16th ICCRTS

"Collective C2 in Multinational Civil-Military Operations"

Title of Paper

Employing Web services between domains with restricted information flows

Topic(s)

Primary topic: Topic 8: Architectures, Technologies, and Tools
Alternative topic: Topic 9: Networks and Networking

Name of Author(s)

Trude Hafsøe and Frank T. Johnsen
Norwegian Defence Research Establishment (FFI)
P.O. Box 25
2027 Kjeller, Norway

Point of Contact

Trude Hafsøe
Norwegian Defence Research Establishment (FFI)
P.O. Box 25
2027 Kjeller, Norway
trude.hafsoe@ffi.no

Abstract

Web services technology is becoming more widespread for military use, in addition to being common in civilian systems. This technology is defined by a number of standards, each addressing a different aspect of communication, integration and interoperability. This makes Web services highly suited for integrating civil and military systems.

In a civil-military scenario, there are strict demands for security and control of information flow between systems. Web services support application level security, and in the long term, security in Web services based systems should be provided using these standards. However, current security policies often demand the use of other, existing security measures, such as the use of a so-called diode. A diode is placed between networks or domains, and ensures that information can only flow one way between these networks, for example from a civil network to a military network domain.

In this paper, we investigate the use of Web services technology in networks where a data diode is in use.

1. Introduction

NATO has identified Web services as the key enabling technology for NATO Network Enabled Capabilities (NNEC). The technology is founded on standards and is currently in widespread use in civil systems. Thus, it makes sense to apply this technology for civil-military operations as well. Web services are currently the most utilized technology for implementing a Service Oriented Architecture (SOA). In a civil-military scenario, there are strict demands for security and control of information flow between systems. Web services support application level security, but current security policies often demand the use of other, existing security measures, such as the use of a so-called diode. A diode can separate low and high classification domains, and provide assurance that no information can flow from the high domain to the low domain.

The most common pattern for Web services is the request/response pattern, but using this across a one-channel (i.e., a diode) is not feasible using standardized software. However, such functionality is mandated in a civil-military operation, where one will want to ensure that e.g., observations posted in the civil domain may reach a service in the military domain. In this paper we discuss Web services in conjunction with a diode in general, and provide a proof-of-concept implementation showing the feasibility of combining commercial off-the-shelf (COTS) Web services clients and services with a diode.

1.1 Web service communication

The communication between the different entities in a Web service SOA implementation is well defined. Figure 1 shows the different entities in a SOA deployment, and how the basic communication between them is organized. The service provider has a software capability know as a service, which it wants to make available to others. The service in question is described in a standardized way, and this service description, or service contract as it is sometimes called, contains all the information a potential service consumer would need in order to use the service. The owner of the service, the service provider, makes this description available to others by publishing it using a service discovery mechanism, in most cases implemented as a service registry. Consumers looking for services contact the service

registry, and query for services that match the consumers' needs. The registry then responds by providing the consumers with the matching service descriptions. The consumer then makes a choice of which service to use, and can contact that service directly using the information found in the service description.

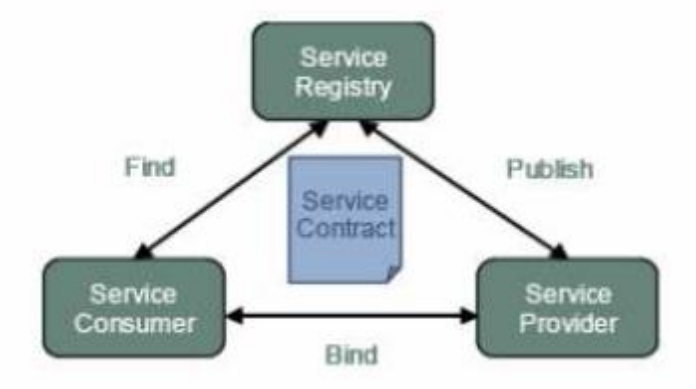

Figure 1: Elements of a Service Oriented Architecture

Based on this information, we see three basic communication steps:

1. The service Provider publishes the service description by sending it to the service registry.
2. The service consumer sends a query to the service registry and gets a set of service descriptions in return.
3. The service consumer invokes the service by sending a request directly to the service, and the service replies.

