SOA Pilot 2011 – demonstrating secure exchange of information between security domains

Raymond Haakseth

Norwegian Defence Research Establishment (FFI)

2 march 2012

FFI-rapport 2012/00117

1176

P: ISBN 978-82-464-2109-4
E: ISBN 978-82-464-2110-0

Keywords

Informasjonssikkerhet

Informasjonsmerking

Informasjonsutveksling mellom sikkerhetsdomener

Guard

Tjenesteorientert arkitektur

Approved by

Rolf Rasmussen	Project Manager
Anders Eggen	Director

English summary

The work described in this document is performed within the context of a SOA Pilot activity undertaken by FFI during the spring of 2011. The overall goal of the SOA Pilot was to show the benefits of Service Oriented Architectures (SOA) and the value added by exposing the information available in different systems through well-defined services, including information from legacy systems. This activity also included the secure exchange of information between security domains using a prototype guard.

Security domains are important constructs used to protect information from unauthorised access and prevent information leakage. The military security domain concept is based on the Bell-LaPadula model and this model can in short be described as allowing information to flow from a low to a high security domain, but not the other way in order to prevent information leakage. A high security domain can still contain information of lower classification. Even though this information can be valuable for users in lower domains and these users are cleared to view it, it cannot be accessed. At the same time the military organisations and thus also information systems are moving towards the concept of Networked Enabled Capability (NEC). In NEC the perhaps most important tenet is the need to share information or the right information to the right user at the right time. In this document a proposed solution for securely exchanging information between security domains is presented. The theory behind the solution is presented as well as a prototype implementation of the solution.

The proposed solution relies heavily on the concept of Object Level Protection (OLP). In short OLP consist of three building blocks; metadata, data and protection mechanisms. Information assurance metadata is used to describe the needs for protection of data, and protection mechanisms use the metadata as basis for their handling of the data. For instance confidentiality metadata attached to data can be used as a basis for a release decision made by a guard placed between a high and a low security domain. In the solution presented in this document, proposed NATO standards for confidentiality metadata (also called a confidentiality label) and the binding of metadata to data (metadata binding) are used. These proposed standards have been developed within the NATO RTO IST-068/RTG-031 research task group and FFI has been the editor and provided substantial contributions to them. The solution presented in this document also features a prototype guard capable of securely exchanging SOAP messages between domains developed by FFI.

Sammendrag

Arbeidet beskrevet i dette dokumentet er utført i sammenheng med SOA pilot aktiviteten som FFI var en svært viktig del av våren 2011. Hovedformålet med SOA piloten var å demonstrere nytteverdien av tjenesteorientert arkitektur (SOA) og den tilgjengeliggjøringen av informasjon gjennom kjente grensesnitt som er mulig med denne teknologien. Eksisterende kommando og kontrollsystemer ble derfor tjenesteorientert og informasjonen som disse innholder ble gjort tilgjengelig for potensielle konsumenter. En løsning for utveksling av informasjon ble også demonstrert ved bruk av SOA-tjenester mellom forskjellige sikkerhetsdomener.

Sikkerhetsdomener brukes i dag for å beskytte konfidensialiteten til informasjon. Sikkerhetsdomener og de tilhørende mekanismene skal hindre uautorisert tilgang til, og dermed også lekkasje av, informasjon. Kort beskrevet gjøres dette ved at informasjon kan flyttes fra et lavere gradert til et høyere gradert domene, men ikke andre veien. Samtidig beveger den militære organisasjonen og dermed også informasjonssystemene seg mot et Nettverksbasert Forsvar (NbF) hvor tilgang til rett informasjon til rett tid for rett bruker er svært viktig. Siden det kan eksistere informasjon i et høyere gradert domene som er av lavere gradering, og som kan være av nytte for brukere i domener med lavere gradering, er det behov for en løsning for å flytte informasjon sikkert mellom domener. Denne løsningen må ta hensyn til både behovet for beskyttelse av informasjon og behovet for å dele informasjon. Dette dokumentet skisserer en foreslått løsning for sikker utveksling av informasjon mellom sikkerhetsdomener. I tillegg skisseres teorien og grunnlaget for den foreslåtte løsningen.

Den foreslåtte løsningen baserer seg på bruk av prinsippene rundt objektnivå sikkerhet (Object Level Protection (OLP)). Kort kan dette forklares ved at hvert objekt blir behandlet av beskyttelsesmekanismer etter dets individuelle behov for beskyttelse, dette behovet er beskrevet ved hjelp av metadata. Beskyttelsesmekanismer gjør sine vurderinger basert på de metadata som er knyttet til objektet og ikke infrastrukturen slik som i dag. En slik beskyttelsesmekanisme kan være en guard som inspiserer all informasjon som skal flyttes fra høyt til lavt domene. I løsningen som er utviklet og skissert i dette dokumentet brukes en foreslått NATO standard for konfidensialitets metadata (også kalt konfidensialitetsmerke) og en annen foreslått NATO standard for å binde metadata til informasjonsobjekter. FFI har vært editor for disse, som begge er tatt fram av forskningsgruppen NATO RTO IST-068/RTG-031. En prototype guard-løsning som kan håndtere SOAP meldinger er også utviklet ved FFI.

Contents

1	**Introduction**	**7**
2	**Object Level Protection**	**9**
2.1	Binding metadata to data	10
3	**Architecture for Cross-Domain Information Exchange**	**12**
4	**Proposed NATO Standards for Information Assurance Metadata and Metadata Binding**	**15**
4.1	XML Confidentiality Label	15
4.2	XML Metadata Binding	18
5	**SOA Pilot 2011**	**19**
5.1	Scenario Overview	20
5.2	Software description	21
5.2.1	Prototype XML/SOAP Guard	22
5.2.2	NFFI Request/Response Service and Client	26
5.3	Applying XML Confidentiality Label and Binding to SOAP Messages	28
5.4	SOA Pilot Execution	32
6	**Future Work**	**35**
7	**Summary**	**36**
	References	**36**
	Appendix A Labelled and signed SOAP Message	**38**

1 Introduction

NATO and NATO nations have embraced the concept of Networked Enabled Capability (NEC) and this concept forms the basis for future development of both the organisations and information systems. One of the key enablers identified for the NATO NEC (NNEC) is Service Oriented Architecture (SOA) and the implementation of this. SOA and its use in military information systems is also the main focus of the FFI project 1176 "Tjenesteorientering og semantisk interoperabilitet i INI" and during the spring of 2011 this project was one of the partners in a pilot activity aiming to service orient existing military information systems. The pilot activity was known as the SOA Pilot.

The overall goal of the SOA Pilot was to show the value added by exposing the information available in different systems, including legacy systems, through well-defined services. Making information available in this way supports the development towards the need-to-share paradigm introduced by NNEC where information is available to whoever needs it, when it is needed. The move from the traditional and strict need-to-know mentality known from today's military systems to the need-to-share paradigm involves sharing information between users in different national security domains and also between allied or partner nations, civilian authorities and non-governmental organisations. At the same time as there is a drive to share more information, it is still important to protect information from unauthorised disclosure.

Demonstrating a solution for securely exchanging information between different security domains was thus an important part of the SOA Pilot. This document describes a proposed solution based on the use of the principle of Object Level Protection (OLP). A demonstrator implementation of the proposed solution was used during the SOA Pilot. The technical solutions and how these fits into the overall scenario are presented by this document, as well as the supporting concepts. It is important to note that none of the software, designs and architectures presented in this document has been formally evaluated or certified. They are all experimental and should be treated as such.

Traditionally security domains are created in order to protect the confidentiality of information, i.e. prevent unauthorised disclosure of information. In a military setting this more or less corresponds to the Bell-LaPadula Model [1;2]. According to the Bell-LaPadula a user (subject) is allowed to read information at her own or lower classification levels (read down), and she is allowed to write to a classification level equal or higher than her own (write up). A strict interpretation of Bell-LaPadula thus prevents information flows from a high to a low domain. However, in real life a strict implementation of this model is not feasible and there is a need for information to flow from high to low domains. It is not usual to have single level domains and as a result domains can contain information up to and including its own classification level, e.g. a secret domain can contain information of classifications unclassified, restricted, confidential and secret. This combined with the fact that users on lower domains may have a need to access the information of lower classification clearly indicates a requirement for information exchange between security domains.

In order to connect and exchange information between security domains, a cross domain sharing solution is needed. Figure 1.1 outlines the main challenges for such a solution. First of all it needs to protect the high domain and the information within from unauthorised information leakage. That is, only information that is allowed according to a given policy can be transported from the high to the low domain by the sharing solution. Second, the sharing solution must also ensure that no malicious code such as viruses, worms and other forms of cyber-attacks are passed to the high domain. The latter issue, although important, has not been the focus of the work presented in this document and will thus not be treated any further.

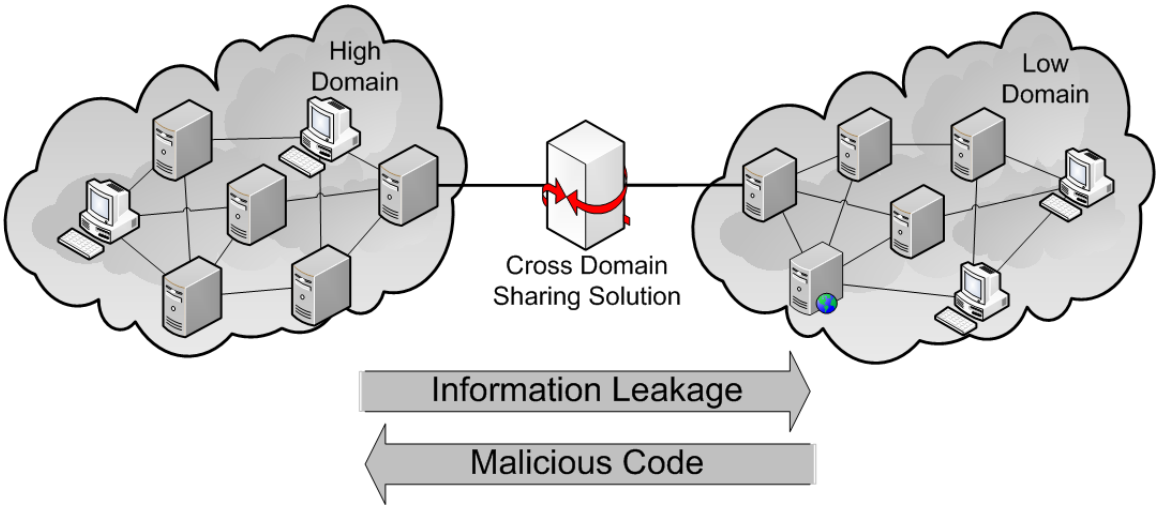

Figure 1.1 The sharing problem

Existing solutions for exchanging information between security domains are often limited either in functionality or bandwidth or both. The most common cross domain solution used today is air gaping. This involves a manual review of the information at the high side, storing the information on some sort of storage medium (e.g. CD or USB memory stick), and then moving this to a computer on the other domain. Moving information from the low to the high domain is done similarly. This process is very time consuming and the bandwidth thus becomes low, and it can also be susceptible to human errors. This cross domain sharing solution is also often denoted the swivel chair solution. A one-way data diode solution can be used to automate and speed up the flow of information from the low domain to the high, but this does not resolve the problem of securely and efficiently moving information from a high domain to a low. In addition, semi-automatic solution with a release function might be used and also fixed format guard solutions. These are however often very limited in functionality, and are not usable for the general purpose information exchange. In addition, it is very expensive to create and change these solutions due to the time consuming and costly process of certification and re-certification. What is needed is a general purpose guard solution that can provide automatic and secure information flow both ways between security domains.

The work presented in this document is heavily based on the results from the now terminated NATO RTO Task Group on XML in Cross Domain Solutions (NATO RTO IST-068/RTG-031). The overall goal of the task group was to improve information sharing between security domains

by facilitating a flexible infrastructure and utilising the power of the eXtensible Markup Language (XML). The group consisted of members from nine NATO nations (Norway, USA, United Kingdom, the Netherlands, Canada, Slovenia, Germany, Poland and France), the NATO C3 Agency and Finland as a partner nation.

The rest of this document is structured as follows; sections 2 through 4 presents the theory, concepts and standards that our proposed solution is based on. Section 2 describes the basic principle of object level protection that our proposed solution relies on. Section 3 explains the general architecture proposed for secure cross domain information exchange. This description is followed by a description of the two proposed NATO standards for XML Confidentiality Label and XML Metadata binding in section 0. Next, section 5 focuses on the contributions to the SOA Pilot and the prototype solution developed for secure exchange of information between security domains. This involves a short description of the scenario, software developed and execution. In section 0 the future work is outlined and finally section 7 summarises this document.

2 Object Level Protection

The concept of Object Level Protection (OLP) is a cornerstone in our proposed solution for exchanging information between security domains. It is also identified as an important part of the puzzle when realising and implementing the future NNEC [3].

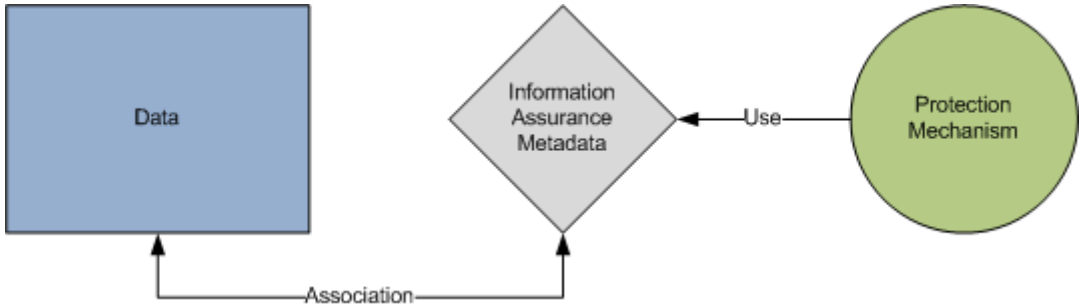

Figure 2.1 The concept of object level protection

Informally OLP can be described in light of the three main components and the relationship between them, as outlined in Figure 2.1. The main components are Data, Information Assurance (IA) Metadata and Protection mechanisms. By data it is here understood a finite piece of information such as documents, files, messages and such. Informally metadata can be defined as data about data and is used to describe some property of the data. Accordingly information assurance metadata is used to describe the need for protection, for instance confidentiality or integrity. Traditionally military systems have focused on the need for confidentiality protection and have used the three metadata attributes policy, classification and categories to describe this. These three together is also often known as a security label. There must be a relationship in form of an association between the data and information assurance metadata. This association simply says that this metadata belongs to this data (see section 2.1 for more details). The last component, protection mechanism, represents a generic mechanism that protects information.

