2006 CCRTS

THE STATE OF THE ART AND THE STATE OF THE PRACTICE

An architecture for experimenting with secure and dynamic Web Services

Topics: C2 Experimentation, C2 Architecture, Information Operations/Assurance

Authors: Rolf E. Rasmussen, Anders Eggen, Raymond Haakseth

Point of Contact: Rolf E. Rasmussen

Organization: Norwegian Defence Research Establishment (FFI)

Complete Address: P.O. Box 25, NO-2027 Kjeller, Norway

Telephone +47 63807000 / Fax +47 63807449

E-mail: rolf.rasmussen@ffi.no; anders.eggen@ffi.no; raymond.haakseth@ffi.no

Abstract

Service Oriented Architectures (SOA) implemented by Web Services are promising technologies for dynamic information sharing between military units. Among the many important requirements in such environments, are dynamic discovery of services and information exchange using publish/subscribe. In addition to addressing these two challenges, we have looked at related end-to-end security solutions. In order to gain some hands-on experience with available technologies, we have built an experimental demonstration system that is referred to in the discussions.

The Service Registry was based on UDDI, with additional functionality in areas like run-time discovery and termination policies, geographical search and security. To obtain end-to-end security we have focused on XML and Web Services security solutions with the addition of XML Security Labels and a Public Key Infrastructure.

This paper describes an experimental implementation of secure and dynamic Web Services. It briefly surveys the theories and standards that our work is based on, and argues that publish/subscribe is an important technique for military purposes. Further, it describes some considerations and delimitations that had to be made, and finally we present our preliminary evaluation of this approach in the direction of secure and dynamic SOA using Web Services.

1 Introduction

The Norwegian Defence has adopted the principles of Network Based Defence (NBD) – which may be compared to Network Centric Warfare and Network Enabled Capabilities – as a major guideline for future development work. At FFI (Norwegian Defence Research Establishment) we have for several years been doing experimental research work in order to clarify the needs of NBD in the future.

Among the most important core functionalities within NBD is dynamic sharing of information. Operational aspects of NBD have been focused in experiments described in [9] and [22]. This paper describes an experimental effort which is primarily technologically focused, but having support of NBD ideas as the overall goal.

Security is often thought of as a challenge with respect to NBD, making sharing of information difficult. In our experiment we focus on application-level end-to-end security, as described in [7] and also highlighted as the long term goal in the NATO NEC Feasibility Study [16]. The use of end-to-end security solutions does not exclude additional use of traditional network and transport level security, but in this paper the latter will not be emphasized.

The military context of our experiment is the process of compiling a situational picture, and sharing it between military units. An NBD-related description of this process can be found in [11]. Technically the system is based on a synthetic environment and a number of autonomous Picture Compilation Nodes (PCNs). The PCNs are interconnected in a peer-to-peer manner [6], but connections are individually configurable by users to form desired information exchange structures.

This experimental implementation is founded on the use of Web Services and the variety of open standards that exist or are being developed in that area. The idea is to use leading edge technologies to build a system that can be evaluated against military criteria. Starting in the lower end when it comes to time and resources, this should be regarded as a simplistic proof of concept. The development of good operational criteria for military evaluation may very well be a subsequent activity. This paper will consider only simple technological evaluations.

Besides security, bandwidth restrictions are very much a limitation when it comes to fully deploying NBD principles. Those restrictions will in most cases remain, and as a means of reducing network traffic, the concept of publish/subscribe may be valuable. Simply speaking, publish/subscribe means that only those who have subscribed to the information will get it. As opposed to a general ”push” mechanism there are obvious benefits. And pure ”pull” principles are not able to notify listeners when events occur.

This paper describes the results of our work within this area. In section 2 we give an overview of the theoretic basis and the standards that our work is based on. UDDI has mostly been used for design-time service discovery – while our focus is on run-time, i.e. dynamic service discovery. The security part surveys SOAP-level security as well as the principles of XML labeling and the use of PKI. For implementing publish/subscribe we chose specifications from OASIS. The MIP data model is being used to define the actual data that is exchanged externally.

Section 3 describes our implementation at an architecture level, not going into details in general, but with a focus on the areas where shortcuts had to be made in order to make it work.

In section 4 we discuss each of the technologies used, highlighting added value and pointing to known and/or potential problems for the implementation.

Section 5 evaluates our goal of implementing dynamic and secure SOA using publish/subscribe and MIP for data exchange, before we give a conclusion in section 6.

2 A brief survey of relevant theories and standards

This section introduces the underlying principles that constitute the theoretic foundation of our experiment implementation.

2.1 Dynamic service discovery

A Service Oriented Architecture (SOA) may be described as ”*A collection of services that communicate with each other*” (simplification of the definition given in [3]). In NBD, the ultimate goal is that all resources in the network are accessible as services. The NBD infrastructure itself can be thought of as an advanced SOA. Being able to find these services and invoke methods on them, in most environments, is vital to a system operating in a NBD. Not only static environments, where service availability is stable, need to be supported. Dynamic environments will also occur, where services and even networks may come and go in a non-deterministic fashion.

Figure 1 illustrates the “SOA triangle” with the basic “publish-find-bind” operations using Web Services
.

