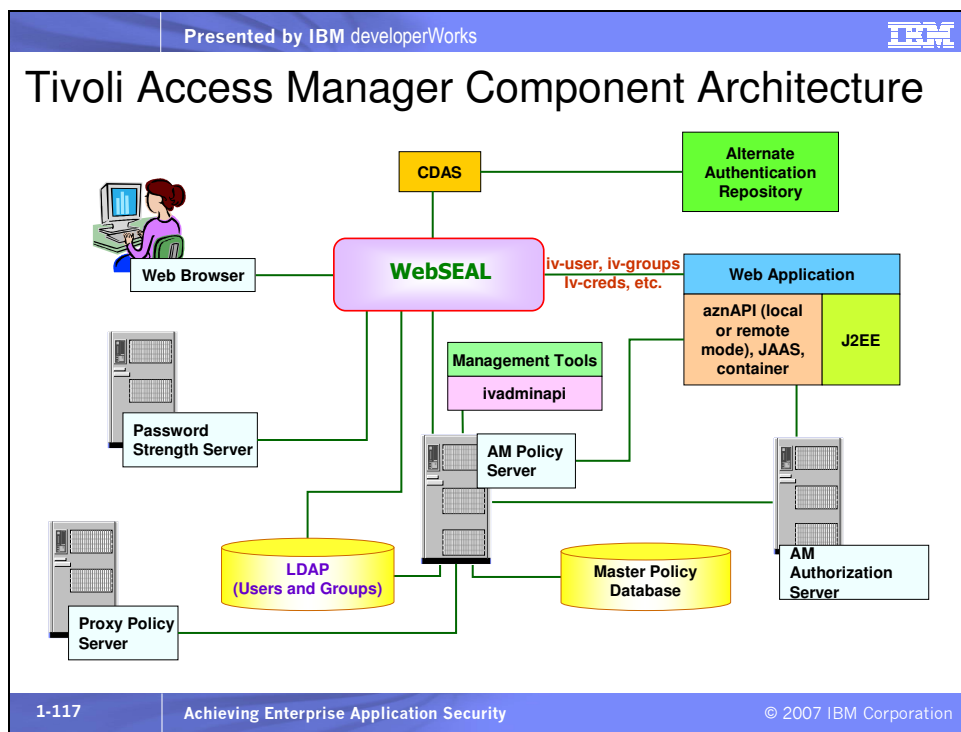
TAMES Desktop Password Reset (DPR) enables end user to reset their Windows password, without calling the help desk, directly from the locked workstation.

TAMES Authentication Adapter allows organizations to use any combination of tokens, smart cards, biometrics and passwords to control access to their applications.

TAMES Provisioning Adapter enables TIM system administrators to directly distribute user credentials — usernames and passwords — to the TAMES Enterprise Single Sign-On solution.

TAMES Kiosk Adapter provides automated termination of inactive sessions and application shutdown for Kiosk or shared workstation users.

Slide 117



The **IBM Tivoli Access Manager** family provides a consistent security implementation for users to flexibly access IT resources. Users can access resources using wired or wireless devices, through flexible authentication methods, such as tokens, digital certificates, or biometrics. The Tivoli Access Manager family also supports different identity data repositories, including the IBM

Directory Server, as well as directories from Sun, Novell, Microsoft, Lotus and Tivoli Identity Manager.

Utilizing a consistent security engine that provides out-of-the-box integration with numerous back-end resources, organizations can realize return on investment through decreased application development and deployment costs, reduced help desk costs and improved administrator productivity.

The IBM Tivoli Access Manager family provides users with a consistent method for accessing their authorized IT resources, including Portal, CRM, ERP and many other Web solutions.

Integration is provided for any Web application server through Java EE, JAAS, and/or the aznAPI (standard authorization API).

Integrated security is available for many WebSphere solutions, including WebSphere Portal, WebSphere Application Server, WebSphere Business Integration, WebSphere Business Integration for HIPAA, WebSphere Everyplace and WebSphere MQ.

Integration with Microsoft is provided in several ways: Microsoft Windows 2000 desktop users can use Single Sign-On to access enterprise portal targets; Tivoli Access Manager leverages SPNEGO (Simple Negotiation Protocol) and also Kerberos and NTLM authentication to extend SSO to applications and middleware; supports Microsoft Active Directory as an identity data repository and can provide Active Directory identity management, provisioning and data synchronization; Microsoft is also working with IBM to define Web Services Security standards (WS-Security).



IBM Tivoli Access Manager provides single sign-on to Lotus Sametime, Team Workplace and e-mail through Lotus Domino. Tivoli Access Manager delivers Web single sign-on, access control for Lotus iNotes Web Access functions, and secure Web browser-based access to the full-featured iNotes functions.

IBM Client Security Solutions offers selected desktop computers and ThinkPad notebooks with built-in cryptographic technologies in both hardware and software that work together to provide a strong level of trust and security in the client PC platform. Tivoli Access Manager for e-business can centrally control and dynamically manage the authentication elements used by these systems.

The Tivoli Access Manager family is comprised of three products:

1. **IBM Tivoli Access Manager for e-business** – provides Web single sign-on and consistent security enforcement across Web and application resources. It provides centralized access control from browsers, PCs and wireless devices to any Web server, using flexible authentication.
2. **IBM Tivoli Access Manager for Business Integration** – Enhances WebSphere MQ's native security services with application-level data protection, without modifying or recompiling MQ applications.
3. **IBM Tivoli Access Manager for Operating Systems** - Helps prevent security violations due to unguarded privileges of UNIX and Linux "root" or "super" users. It can help reduce costs

through centralized administration across AIX, Sun Solaris, HP-UX and Linux systems.

### **Access Manager Policy Server**

The Access Manager Policy Server manages the Master Authorization database where the objects that are used when making authentication and authorization decisions are stored. The Policy Server stores information about objects to be accessed, such as Web pages, JSPs, servlets, and other Web-based application artifacts, and ACLs are associated with these objects.

### **Access Manager Authorization Server**

The Authorization Server is the decision point for authentication and authorization policy decisions and is able to make extremely fast searches as to whether a particular user or group has access to an object, i.e. that there is an ACL containing the user or group, with appropriate action permissions, associated with the object. In addition, a local authorization server can be used as a JACC service provider acting as the decision point whether a Java EE security role is associated with a particular servlet or EJB method.

In WebSphere Application Server 6.x with JACC externalization, the embedded TAM 6.0 Java API builds a local copy of the Master Authorization database (the Master Authorization database is only updated by the Policy Server). Thus the Authorization Server's use is limited, resulting in a faster authorization check since there is no traffic over the network but only the local call.

### **Access Manager WebSEAL**

WebSEAL is the reverse proxy security server that enforces access to resources configured beyond it. See

<http://www.ibm.com/software/tivoli/products/access-mgr-e-bus/>

for much more information about Tivoli Access Manager.

### **Access Manager Policy Proxy Server**

The Access Manager Policy Proxy Server is used to set up a proxy server, which acts as an intermediary between a less trusted network and a more trusted network. This server ensures security and provides administrative control and caching services. It is associated with or part of a gateway server that separates the enterprise network from the outside network, and a firewall server that protects the enterprise network from outside intrusion. In a Tivoli Access Manager environment, the proxy server runs on behalf of the policy server for a given number of authorization applications and administrative functions, such as **pdadmin** commands. (Pdadmin is the command prompt administration tool for Tivoli Access Manager. Access Manager also includes a Web-based administration tool.)

