Editors: John Steven, jsteven@cigital.com
Gunnar Peterson, gunnar@arctecgroup.net
Deborah A. Frincke, deborah.frincke@pnl.gov

Service-Oriented Security Indications for Use

GUNNAR PETERSON
Arctec Group

In evolutionary terms, the information security field is more than a decade behind software development. By that, I mean that we haven't had a single meaningful change in security architecture in 13 years. Developers have evolved, businesses have increasingly bet their entire business models on the Web and networks, and both sides have increased their security budgets. But what has the security architecture (as it's deployed in the field) got to show for all of this? More firewalls and more Secure Sockets Layer (SSL) connections.

Why has information security failed? I think the problem lies with its mission—confidentiality, integrity, and availability are fine statements to make, but they don't lead anywhere. Because information security has proven incapable of evolving, it's time to learn from a discipline that has mastered innovation—software development. In this installment of Building Security In, we'll learn what this field can teach us.

Diagnosis

Software developers began building sophisticated Web applications in the mid 1990s, using CGI and Perl scripts to connect their users to databases and back-end content. Even back then, security people knew immediately that security would be an issue—after all, developers were publishing back-end content from their core business databases and applications onto the Web and letting users post content there as well. In response, the security industry moved as never before or since to build and deploy two security mechanisms. The first was a network firewall to keep the "good stuff" (enterprise data and functionality) separate from the "bad stuff" (the Internet). The second mechanism was the SSL to encrypt the link from the user's Web browser to, ideally, the Web server.

What happened next was the dotcom boom—businesses figured out that they could make buckets of money on the Web, developers began innovating feverishly, Web applications became more sophisticated and personalized, and so on. This led to Java's Java Server Pages (JSPs), Microsoft's Active Server Pages (ASPs), and even greasier Perl scripts, all in an effort to pool enterprise resources and personalized sessions on Web servers. The security people defended this revolutionary new application programming model with their original security architecture—network firewalls and SSL.

Around 1998, developers began building increasingly distributed three-tier applications that separated the business logic, presentation, and data access layers. Among other things, a Web application could now seamlessly integrate data from multiple back-end systems—if you had pricing data in Oracle, order data in SAP, and customer data in a mainframe, for example, you could write separate data access objects, apply business logic in the middle tier, and tie it all together in a friendly user interface. At this point, Web applications began to integrate across departments, business units, and geographic boundaries, with huge critical chunks of the business now connected to the Web. How did the security people defend this vertically and horizontally integrated business architecture? They applied the same exact 1995 security architecture—network firewalls and SSL.

In the 1999 to 2000 time frame, businesses started to rely on Web applications for major parts of their revenue. Software developers responded by building applications in different technologies because the customer didn't care (still doesn't)—the customer wanted (still wants) data access and functionality. To integrate these disparate technologies, developers deployed SOAP and XML so that Microsoft could talk to Java and Websphere could talk to Weblogic and so on. Moreover, developers found they could use SOAP and XML to connect business-to-business networks so that partners in a supply chain or business process could exchange data and interoperate. SOAP and XML presented a fundamentally new programming model, but neither one had a security model by default for authentication, authorization, or confidentiality. How did

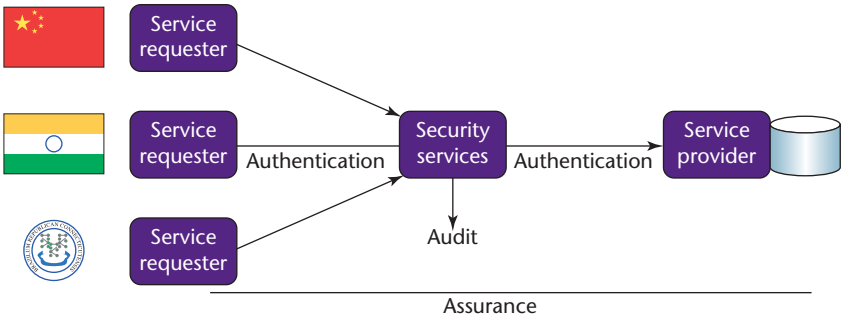

Figure 1. Virtualized service interfaces. By decoupling authentication and authorization, such decisions can be delivered to different locales in the architecture.