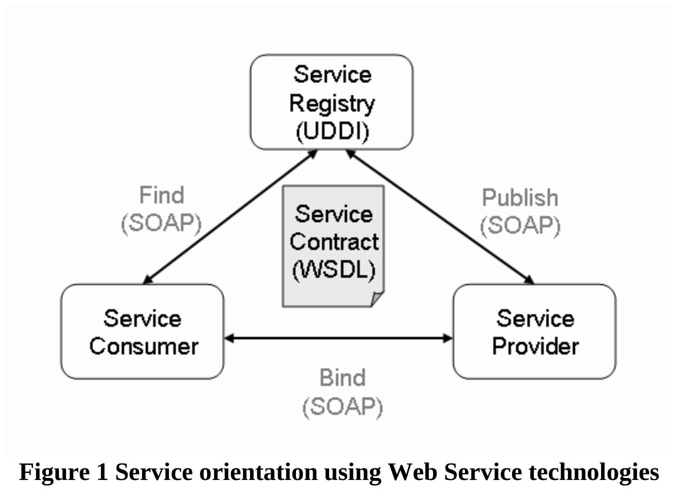

Figure 1 Service orientation using Web Service technologies

Lookup services exist in many different technologies (e.g. RMI, Jini, JXTA). For Web Services the standard lookup registry is called Universal Description, Discovery and Integration (UDDI) [25]. UDDI was chosen as the basis for our experiment, although that leaves us with a need to develop additional functionality to satisfy our requirements in the following areas:

- Security
- Service termination policies
- Extended search capabilities

Architecturally this extra functionality is realized by developing an Abstraction Layer and placing it in front of the UDDI registry. That means no clients will access the UDDI directly – the Abstraction Layer will stand in between.

The internal data model of UDDI is shown in Figure 2. Note that tModels may be used to hold metadata about services and entities, allowing for customization of registry contents.

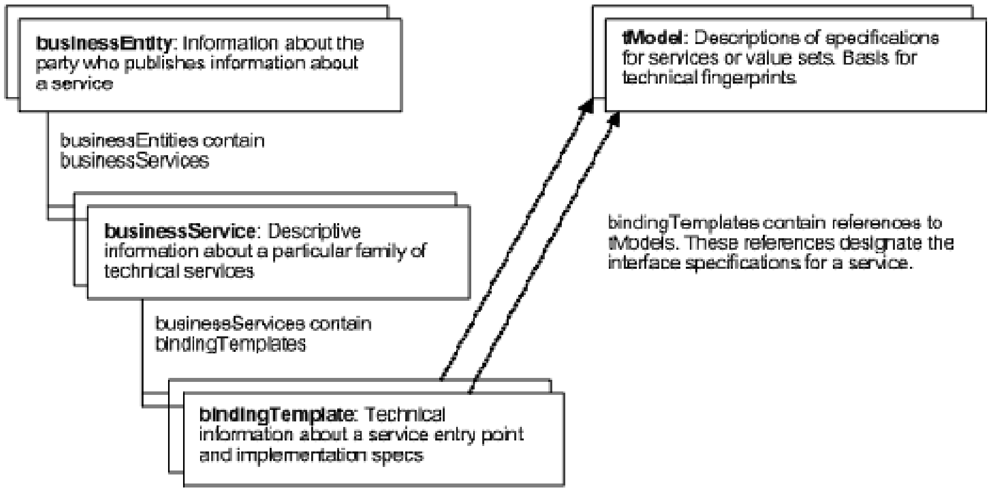
businessEntity: Information about the party who publishes information about a service
businessEntities contain businessServices
businessService: Descriptive information about a particular family of technical services
businessServices contain bindingTemplates
bindingTemplate: Technical information about a service entry point and implementation specs
tModel: Descriptions of specifications for services or value sets. Basis for technical fingerprints
bindingTemplates contain references to tModels. These references designate the interface specifications for a service.

Figure 2 UDDI Data Model

2.2 Securing Web Services

All of the Web Services specifications are based on XML and most often the use of SOAP messages. Therefore XML security specifications may be used for securing the different Web Services components.

Many specifications have been written for securing XML documents. Some of them have become standards. The major standardization organizations in this area are the W3C [26], OASIS [19] and IETF [13]. In addition Microsoft and IBM have developed the Web Services Security Road Map [27], which describes a set of security specifications building on the OASIS WS Security standard [18].

The OASIS WS-Security standard specifies how to extend the SOAP message header in order to achieve message integrity, confidentiality, authentication of originator and replay protection. It is based on the use of the security standards XML Signature [30] and XML Encryption [29] and supports a variety of security token formats (e.g. X.509, SAML and XrML).

What is missing in the wide variety of XML specifications and standards is an XML specification for security labeling of information objects. Security label specifications have earlier been developed for X.400 messaging (X.411 [28]) and SMTP (IETF

S/MIME ESS [12]) and these may be used as a basis for the development of an XML Security Label specification.

The following bullets outline the security functionality that is to be developed. It will be further described in section 3.3.

- All SOAP messages are attached a security label, encrypted and signed
- A "Domain XML Guard" filters all SOAP messages leaving the domain based on the security label
- All advertisements in the Service Registry are attached security labels and signed before storage
- Before any notifications or UDDI records are sent to a requestor, her security privileges are checked against the security label of the information objects.
- A PKI and an LDAP Directory are used for providing the security infrastructure for exchange of certificates and certificate revocation lists.

2.3 Publish/Subscribe

The publish/subscribe pattern is well known and has been used within several different technologies. For this work the obvious choice of technology was Web Services related. An interesting alternative that we did not work with, is defined by Object Management Group (OMG) and is called Data Distribution Service [20]. It is related to CORBA/IIOP in the lower level descriptions. Its higher level descriptions may be considered implementation independent.