Such a mechanism can for instance be an access control system or a guard. Using the OLP concept a protection mechanism performs its tasks by using the information assurance metadata associated with the data as an input and basis for decisions.

Use of the OLP concept provides some key benefits over the traditional approach. It first of all provides a larger degree of flexibility as metadata associated with the data can inform protection mechanisms how to handle the data. In the traditional approach the actions taken by protection mechanisms are pre-configured from policy, which are often dictated by the infrastructure where the data is stored or transported. This often does not reflect the actual need for protection. Also the use of OLP provides the opportunity for a finer level of protection granularity. Individual data objects may be handled individually according to the associated IA metadata, and not as part of a network or computer. In general the use of OLP provides a higher level of flexibility and can easier meet the future requirement for flexible information sharing than the traditional information assurance solutions. It should be said that some existing solutions, like the Military Message Handling System (MMHS) already apply some of the OLP principles.

2.1 Binding metadata to data

As described above the association between data and metadata is one of the key features of OLP. The way this association is implemented is of utmost importance in order for protection mechanisms to know which metadata belongs to which data and whether it can be trusted. There are several ways to implement the association between data and metadata and this section takes a closer look at some of the alternatives. The association is also known as a binding, and this terminology is used for the remainder of this document.

In general there are two types binding; loose and strong binding. A loose binding simply provides a link between the data and the metadata, while a strong binding is created by using cryptographic mechanisms like signatures to ensure the integrity of the binding. Strong binding is therefore also known as cryptographic binding. The principles presented by the reminder of this section apply to both types of bindings.

The first possible solution for implementing the binding is to include the metadata in the actual data object as outlined in Figure 2.2. This alternative provides an implicit binding by the fact that they are located in the same data object, e.g. the same file. The obvious advantage of this solution is that since the metadata is included in the data it is easy to identify which metadata to use and to ensure that it is unique. The disadvantage is that the data object would have to be modified, which is not always easy or even possible. In addition, changing the format of, or adding new metadata would require a redesign of the data object. Even though this might be feasible for some data formats, it is not a good solution for the generic case of applying a binding to any form of data and metadata. This alternative is thus not recommended as it is not always possible or desirable to change the data or data format. Using the same argument, it is not recommended to embed the data object in the metadata either. The same disadvantages apply, and in addition it becomes very difficult to add new types of metadata.

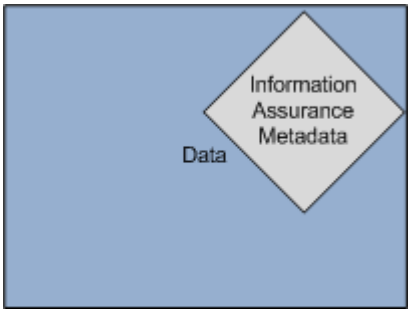

Figure 2.2 Embedding metadata in data

An alternative solution is to include a reference in the data object to the metadata, as outlined in Figure 2.3. This alternative has many of the same advantages and disadvantages, and more specifically there is still a need to modify the data in order to include the reference. However, in contrast to the previous solution, adding new metadata would only require adding a new reference and this reference might be standardised. If so, the need to change the data is now limited to including these standardised references. Even though this is a less severe modification it is still a modification to the data and the applicability of this cannot be guaranteed in the general case. Again this solution is not recommended for implementing the binding between data and metadata since it requires changes to the data object.

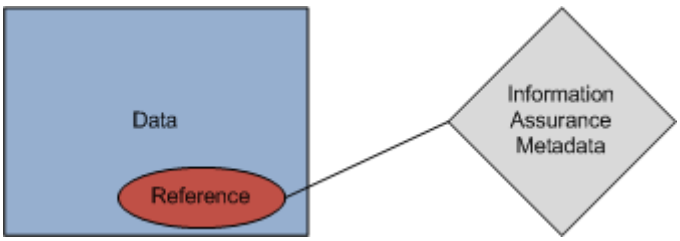

Figure 2.3 Reference to Metadata in Data

Alternatively a reference to the data might be embedded in the metadata object, as outlined in Figure 2.4. Again this implies that all metadata must embed this reference, and this might not be possible for all types of metadata. For instance older metadata types and the systems that use them might not so easily be modified. For this reason this solution is not recommended either.

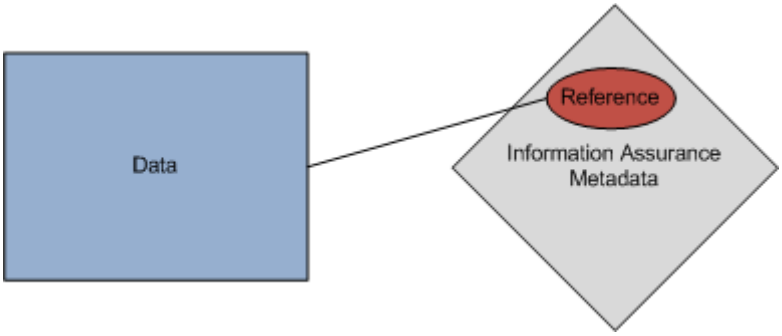

Figure 2.4 Reference to data in metadata

In order to support all (or at least most) types of data and metadata without having to modify either of them, a generic binding is needed. This can be done by introducing a separate binding object, as shown in Figure 2.5. This solution does not present any requirements for change on either the data or metadata. This is also the solution recommended by the NATO RTO IST-068/RTG-031 working group; please see section 4.2 for a description of the proposed NATO STANAG for XML metadata binding.

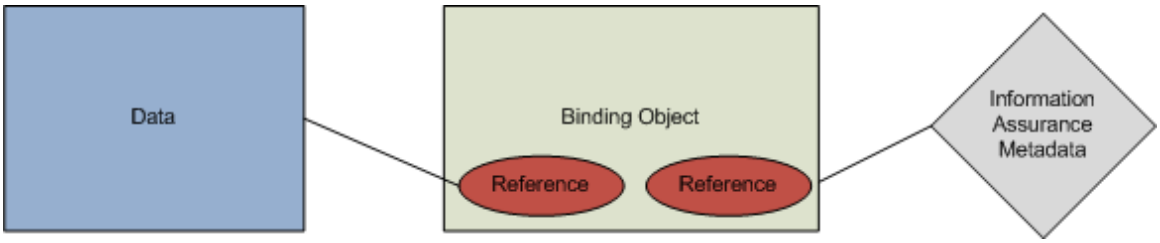

Figure 2.5 Reference to data and metadata in a separate binding object

With this solution there is still a need to specify how to use the elements. For instance where should the different components be stored and how could they be retrieved? And how could one assure the uniqueness of the binding? Standards and handling procedures must be established in the future to handle these issues. This might include general or file/message format dependent profiles describing how to use and include the binding objects. Dependent on the design and requirements, the binding object might for instance be embedded into the data (if a new data format is created or the existing data format can be adapted). A proposed NATO profile for binding [4] has been developed by the NC3A based on the results provided by the NATO RTO IST-068/RTG-031.

This section has focused on the generic binding mechanism and has not made any distinction between loose and strong binding. For loose bindings the solutions presented here can be used directly. To create a strong binding a cryptographic signature must be created. In general this signature must cover the data, metadata and binding in order to be effective. For the recommended solution of having a separate binding object, this would require a signature covering all three objects.

3 Architecture for Cross-Domain Information Exchange

In order to share information between security domains a solution is needed that addresses the challenges presented by section 1. With that in mind the architecture must prevent information from leaking to a lower domain and malicious code from entering the high domain. A simple view of the architecture proposed and demonstrated in the SOA Pilot is shown in Figure 3.1. In this general architecture, each domain is protected by a guard solution. The reason for protecting both domains with a guard is that it is not always clear which domain is the high and which is the low. Typically when the domains are under different administration, like two different nations, both would like to protect its own domain. In terms of OLP the guards become the protection mechanism. Information is labelled using the IA metadata and a binding mechanism, and the

guard is configured to release information of certain classifications. Before releasing any information, the guard compares the IA metadata to the policy in order to make a release decision. In theory, this should prevent information from leaking to the other domain. In addition, the guard must prevent malicious code from entering the domain.

It is important to note that the general guard architecture is not new and it has been utilised in for instance the Norwegian military message handling system (MMHS). The guard architecture is also similar to the Information Exchange Gateway (IEG) architecture developed within NATO. The guard architecture described in this section and the guard solution described in this document can as a result of this also be included as part of the IEG architecture.

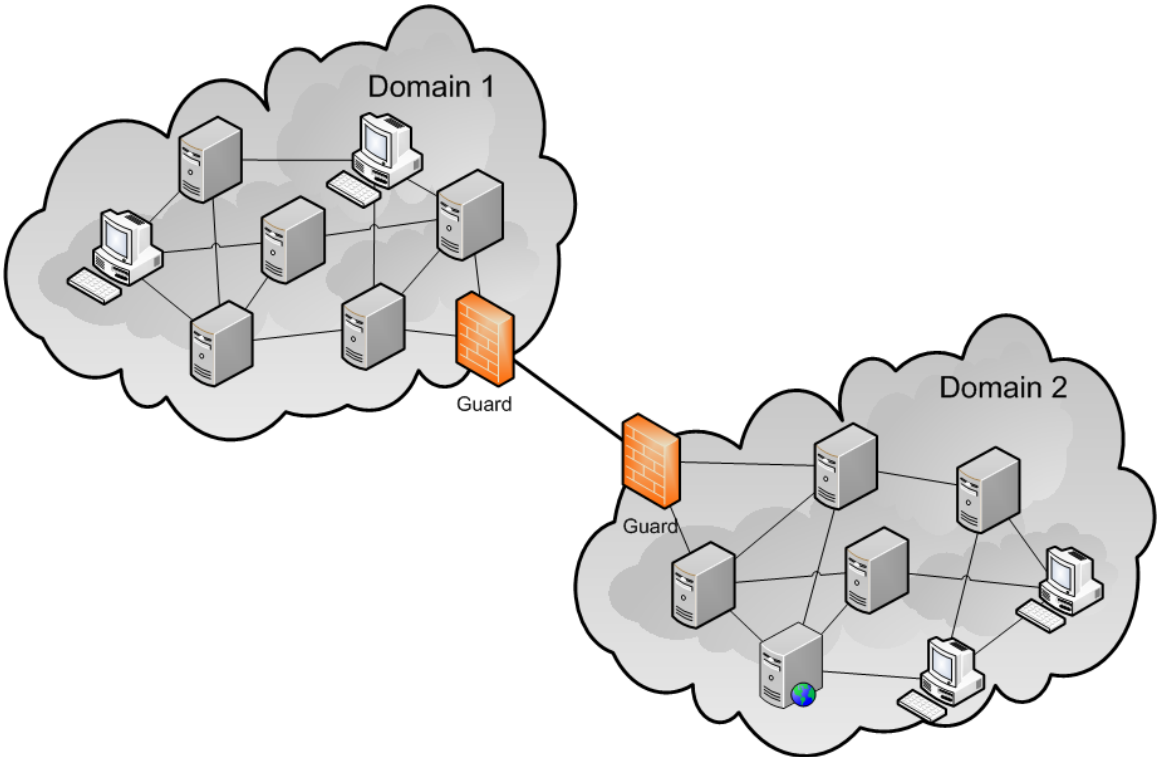

Figure 3.1 Architecture for cross domain information exchange using guards

Use of the guard architecture and the solutions described here would enable interconnection between security domains. However, which domains that can actually be connected will depend on the assurance level provided. In general the required assurance level is a result of the types of domains that are to be interconnected. For instance a solution connecting two unclassified networks would require a lower assurance level than a solution for interconnecting an unclassified and a secret domain. Achieving the highest level of assurance is both costly and time consuming, if at all possible. Assurance levels may be based on the Common Criteria standards[1].

The general architecture described can be deployed and used in the near future, and as mentioned above already is by some existing solutions. However these deployed systems are limited in

[1] http://www.commoncriteriaportal.org/ (29.12.2011)

functionality and can only exchange given formats or limited sets of information. In our proposed architecture any information can be exchanged between the security domains as long as it is associated (labelled) with the correct metadata and guards are used to protect the domain. These guards are placed at the border of networks and as a result one security domain more or less equals one network domain. In the future, the concept of Protected Core Networking (PCN) [5] may be used to build the transport infrastructure. Using this concept the responsibility for protecting information may be moved closer to the end systems, this involves both confidentiality protection and access control. The advantage of this approach is that access decisions can be based on the protection requirements of each individual object of information together with the users' credentials and not by which network the user or the information is currently situated. Two possible solutions for realising this architecture are shown in Figure 3.2. It is highly likely that the implementation of this long term vision would require using Multi Level Secure (MLS) systems to separate information within the end nodes. The time frame for realisation of such a solution is difficult to state, as it is dependent on tackling several challenges such as developing and certifying an MLS system. Revision of current rules and regulations may also be necessary in order to allow for it to be deployed.