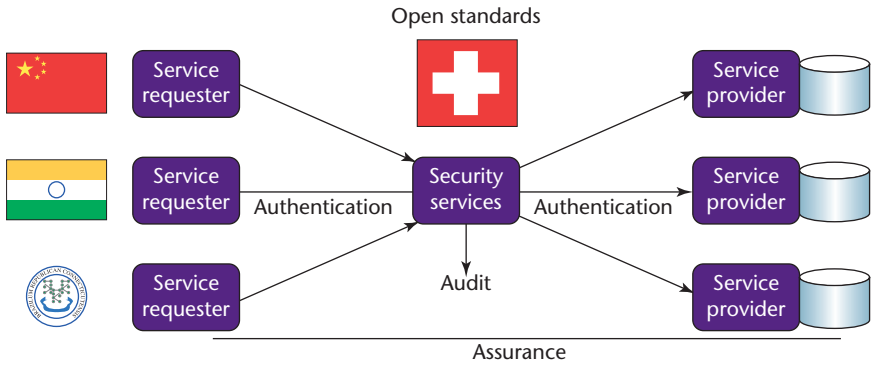

Figure 2. Standards. Consistent policy enforcement and management translates to better security decisions.

Table 1. Comparing field-level software development and information security innovations.

RELATIVE TIMELINE	SOFTWARE	SECURITY
~1995	CGI/Perl	Network firewalls, SSL
~1997	ASP, JSP	Network firewalls, SSL
~1998	COM, EJB, J2EE	Network firewalls, SSL
~1999–2000	SOAP, XML	Network firewalls, SSL
~2001	SOA, REST	Network firewalls, SSL
~2003	Web 2.0	Network firewalls, SSL

the security people deal with this? Sing it with me—network firewalls and SSL.

The software world didn't stop innovating in 2000, of course. In the past few years, we've seen Web services and XML form the basis of powerful service-oriented architectures (SOAs) and simple Representational State Transfer (REST) applications. We've also seen the debut of Web 2.0 and entirely new networked applications built on top of that. Clearly, the time has come to do something to meet all this innovation and somehow protect both its users and developers.

Prescription Patterns

Web 2.0 has no effective security model, so let's pick up the trail with the next most recent innovation, Web services, which have three main goals:

- *Virtualization.* We want Beijing, Bangalore, and Boston to communicate so that we can chop up work and deliver it from where it makes sense.
- *Interoperability.* We want our Java systems to talk to our .NET systems.
- *Reusability.* We want to know how many order, pricing, and customer systems one company needs.

These are goals to keep in mind when building services, so they make perfect starting points for security goals such as confidentiality, integrity, and availability. The way we seek to deliver these properties is through such mechanisms as authentication, authorization, and auditing, but the challenge is deploying these mechanisms as widely and flexibly as possible through services.

Virtualization

In terms of virtualization, we need to be able to authenticate users in one place and authorize them in another—for example, authenticate in Beijing and authorize in Bangalore. To paraphrase Ross Anderson, we need crypto mechanisms that take trust from where it exists to where it's needed. Figure 1 shows that authentication and authorization decisions are delivered to different locales in the architecture.

Interoperability

Security decisions are business, not technical, decisions. Thus, wherever possible, security information must be standards based, allowing for consistent authorization policy enforcement using SAML, XACML, and other open standards. Figure 2 diagrams where standards add the capability to transmit attributes to make security decisions.

Reusability

The perimeter in an SOA is the document, not the network; sim-

ilarly, the security model is defined by the security constructs in the document, not the network firewall. Because security comes from an operational mindset, the inclination is centralized command and control. Figure 3 shows three possible ways to deliver security services.

Unfortunately, this model makes many assumptions from which technical and business realities diverge. In an enterprise today, you can't expect to govern both the subject and the object, as well as the session and data, in one technology or even one business unit.

