Jericho Challenge Architecture

By Gunnar Peterson, CTO, Arctec Group

"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*"

Problem Statement

The primary security mechanisms deployed today rely on notions of perimeters and centralized security models, however the nature of business is moving rapidly towards decentralized "intertwingled-ness" (non-hierarchical connectivity). Malicious attackers exploit the gaps left between the existing security mechanisms deployed based on outmoded assumptions and the reality of the threats to the connected systems on the ground.

Solution

In keeping with current state of the art in the software development world around Service Oriented Architecture (SOA), the proposed solution includes viewing security as service that is decoupled and composeable. In keeping with Jericho's principles, Service Oriented Security architecture does not focus on perimeters, but rather provides a framework to analyze and design for key information security concerns around risk management, identity, data, and so on.

Service Oriented Security Architecture Overview

In software architecture, the word "security" can often do more harm than good. Frequently, stakeholders have differing, conflicting, and overloaded definitions of the term. In order to build a coherent system, the architects must provide specific guidance to the development and operational teams. Service Oriented Security (SOS) Architecture provides a set of software architecture viewpoints that allow security architects to construct a holistic system design based on a set of views. Since security is not a zero sum game, the views provide a framework to conduct security architecture tradeoff analysis and to convey design decisions to development and operational staff. As Kruchten observed [Kruchten95], views enable the software architect to separate concerns in a complex system. The views in Service Oriented Security consist of the following:

* **Identity View**: deals with the claims made about an identity, the identity itself, federated identity, and identity management and services

* **Service View**: deals with the threats and countermeasures for the service, methods and component parts

* **Message View**: deals with threats and countermeasures for the persistent data/service's

message payload

* **Deployment View**: deals with “classic” information security concerns such as the logical and physical administrative and runtime deployment environments

* **Transaction Use Case Lifecycle View**: deals with the key behavioral flows and relationships in a system and its actors from an end-to-end perspective

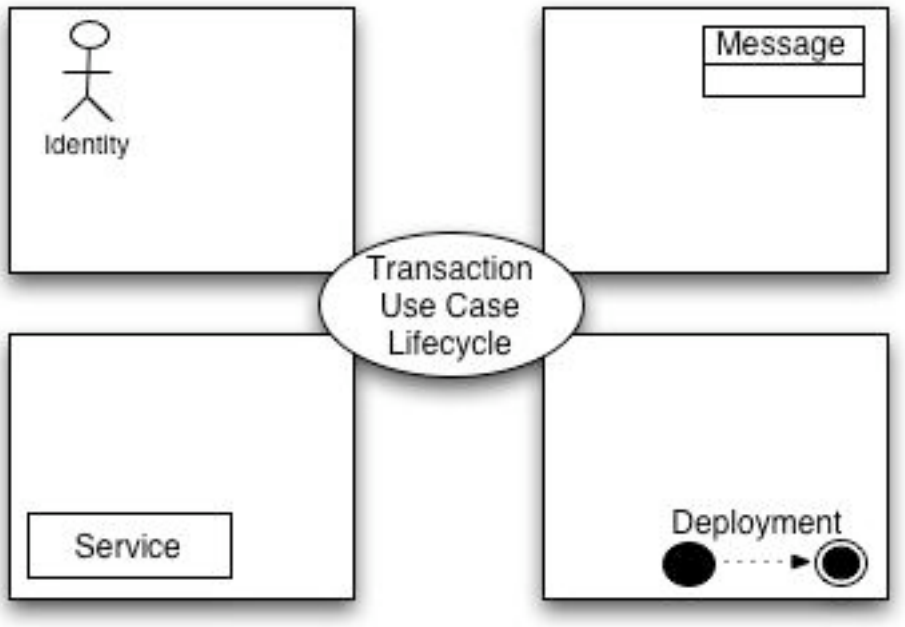

Figure 1 – Service Oriented Security Views

Each of the individual views is composed of domain specific elements, constraints, threats, and countermeasures. Each view also includes a set of key architectural patterns and principles. By partitioning concerns security designers are able to first decouple concerns to analyze security domains and analyze tradeoffs and dependencies among the domains. The resultant architecture takes the concerns from each domain into account and provides holistic solution based on the risk management of digital assets like identities and data. The following sections examine each SOS viewpoint in more detail and provide an example usage.