## Slide 118

Presented by IBM developerWorks

## Why Use Access Manager with WebSphere?

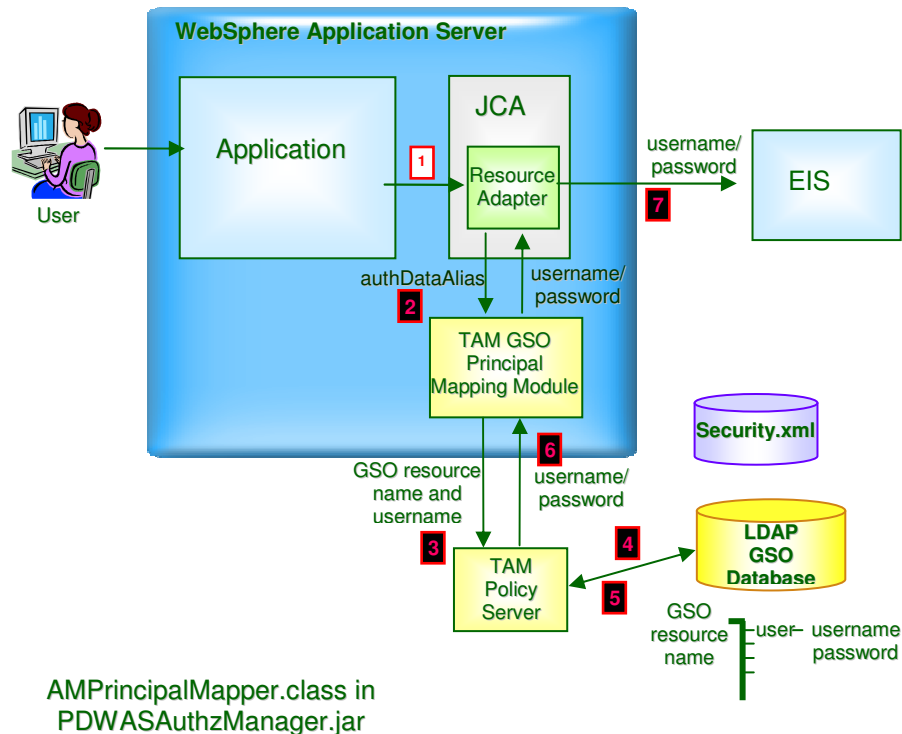
- *Centralize* security management
- *Externalize* security management from WebSphere
- *Manage* access to non-WebSphere resources
- *Authenticate* in the DMZ, not inside the inner firewall
- *Secure* non-HTTP and non-RMI/IIOP access to resources
  - WebSphere MQ Queues
  - OS
- Provide *single sign-on* across old and new applications
- *Container-based* authorization
- *Self-registration* for password and account changes
  
- Access Manager alone does NOT provide access control to EJBs
  - Without custom code in the EJBs
  - WAS still manages J2EE security roles necessary to access EJB / HTTP methods

1-118      Achieving Enterprise Application Security      © 2007 IBM Corporation

There are many integration scenarios for using Access Manager with WAS and there is a great deal of synergy between the two. For more information on the mechanisms and benefits of integrating Tivoli Access Manager with WebSphere Application Server, see the excellent article at

[http://www.ibm.com/developerworks/websphere/techjournal/0206\\_miller/miller.html](http://www.ibm.com/developerworks/websphere/techjournal/0206_miller/miller.html).

## Access Manager GSO Principal Mapping Architecture



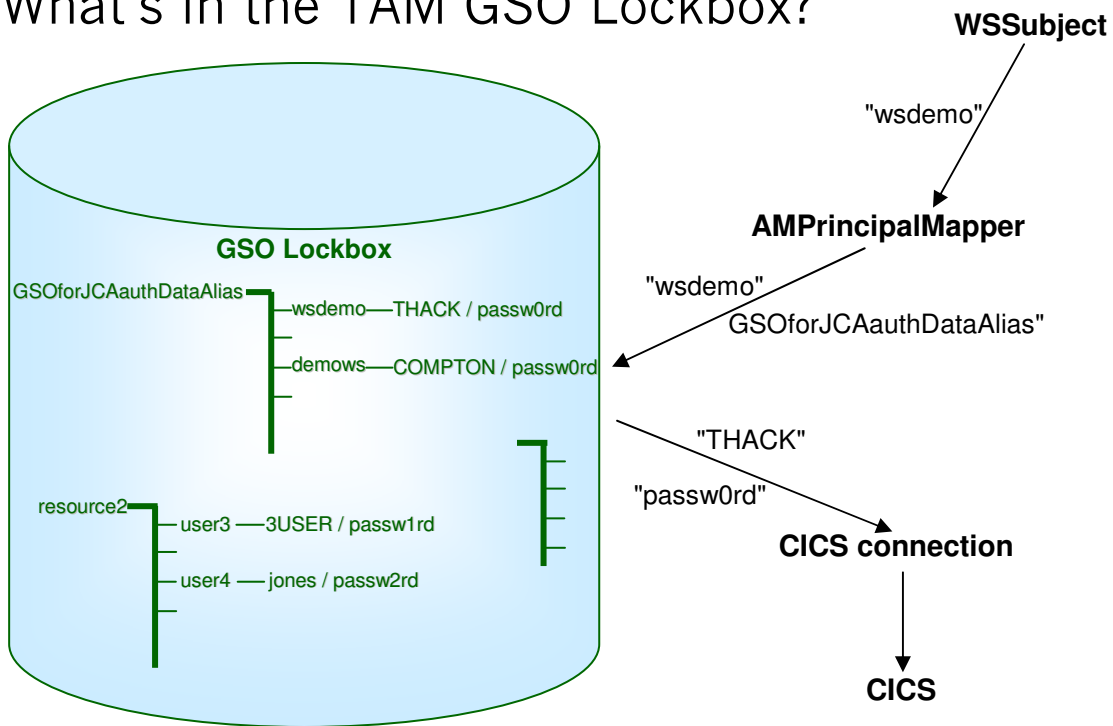
The TAM GSO Principal Mapping Module is the `AMPrincipalMapper.class` in `PDWASAuthzManager.jar`, that comes with WAS V5.x and above. This class can be used to provide individual user credentials per JCA connection rather than a predefined set of user credentials associated with the one and only connection factory for a particular EIS. Using this technique, the credential of the current user can be sent along with the JCA request, rather than a single set of credentials specified as a JAAS alias in the application server's connection factory configuration. The Principal Mapping Module accesses the GSO database and retrieves a configured set of credentials, for the current user, based on the resource to be accessed across the connection.

*Mini glossary:*

**GSO** – Global Sign-On, supported by Tivoli Access Manager, is single sign-on with a different name. The GSO lockbox associates users with resources by specifying appropriate login names and passwords for each resource, for a particular user. Thus if user Jeff can log into application A as jeff with a password of passwOrd (not a good password) but is recognized and authenticated by application B as jeffy with a password of 12345678 (not a good password, either), then Jeff can authenticate to Access Manager's WebSEAL component and WebSEAL will select the proper pair or credentials from the GSO lockbox depending on which application Jeff accesses. The GSO database physically resides in an LDAP registry.