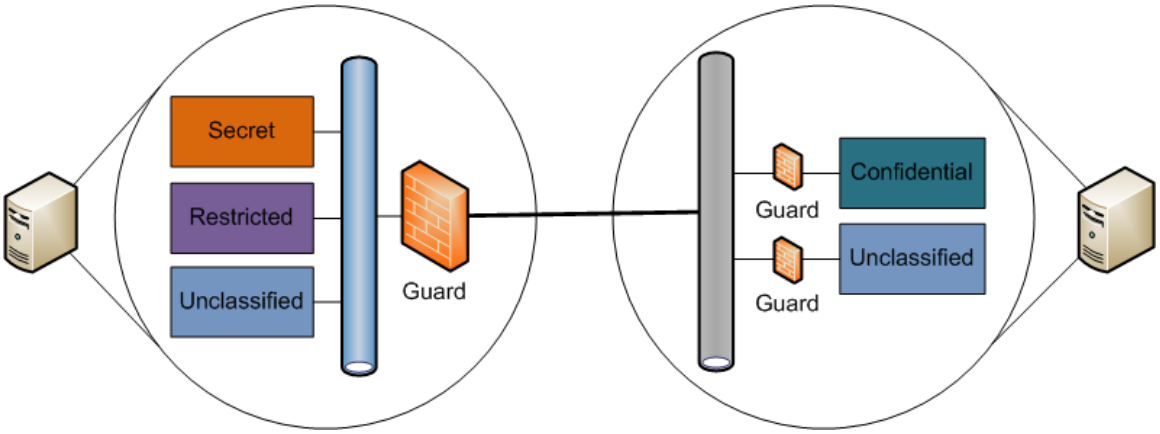

Figure 3.2 Architecture for cross domain information exchange without physical network separation

Introducing an architecture based on OLP and guards as outlined in this document, is important to achieve a higher level of information exchange between security domains. Guards deployed at the network borders are the only viable short term solution. Using a hybrid solution is a natural next step forward in order to reach this goal. A hybrid solution involves keeping the network separation and adding access control at the end systems. All these solutions have unsolved challenges, for instance exchange of user credentials (certificates, authorization etc.) across domain.

4 Proposed NATO Standards for Information Assurance Metadata and Metadata Binding

The now terminated NATO RTO Research Task Group on XML in Cross Domain Security Solutions (NATO RTO IST-068/RTG-031) explored the use of XML and related mechanism in order to improve the sharing of information in a military environment, including between security domains. As part of this work the group used the concept of OLP and defined both IA metadata and a metadata binding. The main result from this group was two proposals for NATO STANAGS, the XML Confidentiality Label and the XML Metadata Binding. Both were used extensively during the SOA Pilot activity and are thus described in this document. These standards are also included in the NISP (NATO Interoperability Standards Profile).

4.1 XML Confidentiality Label

The XML Confidentiality Label is a proposal for an XML formatted metadata that can be used to describe the sensitivity of information. Traditionally sensitivity of information has been described by the three attributes policy, classification and categories. Together these three attributes are often somewhat inaccurately named security label. The XML Confidentiality Label includes these and adds more attributes that can be used to express the need for protection. Since the former security labels only included sensitivity, or confidentiality, constraints, it was more appropriate to use the term confidentiality label for the new metadata. This section provides an overview of the XML Confidentiality Label including a short description of the elements and attributes. For more details the interested reader should read the standard proposal document [6].

The specification of the XML Confidentiality label provides a high degree of flexibility and the fields can be populated in different ways to satisfy different use cases. This flexibility comes with the cost of potential interoperability issues. In order to ensure full interoperability, profiles must be established. A proposed NATO profile for the XML Confidentiality Label has been developed by the NC3A [7] . It is important to note that the XML Confidentiality label can be used to describe the sensitivity of any type of data, both XML and non-XML data formats.

The top level element of the proposed XML Confidentiality Label is named *ConfidentialityLabel*. From Figure 4.1 we can see that it consists of four sub elements, of which one is mandatory. These elements will be described in more detail later in the section. It also has two optional attributes. The Id attribute can be used to provide a unique identifier of an instance of the confidentiality label. In addition, the *ReviewDateTime* attribute can be used to provide a date for when the label shall be reviewed. This can for instance be used to trigger a manual review process.

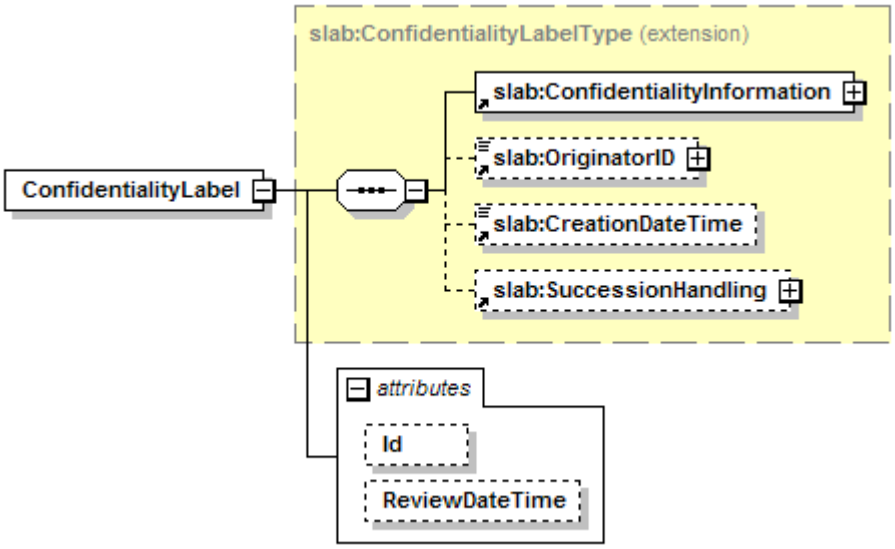

Figure 4.1 Top level components of the XML Confidentiality Label

The only mandatory element of the *ConfidentialityLabel* is the *ConfidentialityInformation* element expanded and shown in Figure 4.2. It consists of the three elements *PolicyIdentifier*, *Classification* and *Categories*. Of these elements the two first are mandatory, while the latter are optional.

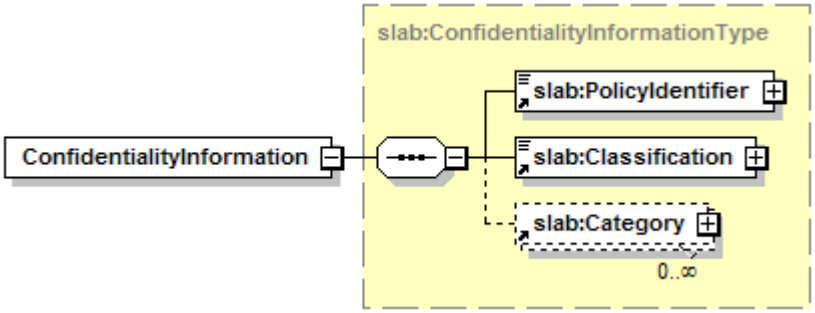

Figure 4.2 ConfidentialityInformation element

The *PolicyIdentifier* element is used to identify the security policy to which the confidentiality label relates. It could either be specified as an URI, text string or both. In the latter case the URI takes precedence in case of inconsistency. Similar the *Classification* element can be specified as an URI, text string, integer value or a combination of an URI and text string or integer. The use and interpretation of the classification element should be specified by the policy.
The final element of the confidentiality information element is the optional and repeatable element *Category*, expanded in Figure 4.3. Categories are used to further refine the granularity of sensitivity, for instance the permissive “releasable to” or restrictive “eyes only”.

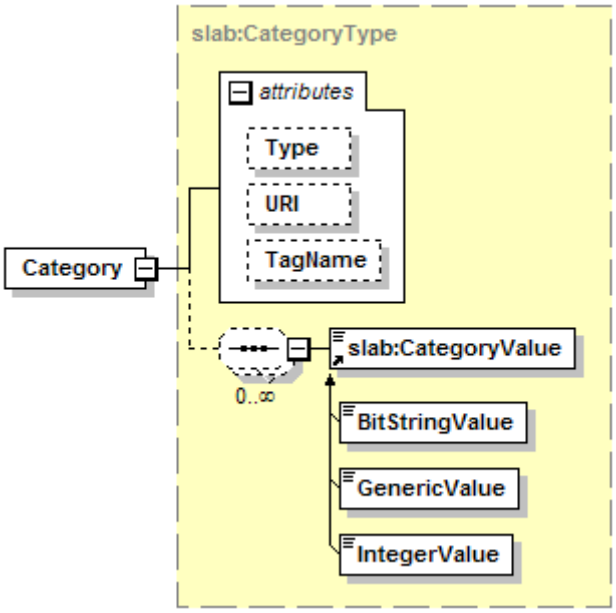

Figure 4.3 Category element

The value of a category can be expressed by a bit string value, an integer value or a generic text string. It can also be expressed by an URI attribute referencing a known category. The mandatory attribute *Type* is used to signalise if the category is permissive, restrictive or informational.

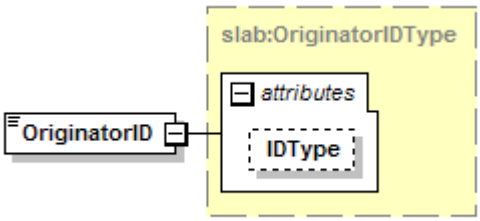

Figure 4.4 OriginatorID element

OriginatorID is the second sub element of the *ConfidentialityLabel* and is shown in Figure 4.4. This optional element may be used to convey the identity of the originator of the label in form of a string. The type of ID can be specified in an attribute, for instance X.509 distinguished name.

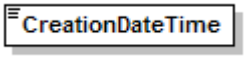

Figure 4.5 CreationDateTime element

The third sub element of the *ConfidentialityLabel* is the optional *CreationDateTime* element shown in Figure 4.5. As the name suggest this element can be used to provide information on when the label was created, i.e. the date and time of the original classification.

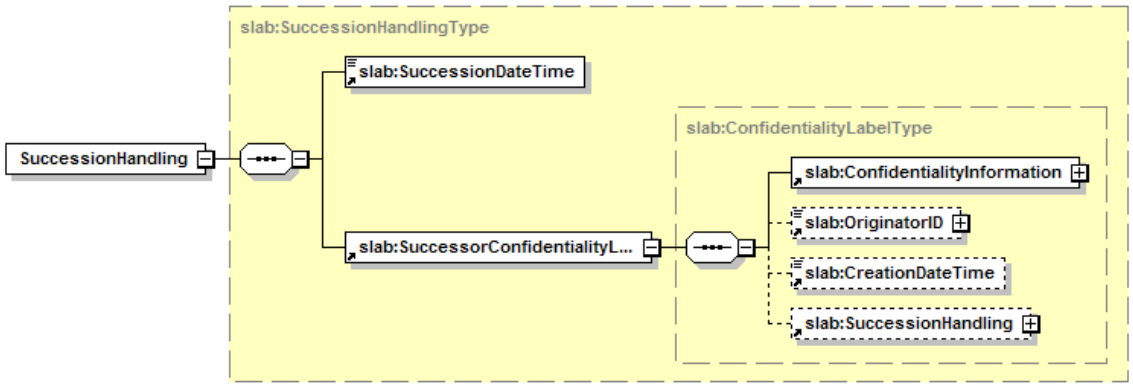

Figure 4.6 SuccessionHandling element

The final element of the *ConfidentialityLabel* is the optional *SuccessionHandling* element shown in Figure 4.6. In essence this element specifies which label to use after a given time as specified by the *SuccessionDateTime* element. The replacement label is given by the *SuccessorConfidentialityLabel* element. The latter element is of the same type as the confidentiality label, and can thus be seen as a nested or recursive label. The typical use-case for this element is for information that has certain sensitivity only for a known period of time, and after that a lower sensitivity. This mechanism can then be used to avoid having to re-label the information.

4.2 XML Metadata Binding

One of the important building blocks of the OLP concept is the association between data and metadata. This association is also known as a binding and informally this binding confirms that the metadata belongs to the data. In section 2.1 different alternatives for realising this association were presented. This section describes the proposed standard for XML metadata binding produced by the NATO RTO IST-068/RTG-031 task group on XML in Cross Domain Security Solutions. Again this document only provides a short description of the XML metadata binding and the interested reader is referred to the STANAG proposal document [8] for more details.

The outline of the XML metadata binding is shown in Figure 4.7, it consists of the top level element named *MetadataBindingContainer* and one or more *MetadataBinding* elements. It is the latter element that actually provides the association between data and metadata. The top level element *MetadataBindingContainer* is a convenience element for collecting more than one metadata binding. This can be useful when for instance binding more than one metadata to different parts of a document, e.g. different confidentiality labels to different sections.

In order for the *MetadataBinding* element to be valid it must have at least one metadata and one data child element. This can either be the actual data and/or metadata embedded in the binding or a reference to it. The *MetadataBinding* element can also have multiple metadata and data elements, which semantically expresses that all the metadata belongs to all data elements. The type of metadata must be specified using the mandatory *type* attribute. This attribute can take one of three values; original confidentiality label, alternative confidentiality label and metadata.

In short, the first attribute is used for the current and prevailing confidentiality label, the second for alternative labels and the third for all other types of metadata. The type original confidentiality label can only be used once within a *MetadataBinding* element. Alternative labels can be used to convey information for how to handle information when the original label is not understood in another domain. For instance it may exist a bilateral agreement between nations that nation A can handle secret information from nation B as its own secret information. The original label would thus state nation B – Secret and the alternative label nation A – Secret.

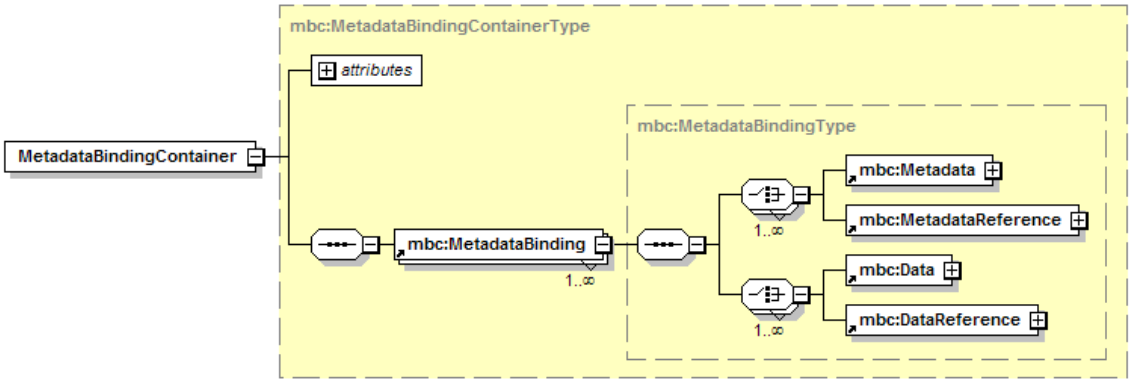

Figure 4.7 Metadata binding

One of the important design considerations when developing the metadata binding was to be data format agnostic. In short the implication of this choice was that the binding must be able to bind any type of metadata to any type of data, also non-xml. In the xml metadata binding proposal this can be achieved by using the metadata and data reference elements, thus referring to data and metadata stored outside this XML object. This can of course also be used when the metadata and data is specified in XML, but not included in the binding.

