Extending Extensible Markup Language Based Services Beyond the Perimeter

Introduction

Across leading enterprises, architects have discovered that convincing a business owner to invest in service-oriented architecture (SOA) and Extensible Markup Language (XML) is most readily achieved when the architect identifies a clear revenue-producing or cost-reducing project that can be demonstrably accelerated by these technologies. The most compelling are often those projects that entail integration between the enterprise and its business partners, suppliers, or even customers. The most significant challenge to those projects is providing security. The vast majority of IT professionals and business people agree that security is the leading concern for SOA and XML messages, and most quickly realize that Secure Sockets Layer (SSL) is limited by its lack of content security, auditability, and reliability.

Industry leaders and analysts assert that addressing security concerns is both practical and possible. In fact, many enterprises can and are extending their XML-based services beyond the perimeter with more pragmatic and graduated approaches than are required for comprehensive internal SOA.

This document discusses how to extend SOA beyond the perimeter through high-performance service access controls, deep content inspection, and alignment of people and processes for sustainable growth

Case Study: Aeroplan

Aeroplan is a highly profitable subsidiary of ACE Aviation Holdings (ACH) whose first SOA initiative securely exposed XML and Message Queue (MQ) based services as Web services (using Simple Object Access Protocol [SOAP] and HTTPS) to third-party partners so that those partners could offer goods and services in exchange for Aeroplan miles. The entire project was deployed in 38 days, and the time to provision a new business partner is now less than 1 hour. The net result is that Aeroplan is realizing 500 percent growth in non-air rewards redemption, making the company more profitable and its customers more satisfied. Following this project, multiple business teams adopted the SOA philosophy, and the use of SOA and XML has expanded throughout Aeroplan and ACH.

XML services (XML, Representational State Transfer [REST], Web services, and Electronic Business using XML [ebXML]) continue to gain momentum as the most efficient and flexible architecture for real-time system-to-system integration. The largest automobile manufacturers are finding that XML-based services are flexible and cost-effective mechanisms for connecting dealer networks. Financial services institutions are expanding their market reach by reducing the costs of integrating their services with employers' portals, increasing growing total revenue while improving their profit margins. Loyalty companies are enabling real-time points-based e-commerce to enhance the value of loyalty programs and increase revenues. These are a few of the many examples of enterprises that have achieved significant business gains by extending XML-based services across the perimeter, often as their first foray into XML- and standards-based SOA.

The virtually instantaneous, open application integration promised by XML services offers organizations the potential to respond rapidly to new business opportunities. The business benefits of connecting and automating mutual processes are clear. XML services technology advances make that goal easier to achieve than ever before. Direct connections to mission-critical functions improve business responsiveness and results; however, they also expose the enterprise to a new class of problems.

To extend XML-based services beyond the enterprise, the service architect must do the following:

- Control service access
- Deeply inspect content
- Align people and processes for sustainable protection

Expanding XML-Based Services Beyond the Perimeter

At the beginning of a SOA initiative, many architects make a design assumption that all services will be within the firewall and consequently have limited to no protection requirements. These projects may demonstrate integration benefits, but they often have trouble getting the enthusiastic support of business teams. The exceptions are those services built for external consumption to achieve a business goal.

By focusing on services that extend outside the enterprise and generate visible and quantifiable benefits, service architects can develop a comprehensive SOA that is deployed gradually with continuous business support. To generate continuous support, the projects delivered through SOA must be faster, cheaper, and at least as reliable as projects delivered through traditional integration technologies. With minimal initial infrastructure, service architects can deliver these benefits and be prepared for the growth that follows initial success.

Protecting the Enterprise When Exposing Services

Every service architect faces choices and trade-offs about where and how to protect the enterprise. Use of XML-based Web services removes the network safety net because messages will transit ports that are open for internet access (ports 80 and 443). Existing network defenses are mostly oblivious to XML and cannot deliver perimeter protection that has the necessary application understanding to be useful. Consequently, service architects must choose between the following alternatives to protect their multiple services perimeters:

- SSL-only protection
- Hard-coded protection
- Platform protection
- Agent protection
- Gateway protection

SSL-Only Protection

The logical first response to securing XML traffic crossing the Internet is to use SSL to secure the transport. This is a viable first step and is often a core element of a protection program for XML-based SOA. SSL is well-understood and broadly supported, making it an attractive alternative, but reliance on existing SSL technologies exposes the enterprise to significant deployment delays and considerable risks.

SSL aggregators and accelerators expedite the SSL handshake so that SSL connections are formed rapidly. These technologies are optimized for one-way SSL connections, and most service architects want to authenticate the third-party connection using two-way SSL. In addition, SSL leaves the enterprise exposed to improperly formatted or malicious XML payloads and XML Schema incompatibilities and provides no message-level access control (so that every message from an authenticated connection has access to every service exposed to that connection). SSL is not designed for message debugging and service provisioning. The associated message debugging costs valuable provisioning time, reducing business support for the initiative. These problems limit the interoperability and security of the exposed XML Web services and consequently limit their benefits. As a consequence, service architects and developers seek message-level security measures to augment their use of SSL.