The next logical step is high-assurance endpoints, but the problem here is that when you have 100,000 of anything, you end up with management problems. You simply don't have enough security gurus to comprehensively address all the distributed endpoints on an ongoing basis.

Next, we go to the hybrid model (remembering that hybrids are the most resilient plants in nature) in Figure 3c. Now we can place various high-assurance intermediaries that can provide some security services to the endpoints. We can tune the intermediaries for their specific services, say, XML encryption/decryption, and environment, say, B2C or B2B. This model is predicated on how successful and scalable enterprise security mechanisms have worked in the past, such as Active Directory, Lightweight Directory Access Protocol, and Federation, which all leverage multiple centers to provide services to a wide variety of endpoints.

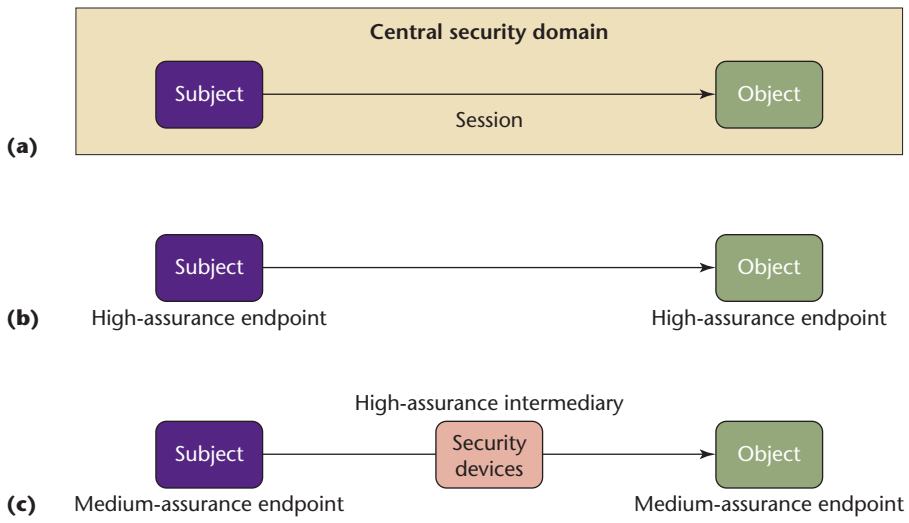

Figure 3. Hybrid models. Pure (a) centralized and (b) distributed security models won't fly in the enterprise, so (c) a decentralized hybrid of security services is the pragmatic way forward.

What Next?

I think this last decentralized model captures information security's best chance to regain some credibility and traction in improving software security. Web security is horribly broken after more than a decade of noninnovation, so it's time to just admit it and look to other models.

How about email systems? They fit the decentralized service-based model well, their endpoints are loosely coupled, and their agreement point is a message, not binary communication. Businesses have already proven that they can deploy this flexible model in the real world, and guess what else? We've actually seen real, live, security innovation in the field. The course your email follows is mediated by many differing security mechanisms, from antivirus tools to spam filters. This is, I think, what information security can leverage in Web applications—designing and deploying decentralized security services that facilitate the delivery of a specific service.

The real question with security as a service isn't about confidentiality, integrity, or availability properties—it's how to distribute the services that enable those properties. Meshing the two concepts together, how can we deliver virtualized, interoperable, and reusable authentication, authorization, and auditing? To move away from the static past and make security services a reality, we need to start from scratch—develop a playbook, and then classify, locate, design, and optimize the controls. Maybe someday soon, we'll catch up with the software development community. □

Reference

1. M. Tanji et al., *Threats in the Age of Obama*, Nimble Books, 2009.

Gunnar Peterson *is a founder and managing principal at Arctec Group, which supports clients in strategic technology, decision making, and architecture. His work focuses on distributed systems security architecture, design, process, and delivery. He maintains an information security blog at http://1raindrop.typepad.com. Contact him at gunnar@arctecgroup.net.*

Interested in writing for this department? Please contact editors John Steven (jsteven@cigital.com), Gunnar Peterson (gunnar@arctecgroup.net), and Deborah A. Frincke (deborah.frincke@pnl.gov).