In itself, the *MetadataBindingContainer* and the *MetadataBinding* element(s) form a loose binding. In order to provide a strong, or cryptographic, binding a digital signature covering the metadata, data and metadata binding must be generated. How this was achieved in the SOA Pilot using XML Digital Signature (XMLDSig) [9] combined with SOAP messages, is the topic of section 5.3.

5 SOA Pilot 2011

The overall goal of the SOA Pilot was to demonstrate how SOA, and in specific Web services, can provide value added to Command and Control Information Systems (C2IS) by exposing information from different sources, including legacy systems. Demonstrating secure two way information exchange between different security domains was a secondary goal. This section describes the proposed solution for how this can be achieved. The proposed solution was presented and demonstrated during the SOA Pilot.

5.1 Scenario Overview

In short, the scenario used in the SOA pilot takes place in an expeditionary operation, were an international coalition is involved. The mandate for this force is to protect civilians and initiate the peace process. The method for achieving this is through symbolic use of force and economical use of force. In more detail, the operation involves use of Tactical Units consisting of ground teams under the command of a Tactical Command HQ. Also other units like a Navy Task force and air capabilities are available. The order of battle (ORBAT) can be seen in Figure 5.1. The details of the scenario can be found in [10].

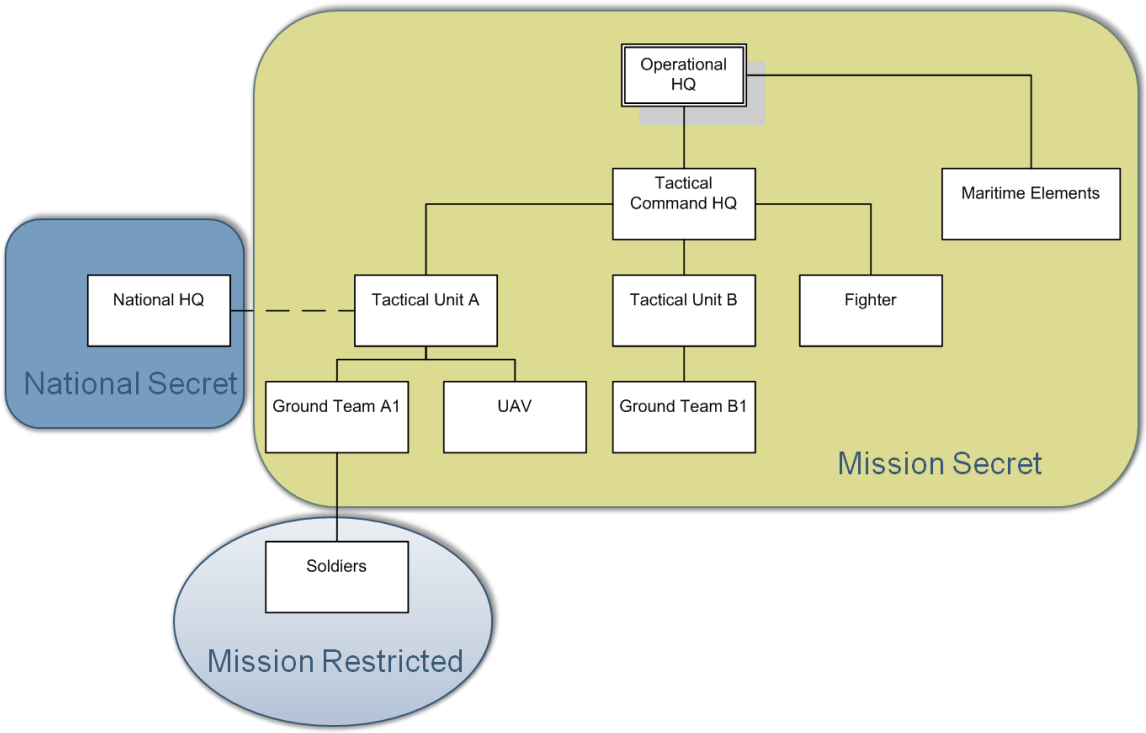

Figure 5.1 ORBAT and security domains

Figure 5.1 also shows the different security domains that were used in the scenario. In general all forces operate in a common security domain at the Mission Secret level. Using one common security domain simplifies administration and minimises the restrictions put on information flows between users and systems within this domain. The drawback of this solution is that there are restrictions on what types of information that can be placed in the domain, for instance national information can't be included. It is however often a need to access and process national information and it is likely that all nations that contribute to the coalition have a need for a reach back to national HQs, and thus to a national security domain. In Figure 5.1 this is shown on the left hand side were the National HQ is placed in a National Secret domain. Notice that the line between the two security domains is dotted since there is no direct information exchange between these two. In the scenario of the SOA pilot access to national secret information was provided by a prototype multilevel terminal solution [11] providing access at both National Secret and Mission Secret. It is important to note that all security domains described here and used during the SOA Pilot were experimental.

The third and final security domain used in the SOA Pilot was Mission Restricted. As explained above the overall goal was to keep most of the systems in a common security domain, however at the lowest level of the ORBAT is a restricted domain. As shown in Figure 5.1 the soldiers and soldier systems are placed in the Mission Restricted domain. The reason for placing these systems in a lower classified domain is that they are inherently more difficult to protect, and thus the risk of security compromise is higher. This includes equipment and systems carried by the soldiers in the field. These are often resource constrained and cannot handle demanding security features. Not to forget the simple risk of losing portable equipment and thus also information.

Since the Mission Secret domain is working by the system high principle, it can contain information that can be shared with the Mission Restricted domain and systems deployed there. This information may be highly valuable for the users of the systems deployed at the lower level and the need for two way information exchange is thus clear. In order to achieve this while ensuring that only the allowed information is passed, a guard is deployed between the Mission Secret and Restricted domains. This guard protects the high domain from information leakage. More specific, the scenario identifies a need for exchanging tracks in the form of NATO Friendly Force Identifier (NFFI) messages between clients in the Mission Restricted and Mission Secret domains. In addition information will flow from the Mission Restricted domain to the Mission Secret domain, for instance will each soldier report her position. This information flow is not restricted or controlled in any way in this experiment, since the focus is on the secure release of information.

The general architecture used for achieving information flow between the security domains is outlined in Figure 5.2. An instance of the prototype XML/SOAP guard is placed between the two different simulated security domains. The task of this guard was to control the information flow from the high to the low domain.

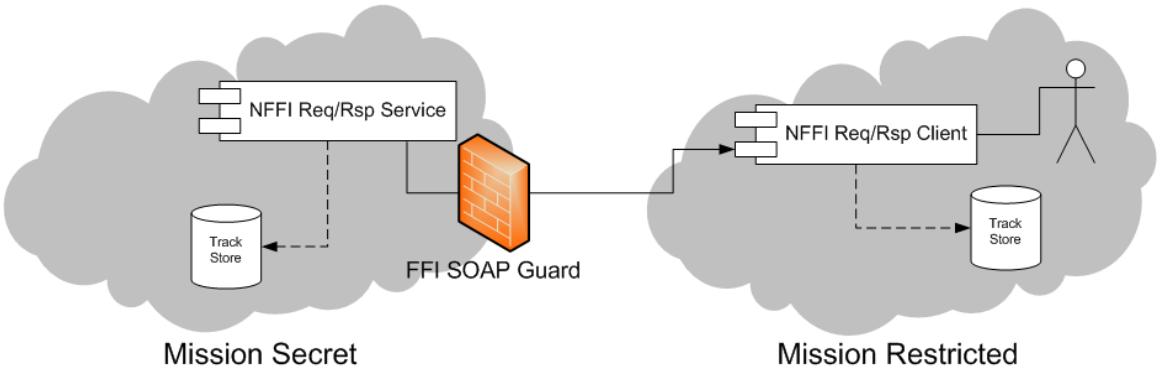

Figure 5.2 Simplified architecture and deployment model

5.2 Software description

This section provides an overview of the software developed and used for exchanging information between security domains in the SOA Pilot. In general this includes a prototype/demonstrator XML/SOAP Guard and the implementation of an NFFI request/response Web service. The latter software also includes functionality to apply XML Confidentiality Labels

and the XML Metadata binding to SOAP messages. The guard includes functionality to verify the same labels and bindings.

5.2.1 Prototype XML/SOAP Guard

The prototype XML/SOAP Guard used during the SOA Pilot is actually the result of a series of experiments and demonstrations. This series started with cross-domain experiments at the NATO Coalition Warrior Interoperability Demonstration (CWID) in 2006 [12], and continued with CWID in 2007, Oasis Final Event in 2008, and CWID in 2009 [13-15] [2]. The prototype XML/SOAP Guard reached its current state with the development for the SOA Pilot. The experiments listed above have all been performed to support, test and develop solutions for secure cross domain information exchange using the Object Level Protection paradigm, and especially the development of the XML Confidentiality Label and the XML Metadata Binding. The prototype guard has been an important tool for testing and verifying the proposed specifications as they were developed. The main reason for developing the prototype XML/SOAP Guard has been to verify the proposed principles and standards, and not to develop a guard product. As a consequence the XML/SOAP Guard has not been evaluated or certified.

Conceptually the prototype XML/SOAP Guard handles the exchange of information between a high and a low domain, and the main functionality is release control for SOAP messages from a high to a low domain. The release control functionality ensures that only information that is allowed by policy is passed to the low domain, thus preventing information leakage. In order to implement the release control the guard depends upon messages being labelled, including the use of cryptographic bindings. In the current version of the guard the release policy is configurable and expressed by an XML Confidentiality Label. The release control functionality compares the label(s) from the message to this policy label in order to decide if the message can be released or not. The main functional components of the XML/SOAP Guard is shown in Figure 5.3 and described below. It is important to note that this figure shows the configuration of the guard used in the SOA Pilot and other configurations can be used if required. As shown in Figure 5.3 messages from the low domain are passed through the guard directly without any modifications while the release handling of messages from the high domain can be broken down into three functional components. In addition, error handling is performed. Protecting the high domain against cyber-attacks and other threats has, although important, been defined as out of scope for this work.

[2] From 2010 CWID was renamed and is now known as CWIX – Coalition Warrior Interoperability Exercise

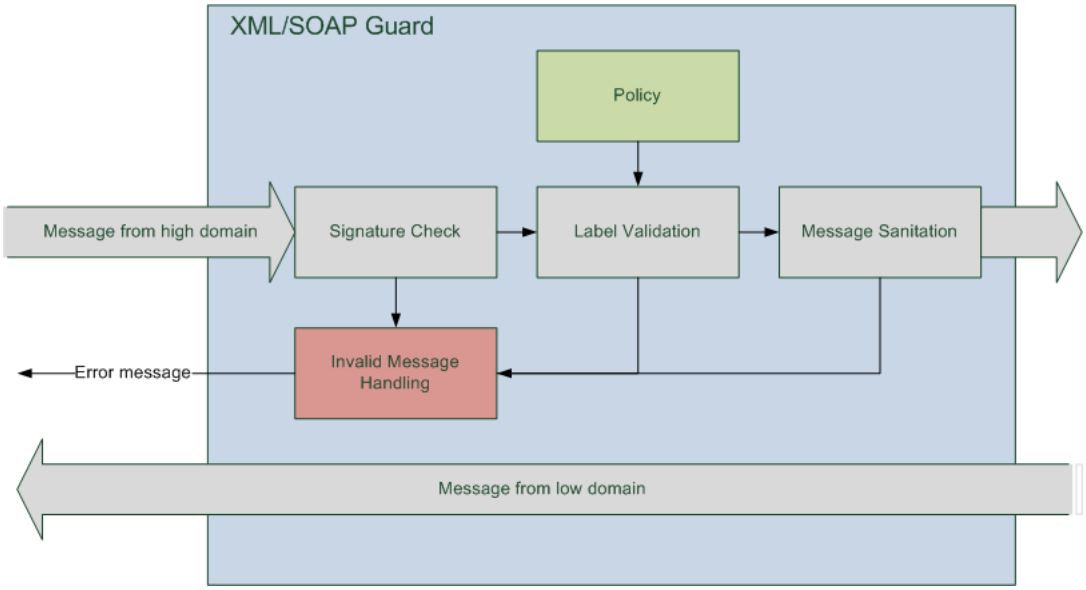

Figure 5.3 The main functional components of the XML/SOAP Guard

A message from the high domain is initially processed by the Signature checker component. In the SOA Pilot, it was a requirement that all information that was to be passed through the guard was labeled and that labels and data were bound using a cryptographic binding. The cryptographic binding is achieved using XML Digital Signature and SOAP Message Security [16]. The signature checker component first of all checks the existence of a digital signature and if not present the message is not allow to be passed and is thus discarded and an error message is generated. An error message is also generated and the message is stopped if the actual signature verification fails. In addition certificate validation can cause the message to be stopped if a certificate is invalid, for instance if it has expired or has been revoked. For simplicity, certification validation was not enabled during the SOA Pilot.

If the message passes the Signature Checker it is forwarded to the Label Checker. The purpose of the label checker is dual, first it checks if the message is correctly labeled and second it checks if the label is approved for release. The first check involves checking if a label is actually present and if so, a check if it is valid. If any of these tests fail, the message is stopped and an error message is generated. If the label exists and is valid, the label is checked versus the policy label. Release is approved if the labels have the same policy identifier and the classification of the message label is equal or lower than the classification of the policy label. In addition, if categories are present these must also be verified. Categories were not used during the SOA Pilot.

A message can potentially contain several labels that mark different parts of the message. If all labels contained in a message fail the policy check, the message is stopped and an error message is generated. Else if all labels are approved for release by the policy check the whole message is released to the low side. Alternatively if some labels are approved and some are disapproved, then the message is forwarded to the message sanitation functional component.