Hard-Coded Protection

During prototype and pilot service development, many developers program protection into the service itself. This approach is often expeditious for demonstrating the service and claiming some level of security. Some developers believe that this is the only way to comply with privacy requirements, but the processing and integration costs to the service are very high, and the policy control is very low. Both service and security architects recognize that it is extremely expensive to audit all programming and verify that protections exist as specified. It is also difficult to modify this code to accommodate new use cases and new risks. Networking and operations specialists indicate that the network bandwidth consumption and the service latency of this approach quickly become unacceptable.

Platform Protection

The next approach usually considered is the use of precoded plug-ins or toolkits within application platforms, packaged applications, message-oriented middleware (MOM), enterprise application integration (EAI), enterprise information integration (EII), and enterprise service buses (ESBs). This use of platform-specific plug-ins or toolkits reduces the probability of a standards implementation error, and platform-specific solutions are often as easily demonstrated as hard-coded ones. However, with platforms, the policies that pertain to the service and the message traffic are hard-coded into the platform component, and the platforms have no protections for network-style risks such as denial of service (DoS) attacks.

The policies programmed into the platform cannot be governed, modified, or optimized centrally which creates significant concerns about their efficacy. In addition, the enforcement of the policy logic occurs on the platform itself, with significant performance and security implications. XML schemas, authentication, authorization, encryption, and digital signing are all extremely processing intensive, and performing these functions on the platform reduces the resources available for actually delivering the intended service. It is inefficient for the SOA to perform repetitive operations on the same message as the message moves between multiple services. In addition, the message is already in a position to cause damage because it is clearly within the perimeter. For reasons of governance, performance, and security, these platform implementations are rarely used independent of other infrastructures and often can be avoided entirely.

Agent Protection

To address the governance concern and help ensure service-level security, many service architects consider agents that enforce policy defined in central policy controllers. This solution does allow considerable insight into and control of policies across multiple services and traffic. The factors that the service architect must consider when implementing an agent architecture are the

platform coverage, the performance effect of platform-based processing, and the level of perimeter protection because the message is on the platform when the agent does its work.

Platform coverage is a significant consideration with XML messages originating and flowing to diverse application development platforms (such as Java 2 Platform, Enterprise Edition [J2EE] and .NET), packaged applications (such as SAP and Oracle solutions), middleware (such as TIBCO, SeeBeyond, and IBM WebSphere MQSeries), and older applications. In fact, analysts estimate that more than 60 XML platforms are currently in wide use, with multiple versions of each. The cost of deploying and maintaining agents across so many platforms is a significant concern for most architects.

The performance effect of agents is the same as for platform and hard-coded alternatives: The processing for expensive operations is performed at the expense of the service itself. The inefficiency of repeat processing is also difficult to address with an agent architecture. Many architects deploy agents as proxies that are independent servers on the network. In this case, the agents are no longer agents but instead have become gateways and should be evaluated in comparison with other gateways.

Gateway Protection

Gateways are network nodes that are centrally controlled for strong governance, and they offload expensive processing to enforce policies. These network nodes can be centralized or distributed depending on the service and network architecture. Often, gateways create a secure buffer between consumers and providers of services, where traffic can be normalized and secured in all directions. Gateways such as the Cisco® ACE Application Control Engine XML Gateway also provide optimizations so that policy can determine whether the same message requires full processing or simple validation as it moves between services. Appliance-based gateways such as the Cisco ACE XML Gateway (as opposed to software proxies) offer strong security as well as maximum throughput for traffic. With the Cisco solution, enterprises experience reduced services latency.

Controlling Service Access

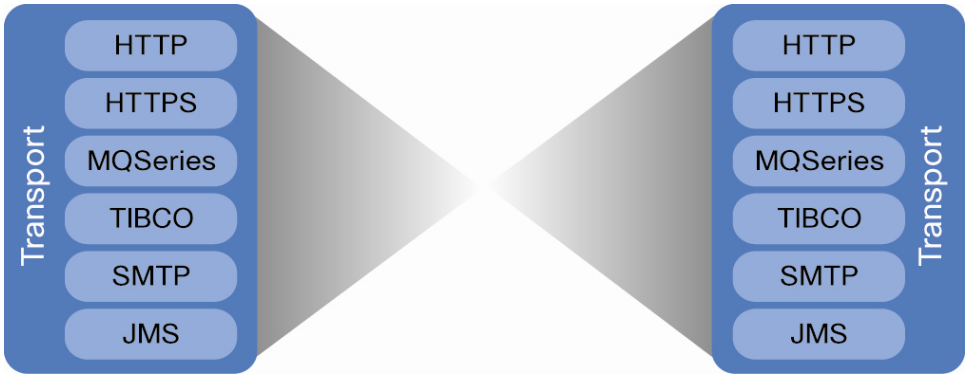
To control service access, service architects need to consider both message-transport security and service-level security.

Message-Transport Security

Much of the success of the traditional Web has been facilitated by creation of the SSL protocol by Netscape and the subsequent broad adoption of SSL by all other browser and server providers. The maturity and availability of the technology have enabled many initial XML Web services to use bilateral SSL to authenticate connections and protect against common transport attacks, such as man-in-the-middle attacks, and data compromise.