Identity View

Using Kim Cameron’s definition we will examine identity as “a set of claims made by one digital subject about itself or another digital subject." [Cameron05]. This definition rewards careful study because it reveals that identity is not a passive entity, but rather the result of an active set of processes that can be judged against a dynamic set of criteria. Key constituents of the Identity View include:

- Authentication mechanisms, events, and principals including Kerberos tickets, X.509, Windows sessions, and web server sessions
- Identity Federations including the portable, strong identities like SAML, Liberty, and WS-Federation identities
- Monitoring and audit systems: provide traceability of identity-related events like authentication

Identity View Pattern: Federated Identity

- 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.

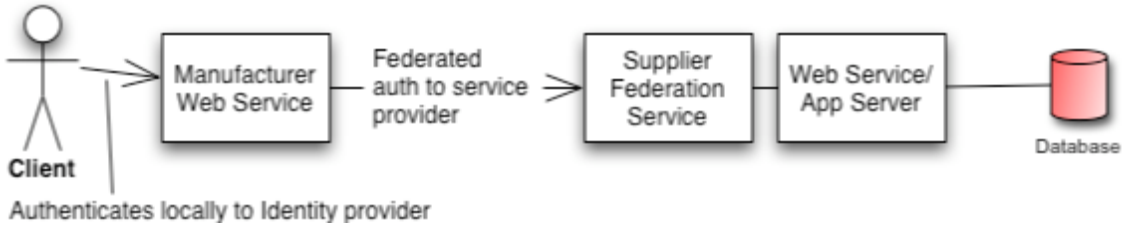

Figure 2: Using Identity Federation

Service View

The Service View is concerned with the security around the service and the service's ability to broker information flows with requisite confidentiality and integrity. Services require access control protection and may consume federated identities from other security domains using cryptographic protocols for verification. From a detection and response standpoint, services require logging mechanisms to vouch for the health of the system. Standard technology specific service hardening and security guidelines apply in the services view, such as the OWASP guidelines for Web Applications.

Service View Pattern: Security Pipeline Interface

- 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 (SPI) [Hoffman90] to enforce the principle of Separation of Privilege [Saltzer75] and reduce risk of data integrity threats. Run SPI in separate physical, process and memory space from business logic aware services. Execute logging of access control and related security event at the SPI.

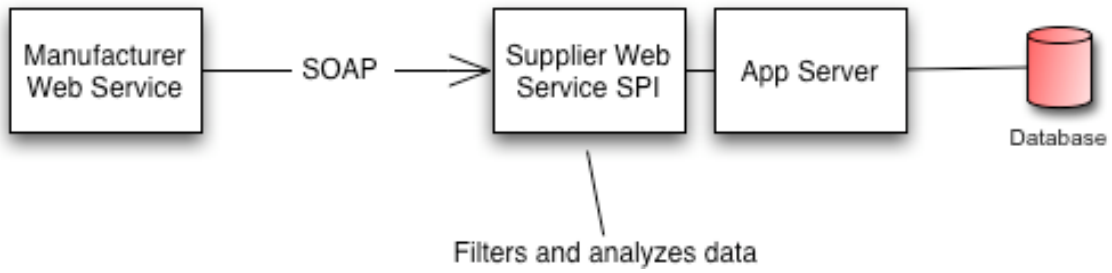

Figure 3: Security Pipeline Interface provides data integrity and logging

Message View

Information security is concerned with protecting valuable digital assets, in many cases the most valuable assets from a risk management perspective is not network access, but rather the company's data. As distributed systems continue to evolve and become more connected to each other in ways not foreseen by their original designers, such as decades old legacy systems being connected to the web, data and messages emerge in ways not intended when their protection mechanisms were implemented. The net result of this evolution is to move security mechanisms closer to the asset level, in this case the data elements. Encryption and related technology standards are used to constrain access to persistent data while it is at rest and ensure integrity and auditability over its lifecycle.

Message View Pattern: WS-Security

- Context: data is increasingly shared across technological and policy boundaries
- Problem: message must be protected beyond the span of control of the service and systems, since it can traverse multiple domains. How does the principle of complete mediation [Saltzer75] apply in a "fire and forget" service oriented world?
- Solution: Use WS-Security standard to sign and encrypt persistent XML documents. WS-Security uses XML Encryption and XML Signature, and can accept tokens such as SAML, Kerberos, and X.509 to provide assurance through authentication, authorization, and validation

Deployment Environment View

The Deployment Environment view is focused on the classic information security considerations such as:

- Firewalls
- Host based and network intrusion detection systems
- Directory services

The Deployment Environment View articulates these concerns and their relationships to the rest of the Information Security picture.

