5 Security Threats and Countermeasures

This specification considers two sets of security requirements, those of the applications that use the WS-RM protocol and those of the protocol itself.

This specification makes no assumptions about the security requirements of the applications that use WS-RM. However, once those requirements have been satisfied within a given operational context, the addition of WS-RM to this operational context should not undermine the fulfillment of those requirements; the use of WS-RM should not create additional attack vectors within an otherwise secure system.

There are many other security concerns that one may need to consider when implementing or using this protocol. The material below should not be considered as a "check list". Implementers and users of this protocol are urged to perform a security analysis to determine their particular threat profile and the appropriate responses to those threats.

Implementers are also advised that there is a core tension between security and reliable messaging that can be problematic if not addressed by implementations; one aspect of security is to prevent message replay but one of the invariants of this protocol is to resend messages until they are acknowledged. Consequently, if the security sub-system processes a message but a failure occurs before the reliable messaging sub-system receives that message, then it is possible (and likely) that the security sub-system will treat subsequent copies as replays and discard them. At the same time, the reliable messaging sub-system will likely continue to expect and even solicit the missing message(s). Care should be taken to avoid and prevent this condition.

5.1 Threats and Countermeasures

The primary security requirement of this protocol is to protect the specified semantics and protocol invariants against various threats. The following sections describe several threats to the integrity and operation of this protocol and provide some general outlines of countermeasures to those threats. Implementers and users of this protocol should keep in mind that all threats are not necessarily applicable to all operational contexts.

5.1.1 Integrity Threats

In general, any mechanism which allows an attacker to alter the information in a Sequence Traffic Message, Sequence Lifecycle Message, or Sequence-related fault, or which allows an attacker to alter the correlation of a RM Protocol Header Block to its intended message represents a threat to the WS-RM protocol.

For example, if an attacker is able to swap `Sequence` headers on messages in transit between the RMS and RMD then they have undermined the implementation's ability to guarantee the first invariant described in Section 2.3. The result is that there is no way of guaranteeing that messages will be delivered to the Application Destination in the same order that they were sent by the Application Source.

5.1.1.1 Countermeasures

Integrity threats are generally countered via the use of digital signatures or encryption at some level of the communication protocol stack. Note that, in order to counter header swapping attacks, the signed and/or encrypted block SHOULD include both the SOAP body and any relevant SOAP headers (e.g. `Sequence` header). Because some headers (`AckRequested`, `SequenceAcknowledgement`) are independent of the body of the SOAP message in which they occur, implementations MUST allow for signature and/or encryption blocks that cover only these headers.

Copyright © OASIS Open 2005. All Rights Reserved.

5.1.2 Resource Consumption Threats

The creation of a Sequence with an RMD consumes various resources on the systems used to implement that RMD. These resources can include network connections, database tables, message queues, etc. This behavior can be exploited to conduct denial of service attacks against an RMD. For example, a simple attack is to repeatedly send `CreateSequence` messages to an RMD. Another attack is to create a Sequence for a service that is known to require in-order message delivery and use this Sequence to send a stream of very large messages to that service, making sure to omit message number "1" from that stream.

5.1.2.1 Countermeasures

There are a number of countermeasures against the described resource consumption threats. The technique advocated by this specification is for the RM Destination to restrict the ability to create a Sequence to a specific set of entities/principals. This reduces the number of potential attackers and, in some cases, allows the identity of any attackers to be determined.

The ability to restrict Sequence creation depends, in turn, upon the RM Destination's ability identify and authenticate the RM Source that issued the `CreateSequence` message.

5.1.3 Sequence Spoofing Threats

Sequence spoofing is a class of threats in which the attacker uses knowledge of the `Identifier` for a particular Sequence to forge Sequence Lifecycle or Traffic Messages. For example the attacker creates a fake `TerminateSequence` message that references the target Sequence and sends this message to the appropriate RMD. Some sequence spoofing attacks also require up-to-date knowledge of the current `MessageNumber` for their target Sequence.

In general any Sequence Lifecycle Message, RM Protocol Header Block, or sequence-correlated SOAP fault (e.g. `InvalidAcknowledgement`) can be used by someone with knowledge of the Sequence identifier to attack the Sequence. These attacks are "two-way" in that an attacker may choose to target the RMS by, for example, inserting a fake `SequenceAcknowledgement` header into a message that it sends to the "AcksTo" EPR of an RMS.