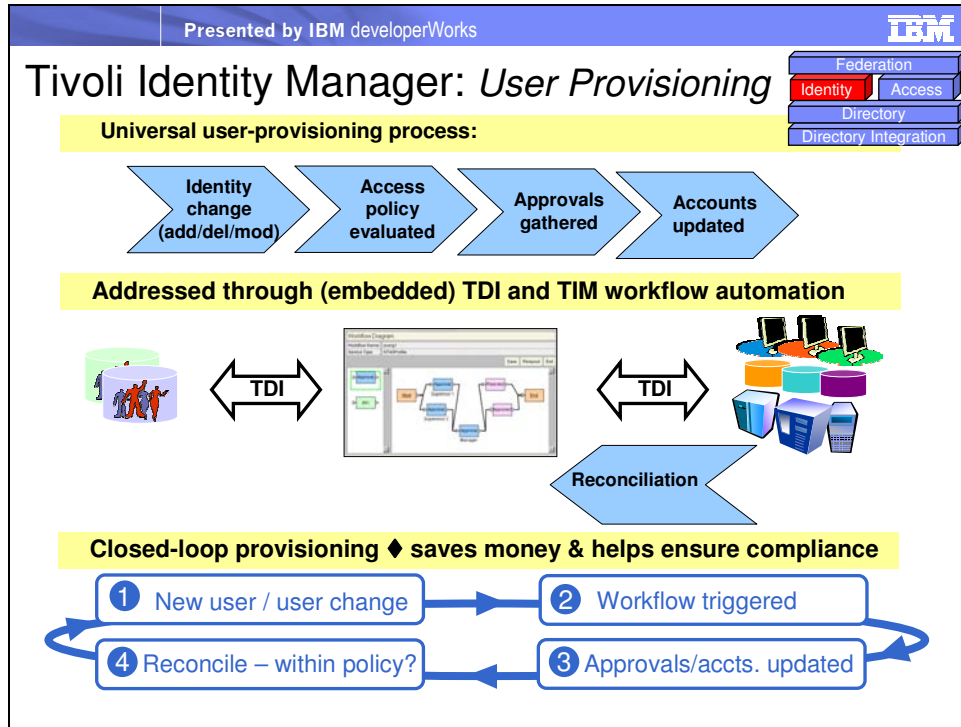
**WebSEAL** – Tivoli Access Manager's reverse proxy security server for Web applications.

## What's in the TAM GSO Lockbox?



We discussed JCA security earlier and mentioned the custom principal mapping module. Here's what it does. The custom **AMPrincipalMapping** module takes the user ID associated with the current thread's **WSSubject**, and using the configured name of the **authDataAlias** (`GSOforJCAauthDataAlias` in this picture) for the particular EIS, retrieves a user ID/password pair for the thread's user ID for that EIS. In the picture, this is `THACK/passw0rd`. In this way, the GSO capability of TAM can provide single sign-on even across JCA connections.

Slide 119



**IBM Tivoli Identity Manager** is one of the products in the IBM integrated identity management solution. Tivoli Identity Manager can automate the management and user provisioning of user identities across the e-business infrastructure. It can help reduce administration costs with delegated administration, centralized user account creation (including self-service interfaces), automated approvals processing that integrates with existing business processes and resource provisioning.

Let's look at an example of automating the steps performed for managing a user.

1. An identity change is requested, such as create a new user account, modify a user account, or delete a user account. This identity change request could be automatically triggered by an



event within your organization. For example, adding a new employee to the Human Resources (HR) Systems could automatically trigger the need to create a user account, based on that employee's role within the organization.


2. Tivoli Identity Manager then evaluates the access policy, using a powerful and flexible automated workflow engine that is GUI based.
3. An automated process is used to gather the required approvals. Automation can simplify and accelerate this process.
4. User accounts are then automatically created, updated, or removed on the corresponding operating systems, databases and applications. Tivoli Identity Manager has the industry's most comprehensive list of supported agents that are used to provision and de-provision user accounts, as well as a toolkit to further extend the solution to additional endpoints.
5. Using bi-directional support, resources are also monitored for any changes made by local administrators, to ensure that those changes are in compliance with your organization's security policies. If a local administrator change is not in compliance, it can be overridden.

Any time an identity change is made to a user, whether it's add, modify, or delete, it is managed by Tivoli Identity Manager to help ensure consistent compliance with your organization's security policy, and to leverage all of the automation and self-service capabilities for improved productivity. Tivoli Identity Manager

helps automate the business processes of creating and provisioning users, managing users and also de-provisioning users.

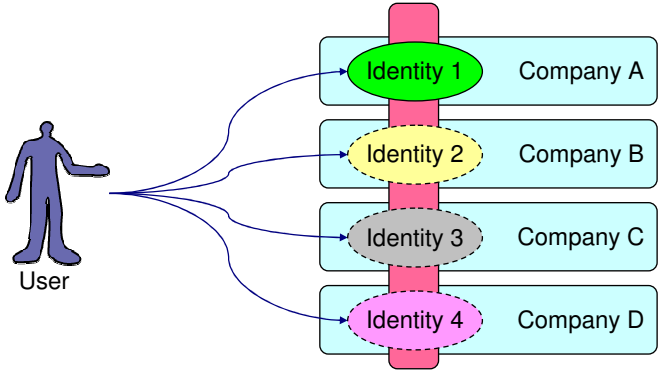
---

## Slide 120

Presented by IBM developerWorks 

### What is Identity Federation?

- The act or process of linking otherwise distinct identities managed by independent parties
  - ▶ E.g. Link multiple virtual identities of a single real user



1-120 Achieving Enterprise Application Security © 2007 IBM Corporation

Federated Identity Management can provide secure business-to-business integration to link identity providers and service providers, while improving the user experience. In this example, the user logs in to their financial institution, which is a trusted identity provider. The financial institution is performing the identity management functions to manage and provision the user on its systems.


When the user wants to access additional Web services that are being provided by other partner providers to the financial institution, the user experiences a seamless access to those Web services. Since the financial institution has already authenticated

the user, the user is able to access the desired Web services from the other partner providers without logging in a second or third time. The Web service providers use the identity information that they are provided from the financial institution, which is the trusted identity provider in this example.

The user benefits by being able to seamlessly access all of the needed services from their financial institution, as well as from other trusted Web services providers. At the same time the Web services providers benefit by being able to provide their services to users without having to incur all of the expenses associated with managing the user's identity.


Now we'll talk about Tivoli Federated Identity Manager.

Slide 121


Presented by IBM developerWorks 

## Roles: Identity Provider and Service Provider

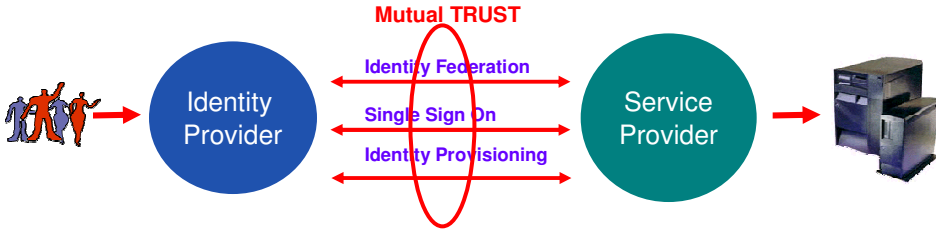
“Vouching” party in transaction



“Validation” party in transaction



**Mutual TRUST**



<ol style="list-style-type: none"> <li>1. Issues Network / Login credentials</li> <li>2. Handles User Administration/ ID Mgmt</li> <li>3. Authenticates User</li> <li>4. “Vouches” for the user’s identity</li> </ol>	<ul style="list-style-type: none"> <li>Service Provider controls access to services</li> <li>Third-party user has access to services for the <u>duration of the federation</u></li> <li>Only manages user attributes relevant to SP</li> </ul>
---	--

Tivoli Federated Identity Manager is an Identity Provider

1-121
Achieving Enterprise Application Security
© 2007 IBM Corporation

Within a federation, organizations play one or both of two roles: *identity provider* and/or *service provider*. The purpose of federated identity management is to link identity providers with service providers.

