Institute of
Public Administration

COMP41530 - Web Services in Cloud Computing

Barry Corish
Associate Lecturer, IPA

Lecture 08

Institute of Public Administration | 57-61 Lansdowne Road | Dublin 4 | Ireland | Ph. +353 1 2403600 | www.ipa.ie

Overview

- Review
- Network Security
- Application Security
- WS- Standards and Security

Overview

- **Review**
- Network Security
- Application Security
- WS- Standards and Security

3 keys to good architecture

- Appropriate use of middleware
- Make the Services easy to find and use
  - One possible solution for this is UDDI – The "phone directory" for Services
- Proper Governance
  - Next Week

Service Discovery and Re-use

- WebServices may be spread around the cloud
- How to make them “discoverable”?
- Can only implement a true SOA if we can find services so they can be re-used
- If we build the same service twice because we don’t know the first instance of it exists, then we’ve failed!

Small Scale

- Easy to keep track of services at first:
  - Small numbers of services to be tracked
  - Just keep a list!
  - Rely on developer/IT Architect knowledge
  - When starting, fewer services to remember

Medium/Larger Scale

- Over time:
  - More services
  - Staff turnover, loss of knowledge
  - Likely to fail to reuse and develop again, unless tightly managed
  - They're easy to reuse – they must be easy to find!
  - We need a registry of services

Why do we do this?

- Maximise service re-use
- Allow us to manage and govern the services we have
- Particularly when growing fast!
- Provides "interfaces" to the information
- All key to successful SOA

Type of Service Discovery (1/2)

- Static Service Discovery
  - Typical pattern
  - Most common with internal provided services
  - At design/build time, do search, make selection
  - "One off" selection
  - Build the selected service into application

Type of Service Discovery (2/2)

- Dynamic Service Discovery
  - Part of the application as built is to pick the service to be used "on the fly"
  - Mainly for "utility" type services
    - Card Payment Processing
    - Computation
    - Storage
  - Pick whoever is cheap/fast/available today

UDDI

- **U**niversal **D**escription, **D**iscovery and **I**ntegration
- A standard for Service Registries
- Designed for use by Developer Tools and applications
  - Not designed for direct usage by humans

UDDI Core Elements

- White Pages
  - Details of Organisations
- Yellow Pages
  - List of Business Services provided by the Organisations
  - Based on industry "standards"
- Green Pages
  - Details of WebServices providing the Services

Types of Service Registry

- Full Public (Dead)
- Affiliate Group (Fairly Common)
- Internal (Most Common)
- Internal with external exposure (Occasional)

Why did Public UDDI fail?

- Complicated
- Hard to categorise products and services
- Hard to get paid to host a public UDDI service
- No dominant Service Registry emerged
  - Remained fragmented
- Never gained critical mass

Where has UDDI worked?

- Large, but controlled environments:
  - Government/Public/Civil Services
  - Specific B2B industry sectors
    - Logistics/Distribution
    - Electronic Components
  - Internal use within large single organisations
    - Specifically as a way to support dynamic binding!

Questions?

?

Overview

- Review
- **Network Security**
- Application Security
- WS- Standards and Security

Security Threats

- WebServces (and SOA) are great!
  - Easy to discover / explore
  - Easy to connect to
  - Open standards
  - Readily available tools
- ... this also makes them vulnerable to attack

SoA and Security

- SOA forces change in Security Architecture...
  - Before SOA, all "actors" and systems inside one security perimeter.
  - With SOA, the is possible, but less likely
- SOA systems more likely to traverse public networks

Cloud/Internet and Security

- Systems are especially vulnerable when available via the public internet
  - Even more so when hosted in a public Cloud
- Much more vulnerable to attack than "older" connection technologies
- Making them secure requires a good security architecture

Risks

- Financial damage
- Business disruption
- Theft of valuable information
- Fines and penalties
- Reputational damage (the big one!)
- Insurance for these risks (if available) is limited, and expensive

SOA and WebServices Use Cases

- Less likely to be "one-to-one"
- More likely to be “some-to-one"
- Very likely to be “many-to-one“
- We can no longer audit, assess, or even know about, every user of our systems
  - This reflects changes in how our organisations work

