SOS: Service Oriented Security

OWASP Europe 2005
Security Briefing by Arctec Group
(www.arctecgroup.net)
Risk Management

“We have no future because our present is too volatile. We have only risk management. The spinning of the given moment's scenarios. Pattern recognition...”
-William Gibson “Pattern Recognition”
Focus

• Fundamental questions:
  • What Service Oriented Architecture (SOA)?
  • What are the security risks and threats in using SOA?
  • What security and risk management options do I have?
  • How can security personnel collaborate with architecture and development, and operations to improve the security in the end product?
  • How do I architect security for SOA?
SOA Usage

- SOA is more than just Web Services
  - This session primarily focuses on Web Services, b/c they are the most mature and well adopted SOA paradigm today
- How did we get here?
  - Web Services are an evolutionary step; evolving from component based systems like EJB and DCOM
- Not just another technology, but the next wave in networked computing
  - All major platforms and 3rd packaged app vendors are going in this direction:
    - Java/J2EE, .Net, SAP, Oracle
- Why Web Services?
  - Interoperability, simplified programming model for networked programs, loose coupling
Web Services Usage

• High Level Web Services Use Case Model View for Trading partners
Evolution to SOA

- Evolutionary steps and security implications
  - Object oriented programming (Java, C++)
    - Abstract models (objects) used to bundle data and methods
    - Security implication: vendor or implementation provides security mechanisms for single process
  - Component based programming (J2EE, COM)
    - Components represent an implementation model for distributed programming (typically vendor specific)
    - Security implication: model provides implementation specific security models (example: container security in EJB)
  - SOA (Web Services)
Evolution to SOA Part II

• Evolutionary steps and security implications (continued)
  • SOA (Web Services)
    • Client and services are decoupled
    • Multiple technologies can be used (Java, .Net)
  • Security implications:
    • Fundamentally different threat model consisting of normal web app-style threats and “asynchronous” type threats
    • Positive trend: current and emerging standards to deal with threat model in a standards-based, cross platform manner with vendor supported tools
Software Development Realities

- Blame Game
Software Development: The Way Forward

- Architecture & a shared understanding of risk - it's our problem
Web Services Architecture

- Semantic Layer
- Security, Workflow, Routing
  - SAML, WS-Security
  - WSDL, UDDI
  - SOAP, XML-RPC
  - XML
  - HTTP, HTTPS, FTP, SMTP

- Domain specific
- Emerging standards/solutions
- Standards-based
WS-* Mumble, mumble

- Current and proposed standards: ASAP (long running transaction support), BPEL4WS (Business Process Execution Language), DIME (Direct Internet Message Encapsulation), ebSOA TC (Electronic Business Service Oriented Architecture), ebXML (Electronic Business XML), FWSI TC (Framework for Web Services Implementation), HTTPR (Reliable delivery of HTTP packets), oBIX TC (open Building Information Exchange), SAML (Security Assertion Markup Language), SOAP (Simple Object Access Protocol), SOAP MTOM (SOAP Message Transmission Optimization Mechanism), Translation WS TC (Automating the translation and localization process), UDDI (Universal Description, Discovery, and Integration), WS-Addressing (enables messaging systems to support message transmission in a transport-neutral manner), WS-AtomicTransaction (provides the definition of the atomic transaction coordination type that is to be used with the extensible coordination framework described in the WS-Coordination specification), WS-Attachments (Superseded by SOAP MTOM), WSBPEL (Business Process Execution Language), WS-CAF (WS Composite Application Framework), WS-CF (WS Coordination Framework), WS-Choreography (Working Group created to address the ability to compose and describe the relationships between Web services), WS-CDL (Web Services Choreography Description Language), WS-Coordination (Describes an extensible framework for providing protocols that coordinate the actions of distributed applications), WS-CTX (WS Context), WS-Discovery (Web Services Dynamic Discovery), WSDL (WS Description Language), WSDM (WS Distributed Management), WS-Enumeration (describes a SOAP-based protocol for enumerating a sequence of XML elements), WS-Eventing (describes how to construct an event-oriented message exchange pattern using WS-Addressing concepts), WS-Federation (Web Services Federation Language), WSIL (WS Inspection Language - WS-Inspection), WS-Manageability (set of capabilities for discovering the existence, availability, health, performance, and usage), WS-MetadataExchange, WS-Messaging, WS-Notification, WS-Policy, WS-Provisioning, WS-Reliability, WS Reliable Messaging, WS-RF, WS-ReliableMessaging, WS-Routing, WSRP, WS-Security, WS-SecurityPolicy, WS-TM (WS Transaction Management), WS-Transaction, WS-Transfer, WS-Trust, XML-Encryption, XML-Signature (Source: http://roadmap.cbdiforum.com/reports/protocols/summary.php)
Web Services Security