We have chosen to rely on the specifications given by OASIS, namely *WS-Topics* and *WS-BaseNotification*, both described in [17]. The same reference also includes *WS-BrokeredNotification*, which we have not used so far.

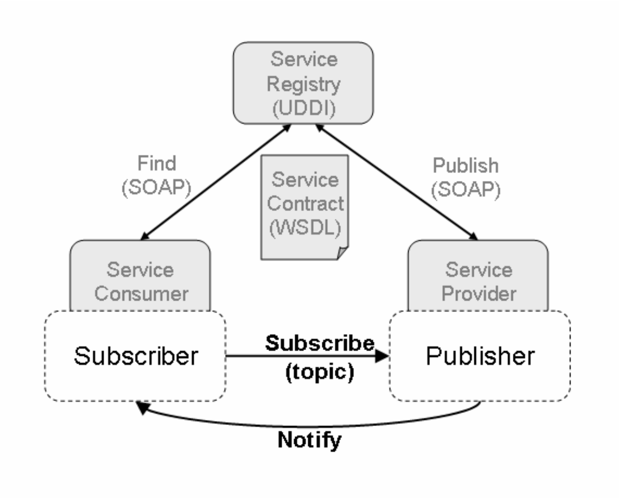

Figure 3 Publish/subscribe basic elements

As shown in Figure 3 the publish/subscribe pattern fits well into the SOA and Web Services picture from Figure 1. The basic elements of publish/subscribe is that a Service Consumer performs a "subscribe" operation for a given topic on a Service Provider, resulting in a series of "notify" operations the other way. The notifications are the actual deliveries, and the notification messages include the information subscribed to.

Prior to the “subscribe” operation, there is a service discovery phase as described in section 2.1.

There is some ambiguity in the use of the term “publish”. In this paper, it means “publishing the service in the registry” – i.e. making it available for others to discover and use. But you may find in literature the term used to indicate the event-based distribution of information (during a subscription session) that in this paper is called “notification”.

2.4 Data exchange format

The data exchange mechanisms of a secure SOA using publish/subscribe are independent of what information is exchanged. In a military context like this, we found it appropriate to make use of the data model defined by the Multilateral Interoperability Programme (MIP) [15]. The model has been developed under many years, starting as a land-oriented model, and later expanded to cover joint environments.

The MIP Data Exchange Mechanism (DEM) is essentially a replication mechanism between similar MIP-databases. This is not what we intend to do, so there is a need for us to define a suitable subset of the MIP model, and do information exchange using XML Schemas containing those subsets.

We need some sort of message oriented exchange mechanism, and the idea is that the object-oriented (OO) XML version of MIP may provide data structures that will be easy to use for demonstration purposes. The process of deriving an experimental exchange format from the OO/XML MIP is described in section 3.4.

3 Implementation considerations and limitations

3.1 Service Registry

To reduce efforts and ensure quality, we chose to buy a commercial UDDI registry from Systinet [24]. No open source implementation was UDDI version 3 compliant at that time. Version 3 implies mechanisms for signing the entries, which was one of the requirements from the security viewpoint.

So far there are no amendments to the Systinet Registry itself. But we needed to develop an Abstraction Layer to place in front of the registry, to handle additional functionality. The Abstraction Layer, as described in section 2.1, is illustrated in Figure 4.

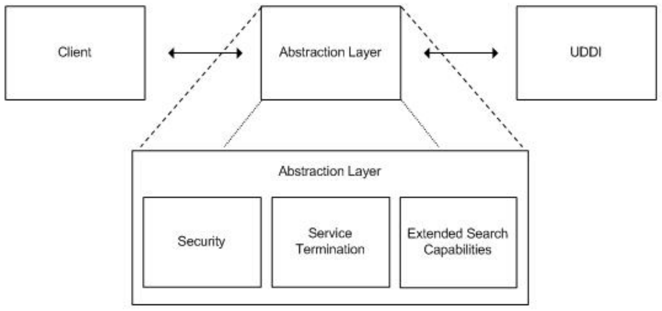

Figure 4 UDDI Registry with Abstraction layer

The security functionality of the Abstraction Layer is described in section 3.3.

Extended search capabilities will in our case mean geographic search. By using geographic search, it is possible to discover services covering a given position or area. We use this as an example of a typical military requirement that is not covered by UDDI. The geographical coordinates must be stored in tModels inside UDDI. To perform a geographic search, the abstraction layer needs to conduct a sequence of inquiries to UDDI.

Run-time discovery implies the ability to de-register services that disappear without performing the necessary cleanup in the registry. For that, the abstraction layer needs functionality to handle termination policies. In essence, they will monitor the presence of all services in the registry, and take specified action when any of them are found inactive.

An extra feature that we chose to implement in the abstraction layer, is the document-based service-publishing option that is explained below. This option represents an alternative where the publishing client does not have to detail the signing of the UDDI content.

In order to keep implementation work focused on the experiment goals, we made a number of simplifications in the demonstration setup and environment. In the area of dynamic discovery, those simplifications were:

- A common (well-known to all systems prior to execution) WSDL library containing the abstract part of the WSDL
- Services are only published through our “special” document-based service-publishing. There, the abstract part of the WSDL is retrieved from the library, while implementation-details like endpoint, port, etc are given as parameters and stored in the UDDI registry by the Abstraction Layer
- As binding we use “SOAP over HTTP” as the one and only binding known by all systems
- A common topic library, containing an XML Schema (XSD) defining the output (notification payload) for each topic
- All BusinessEntities (see Figure 2) in UDDI are initialized prior to execution and remain unchanged, due to the fact that our focus is on handling the services

These points clearly limit the “dynamicity” of our implementation. On the other hand, they are necessary precautions in order to maintain focus within the limits of time and resources. This will be further discussed in sections 4 and 5.