Internal Threats

- "..but we don't expose our systems externally, so we're fine?"
  - Trusting network security?
  - Deniability
  - Risk / audit factors
  - Malcontents
  - Thieves
  - Incompetents

Threat Types

- Broad categories:
  - Unauthorised Access
  - Unauthorised Alteration of Messages
  - Man in the middle
  - Denial of Service Attacks

Countermeasures

- Two broad types of countermeasures:
  - Network Level
    - As per websites etc.
  - Application Level
    - Specific to WebServices

Security Principles

- Paranoia
- Infinite risk, limited resources
- Strength in Depth
- Least possible information
- Block all, then allow as required

Firewalls

- Basic Functions:
  - Limiting incoming and outgoing traffic
  - By source
  - By desitination
  - By port
  - By rate/time
  - Authicated senders of incoming traffic
  - Acting as a router to VPNs etc.

Firewall Topology

- DMZ
- **D**e**m**ilitarised **Z**ones
- Transit the firewall(s) multiple times

Stateful and Stateless Firewalls

- Stateless just examines each packet on it's own
- Stateful examines the whole conversation
  - Handshaking
  - Session creation
  - Request/Reply sequence
  - Session take-down

Application Specific Firewalls

- Read deeper into the network traffic
- Understand (at some level) the application
- Harder to configure than basic "packet only" firewalls
- In our context, typically a “WebService Gateway” or “XML Gateway”

Intrusion Detection Systems (IDS)

- Sits on the network.
- Learns (or is taught) normal traffic patterns
- AI / Expert Systems approach is common
- Notifies Administrators of any significant unusual patterns in traffic

Intrusion Protection Systems (IPS)

- Variation on IDS
- As per IDS, but "shoots first, tell Admins later"
- Proactively Blocks traffic if an unusual pattern is noticed

IDS and IPS

- Both IPS and IDS vulnerable to False Positives
- Both require active and ongoing management

Vulnerability Assessment (1/2)

- Actively scanning for weaknesses in advance of any attack.
  - Penetration testing
  - Network scanning
  - Risk Assessment Exercises
  - "Fuzzers"/Automated testing
- Look for changes, not just weaknesses
  - Mark each difference found against change control document of the cause

Vulnerability Assessment (1/2)

- Perform in advance of go-live, ongoing, and especially before-and after any changes
- Get formal sign-offs from all parties!
- Risk Assessments, Penetration Testing, Vulnerability Assessments should be done:
  - formally
  - frequently
  - both internally and by a third party!

Auditing Requirements

- As well trying to prevent bad things from happening, must record:
  - Who/What/Where/When
  - What happened
  - What failed to happen
  - What we prevented
- Records must be tamperproof
  - Write once, read many (WORM) devices.

Security Requirements in Message Transmission

- Authentication - who are you?
  - Prove it...
- Message Integrity
  - Has the message been altered since it was sent?
- Message Confidentiality
  - Could the message have been read by someone other than the sender and the intended recipient?
- Non-repudiation
  - Can I later deny that it was me who sent a message?

Symmetric Encryption

- Same key both sides
  - "Shared password"
  - e.g. password to open a Word .doc
- Good:
  - Fast
  - Hard to crack
- Bad:
  - Doesn't scale!
  - Every pair/group of users need own key
  - Hard to keep keys secure

Asymmetric Encryption (1/5)

- Different keys both sides
- PKI - Public Key Infrastructure
- "Key pairs" - each participant has a public and a private key
- Anyone can have your public key
  - Only you can have your private key
  - Only have to keep your private key "private"

Asymmetric Encryption (2/5)

- Messages encrypted with one public key can only be decrypted with matching private key.
  - ...and vice versa.
- If a message can be decrypted with my public key, I must have sent it!
  - Can't be changed
  - Must have come from me
  - ...but can be read by anyone (my public key is "public").