The term Web in Web services misleads many into thinking of the client-server approaches based on user interactions using Web browsers. In that model, there are two end points, the browser and the server, and a single transport protocol, HTTP, so a simple point-to-point security model that addresses HTTP can work well. In fact, XML Web services much more closely follow a consumer and producer processing model, in which multiple transports may be used and multiple processing steps may be applied to a transaction (Figure 1). This approach creates new opportunities and challenges in creating a practical threat defense framework.

Figure 1. Typical XML Web Services Transport Protocol Mediations



For example, in addition to HTTP, transports such as the ubiquitous Simple Mail Transfer Protocol (SMTP) (for e-mail) are appropriate for asynchronous or long-lived business transactions and are growing in popularity. SSL does not provide any security support for SMTP or other store-and-forward or message-oriented transports. To expand beyond the enterprise with XML message exchange, the service architect must design or implement mechanisms to accommodate different transports while preserving the policy-determined security of the XML messages themselves.

XML-based services requirements rapidly expand beyond those addressed by a session-oriented security approach like SSL to requirements to address the security of the messages or transactions directly. Ratified standards such as Web Services Security (WS-S) address the broad needs of Web services by directly securing the Web service messages, independent of the transport. Message integrity, privacy, and strong authentication for trusted identities are all provided, independent of the security of the underlying transport.

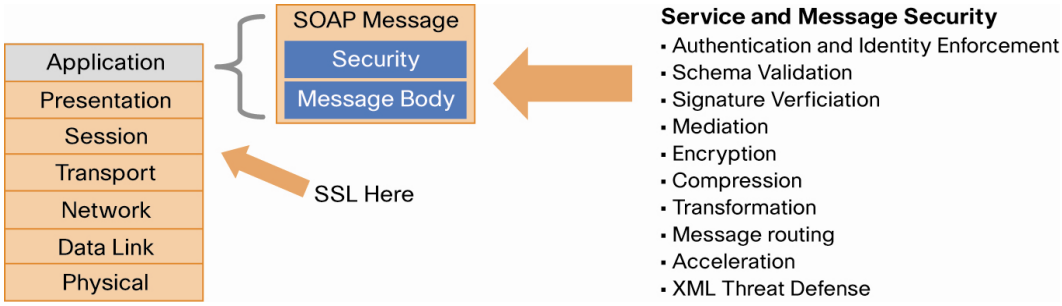
Service-Level Security

At a minimum, every service needs to help ensure that only appropriate messages are accepted and processed. Although SSL can provide transport-level security, service-level security depends on robust authentication of messages as well as dynamic authorization to grant access to services (Figure 2).

Service-level security authentication and authorization functions present several challenges: where to perform authentication and authorization, how to use existing logic, how to minimize network traffic and latency, and how to reconcile the use of different authentication identities by service consumers and providers.

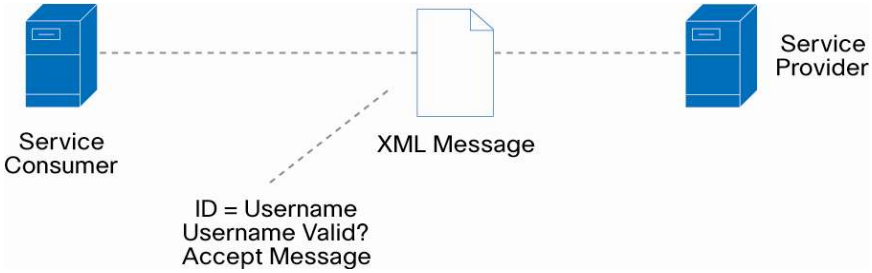
When adopting a SOA, enterprises find that messages originate from systems with different identity services and credentials. To maximize reuse and interoperability while preserving service access control, the destination service must have an efficient mechanism to validate the identity and help ensure authorized access. These challenges are exacerbated in a SOA, where there are likely to be multiple identity systems spanning both commercial and custom products.

Figure 2. SSL vs. Message-Level Security



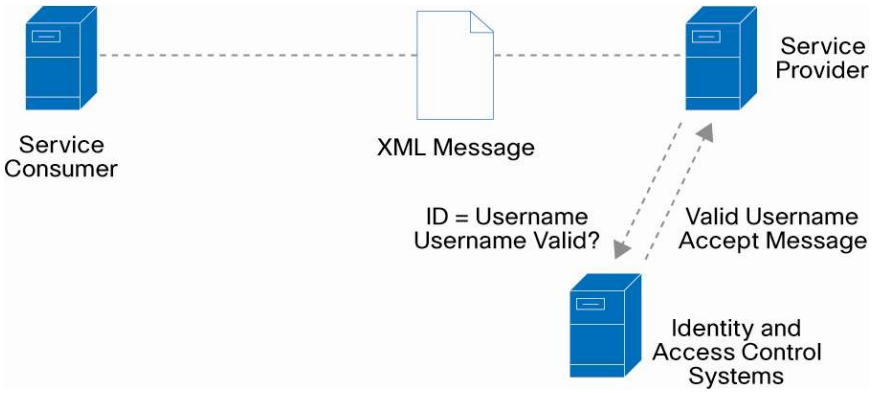
The simplest authentication process is for a service to receive an XML message, parse and recognize the identity, and then verify that the identity is an acceptable one for that service. The authorization process then determines whether that valid identity is allowed access to this service. This approach assumes that the service consumer uses the identity credential (in Figure 3, it is the username) recognized by the service provider.