3.2 Publish/Subscribe framework

Our chosen OASIS specifications exist as open source Java implementations. We made a choice between two, not substantially different ones, namely

- Apache Pubscribe [1].
- Globus Toolkit [8]

Our strategy was to use the open source implementation as a software basis, and identify places to hook in the necessary security modules. The high-level architectural picture was to establish a publish/subscribe “engine” that could be deployed wherever needed in our distributed system, and that would run the standard publish/subscribe mechanisms for us. The specific services (WSDLs) and topics (XSDs) would hopefully be quite simple to parameterize into the “engine”.

Persuaded by the extensive tutorials and documentation material included, we chose Globus Toolkit. Instructive examples was used as a basis and parameterized into using our own WSDLs and XSDs.

3.3 Security mechanisms

As described in section 2.2, the implementation uses a combination of several security mechanisms in order to achieve the goal of end-to-end security at the information object level. Each area is described in the following.

SOAP Security
All information exchange is done using SOAP messages. The security of the SOAP messages is based on the use of the OASIS WS-Security standard with extensions in order to include an XML Security Label. The security label (and other important fields) is bound to the SOAP message by a digital signature. The content of the SOAP messages will be compressed, encrypted, labeled and signed before transmission. Upon arrival the security will be validated and the originator may be identified in order to see if the message comes from a reliable source.

Security Labels and User Security Privileges
A security label is attached to the information objects to be secured. This Security Label gives flexibility in marking the information, and is an XML translation of the IETF S/MIME ESS [12] security label. It has the following fields as defined in [12]:

- **Security Policy Identifier**: A security policy is a set of criteria for the provision of security services. It indicates the semantics of the other security label components.
- **Security Classification**: A Security Classification may have one of a hierarchical list of values. The security-classification hierarchy is defined by the security policy in force. The basic security-classification hierarchy is, in ascending order: unmarked, unclassified, restricted, confidential, secret, top-secret.

- **Privacy Mark**: The content of the Privacy Mark may be defined by the security policy in force which may define a list of values to be used. Alternately, the value may be determined by the originator of the security-label.
- **Security Categories**: The Security Categories provide further granularity for the sensitivity of the information. The security policy in force is used to indicate the syntaxes that are allowed to be present in the Security Categories. Examples of security categories are "releasable to" and "eyes only".

Each user is issued a certificate (X.509), which is extended to include her security privileges. A user in this context may be a person, a role, an application or a process. The users are granted security privileges, which are compared with the security labels of the objects, which the user requests access to. This may be UDDI records of the service registry or notifications, which the user has initiated a subscription for. An illustration of the relations between object security labels and user certificates is given in Figure 5.

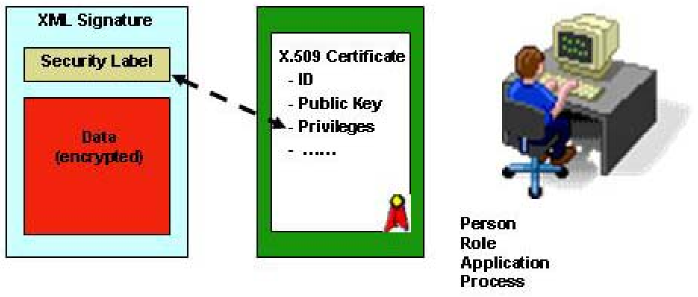

Figure 5 The Security Label bound to the information is compared with the privileges in the user's certificate in for access control to the information object

The security privileges component of the certificate[1] is defined as a "security label", and is named a "privilege label". The semantics of the fields of the privilege label is as follows:

- **SecurityPolicyIdentifier**: Identifies which security policy to be used to evaluate these privileges.
- **SecurityClassification**: Indicates the highest classification this user is allowed to access.
- **PrivacyMark**: Not Used.
- **SecurityCategories**: Indicates which nation, alliance, Role and/or Community of Interest (COI) the user belongs to.

The use of XML Security Labels is proposed in [4]. How to associate security tokens with SOAP messages is specified in WS-Security 2004 [18].

[1] Placing the privileges in the certificate is not a dynamic solution in that one would need to issue a new certificate in order to change a user's privileges. This solution is chosen for simplicity.

Securing the UDDI registry

All records stored in the UDDI registry are labeled and signed in order to indicate their sensitivity and to protect them from being changed during storage.
UDDI v.3 defines APIs for access to the data within the service registry. Two of these are the Inquiry API, which is used for searching for records, and the Publish API, which is used for insertion and updates of records. In order to secure these interfaces and enforce differentiated access control on the stored records, we have introduced a security component called the System Protection Component (SPC) as part of the Security Abstraction Layer in front of the UDDI APIs (see Figure 4). This security abstraction layer will perform the WSS related security processing of the SOAP messages (authentication, signature handling and encryption), in addition to performing differentiated access control on the UDDI records based on the security labels of the UDDI records and the privileges in the user certificates.