The three steps listed above form the basis of communication between Web service entities, but in real deployments the situation can be more complex. The service discovery mechanism can for instance consist of several registries cooperating or sharing service information or the service discovery can be done with a fully distributed, non-registry based solution [1, 2]. Another example is when the communication between the service consumer and the service provider is subscription based, rather than following the standard request/response paradigm [3]. In this paper we look more closely at each of these communication patterns, and how each of them may function in conjunction with a data diode.

1.2 Web service protocols

Web services are based on a set of standards that are in themselves transport agnostic, but the technology relies on standardized bindings that define how various transport mechanisms can be used in conjunction with Web services.

The information exchange between Web service entities is done through the exchange of so-called SOAP messages which are messages expressed using the Extensible Markup Language (XML) and formatted according to the SOAP message format.

The by far most common transport binding is the use of HTTP over TCP, which is a connection oriented protocol that relies on two-way communication being available

throughout the duration of the message exchange between the Web service entities. This transport binding would not function across a data diode, and one would have to rely on other mechanisms instead.

SOAP-over-UDP [4] is a standardized communication protocol that can be used to transport SOAP messages using the UDP transport protocol. This protocol is connectionless, and does not rely on any feedback from the receiver in order to function properly. This means that this binding can be used across a data diode, but it does have the downside of not being able to provide reliable delivery.

It is worth noting that most COTS software and development tools for Web services only support the HTTP over TCP binding for web services, which in turn means that most existing Web service deployments only use this form of communication. Due to this limitation we have looked into how one can make this form of COTS Web services function across a data diode, without having to make significant changes to the client or service implementations.

2. Data diodes

Military communication systems have strict security policies, which in some cases require ensuring that no piece of information, however small, is allowed to flow from one information domain to another. This challenge can be addressed by having networks that are fully disconnected from others, and moving information into this domain using an air gap. However, when there is a need for a steady flow of information into, but not out of, a system, a common approach in military systems is to use a so-called information or data diode. These terms are interchangeable, and use the term data diode in this paper. Such a diode ensures that information can only flow in one direction, from what is called the low side to the high side.

A data diode is in principle a simple concept based around a hardware device that only supports one way communication. In addition to the hardware device, commercial diode solutions also include software functionality that can be used to move data across the diode. One example of such functionality would be file transfer, such as a one-way FTP service.

3. Web Service Communication Patterns

As mentioned previously, the basic communication between Web service entities can be divided into three simple steps. The situation in real deployments can however be more complex, and in this section we present the most common communication patterns in more detail, and discuss if, and how, each of them must be adapted for use across a data diode.

Web services technology is based on two-way communication between nodes, and most communication between parties can involve messages being acknowledged also on the Web service level. This means that the introduction of a data diode might require changes to the message exchange patterns between Web service entities in addition to the changes on the transport level.

3.1 Service Discovery

Service Discovery is a term used for any mechanism that allows a service provider to make its service known to potential service consumers. The traditional way to implement service

discovery for Web services is in the form of a stand-alone registry. In this case the communication with the registry is limited to two request/response patterns, one when the service provider publishes its services, and one when the service consumer queries the registry.

More recent service registries are not limited to function in a stand-alone manner, but have the ability to work together in a federation, or share service information between registries. This leads to a new set of communication patterns that must also be taken into consideration.

When two or more registries form a federation, or set up information sharing, this can be done in one of two ways. Either the registries replicate data between them, or they use so-called federated queries, where the queries sent to one registry is passed on to the other members of the federation. In the latter case, the responses sent by the other registries are then consolidated, and sent back to the requesting consumer by the original registry.