Figure 3. Authentication and Identity Enforcement



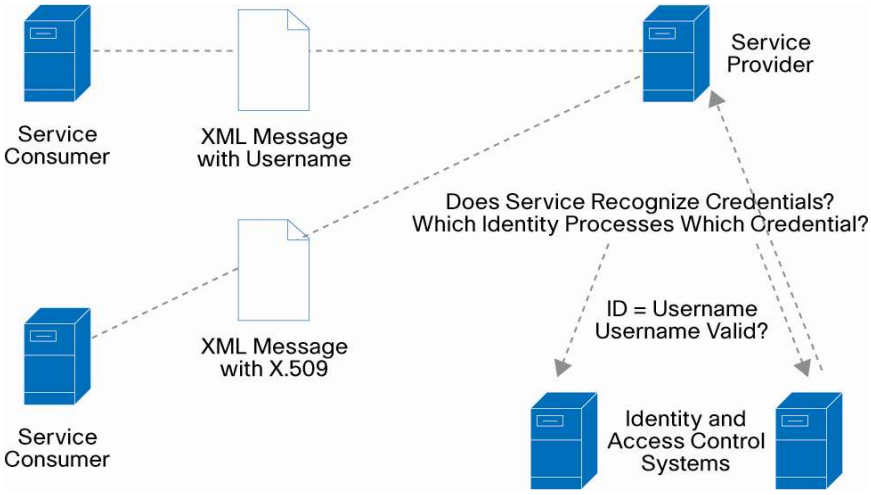
Many enterprises have invested considerably in identity control systems using Lightweight Directory Access Protocol (LDAP), Microsoft Active Directory, or commercial systems such as CA Netegrity or RSA ClearTrust. For consistent governance and rapid deployment, the enterprise may want authentication and authorization policies to reside in these identity management systems. To use such a system, the service provider must integrate with it and, after recognizing the identity credential, send a request to the identity management system for an authentication decision. This process often requires multiple simultaneous integrations for authentication and authorization, with multiple calls between systems for each message (Figure 4).

Figure 4. Leveraging IAM Systems For Authentication and Authorization



Two factors further complicate the authentication process. First, an enterprise may have multiple identity management systems, and the service provider must determine where to route the authentication request. The services architecture needs a scalable model to integrate multiple identity management systems with multiple services. This model must often accommodate custom identity management systems. The rationale for this model is often the aggregation of technologies across the enterprise or the integration of technologies acquired through mergers. In addition, the service consumer may use different credentials than the service provider can accept. A significant concern for the service architect is enabling reuse for the service providers, and to enable reuse the service provider must have a mechanism to accept multiple types of credentials (Figure 5).

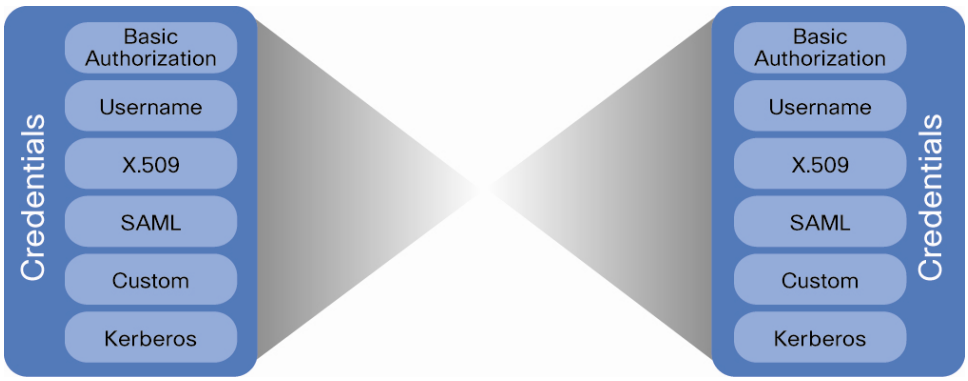
Figure 5. Integration Of Multiple IAM Systems With Multiple Services



Depending on the service, authentication and authorization decisions can result in a very complex set of policies to enforce. For example, a particular service may be accessible only to messages sent from a particular group of machines and domains. The target service may also expect all identity credentials to be delivered to it as Security Assertion Markup Language (SAML) credentials regardless of whether its consumers support SAML.

The services architecture requires a mechanism to mediate between credentials, so that a service consumer can use X.509 certificates, username, Kerberos tickets, or any other identity credential, while the service provider receives credentials only as SAML tokens. The same process is performed in reverse (Figure 6).