Asymmetric Encryption (3/5)

- If you encrypt a message with my public key, only I can decrpyt it!
  - Only I have the private key to decrypt such a message.
  - ...but I don't know who sent it (my public key is "public")

Asymmetric Encryption (4/5)

- Get “total coverage” in two steps:
  - First, I encrypt the message I want to send with my private key
  - Next, I encrypt the message again with your public key
  - Then, I send you the resulting “double encrypted” message.
- When you receive the “double encrypted” message:
  - Only you have the private key to your public key, so only you can decrypt the "outer" encryption
  - You then decrypt the "inner" encryption with my public key
  - If this works, only I could have sent the message, and it hasn't been changed on the way.

Asymmetric Encryption (5/5)

- All solved!
  - ...apart from key exchange!
  - ...and "cancelling" key pairs where private key is stolen.
- Other disadvantage:
  - PKI/Asymetric key encryption/decryption tend to be slower than symmetric key equivalents
    - Calculations are more complex

Digital Certificates and “Signatures”

- Same as a PKI key pairs
  - With the addition that the keys are issued and signed by a third party
  - Third party verifies that the key pair was issued to a specific entity.
  - If you trust the third party, you also trust the entity who signed the message is who they say they are in their keys.
  - Most commonly used to protect client <-> website traffic with SSL.

Secure Sockets Layer (SSL/TLS)

- Commonly used to provide confidentiality and authentication around Websites
- Provides a secured "communications pipe", normally between a web browser and a web server.
- Can offers:
  - Server authentication
  - Client authetication
  - Assurance of Data Integrity
  - Assurance of Data Confidentiality
- Uses PKI/Digital Certificates to provide this.
- Can use some/all of these in protecting our WebServices

Overview

- Review
- Network Security
- **Application Security**
- WS- Standards and Security

So far, so what?

- All of these techniques are commonly used in securing other web based services!

Authentication

- Prove it.
  - Simple: Username and password(s)
  - More complex: 2-factor
- Simple on a single system (e.g. a WebSite)
  - More complex on highly distributed systems
  - Do we flow authentication through all participant systems? How?
  - Do we flow credentials through all participant systems?
  - Do we have a "session" concept, or do we re-authenticate with each request?

Authorisation

- I know who you are.
  - What can I let you do?
- Based on authorisation policies
  - Actions limited to groups of users
  - Number of actions limited by time/rate
  - Actions limited by value.
  - Time of day/day of week etc.

Protection domains (1/2)

- Often, group several systems together into a "protection domains"
- Authenticate to one, you're authenticated to all.
- AKA "Single Sign On" (SSO)
- Authenticate once.
- On authentication, you are given a digital "ticket" to confirm that you've already authenticated
- Multiple systems will accept a valid ticket without requiring re-authentication.

Protection domains (2/2)

- Ticket is normally:
  - Time limited
  - Source IP address limited?
- You present the ticket with each request
- Common standards for tickets are:
  - Kerberos
  - X509
  - LTPA (IBM Tivoli/WebSphere product specific)
  - SAML (Open Standard)

Security Assertions Markup Language (SAML)

- Allows systems to share information
  - Securely
  - Secretly (if required)
- Commonly used to share information that a “user” has already authenticated successfully
  - Equivalent to a "ticket" in the "Protection Domain"

Directory Services

- Use a directory to store/access details of users
- Including details of:
  - Usernames / Passwords
  - PKI Public keys
  - Certificate Status (revoked etc.)
  - Group memberships
  - Attributes
  - Roles
- Directory must be secure!

Lightweight Directory Access Protocol

- LDAP
- Common directory standard
  - Plain Text and SSL support
  - Open standard
  - Many vendors
  - Widely Supported
- Most implementations are an LDAP interface to another Directory
  - There are some "pure" LDAP directories

Proprietary Directories

- Common Proprietary Directories:
  - Microsoft Active Directory ("AD")
  - IBM Tivoli Systems
  - RACF
  - Novell eDirectory
  - Operating System internal user registry
  - …and lots of others..
- Most provide the capability to act as an LDAP server
- Some provide the possibility to act as an LDAP client

XML Security Standards - XML Signature

- Uses PKI to produce a "hash" of part or all of an XML message
- Hash is sent with the message
- Only the private key holder can produce this hash
- Receiver can see that the sender did sign the message, and the message has not been changed.
- Message is in the clear - only proves who sent it and not altered.