• Some security standards we have to work with today
  • SSL:
    • Allows encryption of communication channel for point to point only
    • Not a complete solution for sensitive robust, complex (multi-hop) Web Services applications
  • SAML
    • Subject’s authentication, attribute, and authorization assertions
  • WS-Security
    • Container solution encompassing X.509 Digital Signatures, Username, Password, XML encryption and XML Digital Signature. Provides end to end message confidentiality and integrity for complex applications that have messages traversing multiple hops
  • XACML:
    • Defines element level authorization in XML
  • XML Signature and XML Encryption
Web Services Risks

• Why we are not necessarily in a worse security position than we were with components like DCOM and EJB:
  • Firewall ports were/are being opened for DCOM, RMI-IIOP anyhow
  • Structured data is better than unstructured data
  • Simpler is better for security
  • Interoperable standards
• Software Security Risk Management
  • Trend is towards security involvement in development
  • Developing Web Services Threat Model
  • Use standards to compose security architecture
  • Utilize security as a system service
Web Services Threat Model

- Why is Web Services security hard?
  - Web Services combines security problems from two different paradigms
    - Normal Web Application threats apply
    - Plus additional “Asynchronous” application world
      - Securely persisting state over long running or multi hop transactions
      - No knowledge of client environment
    - Inherits attacks from all the standards and services in Web Services technology stack, and emergent combinations of technologies, example: XDoS
  - Remember the client
  - Revisit design assumptions, I.e. how does principle of complete mediation apply in SOA?
Managing SOA Complexity

• Build up shared understanding of risk and countermeasures
• Utilize architectural views to separate concerns, understand key elements, constraints, and relationships
• Holistic model to base design upon
• Views contain domain specific:
  • Elements
  • Threats
  • Countermeasures
  • Protection, detection, and response
• Views have relationships and dependencies with other views
• Identify patterns in threats (example: OWASP Top Ten), principles (Saltzer and Schroeder’s Principles*), and solutions (standards Principle of Open Design*)
SOS: Service Oriented Security Architectural Views
SOS: Service Oriented Security Architectural Views High Level Definitions

- Identity View
  - deals with the claims made about an identity, the identity, federated identity, and identity management
- Service View
  - deals with the service itself its logical organization and component parts
- Message View
  - deals with persistent data/ service’s message payload
- Deployment View
  - deals with the logical and physical administrative and runtime deployment environment and trust zones
- Transaction Use Case Lifecycle View
  - deals with the behavioral flow of the application and its actors from an end to end perspective
Identity View

• Concerned with authentication, authorization, and principal’s unique security information. SOA implies a multi-centered world: Federated identity is central to most SOA and SOS designs

• Elements
  • Security principals
  • Evidence or set of claims
  • Identity and identity management is a mixture of process, policy, and integration

• Example Threats and countermeasures
  • Spoofing Identity

• Patterns and principles
  • Strong, portable identity through federated identity standards
  • Principle of Open Design(*)
Federated Identity Pattern

• Context: Manufacturer and supplier want to integrate disparate systems with unique policies and management

• Problem: User credentials must be securely ported across domains and security information must be recognizable to both parties

