Institute of
Public Administration

COMP41530 - Web Services in Cloud Computing

Barry Corish
Associate Lecturer, IPA

Lecture 09

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

Overview

- Review
- SoA Development Livecycle

Overview

- **Review**
- SoA Development Livecycle

Security Threats

- WebServices (and SOA) are great!
  - Easy to discover / explore
  - Easy to connect to
  - Open standards
  - Readily available tools
- ... this also makes them vulnerable to attack
- Particularly when hosted in the cloud!

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

Firewalls

- Basic Functions:
  - Limiting incoming and outgoing traffic
  - By source
  - By destination
  - By port
  - By rate/time
  - Authenticated senders of incoming traffic
  - Acting as an endpoint of VPNs etc.

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

Vulnerability Assessment (1/2)

- Actively scanning for weaknesses in advance of any attack.
  - Penetration testing
  - Network scanning
  - Risk Assessment Exercises
  - "Fuzzers“/Automated testing

Vulnerability Assessment (2/2)

- 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!

Security Requirements in Message Transmission

- Authentication - who are you?
- Message Integrity
- Message Confidentiality
- Non-repudiation

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

- 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"

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)

- 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 authentication
  - 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

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.

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!

Main “Security” WS- Standards

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

Overview

- Review
- **SoA Development Livecycle**

SOA Pitfalls

- Taking on too much – giant “design” first
- Starting with the wrong business processes
- Cluttered – building same “old” mess in SOA Services
- Making many services that turn out not to be reusable!
- Adding a SoA skin over broken existing systems
- Adding a SoA skin over broken business processes
- Lack of governance, road map etc.

SOA Management Pitfalls

- No actual commitment to SOA
  - From IT and/or Business management
- Focused on technologies, not on architecture
  - Trying to beat all business requirements into one technology
- Over promised benefits, under emphasised costs
- Lack of testing / support infrastructure

SOA Development Livecycle

- How to start? First questions:
  - What are the core business processes?
  - What services form part of these processes?
  - Which of these services will add immediate value?
  - How can these services be assembled into useful business processed?

Broken Business Processes?

- SoA is not just about technology
  - Or even mainly about technology
- Must have sound underlying business processes first
- Perhaps we need Business Process Re-engineering before we look at SoA?

Software Development Livecycle

- Process of designing, building and implementing software systems:
  - Analysis
  - Design
  - Implementation
  - Maintenance

SDL Models - Waterfall

- Simple sequential run through:
  - Analysis
  - Design
  - Implementation
  - Testing
  - Maintenance
- Rarely to see “Pure” implementation of this in practice

SDL Models: Spiral

- Repeated, iterative Waterfall
- Keep going back around the loop
- Expect requirements to change each time around
- When are you finished?
  - What is “good enough”?

SoA and Object Oriented Development

- SoA and OO both emphasise breaking the problem down into units (modules/services)
- Emphasise keeping clear interfaces between units, with “hidden” implementations
- OO works at a low level (individual class interfaces) – SOA at a higher level (individual service)
- SOA and OO are compatible, but not the same

Process Modelling

- To produce consistent, reusable, extensible services, the must be consistently described
  - This is normally done with a modelling system
  - UML is commonly used
  - This is normally done by Business Analysts or by the end users themselves - the “business experts”
  - Many development tools can “feed” directly from the models, and build a “scaffold”, which is then implemented in detail by the developers

SoA Control and Visibility

- Each service should have an associated SLA
- It must be possible to see if the SLA is being met
  - Implies MIS / Alerting etc.
  - These should be included in the Service from design stage onwards, not added in at the end

SoA Governance is key!

- Operational Governance
  - Is the system/service doing what it should do, i.e. meeting it's SLA?
- Livecycle Governance
  - Is the system/service meeting it's objectives over it's lifetime?
- Organisational Governance
  - Is the system/service providing what the Organisation needs?

SoA Development Principles

- Reuse existing functionality
- Minimise rework of existing systems
- Allow and support incremental integration
- Maximise flexibility
- Scalability
- Operational visibility and control

Approaches to SoA

- Top Down
  - Start with the Organisational Mission Statement
  - Work “down” towards the detail
- Bottom up
  - Start with a specific system, problem or requirement
  - Work back up to a design
- Hybrid
  - Working both Top Down and Bottom Up at the same time
  - Most common approach

Main Steps in SOA Development

- Plan
- Analyse
- Design
- Build
- Test
- Provision
- Deploy
- Manage
- Govern

Planning

- Strategic planning level
- Look at business needs
- Look at current technologies
- Design future SoA architecture
- Perform Cost Benefit Analysis
- Document approach to implementing SoA
  - Which processes first
  - Short, medium and long term targets
  - …and systems/services/processes to be excluded

Analysis

- "Next level down" after planning
- Identify process-by-process targets for SoA implementation, including sequencing
- May include changing the business processes
- Steps:
  - Identify "as-is" physical processes
  - Extract to "as-is" logical processes
  - Plan "to-be" logical processes

Design

- Design service interfaces first
- Design for service granularity, loose coupling, reusability, and to meet standards
- Build process models, down to individual field levels

Build

- Understand existing applications
- Simplify and reduce business logic
- Identify underlying business rules
- Wrap legacy systems to be SoA compliant
- Create and build the service interfaces

Test

- Should be testing throughout the Build phase (Unit testing)
- Dynamic testing
- Interface testing
- Assembly testing
- Performance testing

Provision

- Build the final “physical” systems
- Finalise the SLA
- Assemble the services to provide a business process
- Arrange/assemble the business to make use of the business process provided
- Set up MIS / monitoring etc.

Deploy

- Roll out and advertise the new service to potential clients.
- Publish the WSDL
- Publish the SLA and other supporting documents

Manage

- Watch it!
- Monitor performance, usage, problems etc
- Gather this information over time, and store
- Check conformance to SLA/QoS etc
- Continue to test yourself!

Govern

- Based on information from “Manage” check that the service is supporting the business requirement
- Regularly reassess the business requirement
- Remember: Other people now have a say in how you run your systems!

Key points

- SoA is as much a Business methodology as a set of technologies
- SoA must be implemented within a plan and a methodology to avoid chaos
- Governance is critical!
- SoA != WebServices