**Identity Provider:**

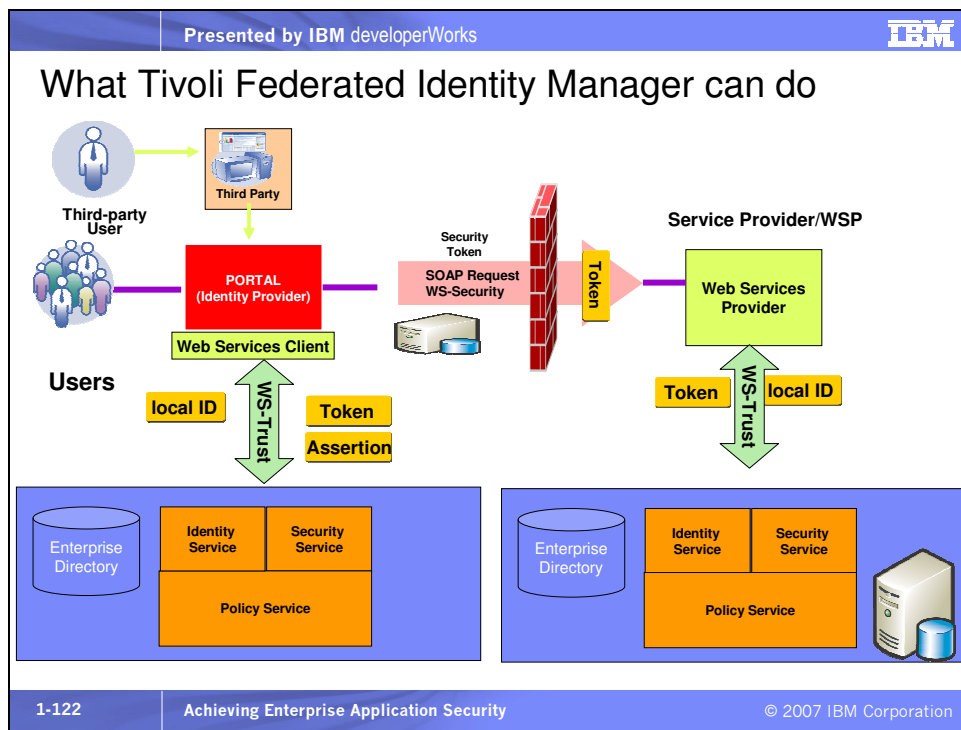
The identity provider (IdP) is the authoritative site responsible for authenticating an end user and asserting an identity for that user in a trusted fashion to trusted partners. The identity provider is responsible for account creation, provisioning, password management, and general account management, and also acts as a collection point or client to trusted identity providers.

**Service Provider:**

Those partners who offer services but do not act as identity providers are known as service providers. The service provider (SP) relies on the IdP to assert information about a user, leaving the SP to manage only those user attributes that are relevant to the SP. In a federation

1. The user authenticates once, to trusted a identity provider
2. The user identity is managed by trusted the identity provider
3. Other Web services providers use identity information from the trusted identity provider

Slide 122



Tivoli Federated Identity Manager, or TFIM, provides the facilities in the two large blue boxes across the bottom. On the left a user attempts to access a page provided by WebSphere Portal.

WebSphere Portal trusts TFIM, on the left, using WS-Trust. TFIM

authenticates the user and returns a token and an assertion about who the user is and what they can do. Now the Portal application needs to call a Web service in another domain for some functionality. In that Web service request, possibly through a firewall, Portal passes along the token and the assertion. The Web service provider receives the request with the token, and passes it to its own implementation of TFIM, at the lower right. TFIM, recognizing the authenticator of the token, says OK, this user is authenticated and oh by the way, this user has a different log in name on our system, and then returns that local ID.

---

### Slide 123

Presented by IBM developerWorks

#### TFIM supported federation standards and efforts

- SAML 1.0, 1.1
  - ▶ Single Sign On
- WS-Federation
  - ▶ Single Sign On/Off
  - ▶ Identity Provider Determination
- Liberty Alliance ID-FF 1.0, 1.1, 1.2
  - ▶ Single Sign On, including Mobile Device SSO
  - ▶ Single Sign Off
  - ▶ Account Linking, De-Linking
  - ▶ Identity Provider Determination

1-123 Achieving Enterprise Application Security © 2007 IBM Corporation

Here we see the standards-based Web single sign-on standards currently supported by IBM Tivoli Federated Identity Manager as of

2005. This list is growing as these standards evolve and as new standards emerge.

Furthering safe online identity management, IBM has recently donated software to the open source community that allows people to keep their identities secret beyond the typical identity management solutions to date, while doing business online. The software, called "Identity Mixer," was developed at IBM's lab in Zurich, Switzerland. Identity Mixer works by creating a pseudonym, or artificial identity information, for online transactions, making them anonymous and eliminating the usual associated data trail. Since only pseudonyms are shared during a transaction, a user's real identity is never exposed.

IBM has donated this technology to the Higgins Trust Framework Project at eclipse.org. See <http://www.eclipse.org/higgins/> for more information.

## DataPower XS40 XML Security Gateway

### *Wirespeed Appliance Purpose-Built for SOA Security*



- Penetration-tested hardened security device
- Appliance Based - “Drop-in” device helps secure multiple applications
- Easy Integration - Interoperates with & augments existing security systems
- Built to differentiate in major areas:
  - ▶ Able to process all data formats including XML and others
  - ▶ Enhanced security - built by experts, 3rd party validation
  - ▶ Performance - no choosing between safety and speed
  - ▶ Agility - future-proof for changing standards, policies, partners

The XS40 is a purpose-built embedded system optimized for SOA processing and provides simplified security management. It can route, transform, and help secure multiple applications without code changes, resulting in lower cost and complexity. It will enable new business with unmatched performance. The XS40 is highly configurable for a simplified SOA architecture. It is standards-based so it works with existing infrastructures.

What is the XS40 not? It is

- Not a general purpose server with some pre-loaded software
- Not running a full standard operating system
- Not traditional network packets (below layer 7)
- Not Java-based

Some of its functionality includes



- XML/SOAP Firewall - Filter on any content, metadata or network variables
- Data Validation - Approve incoming/outgoing XML and SOAP at wirespeed
- Field Level XML Security - Encrypt and sign individual message fields, non-repudiation
- XML Web Services Access Control - SAML, LDAP, RADIUS, etc.
- MultiStep & XML/SOAP Routing - Sophisticated multi-stage pipeline
- Web Services Management - Web services proxy, SLM
- Transport Layer Flexibility - SSL acceleration, WebSphere MQ
- Service Virtualization - Mask backend resources
- Configuration & Administration - Ease of use, Integration for Management

More on the XS40:

- It is a sealed network-resident device that is built with optimized hardware, firmware, and an embedded OS
- It has a single signed/encrypted firmware image and arbitrary software cannot be installed on it
- It has a high assurance, "default off" locked-down configuration; security vulnerabilities are minimized because there are few 3rd-party components
- It provides hardware storage of encryption keys, a locked audit log, has no drives/USB ports, and has a tamper-proof case

- It supports FIPS level 3 HSM as an option, and it is under evaluation by Common Criteria EAL4 (late 2005)
- It is in use by large financial and government customers

For more information see

<http://www.ibm.com/software/integration/datapower/xs40/index.html>.

---

Slide 124

Presented by IBM developerWorks

IBM

# DEMO

Tivoli Federated Identity Manager  
in Action

1-124    Achieving Enterprise Application Security    © 2007 IBM Corporation

We'll look at a short video showing how Tivoli Federated identity Manager enables single sign-on across federated domains.