The third and final processing step performed by the guard before releasing the message to the low domain, is message sanitation. This involves removing the parts of the message that is not allowed to be passed to the low domain. This also involves removing the given confidentiality label from the message. If the label is not removed it might be possible for users at the low domain to infer some information on what type of data that was originally included. The information inferred can also be seen as information leakage. The resulting message from sanitation is released to the low domain. However during message sanitation, the message may become corrupt. If so the message is stopped and an error message is generated. Also as a result of message sanitation the digital signature is broken and thus also the cryptographic binding between the confidentiality label(s) and data. There are essentially three options for handling this, first to drop the whole message, second to resign the new message at the guard or finally remove the signature. The latter alternative was chosen for the SOA Pilot since no formal trust relationship was established between the two domains. The lack of for instance federated identity or federated PKI meant that the signature could not be verified in the receiving domain and the signature was thus less valuable.

The implementation of the prototype XML/SOAP Guard is based on the open-source Apache TCPMon software[3]. Originally TCPMon is a utility that allows users to monitor messages that are passed back and forth in a TCP based conversation. TCPMon can be executed in different modes, either as an explicit intermediate, as a client for Web services or finally as a proxy (either pure TCP proxy or with HTTP Proxy support). For the implementation of the prototype XML/SOAP Guard the TCPMon software is extended with the message and label handling described above. The prototype XML/SOAP Guard utilise the HTTP Proxy mode of TCPMon. The guard acts as an HTTP Proxy and interaction with it must be accordingly, the different interaction patterns are outlined in Figure 5.4. In general for a designer or developer the guard adheres to the proxy programming model, and a client must thus be able to address the guard as an intermediary HTTP Proxy in order to use a service situated in the other domain.

The first interaction shown in Figure 5.4 is between a client in the high domain and a service in the low domain. In order for the client to reach the service, it must add a confidentiality label to the request and send it via the XML/SOAP guard. The guard inspects the label(s) of the request and compares them to the given policy and as a result of this the message is either released, discarded or parts of it are removed/sanitised before it is released. The reply from the service is forwarded without modifications from the service to the client.

[3] http://ws.apache.org/commons/tcpmon/ (29.12.2011)

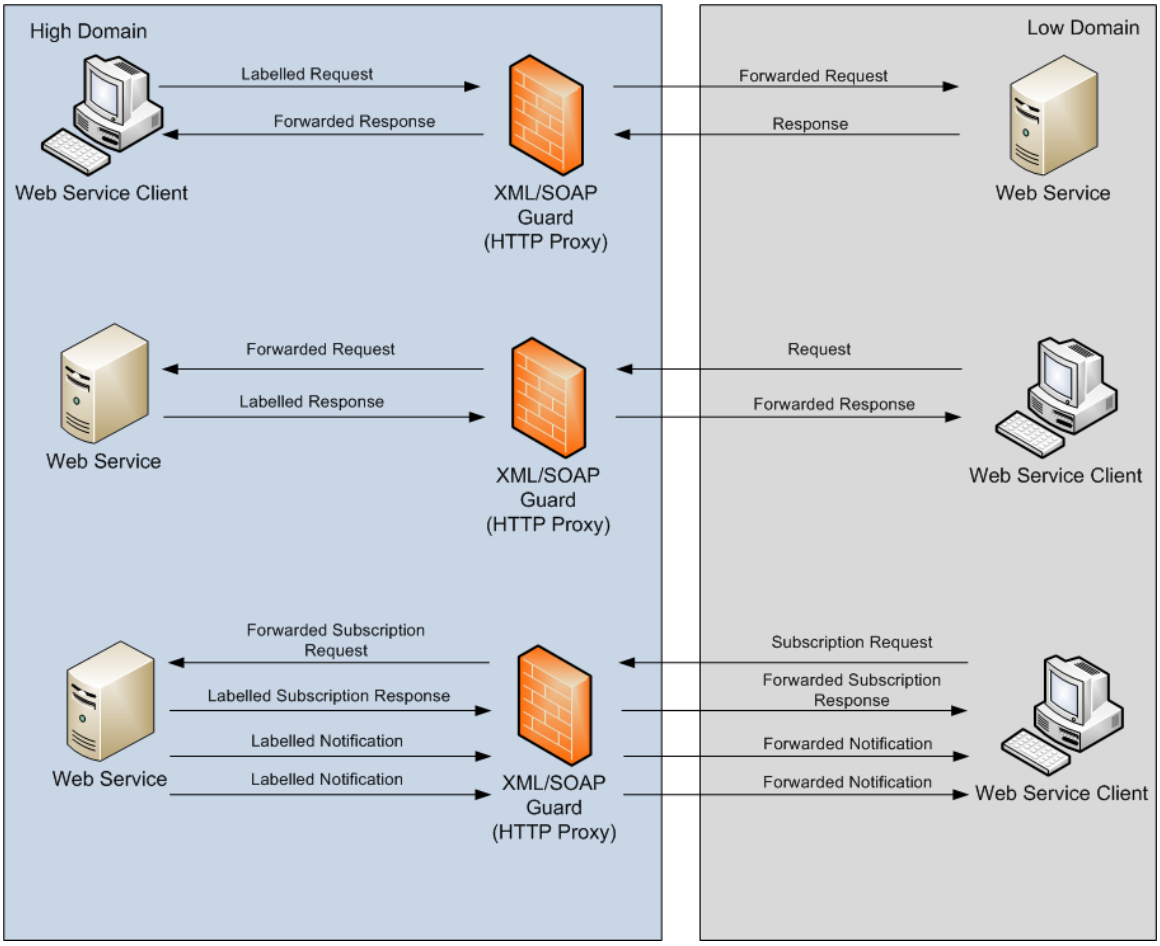

Figure 5.4 Interaction patterns for the prototype XML/SOAP Guard

In the second interaction pattern of Figure 5.4 the client is placed in the low domain and the service in the high domain. In this case the client only needs to direct the original request message to the service using the guard as an HTTP proxy. No labelling is required as the request is forwarded without modifications to the service. In order for the reply to pass the guard, the service must apply confidentiality labels to the generated reply message. When the reply message is intercepted by the guard it is inspected and handled as described above and the message is either stopped, sanitised or forwarded.

While the interaction patterns described above both use request/response Web services, the final involves the use of publish/subscribe Web services. Again, the client is in the low domain and the service in the high domain. In this situation the subscription request message does not need to be labelled as it originates in the low domain. However the subscription response message would have to be labelled in order to pass the guard. Also every subsequent notification message must be labelled in order to pass the guard. Again the guard could either discard, sanitise or forward the message unmodified as a consequence of the given policy. If the client is situated in the high domain only the initial request would have to be labelled. This is not shown in the figure, but is more or less equivalent to the first interaction pattern.

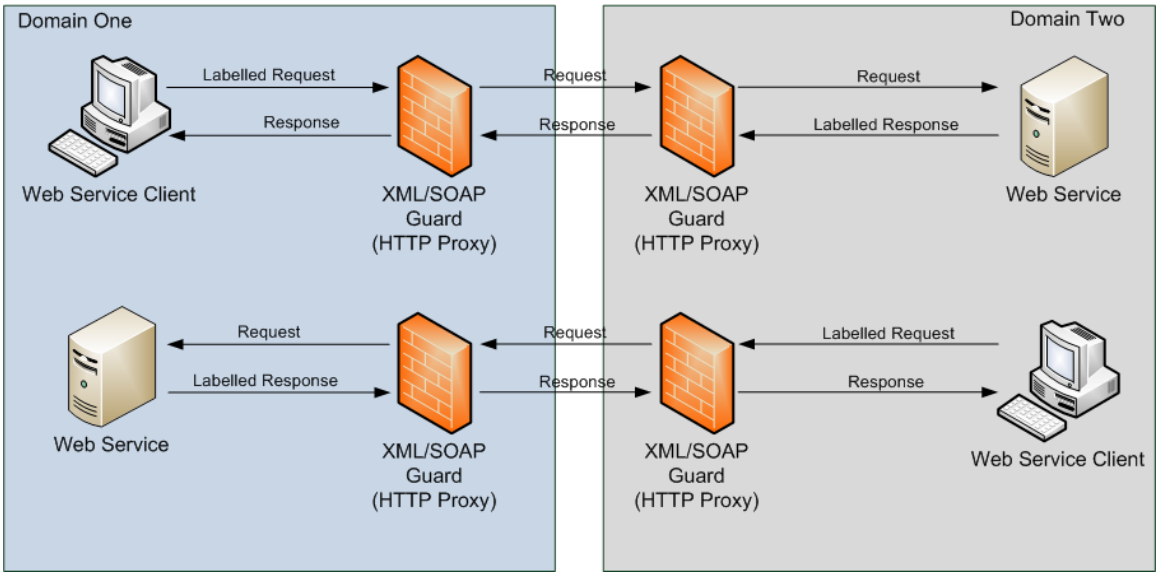

Figure 5.5 Dual guards deployment

The above description of interaction patterns with the prototype XML/SOAP Guard is based on the assumption that only one of the domains is protected. Often the situation is that both domains need to be protected. For instance, in a scenario where a National Secret domain and a Coalition Secret domain are to be interconnected, both could be regarded as the high domain. The reason for this is that from the national perspective the National Secret domain is the high domain, and from the coalition perspective the Coalition Secret domain is the high domain. In this situation it would be natural that both apply local guards to protect their domains as shown in Figure 5.5. The guards would then need to forward messages to the next one in the chain. The prototype supports static configuration of a next guard/HTTP proxy.

It should be emphasised that the prototype XML/SOAP Guard implementation is experimental and should be handled and used accordingly. There are also some implementation issues that must be taken into consideration when using the guard. Known issues include but are not limited to missing support for HTTP chunking.

5.2.2 NFFI Request/Response Service and Client

In the scenario used for the SOA Pilot, a need and requirement to share tracking information between the Mission Secret and Mission Restricted domains is identified (see section 5.1). Also, the NATO Friendly Force Information (NFFI)[17] specification was chosen as the preferred solution for sharing track information. It was also decided to develop a SOA Viewer [18], an independent solution for viewing information available in the network. Remember that the main purpose of the SOA Pilot were to demonstrate the effect of service orientation by making information from legacy systems available as Web services. In short the main functionalities of the SOA Viewer are track and incident handling (including simple GIS capability for showing tracks on a map), chat and service discovery. The SOA Viewer retrieves tracks and incidents by subscribing to publish/subscribe track services. The tracks are stored in the internal track store of the SOA Viewer. In turn the SOA Viewer can publish these tracks to other interested parties, i.e. an aggregated service.

The functionality provided by the SOA Viewer made it an obvious choice for distributing tracking information between security domains as well. Also the fact that the source code was available made it easy to include the labelling and binding software. In terms of information flow a SOA Viewer deployed at the Mission Restricted level would then retrieve tracking information from a designated SOA Viewer at the Mission Secret domain. The track information in the Mission Secret SOA Viewer is aggregated from several different sources, including legacy C2 systems. Use of publish/subscribe for information exchange was one of the important features of the SOA Pilot, and originally it was intended to use this for the cross domain exchange of information as well. However during the development phase there were many elements of uncertainty associated with the choice of publish/subscribe platform and thus also how to include the labelling and binding software. As a result of this uncertainty, it was decided to design and implement a request/response NFFI Web services that include the labelling and binding as part of the SOA Pilot, and to use this for the cross domain information exchange. Since the SOA Viewer was to be used in both domains as described above, it was natural to embed both the service and client software into this. A schematic overview of the SOA Viewer is shown in Figure 5.6. The NFFI request/response Web services extensions are shown in green in this figure.

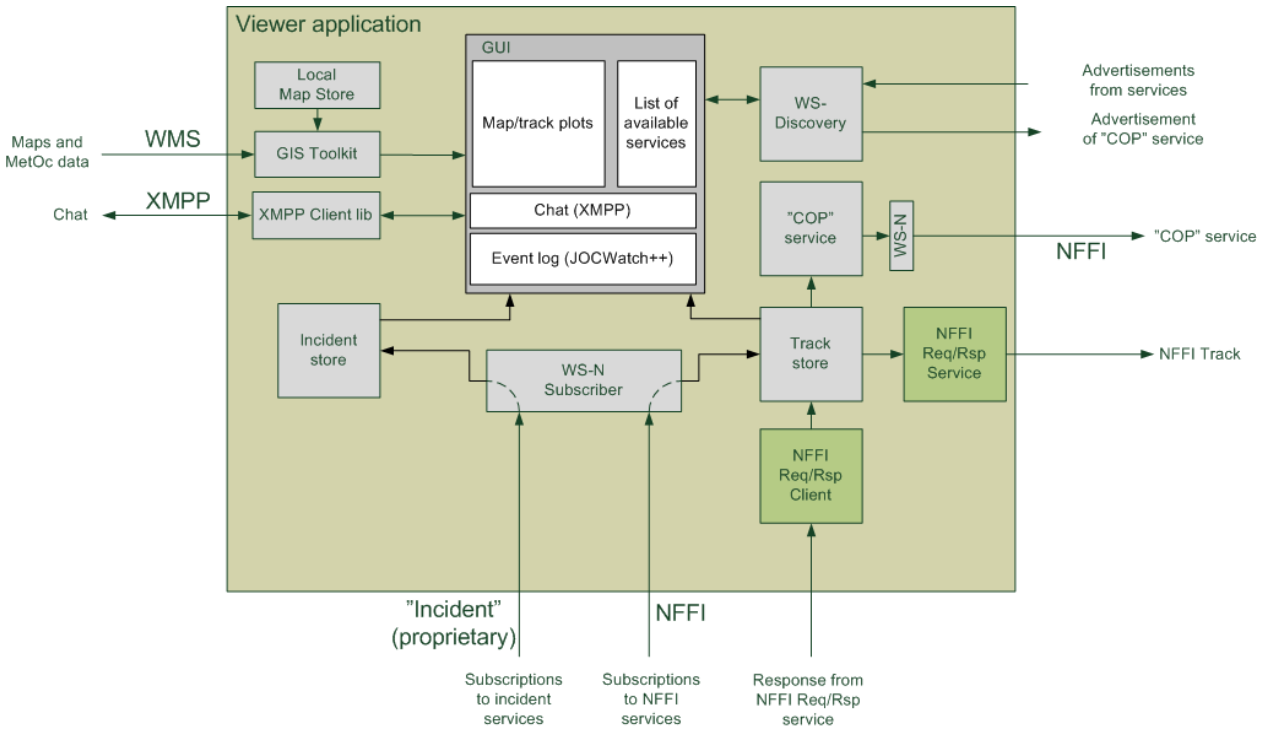

Figure 5.6 The SOA Viewer with NFFI request/response service and client

As can be seen from Figure 5.6 two components were added to the SOA Viewer to enable the NFFI request/response functionality; the actual service implementation and a client. Execution of these components is optional and they can be configured to start as needed. The NFFI request/response client is a simple implementation of a Web service client. It provides functionality to send NFFI SOAP request messages at a configured interval to a NFFI request/response service and to handle the result. The received tracks are stored in the internal

track store of the SOA Viewer and the display is updated with the new tracks. The NFFI SOAP requests are predefined. The client can also be configured to label the request message using the XML Confidentiality Label and the XML Metadata Binding. Labelling a request message is necessary when the client is situated in the high domain and the service in the low domain and it must pass an XML/SOAP Guard. Since information can be embedded in the request it must thus be labelled and protected. This was not used during the SOA Pilot since it was always the client in the low domain who issued the request to a service in the high domain.