Figure 6. Credential Mediation

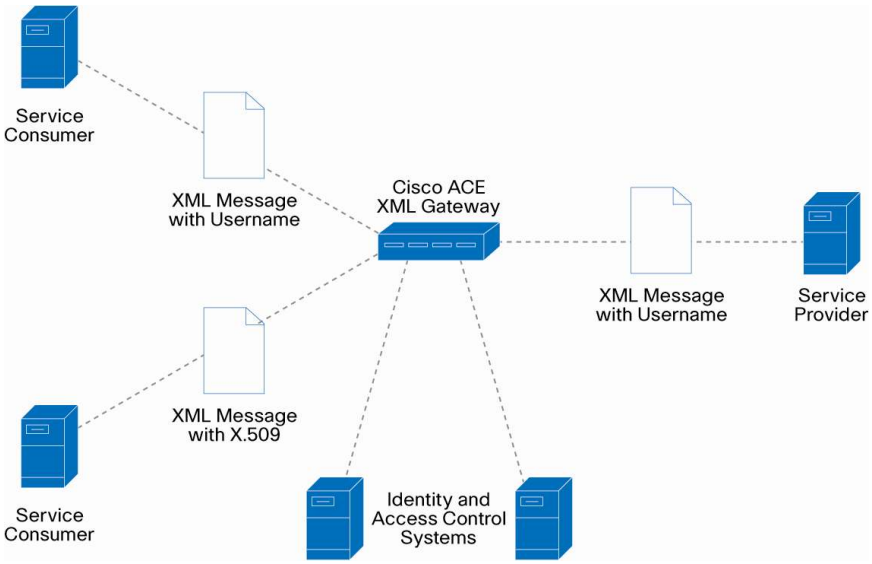


Case Study: Carlson Marketing Group

Carlson Marketing Group (CMG), a subsidiary of Carlson Companies, was connecting to a major credit card company to offer expanded real-time rewards to credit card holders. The identity management system at CMG was a custom-developed system. CMG needed to connect to the credit card company using XML Web services (according to the credit card company's contract) and needed to validate the identity of the individual while replacing it with an identity that would be understood by CMG's systems. CMG used the Cisco Software Development Kit (SDK) to integrate its custom identity management system from the Cisco ACE XML Gateway and transform the credentials between formats. This solution enabled CMG customers to send the credentials they use and CMG to accept the credentials without the need to modify any existing systems. The result was faster time to market, more satisfied clients, and cost savings achieved by minimizing back-end system changes. As a consequence, CMG is expanding its use of SOA.

The time required for all the integration, routing, and authentication decisions can often affect service-level agreements (SLAs) and service availability. In addition, the volume of XML messages either through new connections or through expanding service reuse can cause the service platform to incur significant latency and processing costs in authenticating and authorizing access. Much of the infrastructure integration and authentication optimization can be most efficiently offloaded to policy-controlled gateway appliances such as the Cisco ACE XML Gateway (Figure 7).

Figure 7. Cisco ACE XML Deployment



Application systems composed of XML-based services create additional requirements for the use of identities that can be correlated and verified to support access control, auditing, and trust. The identities of principals initiating a transaction, the service components involved in processing a message, and the roles and other attribute-based authorization and entitlement systems all are important in service implementations. The opportunities for additional-replay attacks, insertion of invalid transactions, and consumption of information by unauthorized users are significant concerns in an application system or SOA.

The Cisco ACE XML Gateway integrates natively with multiple identity systems and can enforce the authentication policies in multiple systems while mediating among formats, terminating SSL, and helping ensure that the XML content is safe and consumable for destination services. For example, the Cisco solution can detect authentication and authorization problems while optimizing service responsiveness by caching authentication credentials and decisions. In combination with native APIs, custom identity system integration, and credential mediation, the Cisco ACE XML Gateway provides a robust authentication and authorization enforcement network for services while helping ensure that service providers have the authentication and authorization information needed to deliver their business logic.

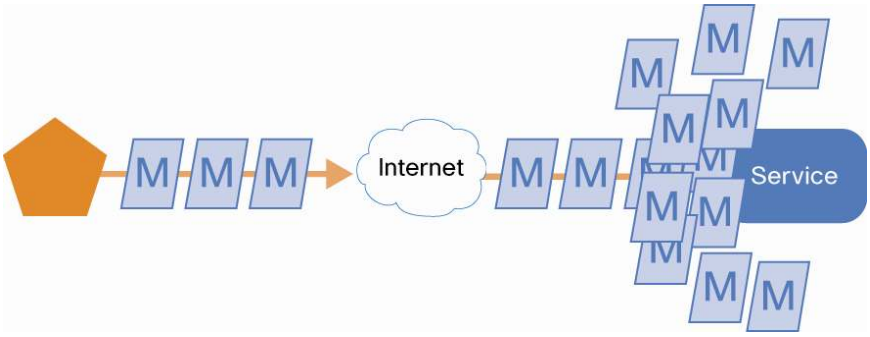
Threat Mitigation

XML DoS and content-based attacks pose particular threats to services.

Protecting Against XML-Based DoS Attacks

For any network-based service, a DoS attack is a serious and common problem. The fundamental approach used in all DoS attacks is for the attacker to initiate a process on the service provider side that entails minimal cost for the attacker but consumes the resources of the provider to the point where the service becomes inaccessible. SSL itself represents a DoS attack vector, because the overhead of mutual authentication is so high that an attacker could use it to consume the service computing resources (Figure 8).