## Attacks and malicious code


## Slide 125

Presented by IBM developerWorks		IBM
<h2>Agenda</h2>		
<ul style="list-style-type: none"><li>▪ Application security basics and core technologies<ul style="list-style-type: none"><li>▶ Security goals for e-business applications</li><li>▶ Confidentiality, Integrity, and Non-repudiation</li><li>▶ Authentication and Authorization/Access Control</li><li>▶ Privacy</li></ul></li><li>▪ Java, Java EE and WebSphere security</li><li>▪ Web services and SOA security</li><li>▪ Security products from IBM</li><li>▪ <b>Attacks and malicious code</b></li><li>▪ Resources</li><li>▪ Q&amp;A</li></ul>		
1-125	Achieving Enterprise Application Security	© 2007 IBM Corporation

## Slide 126

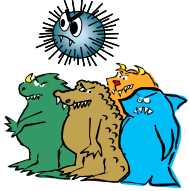
Presented by IBM developerWorks		IBM
<h2>What is the main cause of "malware" infections?</h2>		
<ul style="list-style-type: none"><li>▪ <b>USERS ARE!</b></li><li>▪ <i>Social engineering</i> is a low-tech type of attack in which users are tricked into performing some action or revealing confidential information<ul style="list-style-type: none"><li>▶ Very common, and the easiest way to get someone's password</li><li>▶ <i>Phishing</i> is an example we'll discuss in a moment</li><li>▶ <i>Pretexting</i> means inventing some scenario to obtain another user's information over the phone</li><li>▶ Also includes dumpster-diving, shoulder surfing, looking for passwords under keyboards</li></ul></li><li>▪ Users must be <b>educated</b> how to avoid simple security mistakes like these – don't stick your passwords on your monitor!</li></ul>		
1-126	Achieving Enterprise Application Security	© 2007 IBM Corporation

## Slide 127

Presented by IBM developerWorks 


## Security attacks by malicious code or malware...

- Modify, destroy, steal data or cycles, exploit or damage the system
- May allow unauthorized access
- May come from outside an organization *or from inside!*
  - ▶ Trap Doors – built into system!
    - A developer writes code that allows malicious access into an application while avoiding normal authentication
    - Including Configuration Files
  - ▶ Trojan Horses
    - Harmful code contained in apparently harmless code or data




Further information on newer attempts to deal with and solve the problems of Malware can be found at <http://www.antiphishing.com> and <http://www.ietf.org/internet-drafts/draft-ietf-marid-rationale-00.txt>.

## Slide 128

Presented by IBM developerWorks 


## Security attacks by malicious code or malware

- **Logic Bombs**
  - ▶ Harmful code or logic inserted into a program, designed to execute under certain circumstances. Like a delayed-action Trojan Horse.
- **Viruses**
  - ▶ Code or a program that replicates by being copied or copying itself to another program, document, or computer
- **Worms**
  - ▶ A virus that replicates over the network, usually causing damage when excessive replication causes network and system slowdown



1-128 Achieving Enterprise Application Security © 2007 IBM Corporation

## Slide 129

Presented by IBM developerWorks 

## Malicious behavior continued

- **Man-in-the-middle attacks**
  - ▶ Attacker pretends to be the sender of or responder to a message
  - ▶ Sits between sender and receiver, intercepts sent message, modifies it, and sends it on to target
  - ▶ Solution is to sign messages with Digital Signature or use SSH (Secure Shell) to prevent hacker from sniffing the username/password
- **Spam and fraudulent e-mail**
  - ▶ Unsolicited or unintentionally solicited email
  - ▶ Usually sent over Internet as part of a bulk e-mail by the spammer
- **Phishing**
  - ▶ Phony email or Web site appears to be a trusted source, requests personal information that is then used for identity theft
- **Buffer Overruns**
  - ▶ Code areas of memory existing just above data buffers, overwritten with malicious code that is then executed – less likely with Java code

1-129 Achieving Enterprise Application Security © 2007 IBM Corporation

Phishing is a common online financial scam plaguing users. Emails that claim to be from legitimate businesses like banks and credit card companies, direct recipients to a fraudulent replica of the actual company's Web site. Once they arrive at the site, victims are asked to "update" their personal financial information, such as passwords or PINs, account numbers and Social Security numbers. The information is then used to steal the person's identity, along with their money, and defraud businesses. Phishing sites have grown even more insidious and just landing on a phishing site can sometimes install malicious code without any action from the user. A newer effort to stop phishing is based on building a database of known phishing sites, and then allowing applications, particularly browsers, to programmatically check the database before allowing users to access a potential phishing site. See <http://www.phishtank.com>.

And phishing goes beyond simply fooling people into revealing their financial details. If a phisher does obtain details like that from an unsuspecting user, the phisher must still convert that information into money. In order to do this, a huge worldwide underground network has been established to move and launder money. The people who do the moving are called "money mules." The problem is much more widespread than we realize.

A variation on phishing is called "pharming." Pharming is an attack at the DNS, or domain name server. There are many, many DNSes on the Internet. A DNS is the network server that maps URLs like

www.myCo.com to IP addresses like 111.222.333.444. What makes pharming so dangerous is that there is no way a user can see that the IP address has been switched from the real one, because typically the user only sees the URL when hovering over a link with the mouse or even typing in the URL. One way to notice that there may be a pharming attack underway is if when you access a site with SSL (the small browser lock symbol will be closed), you receive a warning that a server certificate does not match the expected domain for which it is intended. That may appear as an invalid certificate warning from the browser. If you receive a warning like that, be very cautious and careful.

Phishing and pharming are both types of "social engineering" attacks. These types of attacks attempt to manipulate or trick people into doing things, such as revealing private information, that they should not. Social engineering attacks pray on the habits, lack of knowledge, or other vulnerabilities of people, and on the fact that people are not careful enough about what they do on the Internet or their computers in general.

## Slide 130

Presented by IBM developerWorks		IBM
<h2>What is Cross-Site Scripting?</h2>		
<ul style="list-style-type: none"><li>▪ The injection of code into a Web app by a malicious user</li><li>▪ Can gain control of a site or bypass the same-origin policy</li><li>▪ Code can be HTML or scripts, often injected<ul style="list-style-type: none"><li>▶ Via URL request parameters at the end of the URL, or</li><li>▶ By entering malicious code as input to a form that is redisplayed as output</li></ul></li><li>▪ To prevent it<ul style="list-style-type: none"><li>▶ Always validate user input at the server and as much as possible at the client</li><li>▶ Always HTML-escape or entity-encode input data before you write it back out to the browser<ul style="list-style-type: none"><li>▪ E.g. Change the ampersand character from <b>&amp;</b> to <b>&amp;amp</b> or to <b>&amp;#38</b></li></ul></li></ul></li></ul>		
1-130	Achieving Enterprise Application Security	© 2007 IBM Corporation

For a good explanation of cross-site scripting including how to prevent it see

[http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting),

<http://www.devarticles.com/c/a/JavaScript/JavaScript-Security/7/> and

<http://www.devarticles.com/c/a/JavaScript/JavaScript-Security/8/>



## Slide 131

Presented by IBM developerWorks

## Even more malicious behavior

- Denial-of-service attack
  - ▶ A malicious program sends either illegal or a large number of IP packets that confuse and tie up the target system
  - ▶ One solution is to update firewall policies to filter out those packets
- IP address spoofing
  - ▶ An IP packet is sent by a hacker with the source IP address in the packet set to that of a trusted address. The message appears to be from the trusted source
  - ▶ Need to update firewalls, disable weak login authentication
- Impersonation
  - ▶ Hacker with legitimate IP address & stolen password pretends to be user
  - ▶ Need to require use of client side digital certificates for authentication
- Spyware and keystroke loggers
  - ▶ Applications that are secretly or unintentionally installed and that steal and log users' keystrokes, sending them back to a cracker's server for the reuse at protected sites visited by the user
- Adware – advertising-supported software that may double as spyware

1-131 Achieving Enterprise Application Security © 2007 IBM Corporation

The slide states "cracker" instead of hacker, because in the traditional definitions a hacker is a person who writes code, perhaps quickly, sometimes unstructured, trying out lots of ideas, usually successfully, whereas a cracker is someone who breaks into computer systems. Nowadays the term hacker is also used to describe a malicious person.