*Deployment Environment View **Anti**-Patterns: Trusted Versus Untrusted Considered Harmful*

- Do not architect using dualistic concepts like "trusted versus untrusted" deperimeterization renders these definitions meaningless. Instead focus on verification based on available protection, detection, and response mechanisms.
- Use Honeypots for understanding the actual threat profile on the ground of each domain to vet trust zone assumptions. Develop metrics and reporting to feed forward into future security designs.

Transaction Use Case Lifecycle View

Use Cases are used to show the end-to-end view of the system. Use cases provide a synthetic model that correlates requirements from different domains' concerns into a coherent model and flow. Use Case models contain many properties that are critical to secure system design:

- *Stakeholders:* In Information Security, it pays to find allies who have a vested interest in system security. Stakeholders who may be concerned about security implications in the system that is being built include not just the core development staff, but also the legal staff, business owners, domain experts, operational staff, customers, shareholders, and users.
- *Pre and Post Conditions:* Pre and post conditions describe the set conditions that must be satisfied for the Use Case to execute (Pre-conditions) and the set of states that the system can be in after it has completed (Post-conditions). Pre-conditions allow the Information Security team to articulate the security conditions, such as authentication and authorization processes that must be completed before accessing the Use Case functionality so that developers have a consistent spec to build from.
- *Exceptional and Alternate Flows:* A fundamental principle in security design is to design for failure. Development projects are mainly focused on base flows of the system since these implement business valuable features. However from a security standpoint, exceptional and alternate flows highlight paths that often become attack vectors. These flows are worth examination by Information Security to ensure that the system is designed to deal with these exceptions and has deployed security mechanisms such as audit logs and IDS tools to catch security exceptions when they occur.
- *Actors:* Actors can include computer systems, users, and other resources like schedulers. By analyzing the actors involved in the Use Case model, the Information Security team can begin to build a picture of the access control structures such as roles and groups that may be required for design as the system is built. Where delegation or impersonation is used, actors can identify where this is accomplished and what actors are mapped onto other actors at runtime.
- *Relationships:* Much of the power in the Use Case model comes from its simplicity. Use Case models feature two types of relationships: includes and

extends. These have direct security implications, in the includes relationship outcome changes the base flow of the Use Case. In the case of including an access control Use Case like Authenticate User, the outcome of this (usually boolean pass/fail) directly changes the behaviors of the related use case. The extends relationship does not alter behavior of the preceding use case, so if a use case extends to a monitor event use case and that monitor server is down, it may not make sense to alter the flow of the preceding Use Case.

- *Mapping Use Cases to Threat Models:* Security cannot only focus on functional requirements, but must also consider the attacker viewpoint. Threat modeling and abuse cases are techniques used in the development lifecycle to map possible threats, vulnerabilities, and impacts onto the system so that appropriate security countermeasures can be built into the system. The Use Case model allows the Threat Model to refer to functions in a context-sensitive manner.

Architecture Issues

The SOS views describe a way of seeing security architecture across a complex system to make and convey security design decisions. The software security space contains issues that are still being worked to achieve optimal effectiveness.

XML Security

Research has shown various flaws with XML Security [Gutmann05] related to its reliance on XML for encryption and signature as well as replicating a number of problems in legacy technologies. Since a large number of emerging security solutions, particularly WS-* rely on XML security mechanisms it is worth revisting this dependency to see if XMPP or other technology can remedy these issues.

Emerging toolsets and standards

The software security space is evolving at a rapid pace, investment paths are not clear in a long term sense. Deploying resources based on today's assumptions about standards, for example SAML vs. WS-*, implementation, and threat models inserts a higher degree of variability into the system's longevity based on the outcomes of the technical and standards challenges.

Changing Threat Landscape: Attacker Co-evolution

As with any security design, the opponent is *homo sapiens* meaning that the opponent is ever adaptable and resourceful. As security designs become more robust, then business deploy more resources and transactions to the online world, thus attracting more attackers.

References

Cameron05 "*What is a digital identity?"*, Kim Cameron, http://www.identityblog.com/2005/03/07.html#a152, 2005

Gutmann05 "*Why XML Security is Broken",* Peter Gutmann, http://www.cs.auckland.ac.nz/~pgut001/pubs/xmlsec.txt, 2005

Hoffman90 *"Security Pipeline Interface"*, Hoffman and Davis, Proceedings of the Sixth Annual Computer Security Applications Conference, 1990

Kruchten95 *"Architectural Blueprints – The "4+1" View Model of Software Architecture,* Philippe Kruchten, IEEE Software, 1995

Saltzer75 *"The Protection of Information in Computer Systems"*, Saltzer and Schroeder, Proceedings of the IEEE, 1975)