5.1.3.1 Sequence Hijacking

Sequence hijacking is a specific case of a sequence spoofing attack. The attacker attempts to inject Sequence Traffic Messages into an existing Sequence by inserting fake `Sequence` headers into those messages.

Note that "sequence hijacking" should not be equated with "security session hijacking". Although a Sequence may be bound to some form of a security session in order to counter the threats described in this section, applications MUST NOT rely on WS-RM-related information to make determinations about the identity of the entity that created a message; applications SHOULD rely only upon information that is established by the security infrastructure to make such determinations. Failure to observe this rule creates, among other problems, a situation in which the absence of WS-RM may deprive an application of the ability to authenticate its peers even though the necessary security processing has taken place.

5.1.3.2 Countermeasures

There are a number of countermeasures against sequence spoofing threats. The technique advocated by this specification is to consider the Sequence to be a shared resource that is jointly owned by the RM Source that initiated its creation (i.e. that sent the `CreateSequence` message) and the RM Destination that serves as its terminus (i.e. that sent the `CreateSequenceResponse` message). To counter sequence

Copyright © OASIS Open 2005. All Rights Reserved.

spoofing attempts the RM Destination SHOULD ensure that every message or fault that it receives that refers to a particular Sequence originated from the RM Source that jointly owns the referenced Sequence. For its part the RM Source SHOULD ensure that every message or fault that it receives that refers to a particular Sequence originated from the RM Destination that jointly owns the referenced Sequence.

For the RM Destination to be able to identify its sequence peer it MUST be able to identify and authenticate the entity that sent the `CreateSequence` message. Similarly for the RM Source to identify its sequence peer it MUST be able to identify and authenticate the entity that sent the `CreateSequenceResponse` message. For either the RM Destination or the RM Source to determine if a message was sent by its sequence peer it MUST be able to identify and authenticate the initiator of that message and, if necessary, correlate this identity with the sequence peer identity established at sequence creation time.

5.2 Security Solutions and Technologies

The security threats described in the previous sections are neither new nor unique. The solutions that have been developed to secure other SOAP-based protocols can be used to secure WS-RM as well. This section maps the facilities provided by common web services security solutions against countermeasures described in the previous sections.

Before continuing this discussion, however, some examination of the underlying requirements of the previously described countermeasures is necessary. Specifically it should be noted that the technique described in Section 5.1.2.1 has two components. Firstly, the RM Destination identifies and authenticates the issuer of a `CreateSequence` message. Secondly, the RM Destination to performs an authorization check against this authenticated identity and determines if the RM Source is permitted to create Sequences with the RM Destination. Since the facilities for performing this authorization check (runtime infrastructure, policy frameworks, etc.) lie completely within the domain of individual implementations, any discussion of such facilities is considered to be beyond the scope of this specification.

5.2.1 Transport Layer Security

This section describes how the the facilities provided by TLS [RFC 4346] can be used to implement the countermeasures described in the previous sections. The use of TLS is subject to the constraints defined in Section 4 of the Basic Security Profile 1.0 [BSP 1.0].

The description provided here is general in nature and is not intended to serve as a complete definition on the use of TLS to protect WS-RM. In order to interoperate implementations need to agree on the choice of features as well as the manner in which they will be used. The mechanisms described in the Web Services Security Policy Language [WS-SecurityPolicy] MAY be used by services to describe the requirements and constraints of the use of TLS.

5.2.1.1 Model

The basic model for using TLS is as follows:

1. The RM Source establishes an TLS session with the RM Destination.
2. The RM Source uses this TLS session to send a `CreateSequence` message to the RM Destination.
3. The RM Destination establishes an TLS session with the RM Source and sends an asynchronous `CreateSequenceResponse` using this session. Alternately it may respond with a synchronous `CreateSequenceResponse` using the session established in (1).

Copyright © OASIS Open 2005. All Rights Reserved.

4. For the lifetime of the Sequence the RM Source uses the TLS session from (1) to transmit any and all messages or faults that refer to that Sequence.

5. For the lifetime of the Sequence the RM Destination either uses the TLS session established in (3) to transmit any and all messages or faults that refer to that Sequence or, for synchronous exchanges, the RM Destination uses the TLS session established in (1).

