Policy meeting 21 February 2003

Attending:
Burlington:
  Dave Chappell, Sonic Software
  Bob DeWolfe, Tarari
  Yassir Elley, Sun
  Don Flinn, individual
  Frederick Hirsch
  Ron Monzillo, Sun
  Eve Maler, Sun
  Prateek Mishra, Netegrity
  Tim Moses, Entrust
  Andrew Nash, RSA Security
  Rob Philpott, RSA Security
  Rich Salz, DataPower
Santa Clara:
  Doug Bunting, Sun
  Gary Ellison, Sun
  Jeff Hodges, Sun
  Sandeep Kumar, Cisco
  Shivaram Mysore, Sun
  Jerry Schwarz, Oracle
  Frank Siebenlist, Argonne National Labs (later in SCA)
Phone:
  T.J. Pannu, ContentGuard
  Zahid Ahmed, Commerce One
  Steve Anderson, OpenNetwork
  Ugo Corda, SeeBeyond
  Zulah Eckert, Hewlett-Packard
  Komal Lahiri, Sun
  Dale Moberg, Cyclone Commerce
  Gene Thurston, AmberPoint

Agenda:

10:00-10:15 (7:00-7:15 PST) Welcome and Introductions

Ron presented a purpose statement for the meeting.

This meeting will focus on the characterization of the policy related functionality required to support the interoperable use of mechanisms, including, but not limited to, the SOAP message security mechanisms being defined by the Web Services Security TC. The purpose of the meeting will be to capture these requirements in proposed charters for one or more open forums that will serve the industry by providing solutions to these important problems.

This is not a meeting about technical solutions or IP; there should be no expectation of confidentiality. We're trying to stick to problem descriptions. In the event that we subsequently learn that a problem description is encumbered by IP that is not available under RF terms, we would expect any resulting standards activity to adjust its scope as appropriate to preserve the ability of the activity to make its work available under RF terms. We will advocate that the charter of any standards activity resulting from this meeting include in it a clear statement that it shall not produce any specification that it believes would require the use of IP which has not been made available by its owner under appropriate RF terms.

10:15-10:30 (7:15-7:30 PST) Agenda Review and Refinement

The agenda is focused on getting through to the end; it's important to drive towards chartering.

10:30-12:15 (7:30-9:15 PST) Use Cases and Example Scenarios

Ron presented an informal definition of "policy": "The requirements and capabilities that affect the nature of the interactions entities will participate in with other systems." Requirements means "behaviors that an entity demands of its peers". Capabilities means "behaviors that an entity offers to support the requirements of a peer". He also indicated that although the informal definition may be a good match for what we see as the problem space; the use of the word policy tends to complicate agreement on the suggested focus.

Prateek had collected some standard definitions. Informational RFC 3198 on "Terminology for Policy-Based Management" has a more generic definition: "a definite goal, method, or course of action to guide and determine future decisions" and one that is more specific: "~policies as a set of rules to manage, administer, and control access". The focus of this document is network resource management.

Tim: The XACML spec has a definition. Also, Maryann Hondo sent out a paper on the XACML list that said, in part: "WS-Policy is a language that can be used to describe properties and capabilities of many different types of resources. .... This language is used for communicating such information as security requirements, supported features, prefered ways of invoking the service, etc. among Web services." She noted that her definition was broader than is usually used. Her comparison paper is a response to a suggestion from Tim that an authorization style of policy has some of the characteristics that are interesting and important for web services policy.

Ron: AuthZ policy usually require a yes-no answer, and that is what XACML focuses on; but it's use of predicate calculus could be extended to produce more than yes/no answers (as Tim has suggested).

All: Using the word "policy" for this instance may be overloading it. If we could find a synonym, that would be helpful. Let's suspend our preconceptions about "policy" for the time being, and work on getting the right definition.

Ugo: Should we restrict our understanding of this concept to just security policy, or policy in general?