The NFFI request/response service, the second component added to the SOA Viewer, implements a simple version of the request/response Web service defined in the NFFI standard. In short the NFFI request/response service implemented receives an NFFI SOAP request message and all tracks available are then extracted from the internal track store and passed back in an NFFI return SOAP message. The NFFI request is validated against the NFFI XML Schema to ensure that it is a valid request message. The NFFI request/response Web service defined by the NFFI standard also handles user defined filters in a request message. These filters can be used by the client to tailor the returned data set to fit its needs. These filters and the possible values are described in the NFFI standard documents and XML schemas. The filters can for instance be used to specify a given geographical area, or a given system to receive track information from, or to limit the number of tracks to be returned. The simple request/response NFFI service included in the viewer application does not support this and is as such not adhering to the NFFI standard. The NFFI request/response service is implemented using the lightweight, open-source Jetty Web Server and Servlet Container[4].

In addition to the functional parts of the service described above, it also applies XML confidentiality labels and XML metadata bindings to the return messages. As a general rule labelling of information should happen as close to the source of information as possible. The reason for this is that users and systems that generate information also often are best placed to judge the original sensitivity of the information. This might be the original producer of information or users and systems that refine or compose new information. In the SOA Pilot a very simple scheme for defining the values of the labels was chosen. It was decided that information from a given set of systems was to be labelled Mission Secret and the rest Mission Restricted. This simple scheme was chosen since the systems generating track information either did not support labelling of information or access to the systems were limited. Also, the main focus of this activity during the SOA Pilot was on the guard and verification of labels in the guard. The NFFI request/response service applies labels and bindings as outlined in section 5.3.

5.3 Applying XML Confidentiality Label and Binding to SOAP Messages

Sections 4.1 and 4.2 outlined the proposed XML Confidentiality Label and the XML Metadata Binding standards respectively. How these proposed specifications are used to apply labels to SOAP messages in the SOA Pilot are the topic of this section. Since the label and binding specifications are general purpose and can be used to solve many use-cases, separate profiles

[4] http://www.eclipse.org/jetty/ (29.12.2011)

must be established for using them in combination with specific data and message formats. The solution chosen for the SOA Pilot has also been used in previous experiments using labels in combination with SOAP messages. Some of the lessons learned from these experiments have also been incorporated in the two proposed NATO profiles for XML Confidentiality Label and XML Metadata Binding. How to use the label and binding specification in combination with SOAP messages is also part of the CWID 09 demonstrator specification [15].

First a short description of the SOAP message format. As shown in Figure 5.7 a general SOAP message consists of three parts; a mandatory outer SOAP Envelope, which contain a SOAP Header element and a SOAP Body element. The latter is mandatory and used to carry the actual message containing application data. The SOAP Header is optional and as the name suggests used to convey header information such as application specific information and security information.

Figure 5.7 Generic SOAP message

The XML confidentiality label and binding are both in essence metadata describing the data contained in the body part of the message, it is thus natural to place these in a SOAP header element. Since it is allowed to add headers to already existing message formats, this solution also makes it possible to bind labels to already existing SOAP messages without changing the format. This flexibility makes it a more general purpose solution. When defining how the XML binding and confidentiality labels are to be used in combination with SOAP messages reuse of existing standards is an important factor. Even though there are no restrictions on developing and using headers, many are standardised to ensure interoperability. One of these headers is the Security header defined in the Web services Security (WS-Security) Core specification developed by OASIS (Organization for the Advancement of Structured Information Standards)[5]. According to this specification the security header provides a mechanism for attaching security related information. Since the binding and confidentiality label is information assurance related information, it was natural to include these in this header. In addition the confidentiality label(s) was included in the metadata binding (se section 4.2 for details of how to use the binding) since a SOAP message should be self-contained and thus should not have reference to external labels.

SOAP is an XML based message format and it is important to know how to interpret the presence of an XML confidentiality label in this context. XML can be viewed as a tree with a set of parent, child and sibling nodes. The top level node is often called the root node. A label can be applied to each node as can be in seen Figure 5.8. However, care has to be taken when combining labelling

[5] http://www.oasis-open.org/ (29.12.2011)

with automatic filtering. Since XML is tree based, removing a node of a certain classification may have implication on nodes of other classifications since it also implies removing any child nodes. In Figure 5.8, using a filter to remove all nodes of classification restricted also results in removing one unclassified node. Keeping the unclassified node would result in a document that is either not valid XML or invalid according to the defined XML format (for instance by an XML schema).

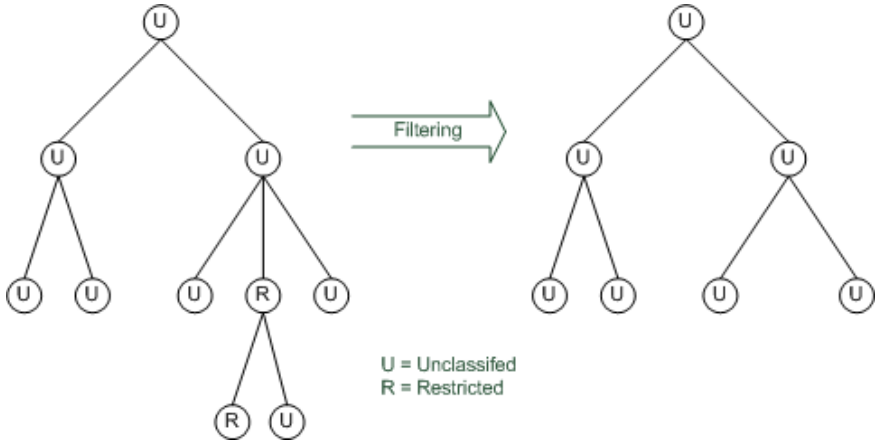

Figure 5.8 Binding semantics when applied to XML

Another effect of this is that using traditional high-water mark labeling, as is known from the world of paper documents, will not work. In short, this can be described as the whole document inheriting the highest classification of each section or paragraph of a document. Applying this to XML and XML labels are shown in Figure 5.9. The effect is that the whole document is actually removed when applying a filter removing restricted nodes. Regardless of this property, if the combined information content of two or more unclassified child elements put together are of a higher classification the parent must be labelled accordingly.

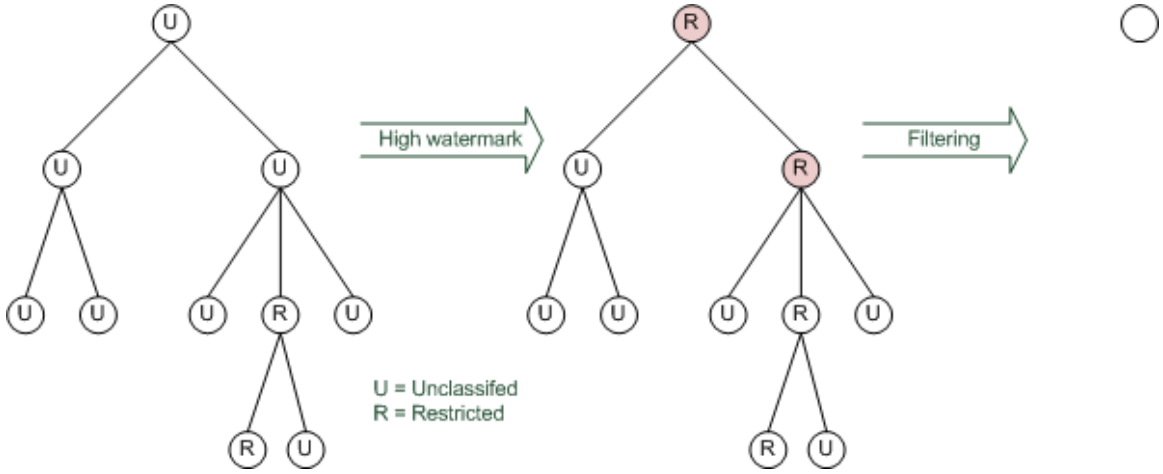

Figure 5.9 Effect of using high watermarking and filtering

The description in this section has so far not made any distinction between loose and strong bindings. Loose bindings may be sufficient in some cases, but in order to achieve a higher level of

trust, a cryptographic binding must be used. Again since the reuse of existing standards and software are important, the mechanisms described in the WS-Security standards are reused, more specifically the signature part of this standard. This describes how the XML Digital Signature standard (XMLDSig) can be used to generate a digital signature over the whole or parts of the message and how to embed it. The Signature part of the WS-Security standard is used to generate a cryptographic binding (also known as a strong binding). For the SOA pilot a single signature were used to cover the whole message since the outer part of the message was labelled. This scheme meant that during message sanitation the signature was broken, and thus also the cryptographic binding. However, for the SOA Pilot activity the release control of information was most important and not propagation of trust in the label and binding to the lower domain. Another scheme must be used if this is important, for instance only sending messages with content with a single classification or having the guard re-bind and re-sign the message. The implications of this have not been considered.

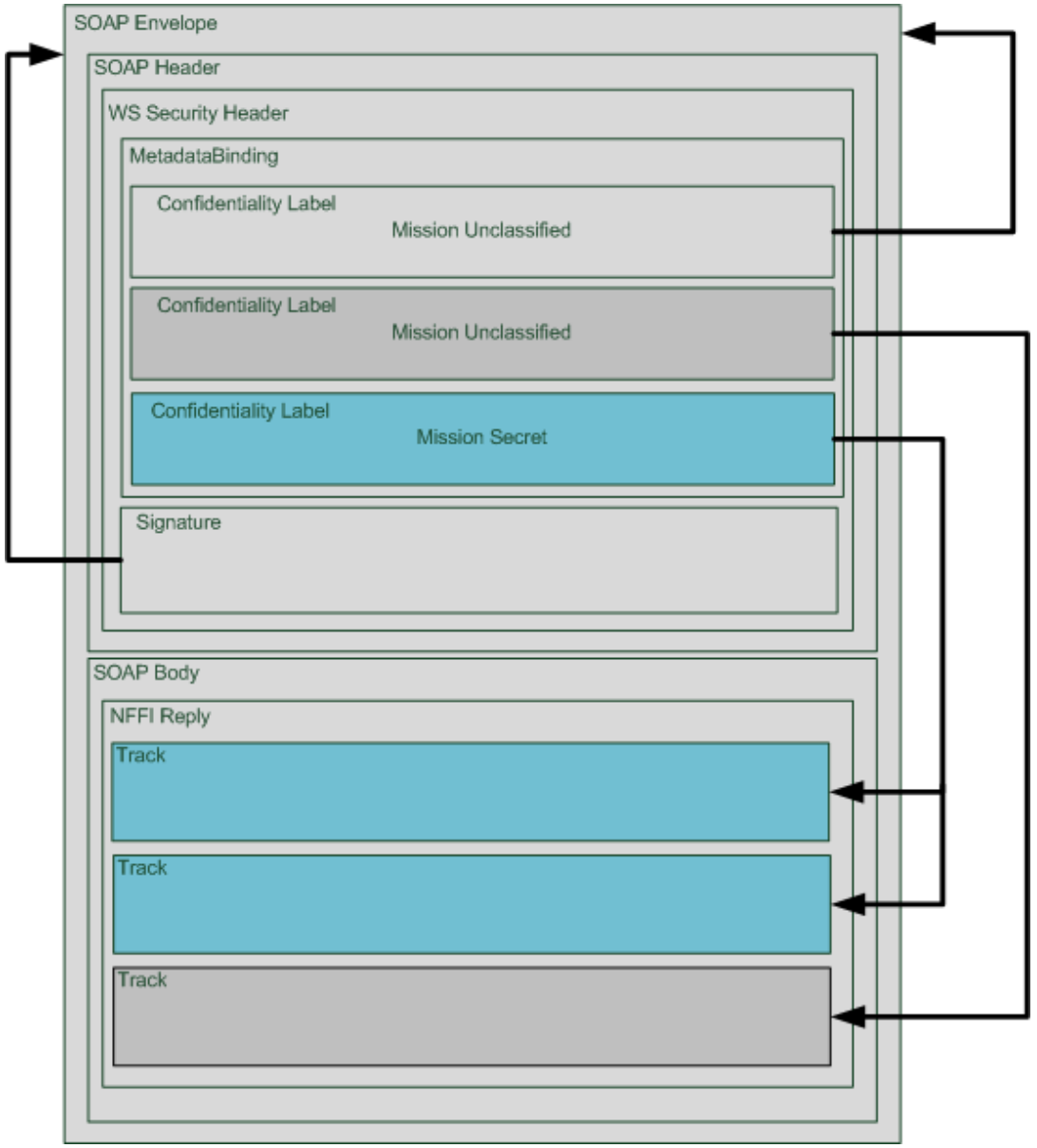

Figure 5.10 SOAP Message with embedded Confidentiality Labels and cryptographic binding

Figure 5.10 shows a schematic view of a SOAP message that is labelled according to the description above. The SOAP message transports tracking information in the form of an NFFI message. In total three tracks are included in the message. Two of these are of classification Mission Secret, as shown by the label in the header, and one track is of classification Mission Unclassified. It is worth noticing that there are two Mission Unclassified labels, one used for labelling the application data objects of this message and one used as a default for labelling the top level of the message. The latter label ensures that the parts of a SOAP message that is not application data cannot be used as a channel for information leakage, for instance by using XML attributes. Also, leaving the top level unlabelled allows a guard or other protection mechanism to decide to stop the whole message since parts of it is not labelled. The guard or other protection mechanisms cannot make the assumption that unlabelled corresponds to unclassified. How to handle unlabelled element should be specified by the policy. A separate Mission Unclassified label is used for the application data in order to be able to preserve the confidentiality information. This way the information can be stripped out of the SOAP message and passed to for instance a tracking application. Since the whole message is labelled, a single signature covering the whole message is used to provide a cryptographic binding between labels and data. For the interested reader with knowledge of XML, an example SOAP message that is labelled using a strong binding can be found in Appendix A.