Inquiry API

Access to the methods of the UDDI Inquiry API Set is restricted to users with a valid certificate, and the SOAP message carrying the inquiry needs to be correctly signed and encrypted. Access to the information in the result set of the inquiry is controlled comparing the security label of the UDDI records with the privileges in the user's certificate. The user certificate is retrieved from the distributed LDAP directory using the X509IssuerSerial and X509SubjectName from the signature of the incoming SOAP message. A list of UDDI records that matches the user's privileges is built and returned to the requestor in a compressed, encrypted, labeled and signed SOAP message. This process is illustrated in Figure 6.

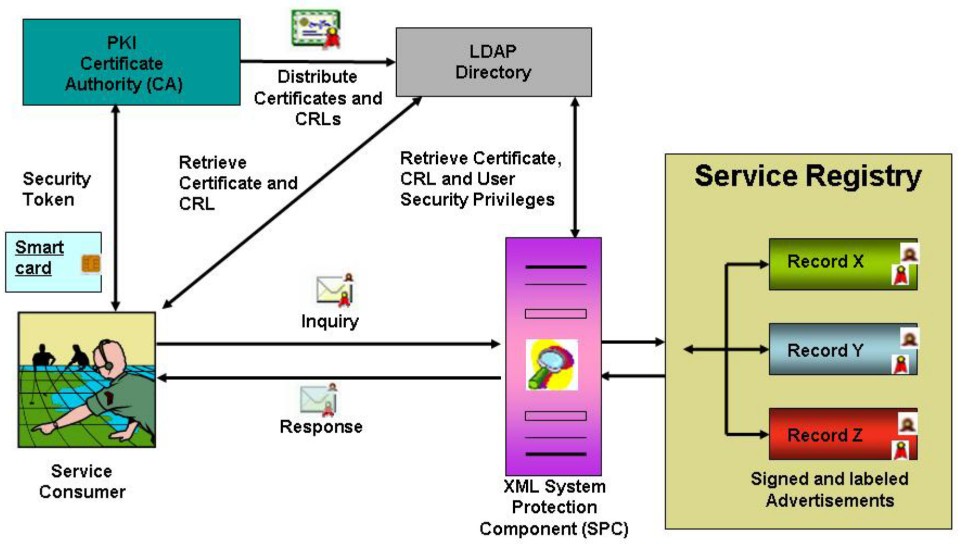
PKI Certificate Authority (CA)
Distribute Certificates and CRLs
LDAP Directory
Security Token
Smart card
Retrieve Certificate and CRL
Retrieve Certificate, CRL and User Security Privileges
Service Registry
Record X
Record Y
Record Z
Inquiry
Response
Service Consumer
XML System Protection Component (SPC)
Signed and labeled Advertisements

Figure 6 Securing the UDDI registry requires support of PKI and LDAP Directories for distribution of Certificates and CRLs

Publish API

In order to be allowed to publish to the registry, a publisher must be listed in the Access Control List of the registry. The publisher must first be in possession of an authorization token by requesting the UDDI Security Policy API. This authorization token then gives her the right to publish using the UDDI registry Publication API. This functionality is a part of the UDDI v3 software from Systinet [24].

All the SOAP messages carrying the UDDI requests and responses must be compressed, encrypted, labeled and signed correctly, if not the message will not be forwarded from the abstraction layer to the UDDI registry.

Several services may be published in the same publish message given that they have equivalent security labels. If the services have different security labels, they must be published using one publish message for each variation of the security label. The security label from the SOAP message used in the publish request, will be used to mark the records put into the UDDI registry. The SOAP message used to send the response will have the same security label as was used for the request.

Securing Subscriptions and Notifications

The WS-BaseNotification standard [17] makes a distinction between the roles NotificationConsumer and Subscriber, but in this context a Subscriber will also be the Notification Consumer. These restrictions influence the specification of the security functionality because it is not allowed to subscribe to a service on behalf of others.

When a subscription request is received in a SOAP message, security processing of the SOAP message is performed (as described above). The X509IssuerSerial and X509SubjectName of the SOAP signature may be used to fetch the certificate with the User Privileges from the LDAP Directory. The NotificationProducer will create a Subscription Resource for the Subscription. The User Privileges found in the certificate will be included in this Subscription Resource for matching against the InformationSecurityLabel of the Notifications.

When the Notification Producer has a notification to distribute, the NotificationProducer will match the InformationSecurityLabel in the SOAP message of the notification against the User Privileges registered for each subscription. For each of the subscriptions it identifies a match, it will issue the Notification to the Subscriber associated with the subscription.

Each Notification sent will be encrypted, labeled and signed by the Notification Producer. The classification of the InformationLabel attached to the SOAP message will be set to the highest classification of the included information.

XML Security Guard

An XML security label is securely bound to each SOAP message (using digital signatures) exchanged in order to allow for release control of the information passing between security domains. XML Security Domain Guards will inspect the SOAP messages being passed between the domains and control the signature and the security label attached, in order to make release control decision. This release control will be required to ensure that only information allowed to be released, is leaving the national domain.

Security Infrastructure

The security infrastructure consists of a Public Key Infrastructure (PKI) used for handling X.509 certificates and CRLs, and a distributed LDAP system. Each nation will have a Certificate Authority (CA) and trust is distributed amongst the systems via the exchange of the National PKI's Root certificates. The Root certificates are placed into a "trust file," which the systems use during digital signature verification. Only the PKI Root CA certificate is needed in the trust file because the CA certificate and the certificate of the system that performed the signature will be available from the LDAP Directory together with the Certificate Revocation Lists (CRLs).