Figure 8. DoS Attack

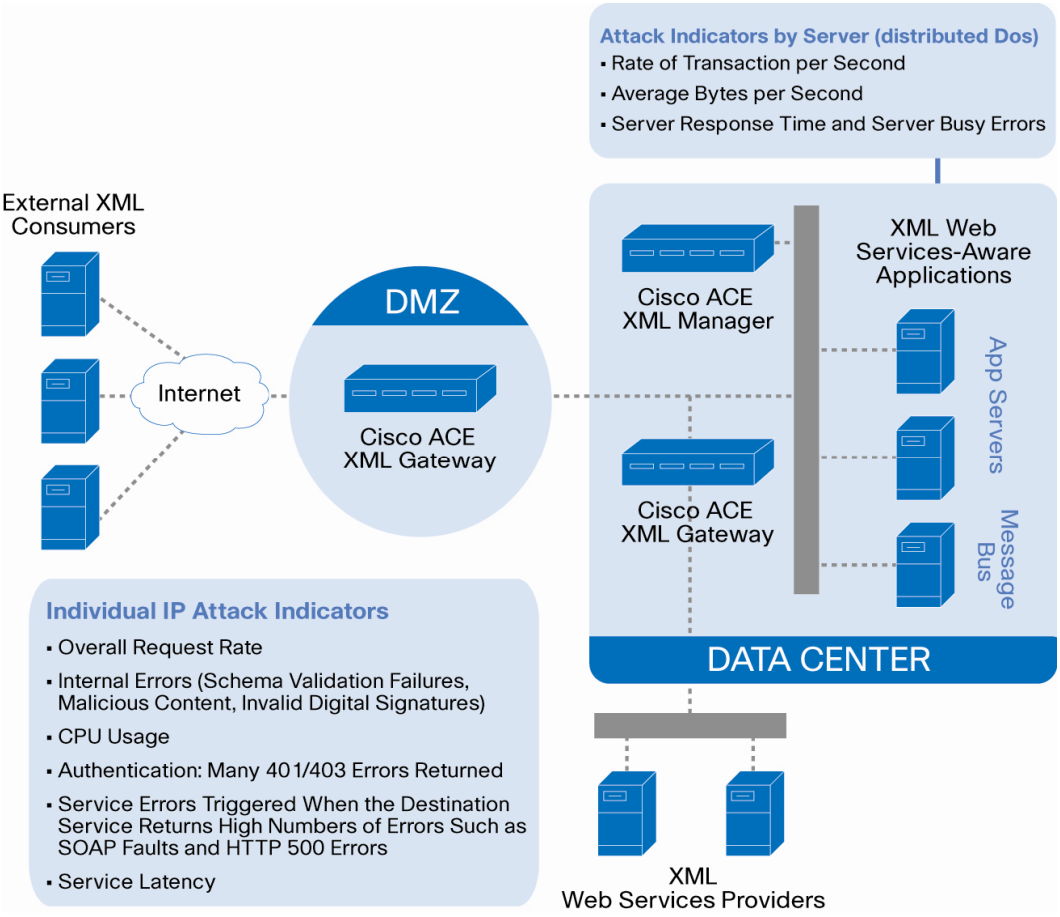


Overwhelming an XML-based service is relatively easy. In comparison to Web servers, which handle thousands or tens of thousands of transactions per second, XML-based services tend to handle tens or hundreds of transactions per second. A partner that simply attempts to interact with the service too enthusiastically can accidentally disable the service. In addition, the relatively limited experience of developers creating XML-based services increased the likelihood of errors such as infinite loops that can also render a service inaccessible to all traffic. Existing network or application infrastructure can neither defend XML-based services from these accidental attacks nor defend against them.

Existing network and Web protection mechanisms do not address DoS attacks that are launched within the application level, as is the case with XML and Web services. To help ensure the availability of these services, a scalable approach is needed that addresses these risks through heuristics and alerting to identify patterns that may constitute a DoS attack against XML and Web services.

XML Web services integrate core business applications with each other and business partners. As a consequence, a number of indicators that might be considered an attack are often legitimate business patterns. For example, with Web services, high message arrival rates can represent an overly simplistic trigger for DoS detection and avoidance. Did your business partner trigger a huge transaction load as a result of the availability of an inexpensive product on its Web site just before a major holiday? This example points to an interesting conclusion about DoS “attacks” in XML-based services: Some of them may be inadvertent and from trusted sources, but the result is the same. It is therefore crucial to assess DoS metrics over configurable periods of time and apply appropriate •heuristics to classify traffic patterns (Figure 9).

Figure 9. Metrics To Detect XML Denial Of Service Attacks



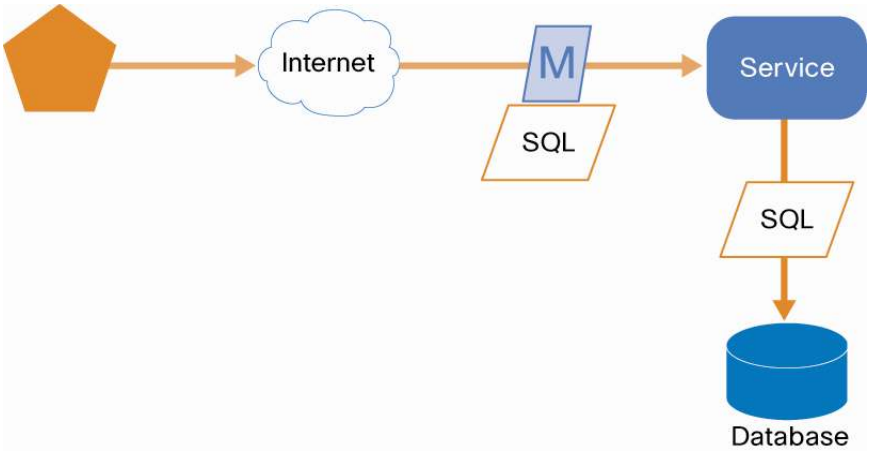
In addition to arrival rates and flow control of messages passed to back-end servers, the richness and complexity of XML itself provides a unique form of DoS attack vector. Extremely complex schemas or expansion of recursively defined entities represent high-overhead processing that ideally suits the needs of a DoS attacker. In this case, a single XML message employing a technique such as recursive entity expansion can consume all the resources of a service. A huge SOAP attachment on a small number of messages can have a similar effect.