XML Security Standards - XML Encryption

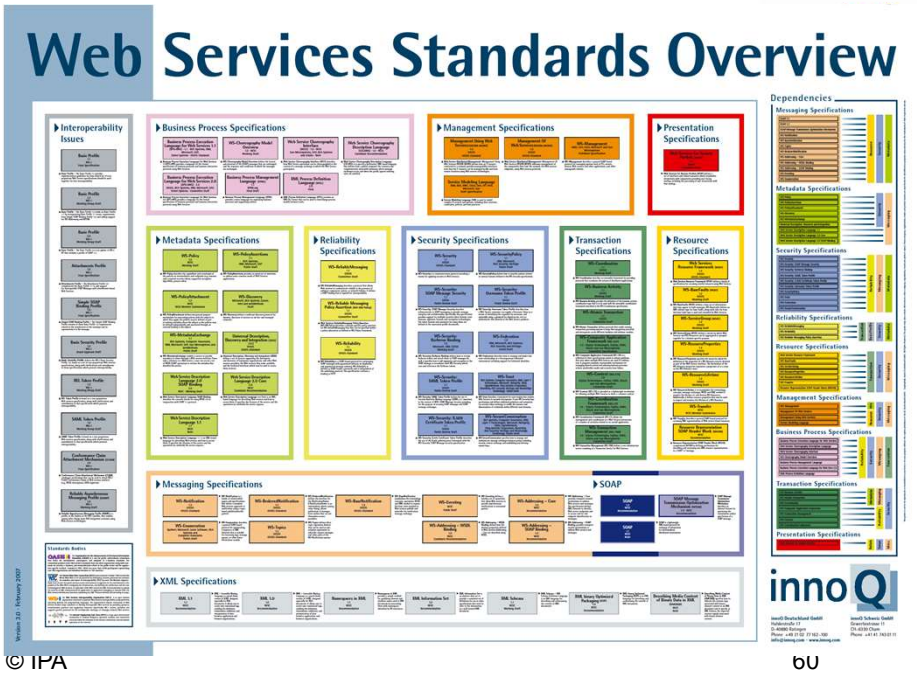
- Uses PKI to encrypt part or all of an XML message
- Produces more XML!
- Usual PKI Rules apply
- Can be done "twice", to sign and then encrypt, as per SSL
- Can be used in conjunction with XML Signature

Overview

- Review
- Network Security
- Application Security
- **WS- Standards and Security**

WS-Policies

- An XML schema
- Defines how web services can include information about their policies
- Policies include security related information



Main “Security” WS- Standards

- WS-Policy
- WS-Security
- WS-Security Policy
- WS-Trust
- WS-SecureConversation
- WS-Federation
- …all built on top of SOAP

WS-Policy

- Describes how a web service can define and advertise it’s polices
- Covers many areas, not just security, e.g:
  - Quality of Service
  - Availability
  - Charging
  - etc.

WS-Security

- Defines a set of SOAP extensions
- Defines how to implement message content integrity and confidentiality
- Doesn't define encryption algorithm etc., just how to implement them in messages
  - Not defining specific algorithims makes this is a little more future proof

WS-SecurityPolicy

- Extends WS-Security
- Binds WS-Security to WS-Policy
- i.e. how to define WS-Security features in an overall WS-Policy

WS-Trust

- Extends WS-Security
- Defines how to request and issue security tokens
- Defines how to pass tokens in messages
- Defines how to build trust relationships between participating parties

WS-SecureConversation

- Extends WS-Security
- Defines how to set up per-session keys to allow “SSL-type” encryption of the messages

WS-Federation

- Extends WS-Security
- Defines how to pass security information between participating parties

Critical Points

- Most fraud is internal fraud
- Build in security from the start
  - Don't add later as an afterthought
- Security is hard
- Security costs money
- **Do not build your own security!**
  - Use established standards and products

Overview

- Review
- Network Security
- Application Security
- WS- Standards and Security