The third standard for Web service discovery, WS-Discovery [5], is a fully distributed solution, which relies on IP multicast for service announcements. In this case, a service Provider will send out a multicast message describing its services. Consumers will then either query a local cache or send a multicast probe which the service providers will respond directly to.

In cases where several independent networks are interconnected, it is possible that the different networks will be using different service discovery mechanisms. In this case it is possible to share information between the different mechanisms using either a common translation format known to both mechanisms, or by using a gateway that translates between the mechanisms. In this case, the translation between mechanisms is performed within one network node, which means that no new communication patterns are introduced by such an interconnection.

Discovery Type	**Communicating entities**	**Pattern Name**	**Pattern Description**
Registry-based	Service Provider Registry	Publish	The service description is sent to the registry
Registry-based	Service Consumer Registry	Lookup/ Find	A query is sent from the consumer, and the registry responds
Registry-based	Multiple registries	Replication	A registry sends its service information to a different registry
Registry-based	Service consumer Multiple registries	Federated query	A query is sent from the consumer, the registries share the query and return a response to the consumer
Distributed	Service Provider Service Consumer (optional)	Service announcement	The service provider sends an announcement, which may be received by one or more consumers
Distributed	Service Consumer Service Provider (optional)	Probe	The consumer multicasts a probe message, and service providers may respond with their matching services

Table 1: Service discovery communication patterns

3.2 Service Invocation

Once a service consumer knows of the existence of a service, it can contact the service directly. The communication between the service and the consumer can be done according to one of two main patterns, namely request/response and publish/subscribe [6].

The simplest form of communication between service and consumer follows the request/response pattern. In this case, the consumer sends a request message to the service, which immediately responds to the consumer.

The publish/subscribe pattern is more complex, and can involve more entities than just the service and the consumer. The basic principle is that the consumer sends a subscription request to a subscription manager, letting it know that the consumer is interested in all or a subset of the information produced by the service. In this case, the subscription manager can be either the service itself, or a separate entity. After the subscription has been established, the service provider will send so-called notification messages to the consumer whenever there is new information available.

Communication type	**Communicating entities**	**Pattern name**	**Pattern description**
Request/Response	Service Provider Service Consumer	Polling	The consumer sends a request, which the provider responds to immediately
Publish/Subscribe	Consumer Subscription Manager	Subscribe	The consumer sends a subscription request to the subscription manager
Publish/Subscribe	Provider Consumer	Notification	The provider sends a message containing new information directly to the consumer.

Table 2: Service invocation communication patterns

4. Adapting Web service communication

When adapting Web services for use with a data diode, it is important to make a distinction between which communication patterns can be adapted, and which patterns it makes sense to adapt. One example is service discovery; while it might be possible to adapt registries to share service information across a diode, one need to consider the value of such information sharing. If a service registry on the low side provides a registry on the high side with information about the services on the low side, potential consumers on the high side will be able to discover the services on the low side as well. These consumers will, however, not be able to invoke these services directly, since they have no way of sending a request to the service. The only time it would make sense to have information about the low end service available on the high side is if there is some other means available through which the low side service can be invoked. One example of such a mechanism could be the end user logging in on a system on the low side, and issue the request on behalf of the high side consumer, e.g., if equipped with a MILS terminal [7, 8].

In this section we consider the communication patterns from Table 1 and Table 2 in turn, and identify those patterns that will function in conjunction with a data diode.

For each pattern, we first need to identify the primary information producer. Due to information only being able to pass from the low side to the high side, the information producer will have to be on the low side of the diode. Note that the information producer is not always the one that initiates the communication between the entities, which and that this might stop some patterns from functioning across a diode.