Practice Safe SOA: Inspect XML Messages for Threats

XML messages and their payloads can easily carry unintentional or malicious service attacks (Figure 10). Content is a well-understood vector for HTML browsers, with attacks using techniques such as Sequential Query Language (SQL) command insertion. Content-based attacks from trusted systems can occur as a result of a user's misconfiguration or compromise of system security in initiating the transaction (for example, infection by a virus).

Specific services, applications, or platforms are each susceptible to a unique set of attacks. The service or application may have vulnerabilities unique to its design. Service-specific vulnerabilities may arise, for example, from the way that the service performs field and type checking to update database entries. An example of a platform-specific vulnerability is the buffer overflow problem associated with a User Datagram Protocol (UDP) port on a well-known SQL server; the problem was exploited by the SQL Slammer virus.

Figure 10. Content-Based Attack



Because attacks are service, application, and platform specific, content inspection must be specific to the service, application, and platform. The service architect must at minimum have a mechanism to help ensure that messages are well formed. In addition, most service architects recognize that validation of XML Schema is a crucial, albeit potentially expensive, mechanism for protecting services from malicious XML. Ideally, the service architect has a policy-controlled mechanism to define unique content screening filters applied across the enterprise as well as to messages intended for a specific service. The processing costs and delays associated with XML Schema validation as well as the ability to deeply inspect all XML content to help ensure message purity is best handled by gateway appliances designed for the task.

Consequently, a gateway must have the flexibility to configure new filters or to securely upload new filters as new vulnerabilities are discovered in standard platforms or new XML Web services applications are deployed. XML content filtering is more analogous to text filtering than to e-mail virus filtering. For example, in some XML use cases SQL should be accepted, but only specific commands; in some situations XML may contain words that constitute a remote procedure call (RPC) statement (for instance, "assign tables") but which are actually innocuous text ("assign tables to Charles for set up").

Types of content-based threats include malicious RPC statements and bad Web Service Description Language (WSDL). Malicious RPC includes SQL injection attacks. Threats can come from the insertion of inappropriate content into a well-formed XML message; such content can be

identified through schema validation and detailed content screening filters that are specific to the service and intended connection. In addition, the service WSDL must be checked to help ensure that it is valid and that the external references from the WSDL are valid and available for the production service.

Content Privacy and Compliance

XML messages are self-describing and human readable. These attributes enable the very broad interoperability that makes XML attractive for extensive system-to-system integration for new applications. These attributes also provide opportunities for confidentiality and integrity violations that can undermine the credibility of those applications.

The mechanisms for helping ensure the confidentiality and integrity of XML messages are well understood and supported for both XML and Web services (through the WS-Security specification). For example, confidentiality can be facilitated through the use of XML Encryption. This mechanism helps ensure confidentiality of the message regardless of transport. But this confidentiality may come at a high performance cost and increase the latency of the system beyond acceptable levels. However, integrity can be facilitated through the use of XML Signatures. In this case, an entire message or just crucial elements can be signed to help ensure that the content is tamper-proof. Again, these operations come with a high performance cost, and both XML Encryption and XML Signatures are complex standards with numerous implementation options, creating opportunities for errors and lack of interoperability.

The Cisco ACE XML Gateway offloads and optimizes these expensive operations while delivering a robust record of the confidentiality and integrity of the XML messages. That record can be used to prove compliance and to repudiate a transaction's processing. With service and message content-specific policy enabled, the Cisco ACE XML Gateway can enforce granular confidentiality and integrity policies at the same level of detail as the destination service, providing end-to-end security.

The Cisco ACE XML Gateway also helps ensure that XML-based services comply with Sarbanes-Oxley 404, Health Insurance Portability and Accountability Act (HIPAA), Gramm-Leach Bliley Act (GLBA), California Senate Bill 1386, European Union Privacy Directive, and corporate compliance and privacy standards by tracking critical Web services traffic and providing alerts to identify abuses. The Cisco ACE XML Gateway enables inspection, action, and reporting on data such as the following:

- User credentials and identities
- Company credentials and identities
- Services and applications accessed
- Requesting parties, servers, and applications

Aligning People and Processes for Sustainable Success

The best perimeter protection for XML services and SOA is meaningless if it cannot be sustained by the organization without the steady participation of service architects. A set of practices and policies must be codified to guide service developers regarding the perimeter protection required for the services. In addition, the technology used to protect the perimeter should enable an enterprise to implement a policy workflow model consistent with the levels of control required by the business as well as compliance (legal) guidelines. Service architects must identify, create, or

select technology that reinforces the practices that protect the perimeter and evolves with expansion in service use or with new risks.