The replication of the LDAP information is done periodically exchanging LDIF files [14] using Publish/Subscribe functionality[2]. By subscribing to the periodical update of each national LDAP server (using the WS-Notification specification), the LDAP information replicated will be protected by the SOAP security functionality, and not bypass the XML Domain Guard. This also shows how a non-XML legacy system like LDAP may be included using Web Services technology.

3.4 Data models and C2IS integration

The MIP model known as Command and Control Information Exchange Data Model (C2IEDM) version 6-15d was chosen as basis for our data structures. The work in this area was performed along two paths:

a) Using the Object-Oriented (OO) XML representation of C2IEDM, we needed to define a subset that covered the necessary information to be exchanged in our demonstration
b) Develop a mapping between the internal data model of the demonstration system (similar to a C2IS in real life) and the MIP element definitions, giving an experimental ”integration” of the two.

Path a) above was an interesting experience. Looking at the Entity-Relationship (ER) diagram of the entire MIP, there are approximately 240 entities. By combining expertise from MIP and our internal data model, they were able to select 30 of them as “core” entities necessary to represent the information internally present.

The ER diagram of these 30 may probably be the best human-readable description of the information exchange contents, but from a programming point of view, the ER diagram describes how to store this information in a relational database. Despite the relative simplicity of the reduced ER diagram, it was not obvious how to construct the messages that should be exchanged.

Again, we took the simple approach. Given that information to be exchanged was about physical object present in the battlefield, we said that for each `object_item` present in the model, we should include all relevant data connected to that entity. And using the OO XML MIP model, that is what you get; the `object_item` XML structure contains all other relevant structures.

This makes every `object_item` structure self-contained, and easily interpreted by recipients. An obvious downside is redundancy, e.g. the object type information that will be repeated over and over. Further enhancements of this approach has been defined as “out of scope” of the first version.

[2] thanks to our colleagues in Thales France for the idea of replication using publish/subscribe

Path b) above was handled as a “brute force” Java programming exercise. More sophisticated tools such as XML translations (XSLT) could probably also have been used in a good way. Valuable theory on how to integrate MIP with a C2IS in a more general perspective may be found in [23].

4 Discussion

In this section, we discuss the technologies used, highlight benefits and point to potential problems for the implementation.

4.1 Dynamic UDDI registry

Normally, a UDDI registry is used more at design-time than at run-time, stating a moderate starting point for UDDI in this dynamic context. Also, the Abstraction Layer that we found necessary to build, pinpoints several shortcomings of the standard UDDI registry.

There are currently a number of open source implementations of UDDI version 2, but very few available registries are version 3 compliant. The extra features available in UDDI version 3 include:

- Support for digital signatures
- A subscription API, enabling tracking of registry activity
- Support for multi-registry environment
- Better search API (single-step instead of multi-step queries)
- Possible to categorize bindingTemplate-entities

Generally speaking, any dynamic service registry containing metadata will have to meet the challenge of complying with metadata that in itself is dynamic. A good example is the position of a military sensor – an important piece of information – that may vary from being statically “bolted to the ground” to flying around as part of an airplane. Continuous updates to a registry will kill any performance in the latter case.

This challenge may be solved in the registry by storing the position as a URL to an external service that delivers the updated position. In that way the performance burden of frequent updates to the registry, is removed.

Advanced service discovery will eventually also involve semantics. A common vocabulary is necessary to enable extended use of metadata. The work described here does not take semantics into account. A good description of the main components in a service discovery architecture can be found in [5].

4.2 Security

The security concept described in sections 2.2 and 3.3 results from our work with [7], and is also in line with in the long term goal of the NATO NEC Feasibility Study [16], where the security is moved to the end systems.

An important goal of the NATO NEC FS is to introduce information domains based on user needs (Communities of Interest, COIs) and not on security classifications. There will be a long process in order to develop security policies, directives, procedures and management solutions that will allow for pure end-to-end security as described so far in this paper. Dynamic risk/threat management and adaptive protection levels may be something to look at. However, in the migration towards this long term solution, we still have to face the challenges of information exchange between security domains. One possible solution might be to try to automate the access control of the information passing between the security domains.