Zula: She represents the needs of web services management here, so doing just security won't be as interesting to her.

Ugo: He points to WS-Reliability as an example. Would reliable conversations be part of handling policy? Dave: It should be. Prateek: Is all of WSDL policy? Ugo: SOAP has a general notion of "features" or "properties"; a W3C task force of WSDL and WSA people is working on how to specify SOAP features in WSDL.

Ron: A differentiator might be that we want to express policies that are generic in their expression. When you talk about WSDL, you imply those policies' interpretataion. Tim: The relation between service features and policy might not be very remote. Specifying that the language "English" is supported could be seen as either, depending on whether you declare it first or are asked about it. Zahid: He listed several SOAP feature extensions: header extensions for choreography,

reliability, security, routing, and SOAP manifests.

Don: Whether the policies are runtime vs. static makes a difference. Ron: Are any policies static? Dale: An example from CPPA is "How often should the CRL list be checked?" It's a feature of the PKI administration, but doesn't manifest itself on the wire. Ron: You are advertising obligations that you're imposing. Don: The difference manifests itself in terms of how you store it (e.g., in LDAP or whatever). During runtime, you may not send it anywhere.

Dave: Should we stick to the scope of a conversation between two endpoints? Andrew: There would be a need for policy management. If there is a set of selections you can make within a generic implementation, then there's a need for a policy sense in which you set the expectations for a particular instance. Tim: One aspect is how a client and service can produce messages that the other will expect; the other aspect is administration and management. The latter shouldn't be in scope.

Sandeep: If you want to enforce a privacy policy, that seems mostly static. Non-repudiation is another static kind of policy. Ron: When you invoke the service, you want it to support your reasonable expectations; a service can have the capability of "keeping a secret". It's a necessary part of the structure of policy to designate that certain rules are intended for a level of negotiation.

Ron: If we only talk about interactions between systems, is there more to the problem than that?

Andrew: How do I set enterprise policy, Tim recommends out of scope, as mgmt interfaces will be a point of differentiation among vendors.

Don: Static policies are ones that I can declare independent of any interaction that I have with others; dynamic (runtime) policies are those that need to interact in order to have an outcome. Jerry: Static policies don't affect the nature of the interaction. Ron: The static ones represent a pre-calculation of a solution. Dave: You still need to have the assertions, in the end, to verify that you're doing what you and a peer had a pre-agreement about.

Eve: Ron's "definition" (which is undergoing some revision now) is better as a scope statement, so that we don't have to worry about what "it" is called until later.

> What types of policy negotiation problems do we seek solutions for?
>
> What is the nature of the interactions whose properties are being negotiated?
>
> What is the relationship of the negotiation to the interaction whose properties are being negotiated?
>
> Where in the interactions is there a need to apply the policies of the various actors in an interaction?

We agreed to keep these four questions in mind as Tim makes his presentation: "Requirements for policy-management in distributed systems":

This comes out of his QOP work and the XACML work he's been doing over

the last 18-24 months. The XACML examples here are purely just to address objections that it's far too complicated and can't happen.

Examples of types of policy: authorization (is this request properly authorized?), cryptographic security (does this request have the required security attributes?), privacy (is the requested disclosure properly authorized?), trust (is the key acceptable for this purpose? also CRL issues), others (reliable messaging, transactions, etc.). He has reconciled himself to the use of the word "policy".

Don: Notices that all these questions are boolean. Tim: They are deliberately so. But this question needs to come up later. Ron: Another example, in the area of privacy, is about representations being made as opposed to authorizations being decided.

XACML uses SAML-style namespace URIs to indicate different testing functions (e.g., testing that key sizes are greater than or equal to the required size), as well as to indicate the meanings of the attributes themselves (e.g., AttributeId="wssqop:key-size").

An alternative view (Tim is trying to represent Maryann as best hs can here) might be that policy doesn't have to be executable; it's just a data container. But he believes that even "just data" will tend to have semantics that can be seen as "executable".