4.1 Service Discovery

For the first pattern, the *publish* pattern, the information producer is the service, which has a service description that it needs to pass to the registry. Having the service on the high side and the registry on the low side would make this pattern impossible, so we concentrate on the opposite case. A service on the low side could, by using this pattern, make its existence known to potential consumer on the high side. The high side consumers on the other hand, would not be able to contact the service directly. This means that while possible, this pattern is only useful if the consumers either simply want to know which services are available (but do not plan on using them), or if they have another way of contacting the service on the low side.

A similar pattern is the *replication* pattern, in which cooperating registries share service information. This pattern can be used for one-way information sharing, in which all or parts of the content of the low side registry is copied into its partner registry on the high side. Interconnecting registries in this manner does not allow for automatic usage of the services on the low side (as the high side consumers cannot contact these services), but it can still be useful. It makes the high side users aware of which capabilities exist on the low side, which enables them to perform manual set-up of communication from the low side to the high side by for instance contacting the service owner via other means.

The two other registry related patterns, namely the *lookup/find* pattern and the *federated query* pattern, both rely on information being provided as a direct response to a query initialized by the service consumer. The strict two-way demand of these patterns means that neither of them will function in conjunction with a diode.

Distributed service discovery is intended for use within a single network, and relies on multicast. Under the assumption that cross-diode multicast traffic is supported, the *service announcements* may cross from the low to the high domain. It does however suffer from the same usage limitations as the registry publish pattern.

WS-Discovery, the only standardized distributed Web service discovery solution, uses a pattern similar to the request/response invocation pattern for its *probes*. Both the probe and the probe match are of equal importance, and none can be omitted. This means that probing is not possible across a diode.

4.2 Service Invocation

The *request/response* pattern, which is the most common invocation pattern for Web services, relies on the consumer first issuing a request which the service then responds to. In most cases, the primary information producer is the service, whose response supplies the consumer with the requested information. The service cannot be made responsible for initiating the communication because it has no way of knowing which consumers will be

interested in the information it can supply. Because of the synchronous nature of this pattern, its usefulness across a diode is fairly limited.

However, there exist a few potential service types where the majority of the information is carried in the request. These services, such as event logging services (reporting alarms from a software system etc), could function even if the response is not delivered to the consumer. These services would then function on a best-effort basis, since reliable delivery cannot be guaranteed, making it unsuitable for critical systems.

The other Web service invocation paradigm, publish/subscribe, relies on two patterns that are executed in order. The first step is the *subscription* pattern, which is generally executed once, which leads to the *notification* pattern being executed a number of times in the opposite direction.

In the subscription pattern the eventual service consumer acts as both the information provider and the initiating entity, and supplies either the service or an external third party, a subscription manager, with its request for information. Even if the subscription pattern, seen isolated, can function from the low to the high side of a diode, its usefulness is limited when executed in this manner. This is because the subscription pattern is only an initial step in setting up the *notification* pattern, in which the information flow goes in the opposite direction of the subscription pattern.

The notification pattern is a one-way concept, where the service periodically sends a notification message to the service consumer without needing a reply. This means that notifications are well suited for use across diodes, but the service does rely on first having received a subscription message (either directly, or by delegating this responsibility to a subscription manager).

The WS-Notification standard, which is one of the two Web service notification standards, does allow for a third party to issue this subscription request on behalf of the service consumer [6, references therein]. This means that it is possible for a user on a high system to receive notifications from a low system without having initiated the subscription herself. If the user can either log in to the low system to issue the request, or have a user in the low system issue the request for them, notification can flow from the low to the high systems without requiring any feedback from the high side.

5. Proof-of-Concept Implementation

As previously mentioned, making Web service technology function one way across data diodes is best done by using an alternative transport binding (e.g., SOAP-over-UDP), in addition to the adaptations that have to be made on the Web service level. However, since most existing Web service solutions use the standard HTTP over TCP binding, we have implemented a solution that shows that is it possible to make these Web services function without any significant modifications to the service or consumer software. The only modification necessary is to adapt the addresses used so that the communication goes via a Web service proxy.