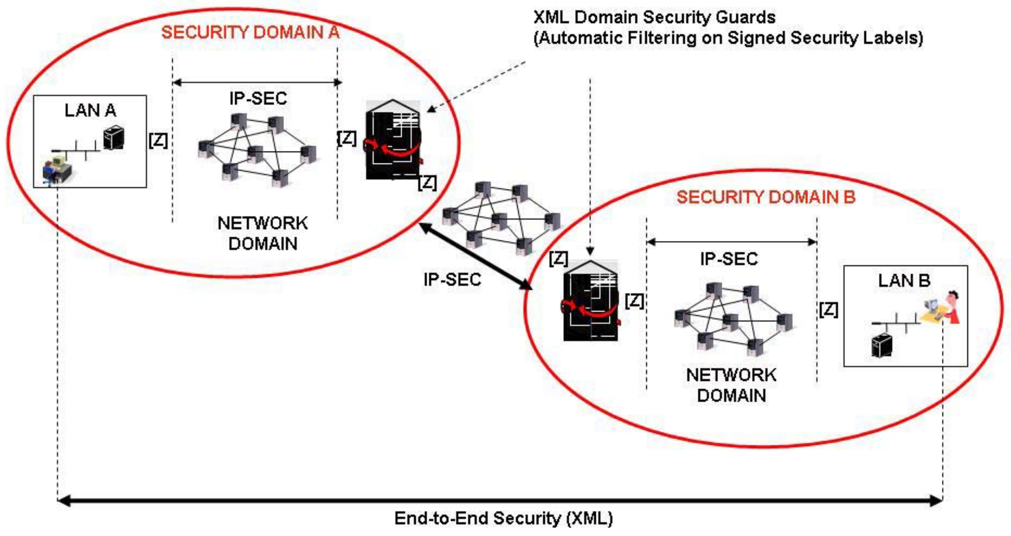

Figure 7 Security Domains

Figure 7 shows an example of information flow between the two security domains A and B. Within each security domain there is an application domain and a network domain. The information in transit (within each network domain), is protected by IP crypto devices forming secure Virtual Private Networks. Within the application domain the XML information is attached a security label, which is securely bound to the information using a digital signature. The signature and security label may be checked when leaving the domain, in order to control if the information is allowed be sent to the other domain according to the security policy in force. At the border of the other domain the signature and security label may be controlled again, in order to make sure that the information is sent by a trusted originator and that the information is labeled according to the security policy in force for this domain. The XML information may also be encrypted using XML Encryption for providing end-to-end confidentiality in order to enforce stronger need to know separation of the information and for better protection against CNA (Computer Network Attacks).

As an example, restricted information in a system high secret domain (A) may automatically be released to a restricted domain (B) based on the inspection of the security label (cryptographically bound to the information) by the XML Domain Security Guard. One of the greatest challenges with this concept of automatic filtering is probably not the issue of trusting the Security Guard itself, but to find a solution for the labeling and signing process that is trusted enough in order to allow for the Security Guard to do the automatic processing.

4.3 Publish/Subscribe

The basic principles of the publish/subscribe concept are easy to understand, and intuitively there are great benefits in delivering event-based updates to exactly those who have signed up for it. However, at this point in time we lack experience with systems based on the WS-notification specification. In theory, a subscription service is a proper superset of a traditional request/response web service, because you can always use the "get current" operation to achieve the one response. The downside, of course, is the resource and performance issue of having a publish/subscribe "engine" running to serve the subscription part.

One of the challenges within the publish/subscribe area is how to deal with variants of services. A number of variants can be imagined along the axis of notification frequency; the basic rule may be "*for every event*", another may be "*...but not more often than every x seconds*", and so on. Subscription parameters or eventually optional filtering mechanisms are necessary to avoid creating a large number of individual services to cover the variations.

Another issue is how using SOAP over HTTP will be handled by different Commercial Off The Shelf (COTS) technologies when it comes to low level use of sockets. Default HTTP behaviour is to "piggyback" the SOAP response with the HTTP ACK to the sender's already open socket. Given a half duplex asynchronous WAN environment, chances are that the sender does not wait for the ACK. In theory, the result is undefined, and must be focused in future work.

5 Evaluation of our technology goals

In today's military solutions for information exchange, the information flow is typically defined by a set pattern from sender to recipient. It is pre-planned and pre-configured, and changes normally require manual assistance. Therefore we need new concepts in order to achieve more flexibility and adaptivity. The benefits of using this secure and dynamic SOA solution in a military environment are as follows:

- Military resources can be made available as services, that may be accessed over a communication infrastructure
- Information is characterized by metadata and published in the network
- Efficient discovery, downloading and subscription to relevant information
- Faster deployment of new technology and functionality
- Dynamic reconfiguration of functionality within a relatively short timeframe
- Integration of functionality over different networks and heterogeneous technologies

The increased information sharing in SOA may lead to increased vulnerability if security is not properly integrated. The situation of today is that separate networks protect information of different classification using physical, cryptographic and administrative separation. Introduction of security mechanisms which allows for dynamic and seamless exchange of information between units, will be a challenge in NBD. IP level security will give confidentiality between systems, but will not prevent unauthorized access from within the systems or LANs. Computer Network Attacks (CNA) will focus on attacks behind the firewalls (crypto devices) within the

LANs/Systems. Therefore, end-to-end security services are required in order to secure the information in the NBD systems and LANs.

The overall technology goal has been to try out the combination of secure and dynamic publish/subscribe Web Services for military purposes. This paper covers the initial part of the implementation process. That is, the architecture is verified and discussed, but the experiences and results from running the system in experiment mode are parts of future work.

Each part of the combination (secure, dynamic, publish/subscribe) has to some extent been proved viable as separate technologies. The combination of them may introduce new problems. So far no major showstoppers have been identified, but one can think of potential problems in the following areas:

- Mutually exclusive behaviour at the technical level. As well as flexibility and security are well known as “opposing forces”, there may be incompatibilities between the technologies involved that appear only at run-time.
- Use of COTS and open source products may introduce unexpected problems
- Performance issues must be expected. All the three above mentioned parts require a lot of processing power per se (signing, encrypting, registry-polling for activity, etc.).
- Scalability is an important potential issue.
- Bandwidth consumption must be evaluated.
- Military conditions may expose previously unexpected challenges.

Use of the security technology described in this paper depends on adequate security policy and management procedures, which are assumed to be in place. So the evaluation must be held at a technical level, and from that perspective the outlook is still promising.

At this point in time, our evaluation of the concept has to be preliminary. Future experiments will bring more results as basis for a more detailed evaluation.

The work described in this paper has been performed while being member of the NATO RTO IST-061 research group. The group has developed a common set of specifications, and intends to perform a technology demonstration during NATO Coalition Warrior Interoperability Demonstration (CWID) at Camp Jørstadmoen, Lillehammer/Norway in June 2006. This experimental implementation is registered to be a part of the Norwegian contribution. We believe that NATO CWID 2006 will be an excellent opportunity for a proof-of-concept on secure, dynamic publish/subscribe Web Services.