5.4 SOA Pilot Execution

The actual execution of the SOA Pilot took place in the lab during the spring of 2011. During the SOA Pilot execution clients in the low domain, representing the soldier systems as described in the scenario, requests tracking information from the NFFI service in the high domain. The latter service exposes the track store of the given SOA Viewer and thus also this nodes view of the Common Operational Picture (COP). A screen shot of the SOA Viewer at a given time in the scenario is shown in Figure 5.11, and shows its version of the COP. At this point in time there are five tracks in view; one vehicle, three dismounted soldiers and one UAV flying over the area. These tracks are returned by the NFFI request/response service after being labelled and signed.

The low side NFFI request/response client sends its requests through the XML/SOAP guard to the high side service. This service in turn builds a return message containing the tracking information available in its track store, labels and signs this message and passes it back to the guard. The guard applies its policies and mechanisms to the message and removes information that is not allowed to pass. The resulting COP on the low side SOA Viewer can be seen in Figure 5.12. The low side COP consist of four tracks; one vehicle and three dismounted soldiers. So at the same time in the scenario the two COP views differ. The soldier can see his own position, the position of his team mates and the position of the vehicle. In the viewer at a higher level tracking information about the UAV is available as well. The reason for this difference is that the UAV track information is classified as Mission Secret and is thus removed from the reply message to the low side client. The net result is that only information of classification Mission Restricted or lower is shown in the low side SOA Viewer.

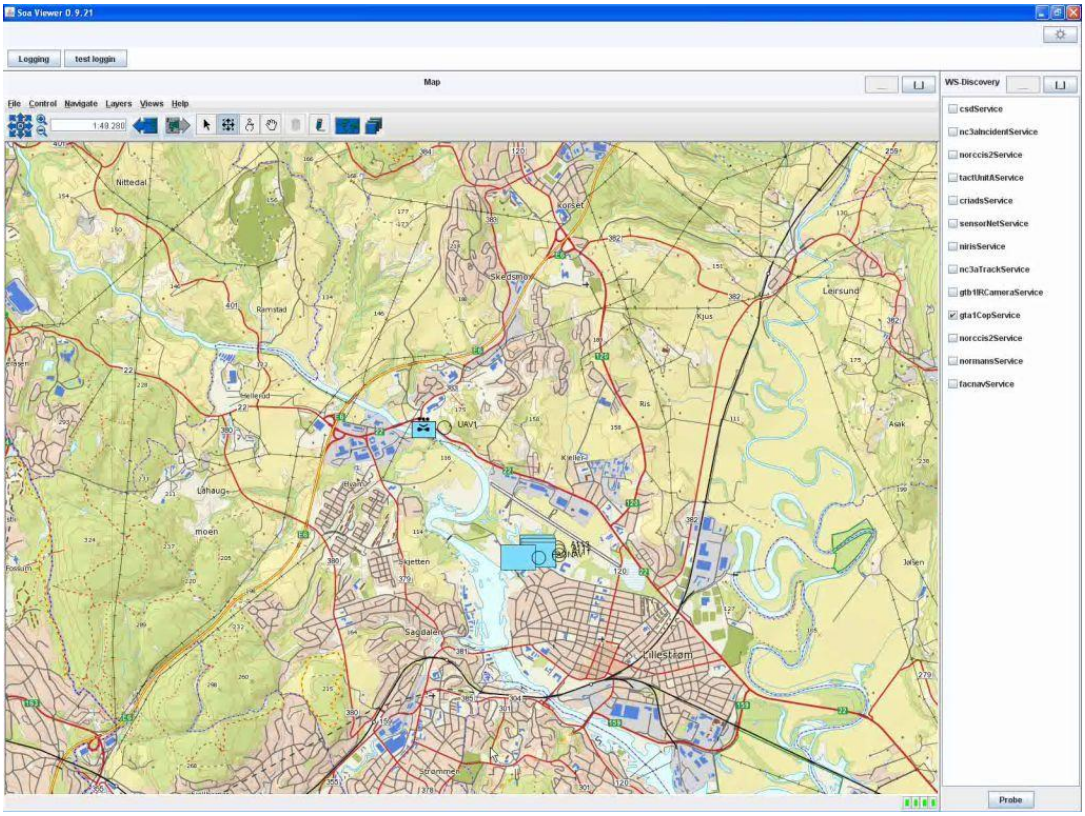

Figure 5.11 High side view of the COP

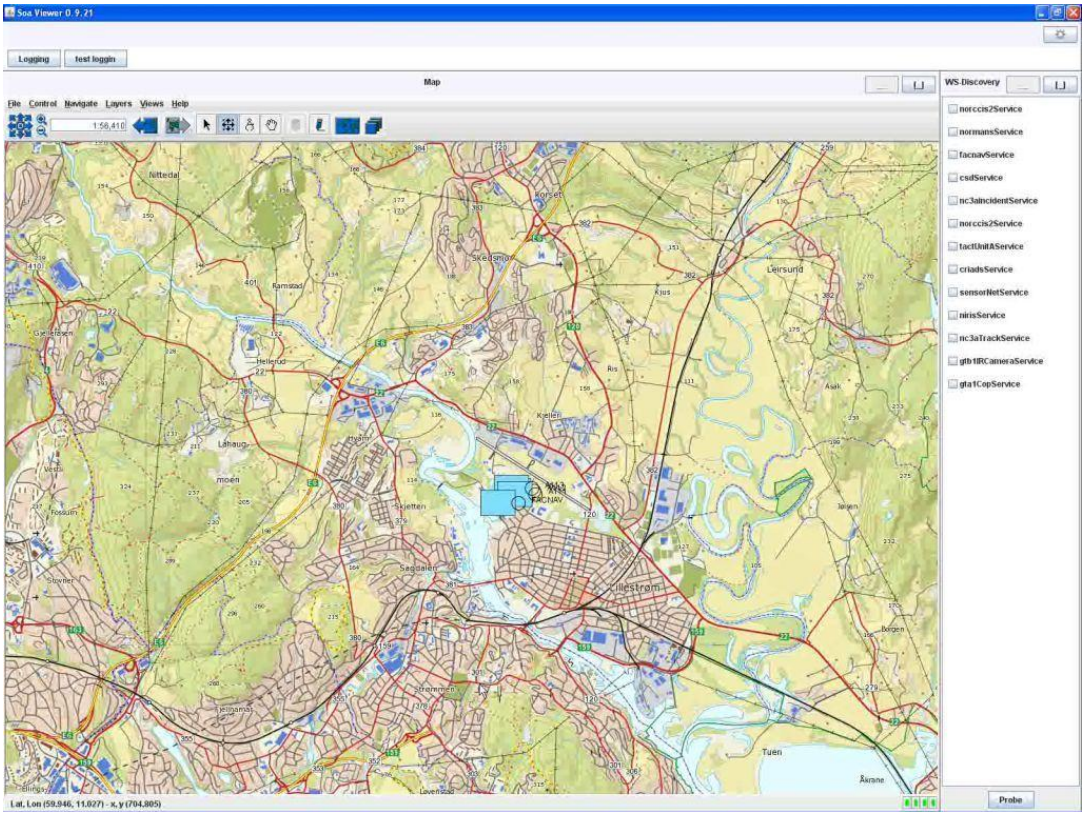

Figure 5.12 Low side view of the COP

The role of the XML/SOAP guard is essentially to stop information from leaking from the high side to the low side, while still enable the flow of permissible information. Figure 5.13 shows a screen shot of the guard UI while in action. The main functionality of the guard UI is to start and stop guard instances, and also to inspect the information flow. It is important to note that the guard UI is only used for inspecting the result of the actions taken by the guard and for debugging purposes. As a result user requirements have not been considered when developing the interface.

At the top level the user can choose which message dialogue to inspect, the chosen message is shown in the lower two panes. The left hand side shows the request message while the right one shows the return message. The guard UI always show messages that are processed by the filtering mechanism, i.e. it shows the messages that are actually released to the low domain. The message processed by the guard can either be the request or the response dependent on which side the request is generated. If the initial request comes from the high side, the left side pane shows the request as processed and filtered by the guard. Or, as the case was in the SOA Pilot, if the request comes from the low side, the right hand side shows the processed and filtered return message. In Figure 5.13, a Mission Unclassified request message (left hand side) is sent from the high side to the low side. And the low side service response is shown on the right hand side. Note that this particular example shows interaction with a different service than the NFFI service described in this document.

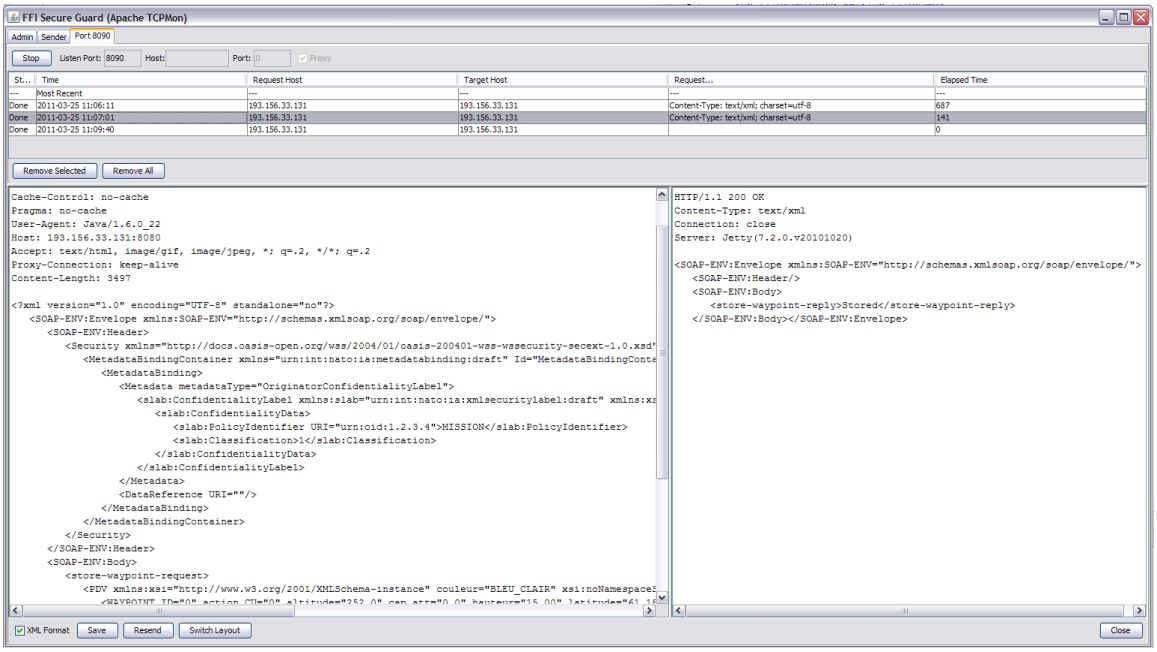

Figure 5.13 Screen shot of prototype XML/SOAP Guard

6 Future Work

The work presented in this document represents an important contribution to the possible future implementation of secure exchange of information between security domains. Nevertheless, this does not constitute a full solution that can be used in an operational setting and there is still unsolved issues that must be addressed in future work. These includes, but are not limited to topics like cross domain key exchange and trust propagation and also cross domain identity management and identity federations.

Standardising the XML Confidentiality Label and XML Metadata Binding specification should be a priority in order to stop the development of proprietary solutions. Thus, standardisation of these specifications is important to ensure interoperability between implementations. Actually the standardisation process has been started and is aiming for NATO standardisation as a first step. The specifications have been handed over to the NATO HQ C3 staff for evaluation. However, due to the re-organising of the NATO C3 Board (NC3B) structure it has been difficult to task the correct working group to perform the actual evaluation. The standardisation process has as a result been lingering in the system. Registration in the NATO Interoperability Profile (NISP) and NATO Metadata Registry & Repository (NMRR) are as a result also pending. As the new structure of the NC3B is slowly emerging it is likely that the standardisation process will pick up momentum.

The next steps planned for the work performed at FFI are the development of a high assurance guard solution and the development of a concept for trusted labelling and signing. The guard solution presented in this document is a low assurance prototype guard that is not certified according to any process like the Common Criteria (CC). Since the guard has not been certified it cannot be used in operational systems. Due to the way it is implemented, it is not feasible to get the current version of the guard certified either. We aim to develop a new version of the guard with more or less the same functionality that should be certifiable to EAL-5. In order to achieve this we plan to utilise the MILS architecture [19].

A high assurance guard solution would need to trust the input it is provided in order to make the correct decision. Input to the guard includes labels, bindings and actual data. Generating trusted labels and signatures is as a consequence very important. The primary goal is to produce a concept that describes how this can be realised. This also involves adding labels in a trusted way to ensure that data does not receive a lower classification than intended. Also a trusted signature process is important to ensure that the signature covers the data needed and not more. This is equal to the problem of knowing that what is intended to be signed is actually signed when a user presses sign in an application. What is probably needed is some kind of What You See Is What You Sign (WYSIWYS) application. Trust, certificate distribution and identity management would also be affecting the trusted labelling and signing.