One way that we can reduce the invasion of malware into our enterprises is with Intrusion Detection. This is the process of identifying and responding to malicious activity targeted at computing and networking resources. Intrusion Detection Systems (IDS) monitor a target system and report information.

Some of the identifying methods are

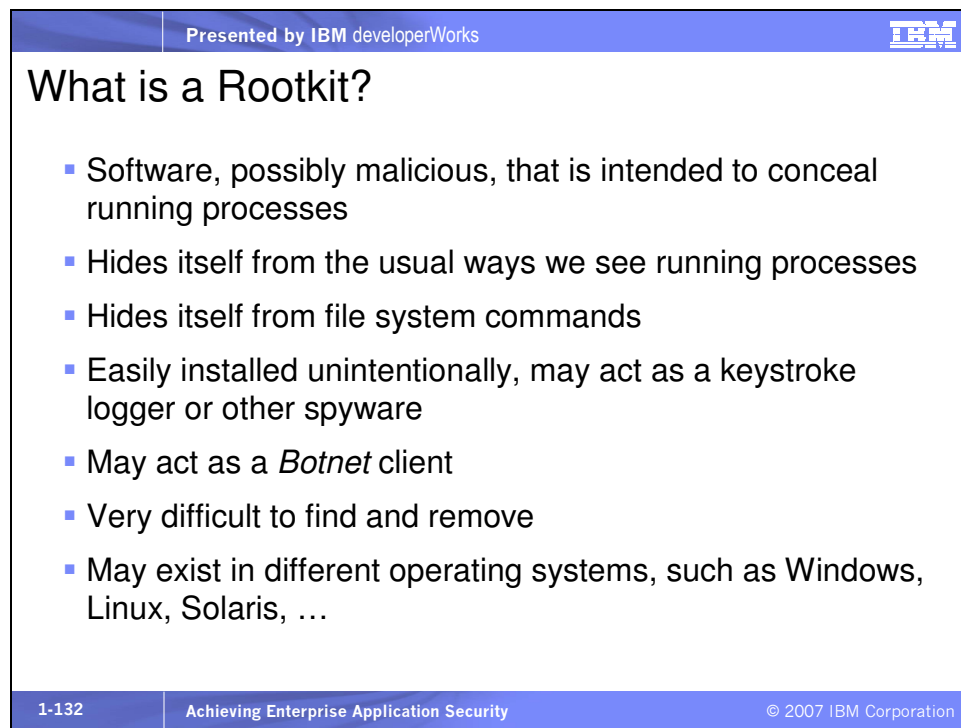
- Audit trail processing

- Real-time traffic monitoring
- Profiles of "normal" behavior
- Signatures of "abnormal" behavior

Intrusion Prevention Systems (IPS) are a newer type of product that attempts to be proactive about intrusions.

---

## Slide 132



Presented by IBM developerWorks

### What is a Rootkit?

- Software, possibly malicious, that is intended to conceal running processes
- Hides itself from the usual ways we see running processes
- Hides itself from file system commands
- Easily installed unintentionally, may act as a keystroke logger or other spyware
- May act as a *Botnet* client
- Very difficult to find and remove
- May exist in different operating systems, such as Windows, Linux, Solaris, ...

1-132 Achieving Enterprise Application Security © 2007 IBM Corporation

Rootkits are not necessary malicious. Some laptop rescue and recovery software installs as a rootkit. Some virus checking and spyware prevention software has rootkit components. Sony used a rootkit to implement their CD copy protection scheme. The Sony rootkit, unfortunately, had a security flaw that allowed a cracker to control the rootkit and repurpose it for malicious activity.

A free, excellent tool to detect rootkits on Windows is RootkitRevealer, from sysinternals.com. Go to

<http://www.sysinternals.com/Utilities/RootkitRevealer.html> and read the entire page to understand the functionality.

BotNets can cause widespread attacks. And it is getting more and more difficult to stay ahead of the BotNets. They are spreading rapidly.

*Mini glossary:*

**Botnet** – you can think of a botnet, from software robot net, as a zombie army, all doing the bidding of a malicious hacker. This is a group of networked computers that have been compromised, possibly by a rootkit, spyware or other malicious code. Upon command all the computers begin to send out denial-of-service attacks, or spam, or worms, or other bad things. When a botnet is large and distributed around the world, it is very hard to eliminate.

## Slide 133

Presented by IBM developerWorks		IBM
<h2>Zero-day attacks</h2>		
<ul style="list-style-type: none"><li>▪ A Zero-day attack involves a cracker/hacker exploiting a vulnerability before it is publicly known</li><li>▪ We have to accept that vulnerabilities exist</li><li>▪ We have to accept that people will always try to exploit them</li><li>▪ You will get hit</li><li>▪ You have to deal with it</li> <li>▪ Recent examples:<ul style="list-style-type: none"><li>▸ MyDoom.ag – buffer overflow triggered by an IFRAME or EMBED tag – oversized SRC or NAME attribute</li><li>▸ Zero-day bot attack – 28 Jan 2005 – MySQL Dynamic Library Exploit</li></ul></li><li>▪ OASIS Application Vulnerability Description Language<ul style="list-style-type: none"><li>▸ For Web application vulnerabilities</li></ul></li><li>▪ Open Web Application Security Project (OWASP) has more information</li></ul>		
1-133	Achieving Enterprise Application Security	© 2007 IBM Corporation

Be vigilant about the security of your systems.

## Slide 134

Presented by IBM developerWorks		IBM
<h2>What you must do:</h2>		
<ul style="list-style-type: none"><li>▪ Pre-establish trust model and security policies with Web services stakeholders and provide auditing controls</li><li>▪ Harden systems and remove any peripheral or superfluous services or applications that are not essential</li><li>▪ Conduct code audits to minimize buffer overflow and SQL insertion attacks</li><li>▪ Ensure confidentiality at the application layer by applying XML Encryption standards</li><li>▪ Prevent message tampering and protect integrity by applying XML Signature</li><li>▪ Carefully select “Web services aware” network security tools, i.e. IPS and firewalls</li><li>▪ Keep patches current</li><li>▪ Perform regular security audits</li><li>▪ <b>EDUCATE YOUR USERS!</b></li></ul>		
1-134	Achieving Enterprise Application Security	© 2007 IBM Corporation

Make sure your users use complex passwords and change them periodically.

## In addition

- Secure the transport layer
- Implement XML filtering
- Mask internal resources
- Protect against XML Denial of Service attacks
- Validate all messages
- Sign all messages
- Time-stamp all messages
- Encrypt all message or parts fields
- Implement secure auditing

To scan your computer for vulnerabilities, go to [www.grc.com](http://www.grc.com) and run ShieldsUP!!, a tool that will attempt to find every available entry point into your machine. While you're there, Steve Gibson and Leo Laporte offer a terrific series of podcasts on security, entitled SecurityNow.