The data diode we used in our experiments is a proprietary prototype, which in addition to the fiber-based hardware, has software that supports automatic duplication of files across the data diode. Commercial diodes can support more advanced communication patterns than

this, but the prototype we used for our experiments was sufficient for our proof-of-concept implementation.

Our proof-of-concept implementation and test use a piece of diode hardware, which ensures that information can only flow one way through it. In other words, it is a data pump capable of moving data from a domain of low classification to a domain of high classification. The diode comes with custom software, which can be configured to achieve the desired operation. The make and model of the diode and software is not important, because several products which all offer similar characteristics and high assurance exist. It suffices to say that we configured the software to take immediate action whenever new data was available, so that when new data became available in the low domain, it would be pushed to the high domain as soon as possible.

The diode is traditionally used to move data files across domain boundaries, often with a manual review and release inspection step along the way. Since we wanted to allow Web services traffic to flow across the domain, we set up a directory with the proper rights so that any new file (or update to an old file) which was placed in that directory would be duplicated in the high domain in a corresponding directory structure. This gave us the necessary infrastructure to enable one-way Web services invocations across classification domains, by going through an intermediary file which could be moved by the diode.

One of the main reasons for using Web services is that the technology is founded on standards, which ease interoperability and reduce chances of vendor lock-in. Thus, for our proof-of-concept implementation we have strived to employ standards where possible.

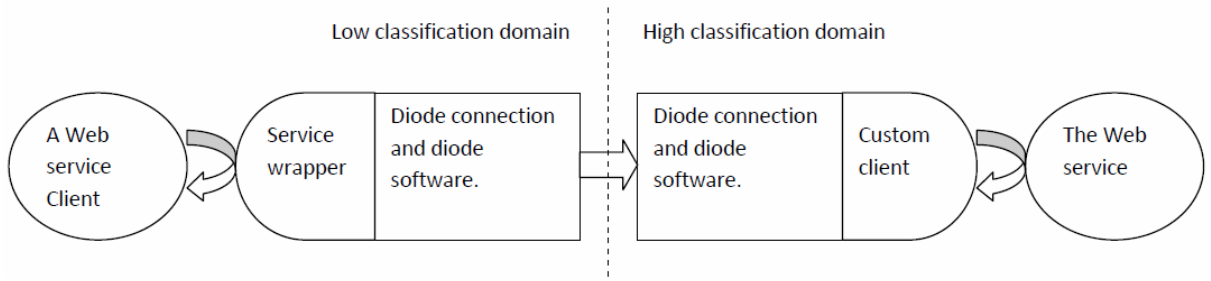

Figure 2: Proof-of-concept architecture

Figure 2 gives an overview of our setup, where the circles indicate COTS implementations following Web services standards (i.e., W3C Web services where the interfaces are defined by a WSDL, and information exchange is performed over the SOAP protocol). Typically, Web services use HTTP/TCP as transport for the SOAP messages. However, when we have a diode as a gateway between networks, TCP (being bi-directional) cannot function. Along the same lines, even Web services which do not have a return parameter usually need some acknowledgement on the HTTP layer for the completed transaction. As a consequence, we need a piece of software on both sides of the gateway that can function as a wrapper service and a wrapper client, respectively. We implemented a service wrapper in the low domain using the WSDL from the Web service we want to submit data to in the high domain. This ensures that to a client, the two are not distinguishable and can be used interchangeably. Thus, the COTS Web service client in the low domain is able to invoke our service wrapper without modification. Conversely, in the high domain we have created a client using the same WSDL, so that data posted by the COTS client can be submitted to the COTS service in the

high domain once the data has passed through the diode. This means that the COTS client can use SOAP over HTTP/TCP when connecting to our service wrapper, whereas our custom client can use the same protocol suite when connecting to the COTS Web service. This allows us to use existing services and clients over a diode, provided the services are information sinks only (i.e., they do not need to issue any data as a response to the invocation). For each such service that we want to expose in the low domain, we must create a service wrapper and custom client that can accommodate SOAP messages over the diode. Since the diode we used operated on file structures, the service wrapper would take incoming requests, serialize the XML to a file, and then place this file in the directory structure the diode software was monitoring. In the high domain, our custom client was monitoring the corresponding directory structure, so that when the diode software had moved new data across from the low domain it would be read by our custom client. Our client could then de-serialize the XML, and use this as its data when invoking the Web service.