Jerry: He doesn't like anything that isn't statically interpretable. He's concerned that the list of attributes and functions is extensible. Eve: Even though the URIs are extensible, you can profile the technology to support only some subset.

There is no single point at which all the aspects of policy get implemented or acted on. Cryptographic security policy might apply at the "receive" stage, authorization policy at the "process" stage, and privacy policy at the "forward" stage.

A service consumer might have these questions:

- Does my request satisfy the provider's requirements?
- How can I form a request that satisfies the provider's requirements?
- How can I find out what the provider's requirements are?
- Are the provider's requirements compatible with my requirements?
- How can I form a request that satisfies both the provider's requirements and my requirements?

You need to take a policy, turn it into execution instructions, and interpret them to produce a message that conforms to the policy. (Prateek, Jerry, Prateek, and others had questions and concerns about the notion of execution instructions.) BPEL4WS has constructs like <sequence> and <switch>, which are similar to XACML's <apply and> /<apply or> and WS-PolicyFramework's <All> and <OneOrMore>.

Provider policies and consumer policies may both place constraints on a message. So they need to be combined and renormalized to produce an "applied policy". Sandeep: Combination is a very large problem. Shivaram: In addition, a policy might request that it not be combined with other policies. Ron: We may just be talking about set arithmetic, ORs and ANDs, which is doable. Don: There may also be a negotiation phase in which the number of combinations can be reduced. Gary: If the set arithmetic has negation, the negotiation may have to be iterative to avoid the empty set of applied policies.

Ron: Perhaps "negotiation" should be reserved for cases where policies have to be changed because there's a non-intersecting set of policies. Once each sides requirements and capabilities have been exchanged, if there is a non-null set of possible applied policies, it's not so much negotiation as unilateral selection of options. Tim: There's an analogy to cases where the trust list is empty. You either decide to go ahead anyway (default policy), or get someone on the phone, or something else more closely resembling true negotiation. Ron and Rob: Negotiation should be in scope, or at last what you do to get an updated representation of the requirements and capabilities of your peer.

The semantics of <or>, as we're discussing them here, mean "one or both" (not XOR). Renormalization would involve things like collapse identical adjacent operators, combining identical set operators, and reordering sequences.

A service consumer receiving a response might need to test it to see that it conforms to its policy, just as much as a service provider might want to do the same for a request. Both of these cases presume that each side has been given access to the other's policies. This also applies to delegation, when the provider goes off and gets some other service on behalf of the consumer.

Policy distribution might need to be accomplished by several means: WSDL (provider policy, through the <wsdl:operation> element), SOAP (consumer policy, though the <wsse:security> element), LDAP, HTTP, etc.

If there's a requirement that a patient number in a request NOT be encrypted, then any request without a patient number in the clear would be rejected. If it's acceptable but not required to encrypt a field, the XACML "anyURI-superset" function could be used. Then there's the notion of preferences, e.g. to indicate which options are less costly. They can guide the selection/negotiation. For example, this could be expressed numerically as WS-PolicyFramework does it, or policies could be listed in order of preference as CPP/A does it.

Dale: Where preferences are mismatched, it may require negotiation in order to get an optimized solution, as opposed to mere unilateral selection from the options at hand.

Some miscellaneous issues here: sequential application of requirements ("pipelining" of encryption and signing), whether attributes are identified by name or location, policies that specify behavior in the event of unavailable attributes (defaults), and mechanisms for locating and retrieving policies.