## Finally

- For applications which require the creation and validation of large volumes of XML Signature data, consider the use of hardware acceleration
- Know when XML Encryption is appropriate, and when it is not needed. XML Encryption is useful when a portion of an XML document must be encrypted, and other parts left unencrypted. XML Encryption is also useful when XML data must be *persistently encrypted*
- Remember that key management is vital for XML Encryption. Do not store keys on the file system of an untrusted machine, for example a server located in a DMZ. Unlike XML Signature, where a signature can be verified without the use of a private key, XML Encryption requires a private key for decryption. The safety of this private key is vitally important

These are some additional considerations about Web services and XML security.

## Resource and Q&amp;A

## Slide 135

Presented by IBM developerWorks		IBM
<h2>Agenda</h2> <ul style="list-style-type: none"><li>▪ Application security basics and core technologies<ul style="list-style-type: none"><li>▸ Security goals for e-business applications</li><li>▸ Confidentiality, Integrity, and Non-repudiation</li><li>▸ Authentication and Authorization/Access Control</li><li>▸ Privacy</li></ul></li><li>▪ Java, Java EE and WebSphere security</li><li>▪ Web services and SOA security</li><li>▪ Security products from IBM</li><li>▪ Attacks and malicious code</li><li>▪ Resources</li><li>▪ Q&amp;A</li></ul>		
1-135	Achieving Enterprise Application Security	© 2007 IBM Corporation

## Slide 136

Presented by IBM developerWorks		IBM
<h2>Web resources...</h2> <ul style="list-style-type: none"><li>▪ Computer Emergency Response Team (CERT) – Internet security bulletins<ul style="list-style-type: none"><li>▸ <a href="http://www.cert.org">http://www.cert.org</a></li></ul></li><li>▪ Security Focus – network security, tools, downloads, white papers<ul style="list-style-type: none"><li>▸ <a href="http://www.securityfocus.com">http://www.securityfocus.com</a></li></ul></li><li>▪ Gibson Research Corporation – home security, security scanner testing<ul style="list-style-type: none"><li>▸ <a href="http://www.grc.com">http://www.grc.com</a></li></ul></li><li>▪ System Administration, Security, and Networking Institute – news and vulnerability lists<ul style="list-style-type: none"><li>▸ <a href="http://www.sans.org">http://www.sans.org</a></li></ul></li><li>▪ Vmyths.com for finding hoaxes<ul style="list-style-type: none"><li>▸ <a href="http://www.vmyths.com">http://www.vmyths.com</a></li></ul></li><li>▪ Common Vulnerabilities and Exposures – standard list of names for vulnerabilities<ul style="list-style-type: none"><li>▸ <a href="http://cve.mitre.org">http://cve.mitre.org</a></li></ul></li><li>▪ NTSecurity (and sister site) NTBugtraq for Windows<ul style="list-style-type: none"><li>▸ <a href="http://ntsecurity.ntadvice.com">http://ntsecurity.ntadvice.com</a> and <a href="http://www.ntbugtraq.com">http://www.ntbugtraq.com</a></li></ul></li><li>▪ Security Portal – DSL security, latest news on MS hotfixes, viruses, hoaxes<ul style="list-style-type: none"><li>▸ <a href="http://www.securityportal.com">http://www.securityportal.com</a></li></ul></li><li>▪ RSA Security – loads of technical security information from inventor of PKI<ul style="list-style-type: none"><li>▸ <a href="http://www.rsasecurity.com/rsalabs/">http://www.rsasecurity.com/rsalabs/</a></li></ul></li><li>▪ OASIS Web Services Security: SOAP Message Security spec<ul style="list-style-type: none"><li>▸ <a href="http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss">http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss</a></li></ul></li></ul>		
1-136	Achieving Enterprise Application Security	© 2007 IBM Corporation

## Slide 137

Presented by IBM developerWorks		IBM
<h2>Web resources</h2> <ul style="list-style-type: none"><li>▪ IBM Tivoli – tech resources for security (and other Tivoli) products<ul style="list-style-type: none"><li>▶ <a href="http://www.ibm.com/developerworks/tivoli/">http://www.ibm.com/developerworks/tivoli/</a></li></ul></li><li>▪ Download the WAS V6.1 documentation security portion as a PDF<ul style="list-style-type: none"><li>▶ <a href="http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp">http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp</a></li></ul></li><li>▪ All about Java 6 security<ul style="list-style-type: none"><li>▶ <a href="http://java.sun.com/javase/6/docs/technotes/guides/security/">http://java.sun.com/javase/6/docs/technotes/guides/security/</a></li></ul></li></ul>		
1-137	Achieving Enterprise Application Security	© 2007 IBM Corporation

## Slide 138

Presented by IBM developerWorks		IBM
<h2>Books and periodicals</h2> <ul style="list-style-type: none"><li>▪ <i>Web Security</i> by Lincoln Stein, Addison-Wesley (great read, good overall overview)</li><li>▪ <i>Network Security 2<sup>nd</sup> Ed.</i> by Charlie Kaufman, Radia Perlman &amp; Mike Speciner. Prentice Hall (superb and great read)</li><li>▪ <i>Practical Intranet Security</i> by Paul Ashley and Mark Vandenwauver. Kluwer Academic Press</li><li>▪ <i>SG24-6316 – WebSphere Application Server V6.1 Security Handbook</i><ul style="list-style-type: none"><li>▶ Available from <a href="http://ibm.com/redbooks">ibm.com/redbooks</a></li></ul></li><li>▪ <i>Core Security Patterns</i> by Steel, Nagappan, &amp; Lai. Prentice Hall (Sun)</li><li>▪ <i>Security+ Certification for Dummies</i> by Miller &amp; Gregory. Wiley</li><li>▪ <i>Java &amp; Internet Security</i> by Nadalin, Rich, &amp; Shrader. iUniverse.com</li><li>▪ <i>Inside Internet Security</i> by Jeff Crume. Addison-Wesley</li><li>▪ <i>Enterprise Java Security</i> by Koved, Nadalin, Nagaratnam, &amp; Pistoia. Addison-Wesley</li><li>▪ <i>2600 Magazine - The Hacker Quarterly</i>. Superb. <a href="http://www.2600.com">http://www.2600.com</a></li></ul>		
1-138	Achieving Enterprise Application Security	© 2007 IBM Corporation



Slide 139

Presented by IBM developerWorks 

# ibm.com/developerWorks

Country/region [select] Terms of use

All of dW Search

Home Products Services & industry solutions Support & downloads My IBM

developerWorks > **SOA and Web services** Updated 26 Jan 2007

**Top story**



**Invoke Web services with WebSphere MQ and WebSphere Enterprise Service Bus**  
Learn how to use an existing WebSphere® MQ system to invoke services hosted on WebSphere ESB. [More >](#)

**Defining SOA as an architectural style:** Discover how to define SOA as an architectural style to promote business-aligned enterprise services as the fundamental unit for designing and building solutions.

**Building a secure SOAP client for J2ME, Part 3:** This tutorial covers secure Web services API stub classes. You'll implement a Base64 encoding algorithm, test secure Web services clients, and build a stub enhancer tool.

**Business briefing: SOA connectivity:** This technical briefing provides a technical, practical overview of SOA connectivity, including the benefits and importance of an enterprise service bus (ESB) in creating and maintaining a flexible IT infrastructure.

**Invoke a WebSphere Application Server Web service from CICS:** Use the SOAP/MQ transport to send requests from a CICS-deployed app to one deployed in WebSphere Application Server.