For example, the technology should support a large number of configurable roles that deliver different privileges to users. In addition, the technology should provide a visual mechanism to compare proposed policy changes to existing policy so that administrators with approval and deployment rights can easily approve and deny requests as well as track a workflow of policy development, approval, and deployment. The technology should also have functions that encourage its use by application developers as they create new XML-based services and provision new connections to existing services: for example, functions that enable an isolated application development team to create services, connections, and policies to test, with the policies easily exported from the developers' environment and imported into the product environment.

Finally, the technology should provide mechanisms to help ensure that protected services are inaccessible unless the user has been properly authenticated and screened. This crucial protection can be achieved through a variety of mechanisms: using network address access control within the stack, maintaining SSL connections between the technology and the services, or inserting signed SAML assertions on every processed message, with accompanying logic at the service to accept only messages so marked.

Applying the Lessons of External Integration to Other Services Perimeters

XML-based services used within the enterprise often incorporate reused and composite services that commonly cross boundaries inside the enterprise. These boundaries can be physical, such as the boundary between data centers or remote offices and headquarters. These boundaries can also be virtual: between different services platforms and applications, with different trust practices for access and data review. Service architects must have a robust view of service perimeters and recognize that any boundary crossing by definition constitutes crossing a perimeter. Every perimeter usually requires some protection, if only to help ensure the availability and responsiveness of the service itself.

Types of Services and Associated Perimeters

Successful XML service and SOA implementations have a business goal that often shapes the type of services deployed and their associated perimeters.

An obvious perimeter surrounds services that enable third-party access to enterprises. These services are created to enable real-time interaction with partners, suppliers, and customers that provides incremental revenues. The perimeter surrounding the service and the third-party connections clearly must be protected.

Standalone services provide central access to shared data or functions such as a customer information store. Consequently, the perimeter for a standalone service is the edge of the service itself. This is the most constricted perimeter possible and often the easiest for which to implement the first service, but the hardest to scale to accommodate multiple services and multiple connections.

ESBs are used to expose older or MOM technologies as services and integrate applications more efficiently. Within the ESB, the ESB helps ensure that every service on the bus is protected, and that the messages are acceptable and compatible across the bus. However, enterprises commonly have multiple MOM, EAI, EII, and ESB technologies, both standalone and integrated into application platforms and packages. Consequently, there is a boundary between multiple ESBs. In

addition, there is a boundary between ESBs and external connections to the services offered on those ESBs.

Composite services are representations of a business process as a service to provide efficiency and reduce errors. A composite service can be implemented using an orchestration server, ESB, EAI, EII, or other technology, which determines whether there are one or more perimeters. If the implementation mechanism provides security between services in the composite service, then the only boundary is between composite services and their consumers. However, the most loosely coupled composite applications rely on an orchestration server (or service) to coordinate the messages between services in the composite service, which introduces perimeters around the individual services within the composite service.

Eventually, the enterprise must address each of these perimeters with a policy-controlled infrastructure that can accommodate the practices required by each type of perimeter, just as virtual firewalls provide protection at the IP level.

Perimeter Protection Excellence: Cisco ACE XML Gateway

Helping ensure that XML messages securely and efficiently reach their intended targets with minimal latency and comprehensive policy control requires dedicated infrastructure that understands XML messages. Gateways provide the critical protection needed at each service perimeter, between untrusted and trusted zones. The Cisco ACE XML Gateway fits transparently with other network infrastructure and can be administered by both application and operations staff. The Cisco ACE XML Gateway integrates with existing infrastructure such as directories, single sign-on (SSO), public key infrastructure (PKI), and network system management.

Exposure of services beyond the enterprise must take into account several types of risks. As a base, service architects must provide access control, prevent DoS attacks, and deeply inspect content, and this evaluation is application and architecture dependent. An understanding of people and processes and their roles in threat prevention is also important to an appreciation of solutions that provide valuable functions such as flexible authentication policies for users and service requestors and role-based administration for gateway administrators. The Cisco ACE XML Gateway adapts to the evolving perimeter protection needs of XML services.

Cisco provides the crucial XML gateways used by enterprises to achieve the benefits of XML services and SOA. The Cisco ACE XML Gateway enables businesses to secure, accelerate, and integrate XML Web services efficiently with the market's most extensive policy control and end-to-end performance solution.

Cisco customers accelerate their time-to-market and gain competitive advantage in their businesses through the use of secure, reliable, and responsive XML-based services. For more information about the Cisco ACE XML Gateway, visit http://www.cisco.com/go/ace.

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Asia Pacific Headquarters
Cisco Systems, Inc.
168 Robinson Road
#28-01 Capital Tower
Singapore 068912
www.cisco.com
Tel: +65 6317 7777
Fax: +65 6317 7799

Europe Headquarters
Cisco Systems International BV
Haarlerbergpark
Haarlerbergweg 13-19
1101 CH Amsterdam
The Netherlands
www-europe.cisco.com
Tel: +31 0 800 020 0791
Fax: +31 0 20 357 1100

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **www.cisco.com/go/offices.**

Printed in USA C11-409465-00 05/07