6 Conclusion

The experimental implementation described in this paper covers several interesting technologies. Each of them is valuable contributions to the effort of solving the "flexible but secure" requirements of NBD. In combination, they have the potential of solving many of the NBD challenges. Shortcomings and limitations have been identified and discussed, but so far there are no major showstoppers.

Regarding the use of UDDI as a dynamic service registry, several issues are identified. But among standardized specifications we are currently not able to find a better alternative than UDDI. Hopefully the future will bring solutions that not only take into account the dynamic aspects, but also allow for shared taxonomies and extended use of metadata and other semantic technologies. The ultimate requirement down that line seem to be what could be called a "*Secure, Dynamic and Semantic Service Discovery*".

Web Services and XML are extremely flexible tools. On the lower tactical levels there may not be sufficient bandwidth and processing power for their normal use. This paper does not answer questions as to pinpoint those limits. Hopefully, the implementation described here may be used as an experimental tool to gather results in the future.

Finally, it is necessary to state that technology is not everything. The technical solutions described here to secure the information exchange, are not completely in line with current security policies. To achieve the full potential of NBD, there is a need to revise and re-work security policies. And there are important challenges in the area of security management. Knowledge of current technological possibilities should be a major input to future work on security policies.

The conclusion to the work described in this paper must be that these technologies constitute a substantial potential for the construction of an NBD. The experimental implementations outlined here look very promising, and should be pursued.

References

[1] Apache Pubscribe: http://ws.apache.org/pubscribe/

[2] Booz Allen Hamilton (2004): *Service Discovery Core Enterprise Services - Concept of Operations* - Version 0.4 (Pilot) http://www.w2cog.org/documents/NCES_SDCES_CONOPS_v0.4.pdf

[3] D. K. Barry (2003): Web Services and Service-Oriented Architectures - *The Savvy Managers Guide*, Morgan Kaufman Publishers, San Francisco, USA.

[4] Eggen A, Haakseth R (2005): *A Survey of solutions for end-to-end security when using JXTA or Web Services*, FFI/NOTAT-2005/00332.

[5] Gagnes T, Bjørnstad R, Langmyr A (2006): *An Architecture for Service Discovery in a Network Based Defence*, FFI/NOTAT-2006/00115

[6] Gagnes T, Bråthen K, Hansen B J, Mevassvik O M, Rose K (2004): *Peer-to-Peer Technology – An Enabler for Command and Control Information Systems in a Network Based Defence?*, Proceedings of the 9th International Command and Control Research and Technology Symposium, San Diego, USA, 2004.

[7] Gagnes T, Eggen A, Hedenstad O E, Rasmussen R, Sletten G (2005): *Information and integration services for command and control – future Network Based Defence*, FFI/REPORT-2005/03584. (In Norwegian).

[8] Globus Toolkit: http://www.globus.org/toolkit/

[9] Hafnor H, Olafsen R (2005): *Ad hoc Organization of Distributed Picture Compilation and Support for Situation Awareness in Network Based Defence – An Exploratory Experiment*, Proceedings of the 10th International Command and Control Research and Technology Symposium, Washington, USA, 2005.

[10] Hansen B J, Mevassvik O M, Bråthen K (2003): *A Demonstrator for Command and Control Information Systems Technology Experimentation* Proceedings of the 8th International Command and Control Research and Technology Symposium, Washington, USA, 2003.

[11] Hansen B J, Mevassvik O M, Bråthen K, Rose K (2004): *A Picture Compilation Concept for Network Based Defence*, FFI/REPORT 2004/00983, Unclassified

[12] IETF Enhanced Security Services for S/MIME: http://www.ietf.org/rfc/rfc2634.txt

[13] IETF: http://www.ietf.org/

[14] LDAP/LDIF: http://www.openldap.org/doc/admin22/dbtools.html#The%20LDIF%20text%20entry%20format

[15] MIP: www.mip-site.org

[16] NATO NEC Feasibility study, version 2.0 (2005), NATO Unclassified

[17] OASIS WS-notification: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn

[18] OASIS WS-security: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

[19] OASIS: http://www.oasis-open.org/

[20] OMG DDS: http://www.omg.org/technology/documents/formal/data_distribution.htm

[21] PKI: http://www.ietf.org/html.charters/pkix-charter.html

[22] Rasmussen R, Gagnes T, Gustavsen R, Hafnor H, Hansen B, Haakseth R, Mevassvik O M, Olafsen R, Rose K (2004): *Exploratory Experiment "Ad hoc Organization of Picture Compilation" Conducted during Blue Game 2004: Evaluation Report*, FFI/REPORT-2004/01940. Norwegian Defence Research Establishment (FFI).

[23] Schmitt, Michael (2005): *Integration of the MIP Command and Control Information Exchange Data Model into National Systems*, Proceedings of the 10th International Command and Control Research and Technology Symposium, Washington, USA, 2005.

[24] Systinet: http://www.systinet.com/

[25] UDDI: http://www.uddi.org/

[26] W3C: http://www.w3.org/

[27] WS Security Roadmap: http://www.ibm.com/developerworks/library/ws-secroad/

[28] X.411: http://www.itu.int/ITU-T/asn1/database/itu-t/x/x411/

[29] XML Encryption: http://www.w3.org/Encryption/2001/

[30] XML Signatures: http://www.w3.org/Signature/