Tim summarized these candidate requirements for consideration (edited to reflect the discussion around these points):

1. Publish provider policies for requests
2. Transfer or publish (e.g. through an identity service) consumer policies for responses
3. Combine provider and consumer policies
4. Translate to execution instructions
5. Express capabilities as well as requirements
6. Express preferences
7. Identify result of an execution step so it can be referenced in later ones
8. Use single formal logic system

12:15-12:45 (9:15-9:45 PST) Lunch/Break (was taken 12:30-1:00)

12:45-2:15 (9:45-11:15 PST) Detailed Problem Characterization and Decomposition

First, Ron presented some slides:

In CORBA (CSIv2) today, an IOR (interoperable object reference) is sent from the EJB container to the client container if the client has somehow misapplied the provider's policies. Bitmasks for "required" and "supported" policies are exactly aligned at all the different levels of policy to allow for semantics such as EXACTLY ONE, ALL, ANY OR NONE, etc. at each bit position (where a position represents a behavior, such as that the client must authenticate). The bitmask mechanism is extremely simple and compact, but Ron is hoping for a big improvement in this next generation. (Others commented that this basic mechanism has been independently invented in many systems.)

Prateek: Interface and policy versioning is something that needs to be handled. Tim: Also, in the WSS-QOP discussions, the question of a "naked WSDL" that gets augmented by someone else came up.

A top-level scenario contributed originally by Yassir: Alice wants to have a secure exchange with Bob:

1. Bob associates Bob's policy with Bob's service description. (One way to do this is to embed the policies in the stub, a la Jini.)

2. Alice combines her policy with Bob's in preparation for interacting with Bob.

3. Alice selects a solution if there's a non-empty intersection of Bob's requirements and Alice's capabilities and of Alice's requirements and Bob's capabilities.

There may also be a need not to compute the intersection because of dependent constraints or other computational reasons, so instead what you do is query (on your own side) whether message configuration X would be acceptable; with a Yes response, you go ahead and send it. This is Jini's approach. Tim: It would be a mistake to define a language that was locked into a particular combination approach.

4. Alice prepares and sends a request, along with Alice's policy.

5. Bob enforces and satisfies Bob's policy and the portions of Alice's policy that were sent.

6. Bob responds to Alice or invokes on Alice's behalf. The potential need for delegation came up in EJB/IIOP. Don: This is pretty important in the case of web services, rather than traditional client/server.

> Consideration of the following aspects and others as appropriate.
>
> The distribution or other communication of policy between interacting entities

Ron: This seems to have been covered by Tim's and Ron's presentations.

> The grouping of related policy statements or the

composition of policy from smaller perhaps reusable statment groups.

Ron: The CSIv2 example showed this with its different layers and bitmasks. Tim's slides also showed cases where different policies get applied at different stages. Eve: Can it be determined a priori, and thus be made a requirement, that policies be grouped by which stage they apply to? Ron: An enforcement point needs to be allowed to find the policies that apply to it. For example, you might have transport-layer vs. messaging-layer signing policies or you might have a generic policy that can be satisfied in either of the two ways; he believes you should be specific as to the layer the policy applies to. Jerry: In language design, you need to be able to identify each policy in such a way that it can be referred to from multiple places (i.e a reference must be properly scoped to a context).

Do we need a requirement to identify the specific layer to which each policy applies? Jerry: Does Alice need to be able to say that an intermediary must talk to the ultimate recipient using SSL?
Ron: Yes, that's what the slides are showing in the delegation case.

There remains a concern about "policy forwarding" among us; who does Alice allow to be delegated to (directly by her or indirectly by other intermediaries)?

The reconciliation of policy negotiation with a layered processing model

(Here, "policy negotiation" means just the combination (intersection calculation) step.) This means that only policies that apply to the respective layer need to be combined; you don't take a big bag of policies and combine them. You don't have a complete semantic model until you have a clear understanding of the layers.

Frederick: Uncomfortable with explicit layering in the semantic model, because you can express layer-specific details in the policy itself.