7 Summary

In general the SOA Pilot demonstrated how SOA, and in specific how Web services, can be used to make information available from different sources. These sources may include existing legacy systems and other sources. Exchanging this information between different security domains in a secure way has been the topic of this document. Security domains are typically used to protect the confidentiality of information and to avoid disclosure by not allowing information to flow from a high to a low domain. However, since the high domain can contain information of a lower classification there is a need to share this with users in lower domains.

This document has presented a proposed solution that enables secure information exchange between security domains. It relies heavily on the concept of Object Level Protection (OLP) and the use of metadata to inform the security mechanisms of how to handle it. The proposed NATO standard for the XML Confidentiality label is used as metadata in the solution to describe the sensitivity of information. The proposed NATO standard for XML Metadata binding is used to cryptographically bind the metadata to the data. In the SOA Pilot the secure release of information from a high to a low domain was successfully demonstrated. The solution used a prototype guard processing labelled SOAP messages, releasing only information with labels that are allowed to be released. Information with labels of a higher classification, or unlabelled information, was stopped at the guard.

It is important to note that the software and also the concepts described in this document and demonstrated during the SOA Pilot are not certified or formally evaluated. The software should thus be handled accordingly. Through the experimentation and demonstrations performed during the SOA pilot, the potential for the proposed solution for solving the problem of securely exchanging information between security domains has been shown. This has also been verified by previous experiments and demonstrations. Using the concepts described in this document should provide a viable path for implementing automatic two way information exchange between security domains.

References

[1] D. Bell and L. LaPadula, "Secure Computer Systems: Mathematical Foundations," MITRE Corporation, Bedford, MA,Technical Report MTR-2547, Vol I, 1973.

[2] D. Bell and L. LaPadula, "Secure Computer System: Unified Exposition and Multics Interpretation," MITRE Corporation, Bedford, MA,Technical Report MTR-2997 Rev. 1, 1975.

[3] P. Bartolomasi, T. Buckman, A. Campell, J. Grainger, J. Mahaffey, R. Marchand, O. Kruidhof, C. Shawcross, and K. Veum, "NATO Network Enabled Capability Feasibility Study, Version 2.0," 2005.

[4] S. Oudkerk, "NATO PROFILE FOR THE BINDING OF METADATA TO DATA OBJECTS,"NC3A Reference Document 2977 (NATO UNCLASSIFIED), 2010.

[5] G. Hallingstad and S. Oudkerk, "Protected core networking: an architectural approach to secure and flexible communications," *IEEE Communication Magazine*, vol. 46, no. 11, pp. 35-41, Nov.2008.

[6] A. Eggen, R. Haakseth, S. Oudkerk, and A. Thummel, "XML Confidentialty Label Syntax - a proposal for a NATO specification,"FFI-rapport 2010/00961 (NATO UNCLASSIFIED), Apr.2010.

[7] S. Oudkerk, "NATO PROFILE FOR THE XML CONFIDENTIALITY LABEL SYNTAX,"NC3A Reference Document 2903 (NATO UNCLASSIFIED), 2009.

[8] A. Eggen, R. Haakseth, S. Oudkerk, and A. Thummel, "Binding of Metadata to Data Objects - a proposal for a NATO specification,"FFI-rapport 2010/00962 (NATO UNCLASSIFIED), Apr.2010.

[9] D. Eastlake, J. Reagle, D. Solo, F. Hirsch, T. Roessler, M. Bartel, J. Boyer, B. Fox, B. LaMacchia, and E. Simon, "XML Signature Syntax and Processing (Second Edition)," World Wide Web Consortium (W3C), http://www.w3.org/TR/xmldsig-core/,2008.

[10] R. Rasmussen, "Experiment Report: SOA Pilot 2011," FFI-rapport 2011/02407 (U), 2011.

[11] N. A. Nordbotten and T. Gjertsen, "Towards a certifiable MILS based workstation," FFI-rapport 2012/00049 (U), 2012.

[12] R. Rasmussen, A. Eggen, D. Hadzic, O.-E. Hedenstad, R. Haakseth, and K. Lund, "Experiment report: "Secure SOA supporting NEC" - NATO CWID 2006," FFI rapport 2006/00325 (U), 2006.

[13] R. Haakseth, T. Gagnes, D. Hadzic, T. Hafsøe, F. T. Johnsen, K. Lund, and B. K. Reitan, "Experiment report: "SOA - Cross Domain and Disadvantaged Grids" - NATO CWID 2007," FFI-rapport 2007/02301 (U), 2007.

[14] R. Haakseth and M. Andreassen (Thales Norway), "Oasis demonstration - secure information exchange between military and civilian systems," FFI-rapport 2009/00319 (U), 2009.

[15] R. Haakseth, M. Andreassen (Thales Norway), and J. Craigie (Clearswift), "CWID 09 demonstrator specification," FFI-notat 2009/02211 (U), 2009.

[16] OASIS, "Web Services Security: SOAP Message Security 1.1," 2004.

[17] R. Malewicz, "NATO Friendly Force Information (NFFI) (version 1.2) Interface Protocol Definition IP3, NC3A Working Document," 2006.

[18] K. Lund, F. T. Johnsen, T. H. Bloebaum, and E. Skjervold, "SOAPilot 2011: Web service," FFI-rapport 2011/02235 (U), 2011.

[19] T. Gjertsen and N. A. Nordbotten, "Multiple independent levels of security (MILS) : a high assurance architecture for handling information of different classification levels," FFI rapport 2009/01137 (U), 2009.

Appendix A Labelled and signed SOAP Message

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <SOAP-ENV:Header>
    <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-   200401-wss-
wssecurity-secext-1.0.xsd"
      xmlns:ns0="http://schemas.xmlsoap.org/soap/envelope/"
ns0:mustUnderstand="1">
      <MetadataBindingContainer xmlns="urn:int:nato:ia:metadatabinding:draft"
Id="MetadataBindingContainer">
        <MetadataBinding>
          <Metadata metadataType="OriginatorConfidentialityLabel">
            <slab:ConfidentialityLabel
xmlns:slab="urn:int:nato:ia:xmlsecuritylabel:xmlconfidentialitylabel:draft"
Id="defaultLabel">
              <slab:ConfidentialityInformation>
                <slab:PolicyIdentifier
URI="urn:oid:1.2.3.4">MISSION</slab:PolicyIdentifier>
                <slab:Classification>1</slab:Classification>
              </slab:ConfidentialityInformation>
            </slab:ConfidentialityLabel>
          </Metadata>
          <DataReference URI=""/>
        </MetadataBinding>
        <MetadataBinding>
          <Metadata metadataType="OriginatorConfidentialityLabel">
            <ConfidentialityLabel
xmlns="urn:int:nato:ia:xmlsecuritylabel:xmlconfidentialitylabel:draft"
Id="label-1">
              <ConfidentialityInformation>
                <PolicyIdentifier
URI="urn:oid:1.2.3.4">MISSION</PolicyIdentifier>
                <Classification>3</Classification>
              </ConfidentialityInformation>
            </ConfidentialityLabel>
          </Metadata>
          <DataReference URI="#BodyElem-1"/>
        </MetadataBinding>
        <MetadataBinding>
          <Metadata metadataType="OriginatorConfidentialityLabel">
            <ConfidentialityLabel
xmlns="urn:int:nato:ia:xmlsecuritylabel:xmlconfidentialitylabel:draft"
Id="label-2">
              <ConfidentialityInformation>
                <PolicyIdentifier
URI="urn:oid:1.2.3.4">MISSION</PolicyIdentifier>
                <Classification>2</Classification>
              </ConfidentialityInformation>
            </ConfidentialityLabel>
          </Metadata>
          <DataReference URI="#BodyElem-2"/>
        </MetadataBinding>
        <MetadataBinding>
          <Metadata metadataType="OriginatorConfidentialityLabel">
            <ConfidentialityLabel
xmlns="urn:int:nato:ia:xmlsecuritylabel:xmlconfidentialitylabel:draft"
Id="label-3">
              <ConfidentialityInformation>
                <PolicyIdentifier
URI="urn:oid:1.2.3.4">MISSION</PolicyIdentifier>
                <Classification>1</Classification>
              </ConfidentialityInformation>
            </ConfidentialityLabel>
          </Metadata>
```

```
          <DataReference URI="#BodyElem-3"/>
        </MetadataBinding>
      </MetadataBindingContainer>
      <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
        <SignedInfo>
          <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315#WithComments"/>
          <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1"/>
          <Reference URI="">
            <Transforms>
              <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/>
            </Transforms>
            <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <DigestValue>zPmCAdRCnLhfCeE+62ukUA2TWS8=</DigestValue>
          </Reference>
        </SignedInfo>

<SignatureValue>bIIekVcIICw9bE4VzFoZH+JaFAQeV3j1M0nreyhLUPObfFIDIujYN/fAH18y+mzo
c9b5HBu52KLS

YGlNGl7WavcZmQavf5iEQWoh7HLuTTsEqCIMhKHsBhh/w8CRWtQN/ryFjoEibgEAAXfS0ee1y2On
          HRuTT8oYc4pe/hWHqdQ=</SignatureValue>
        <KeyInfo>
          <X509Data>
            <X509Certificate>
MIICUzCCAbygAwIBAgIBCDANBgkqhkiG9w0BAQUFADA8MQswCQYDVQQGEwJOTzEMMAoGA1UEChMD
RkZJMQ0wCwYDVQQLEwQyMDA4MRAwDgYDVQQDEwdSb290IENBMB4XDTA5MDUxNDExMjUwNVoXDTEw
MDUwNTExMjUwNVowRzELMAkGA1UEBhMCVUsxEzARBgNVBAoTCkNMRUFSU1dJRlQxDTALBgNVBAsT
BDIwMDkxFDASBgNVBAMTC25mZmktY2xpZW50MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDZ
kSRlxrcQr1Ml5J6jdas+CqbsMGDwsBI8OOQRBIOHulsvJrOjeSKonx3LGJkHmM3hvqYDkYFA0pgQ
0rvMJa1iXbdfbH7V9SHDGrvMmjM+acp9OrlWJDFj/+lVFyMYhDrPDYViuw2tAhY8LUq4L6c0Vro2
MDiN1XpRxwsPcnwisQIDAQABo1owWDAdBgNVHQ4EFgQUs+RxD4zMbe+DIfbnD0k5K5uQoNAwHwYD
VR0jBBgwFoAULePS3tA7Zym+G/rHMgIqYavNSJgwCwYDVR0PBAQDAgbAMAkGA1UdEwQCMAAwDQYJ
KoZIhvcNAQEFBQADgYEAEg4Ym/aVnUtU1/WzLbKaRzQPr8G9EsZ9S7w2Q/wlEd68LyxFFd6q5jEp
OwCDerHAC1rRYPrJf4kPm29t+QuOiOk0lSjtsK2Vu13ooU5N0wboF6h/XCCELq6xlQUbI4QQrxgx
hoR7Qs3bKmtYg+U02pe99uxRfO4VJjvSiFwM2aA=</X509Certificate>
          </X509Data>
        </KeyInfo>
      </Signature>
    </Security>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <pullNFFIResponse xmlns="urn:nato:fft:protocols:nffi_ip3"
xmlns:ns2="urn:nato:fft:protocols:nffi13">
      <ns2:NFFIMessage>
        <ns2:track Id="BodyElem-3">
          <ns2:positionalData secClassification="1"
secPolicyName="urn:oid:1.2.3.4">
            <ns2:trackSource>
              <ns2:sourceSystem>
                <ns2:system>FFI</ns2:system>
              </ns2:sourceSystem>
              <ns2:transponderId>331-A1</ns2:transponderId>
            </ns2:trackSource>
            <ns2:dateTime>20090417141046</ns2:dateTime>
            <ns2:coordinates>
              <ns2:latitude>52.36605</ns2:latitude>
              <ns2:longitude>-2.117851</ns2:longitude>
              <ns2:altitude>2000.0</ns2:altitude>
            </ns2:coordinates>
          </ns2:positionalData>
          <ns2:identificationData secClassification="1"
secPolicyName="urn:oid:1.2.3.4">
            <ns2:unitSymbol>SFAPMFR--------</ns2:unitSymbol>
            <ns2:unitShortName>331-A1</ns2:unitShortName>
          </ns2:identificationData>
```

```
        </ns2:track>
        <ns2:track Id="BodyElem-2">
          <ns2:positionalData secClassification="2"
secPolicyName="urn:oid:1.2.3.4">
            <ns2:trackSource>
              <ns2:sourceSystem>
                <ns2:system>FFI</ns2:system>
              </ns2:sourceSystem>
              <ns2:transponderId>331-A2</ns2:transponderId>
            </ns2:trackSource>
            <ns2:dateTime>20090417141045</ns2:dateTime>
            <ns2:coordinates>
              <ns2:latitude>52.35605</ns2:latitude>
              <ns2:longitude>-2.107851</ns2:longitude>
              <ns2:altitude>2000.0</ns2:altitude>
            </ns2:coordinates>
          </ns2:positionalData>
          <ns2:identificationData secClassification="2"
secPolicyName="urn:oid:1.2.3.4">
            <ns2:unitSymbol>SFAPMFR--------</ns2:unitSymbol>
            <ns2:unitShortName>331-A2</ns2:unitShortName>
          </ns2:identificationData>
        </ns2:track>
        <ns2:track Id="BodyElem-1">
          <ns2:positionalData secClassification="3"
secPolicyName="urn:oid:1.2.3.4">
            <ns2:trackSource>
              <ns2:sourceSystem>
                <ns2:system>FFI</ns2:system>
              </ns2:sourceSystem>
              <ns2:transponderId>331-A3</ns2:transponderId>
            </ns2:trackSource>
            <ns2:dateTime>20090417141045</ns2:dateTime>
            <ns2:coordinates>
              <ns2:latitude>52.34605</ns2:latitude>
              <ns2:longitude>-2.097851</ns2:longitude>
              <ns2:altitude>2000.0</ns2:altitude>
            </ns2:coordinates>
          </ns2:positionalData>
          <ns2:identificationData secClassification="3"
secPolicyName="urn:oid:1.2.3.4">
            <ns2:unitSymbol>SFAPMFR--------</ns2:unitSymbol>
            <ns2:unitShortName>331-A3</ns2:unitShortName>
          </ns2:identificationData>
        </ns2:track>
      </ns2:NFFIMessage>
    </pullNFFIResponse>
  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```