**Rule-based access control:** Improve security and make programming easier with an authorization framework.

**Building SOA composite business services, Part 2:** Migrate business integration projects from WebSphere Application Developer-IE v5.1 to WebSphere Integration Developer v6.0.1.

→ More content 

**My developerWorks**  
Welcome guest  
→ Sign in  
→ Register

**Spotlight**


- Join the developerWorks wiki community
- Attend a tech briefing: Laying the foundation for a Service-Oriented Architecture
- Celebrating 5 years of eclipse
- Beta program - delivering on reliable Web services

**Editor's picks**

- IBM SOA Business Catalog Forum
- IBM Certified SOA Solution Designer certification prep - part of the Enterprise Architect kit/text


1-139 Achieving Enterprise Application Security © 2007 IBM Corporation

Slide 140

Presented by IBM developerWorks 

## Next steps

- Visit the Tivoli security zone on developerWorks  
▪ <http://www.ibm.com/developerworks/tivoli/security/>
- Download a whitepaper on Security And Management for SOA Environments  
▪ <ftp://ftp.software.ibm.com/software/tivoli/whitepapers/GC28-8455-00.pdf>
- On demand demos  
▪ [ibm.com/developerWorks/offers/tp/demos](http://ibm.com/developerWorks/offers/tp/demos)  
▪ <http://www.devx.com/OnDemandDemos/Door/30314>
- The IBM Academic Initiative  
▪ [ibm.com/university/scholars/academicinitiative/](http://ibm.com/university/scholars/academicinitiative/)
- Alphaworks emerging technologies  
▪ [alphaworks.ibm.com](http://alphaworks.ibm.com)
- User group communities  
▪ [ibm.com/developerworks/usergroups/](http://ibm.com/developerworks/usergroups/)
- IBM Redbooks  
▪ [ibm.com/redbooks](http://ibm.com/redbooks)
- DeveloperWorks podcasts  
▪ [ibm.com/developerworks/podcast/](http://ibm.com/developerworks/podcast/)





1-140 Achieving Enterprise Application Security © 2007 IBM Corporation


## Slide 141

Presented by IBM developerWorks		
<h2>Complimentary events from developerWorks</h2>		
<h3>IBM Software Delivery Platform webcast series</h3>		
<ul style="list-style-type: none"> <li>▶ View technical webcasts covering the IBM Rational Software Delivery Platform. See what's coming up in our 2007 schedule or browse the library of more than 30 recorded titles. Topics include           <ul style="list-style-type: none"> <li>▪ Using Tivoli Federated Identity Manager to provide Identity Integration Services within an SOA</li> <li>▪ Optimizing the Role of Compliance in IT Governance Efforts</li> </ul> </li> </ul>		
<p>IBM developerWorks  Briefings</p>		
<ul style="list-style-type: none"> <li>▶ Register for complimentary live, worldwide events covering PPM, the IBM Rational SDP, SOA, Linux, WebSphere, Information Management, Integration &amp; Infrastructure, IBM Workplace, and more!</li> </ul>		
<p><a href="http://ibm.com/developerworks/offers/techbriefings/events.html">ibm.com/developerworks/offers/techbriefings/events.html</a></p>		
1-141	Achieving Enterprise Application Security	© 2007 IBM Corporation

## Slide 142

Presented by IBM developerWorks		
<h2>2007 Briefings...</h2>		IBM developerWorks 
<ul style="list-style-type: none"> <li>▪ <b>Accelerating Global Software Delivery:</b> This briefing provides an overview of the need for Business Driven Development in addressing topics including Software delivery challenges, Compliance and Monitoring. It introduces the IBM Rational Software Development Platform's team products and focuses on their value of providing the development team's capability to better manage value, maintain flexibility, and control risk and change – ½ day</li> <li>▪ <b>Achieving Enterprise Application Security:</b> This briefing covers application security end-to-end. It discusses the goals of application security with a review of basic security concepts and shows how the basics are applied to building of security stacks, to add additional layers of defense, using Java / Java EE as the programming language and environment to illustrate technologies and techniques. – ½ day</li> <li>▪ <b>Architecture, Design, and Construction using the IBM Rational Software Delivery Platform:</b> This briefing will demonstrate the latest version of the IBM Rational Software Development tools, their broad range of functionality and their use throughout the entire software development process, focusing on application modeling, design, development, coding, and testing. – ½ day</li> <li>▪ <b>Eclipse: Empowering the Universal Platform:</b> This briefing takes a deep dive into some of the most important and feature rich projects that the Eclipse community is developing. From multi-language support to plug-in development, it shows how Eclipse has evolved into a universal platform complete software development. – ½ day</li> <li>▪ <b>Effective Software Testing:</b> Focus on variety and thoroughness of IBM Rational's Software Quality solutions and best practices and demonstrates the framework and tools needed in software testing as a strategic business advantage. – ½ day</li> </ul>		
1-142	Achieving Enterprise Application Security	© 2007 IBM Corporation

Slide 143

Presented by IBM developerWorks 

## 2007 Briefings IBM developerWorks

- **IBM Middleware On Linux:** This briefing includes an overview of Linux as the premier open computing platform and the value it brings to any business. It introduces key IBM middleware products that run on Linux with a focus of products in the development space. – ½ day
- **Information on Demand Live! - Building the Next Generation of Database Applications:** A demonstration of IBM Information Management tools & services available to support rapid development of integrated solutions for managing expanding volumes of data by efficiently handling XML and relational data with ease to build agile applications faster. – 1 day
- **Open Community Tools: An Open Stack Development Platform:** This briefing explores the innovation of Open Source solutions available from IBM and illustrate the power and flexibility of IBM's Open Community development tools.– ½ day
- **Open Source Development: Tools and Open Standards:** Focus on IBM's role in "Open" source tools and Open standards – ½ day briefing/workshop
- **SOA Governance: Implementing the IBM Method:** Provides an in-depth look at Service Oriented Architecture governance, from the basics to the business case focusing on IBM Rational's tools for the designing and constructing of an SOA governance framework to span the entire application development lifecycle – ½ day

[ibm.com/developerworks/offers/techbriefings](http://ibm.com/developerworks/offers/techbriefings)

1-143 Achieving Enterprise Application Security © 2007 IBM Corporation

Slide 144

Presented by IBM developerWorks 

## IBM Rational Software Development Conference



 What keeps me **Rational**?    

*June 10-14, 2007 - Walt Disney World Swan and Dolphin Orlando, FL*

- Over 275 Sessions – 12 tracks
- 3 & 5 hour Technical Workshops
- Keynotes with industry leading experts
- Exhibit Hall showcasing complimentary product and services
- Access to IBM engineers & IBM Research
- Unlimited network opportunities
- IBM Solution Center
- Interactive Birds-of-a-Feather sessions
- Luncheon discussion tables
- Evening receptions
- Over 2,500 customers and partners

Register with this discount code and receive \$100 off your registration fee!  
**TECHBRF**

Visit [ibm.com/rational/rsdc](http://ibm.com/rational/rsdc) for more information

1-144 Achieving Enterprise Application Security © 2007 IBM Corporation

Slide 145

Presented by IBM developerWorks 

# Questions Thank You

Download a PDF of this technical presentation from  
[ibm.com/developerworks/offers/techbriefings/details/security.html](http://ibm.com/developerworks/offers/techbriefings/details/security.html)

1-146 Achieving Enterprise Application Security © 2007 IBM Corporation

## **Achieving Enterprise Application Security**

Brought to you by IBM Corporation 2007

For the latest version of these materials, visit us on the Web at

**[ibm.com/developerworks/offers/techbriefings  
/details/security.html](http://ibm.com/developerworks/offers/techbriefings/details/security.html)**