Dale: The ebXML folks looked into this. CPPA encapsulates characteristics akin to "SOAP features". They broke through this issue by reaching agreement on sets of features, which pragmatically made reference to layers as appropriate . So the language includes constructs for "transient confidentiality" and "persistent confidentiality" and "both" as security features. It closely matches Tim's slide 9 (combining policies 1 and 2 into applied policy 3). EbXML has two models: take-it-or-leave-it (CPP Template) and negotiated (CPA). Didn't focus on the procedural creation of CPA, but it can be done.

Summary of the discussion: Even if we're not willing to say that it's a requirement to "layer" policies, it may still be useful or necessary to "group" policies. Need some assurance that someone is enforcing each of the policies.

The need for a semantic model in which to configure, manage, and interpret (including in combination) groups of policy staments.

Summary: This seems uncontroversial, given the previous point.

The need to allow for (perhaps decentralized) evolution of the policy framework or statement model.

Ron: In cases where there are relationships between things (like the "pipelining" of encryption vs. signing, or when you have two policies about the same data field), there may be some complexity that the solution needs to handle. how might the introduction of a new constraint in the semantic model affect existing constraints? Related to version control. Policies may be out-of-date. Policies may get relaxed Jerry suggests that (when a request does not conform) the provider simply returns its policy. Ron: When a request is made in a fashion that is inconsistent with the current policy of the recipient, we need a way for the recipient to inform the client that it is operating with out date policy info. Perhaps we need to revise the SOAP protocol or profile its use so that this type of feedback need not be an application layer concern.

Jerry: We need a denotational semantics for our language.

Ron: Do we have any requirements about handling constraints (in the sense of limitations) imposed by the SOAP invocation model? Jerry: You can creatively use the mechanisms SOAP does contain by the client sending an encrypted message that it knows the responder can't decrypt, causing the responder to return its policies for a correct message. Frank: This is like the familiar fake-ping mechanism.

2:15-2:30 (11:15-11:30 PST) Break/Lunch (taken 2:30-2:50)

2:30-4:00 (11:30-1:00 PST) Presentation and collaborative development of Work descriptions suitable for inclusion in the charters of one or more forums.

Rob - Let's not reinvent the wheel. Get familiar with existing work (EbXML, XACML, WS-Policy-*).

Rob - emphasizes need for use-case analysis after a TC is chartered.

Ron combines requirements from Tim's presentation with his list of work items and rationalized the result.

(some folks planning to leave early so we sut problem description short and moved on to the next agenda item)

3:30-4:30 Charter Discussion

What forums (eg. TCs), if any, do we perceive as the appropriate host for a particular piece of work?

Definition of next steps:

How do we transfer problem description(s) to the appropriate forums, including potentially by deciding to charter a new forum?

Discussion of forum specific requirements, collection of advocates, sponsors, etc, to meet the requirements of the standard organization.

Frederick asks if we need to consider IPR issues. Everyone agreed. Ron has candidate language for an OASIS charter addressing the need for a royalty-free specification.

Jerry asks if we are planning to define a language from scratch. If we like XACML or WS-Policy we don't have to define a new language.

Rob suggests we can't predict what the work products are. Ron suggests that Attachment is the subject of an existing TC. Rob suggests chartering a Discussion List, rather than going directly to a TC.

Various approaches to taking on the work were discussed:

- Further ad hoc requirements-gathering meetings
- An OASIS discussion group
- An OASIS TC to both define specific flows/use cases and develop the solution
- Get more input from other efforts on chartering before proceeding

One concern is that the discussion so far has been a little biased towards security policy. It's important to get wider input, for example from web services management efforts.

Discussion of the IPR question. Ron feels that the other major participants will not participate until an RF IPR regime is established. Ron presented a wording for a TC IPR clause.

Ron presents a template charter.
Fairly broad sense that it an ineffective use of our time to work on writing charter wording as a group, but that sample text should be
sent our and collaborated on in email, before a next teleconference.

4:30 Wrap up

Decided that some of us will propose a draft charter, which will be circulated by email before being discussed in a teleconference, and then published more widely for comments and to determine support for submission.