6. Conclusion

In this paper we have investigated if it is possible to make Web services, a technology that is common to both military and civilian systems, function across a data diode. We have discussed the various communication patterns that are common in Web services, and identified the publish/subscribe pattern as the most promising for use when interconnecting military and civil systems using a data diode. In summary, the publish/subscribe pattern will allow one way information sharing, but it relies on an external mechanism/entity to set up subscription information. Some other patterns, such as registry replication and the request step of the request/response pattern can also function across data diodes, but their usefulness is more limited than that offered by the publish/subscribe pattern. Consequently, the request/response pattern may also be used across a diode provided that the response is of no concern to the client. We have shown the feasibility of such use of Web services in our proof-of-concept implementation.

References

[1] Frank T. Johnsen, Trude Hafsøe, Magnus Skjegstad, *"Web Services and Service Discovery in Military Networks"*, 14th International Command and Control Research and Technology Symposium (ICCRTS), Washington DC, USA, June 2009.

[2] Frank T. Johnsen, Trude Hafsøe, Anders Eggen, Carsten Griwodz,and Pål Halvorsen, *"Web Services Discovery across Heterogeneous Military Networks"*, IEEE Communications Magazine, October 2010.

[3] Espen Skjervold, Trude Hafsøe, Frank T. Johnsen, Ketil Lund, *"Enabling Publish/Subscribe with COTS Web Services across Heterogeneous Networks"*, 4th International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing (ACT4SOC 2010), Athens, Greece, July 2010.

[4] Ram Jeyaraman (ed.),*"SOAP-over-UDP version 1.1 OASIS standard 1 July 2009"*, http://docs.oasis-open.org/ws-dd/soapoverudp/1.1/os/wsdd-soapoverudp-1.1-spec-os.pdf, accessed 2010-02-23.

[5] Vipul Modi, and Devon Kemp (eds.), *"Web services dynamic discovery (WS-Discovery) version 1.1 OASIS standard 1 July 2009"*, http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.pdf, accessed 2009-12-28.

[6] Trude Hafsøe, Frank T. Johnsen, Ketil Lund, Anders Eggen, *"Adapting Web Services for Limited Bandwidth Tactical Networks"*, 12th International Command and Control Research and Technology Symposium (ICCRTS), Newport, RI, USA, June 2007.

[7] G.M. Uchenick and W.M. Vanfleet, *"Multiple independent levels of safety and security: high assurance architecture for MSLS/MLS"*, IEEE Military Communications Conference (MILCOM) 2005, Atlantic City, NJ, USA, October 2005.

[8] T. Gjertsen and N.A. Nordbotten, *"Multiple independent levels of security (MILS) – a high assurance architecture for handling information of different classification levels"*, FFI report 2008/01999, http://rapporter.ffi.no/rapporter/2008/01999.pdf

Abbreviations

COTS	Commercial off-the-shelf
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
MILS	Multiple independent levels of security
NNEC	NATO Network Enabled Capabilities
SOA	Service Oriented Architecture
SOAP	Called the "simple object access protocol" up to and including its release as a W3C version 1.1 note, but this name did not describe exactly what SOAP was, and so it was later dropped. In its current version, the W3C version 1.2 recommendation, the protocol is just called "SOAP".
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
W3C	World Wide Web Consortium
WS-Discovery	Web Services Dynamic Discovery
WSDL	Web Services Description Language
XML	Extensible Markup Language