• Solution: Use federated identity for SSO. Client logs onto local system, receives/sends encrypted SAML token to the Web Service. Web Service validates assertions for authentication and authorization.
Service View

• Key Elements:
  • Web Service
  • Security model includes: Access control model, audit log
• Example Threats & Countermeasures
  • See OWASP Top Ten (Input validation, et. al.) -- all of them apply here
• Principles & Patterns:
  • Principle of Least Privilege, Fail Safe Defaults (*)
  • Security as a service
  • Security Pipeline interface
Security Pipeline Interface Pattern

• **Context:** host must mediate activity between remote client system and back end resources

• **Problem:** host system cannot trust incoming requests and data

• **Solution:** Use Security Pipeline Interface (1) used to enforce the principle of Separation of Privilege and reduce risk. SPI Runs in separate process and memory space
Message View

- Elements
  - Persistent data
    - Header, tokens
  - Audit log
- Example Threats
  - Integrity threats
  - Information disclosure
  - Threats not limited to a certain session
- Principles and Patterns
  - WSS for message integrity and confidentiality
Message View Pattern

- Encrypting message payload with WSS
- SOAP Only
Deployment View

• Elements
  • Logical and physical entities: nodes, firewalls, IDS, servers, Directory services, et. al…
  • Logical Trust zones
• Example Threats
  • Attacks on the environment that the service executes in
• Principles & Patterns
  • Trust zones
  • Defense in depth
  • System logs, many others present in security architecture
Deployment View Pattern

• Trust Zones

  • Do not architect using concepts like trusted vs. untrusted. Instead use levels to separate untrusted vs. more trusted

  • Zone definition: Define trust zones based on risk versus available protection, detection, and response mechanisms deployed in that zone

  • Input/external to zones: Protect information sources that give information on what the Web Services expects, such as WSDL, from attackers. Trust only what you can validate and verify requests and input data

  • Consider Honeypots for understanding the threat profile of each security zone to vet trust zone assumptions.
Deployment View Pattern

- Trust Zones
Transaction Use Case Lifecycle View

• Elements
  • Use Cases (or User Stories and requirements are a rich data set)
  • Usability Constraints
  • Actors
• Example Threat Analysis
  • Can illuminate logical flaws in order of execution/assumptions
• Principles and Patterns
  • Synthetic not analytic
  • Principle of Complete Mediation (*)
  • Principle of Psychological Acceptability (*)
Transaction Use Case Lifecycle View Pattern

- Includes and extends in Use Case Modeling
Abuse Case Pattern
SOS: Service Oriented Security Architectural Views
Building Web Services & SOAs

• Security in the Software Development Process
  • To communicate its message precisely: Security must learn to speak the “language of the tribe”
  • Use Threat Models and Abuse Cases to drive security analysis, design and testing. Suggestions on how to do this at: www.arctecgroup.net/articles.htm
• Use architectural models to navigate risk management tradeoffs and communicate design decisions
• Understand Web Services features and limitations with regard to security and articulate security concerns
• Help developers and architects understand and find failure modes and points
• Use OWASP to educate team - power of standards
Building Web Services & SOAs

• Implement Saltzer and Schroeder’s Principles*
  • Principle of Least Privilege
  • Principle of Fail Safe Defaults
  • Principle of Economy of Mechanism
  • Principle of Complete Mediation
  • Principle of Open Design
  • Principle of Separation of Privilege
  • Principle of Least Common Mechanism
  • Principle of Psychological Acceptability

References & Future Directions

• Track Web Services and Identity-related Standards and implementations
• Participate in the development process
• References & Resources:
  • OWASP
  • Saltzer & Schroeder (*“The Protection of Information in Computer Systems”, Saltzer and Schroeder, Proceedings of the IEEE, 1975)
  • Kim Cameron’s “Laws of Identity”
  • Blog (more SOS information): 1raindrop.typepad.com
  • Free architectural newsletter at: www.arctecgroup.net/views.htm
Wrap Up

• Thank you for your time
• Questions?