5.2.1.2 Countermeasure Implementation

Used in its simplest fashion (without relying upon any authentication mechanisms), the per-packet encryption performed by TLS provides the necessary integrity qualities to counter the threats described in Section 5.1.1. Note, however, that the nature of TLS limits the scope of this integrity protection to a single transport level session. If TLS is the only mechanism used to provide integrity, any intermediaries between the RM Source and the RM Destination MUST be trusted to preserve the integrity of the messages that flow through them.

As noted, the technique described in Sections 5.1.2.1 involves the use of authentication. This specification advocates either of two mechanisms for authenticating entities using TLS. In both of these methods the TLS server (the party accepting the TLS connection) authenticates itself to the TLS client using an X.509 certificate that is exchanged during the TLS handshake.

- **HTTP Basic Authentication:** This method of authentication presupposes that a SOAP/HTTP binding is being used as part of the protocol stack beneath WS-RM. Subsequent to the establishment of the the TLS session, the sending party authenticates itself to the receiving party using HTTP Basic Authentication [RFC 2617]. For example, a RM Source might authenticate itself to a RM Destination (e.g. when transmitting a Sequence Traffic Message) using BasicAuth. Similarly the RM Destination might authenticate itself to the RM Source (e.g. when sending an acknowledgement) using BasicAuth.

- **TLS Client Authentication:** In this method of authentication, the party initiating the connection authenticates itself to the party accepting the connection using an X.509 certificate that is exchanged during the TLS handshake.

To implement the countermeasures described in section 5.1.2.1 the RM Source must authenticate itself using one the above mechanisms. The authenticated identity can then be used to determine if the RM Source is authorized to create a Sequence with the RM Destination.

This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring an RM node's Sequence peer to be equivalent to their TLS session peer. This allows the authorization decisions described in section 5.1.3.2 to be based on TLS session identity rather than on authentication information. For example, an RM Destination can determine that a Sequence Traffic Message rightfully belongs to its referenced Sequence if that message arrived over the same TLS session that was used to carry the `CreateSequence` message for that Sequence. Note that requiring a one-to-one relationship between TLS session peer and Sequence peer constrains the lifetime of a TLS-protected Sequence to be less than or equal to the lifetime of the TLS session that is used to protect that Sequence.

This specification does not preclude the use of other methods of using TLS to implement the countermeasures (such as associating specific authentication information with a Sequence) although such methods are not covered by this document.

Issues specific to the life-cycle management of TLS sessions (such as the resumption of a TLS session) are outside the scope of this specification.

5.2.2 SOAP Message Security

The mechanisms described in WS-Security [WS-Security] may be used in various ways to implement the countermeasures described in the previous sections. This specification advocates using the protocol

Copyright © OASIS Open 2005. All Rights Reserved.

described by WS-SecureConversation [WS-SecureConverstaion] (optionally in conjunction with WS-Trust [WS-Trust]) as a mechanism for protecting Sequences. The use of WS-Security (as an underlying component of WS-SecureConversation) is subject to the constraints defined in the Basic Security Profile 1.0 [BSP 1.0].

The description provided here is general in nature and is not intended to serve as a complete definition on the use of WS-SecureConversation/WS-Trust to protect WS-RM. In order to interoperate implementations need to agree on the choice of features as well as the manner in which they will be used. The mechanisms described in the Web Services Security Policy Language [WS-SecurityPolicy] MAY be used by services to describe the requirements and constraints of the use of WS-SecureConversation.

5.2.2.1 Model

The basic model for using WS-SecureConversation is as follows:

1. The RM Source and the RM Destination create a WS-SecureConversation security context. This may involve the participation of third parties such as a security token service. The tokens exchanged may contain authentication claims (e.g. X.509 certificates or Kerberos service tickets).

2. The RM Source SHOULD reference the token of the security context created by (1) within the `CreateSequence` message to identify the security context that will be used to protect the Sequence. This is done so that, in cases where the `CreateSequence` message is signed by more than one security context, the RM Source can explicitly indicate which security context should be used to protect the newly created Sequence.

3. For the lifetime of the Sequence the RM Source and the RM Destination use the session key(s) associated with the security context to either sign or encrypt (as defined by WS-Security) at least the body and any relevant WS-RM-defined headers of any and all messages or faults that refer to that Sequence.

5.2.2.2 Countermeasure Implementation

Without relying upon any authentication information, the per-message signatures (or encryption blocks) provide the necessary integrity qualities to counter the threats described in Section 5.1.1.

To implement the countermeasures described in section 5.1.2.1 some mutually agreed upon form of authentication claims must be provided by the RM Source to the RM Destination during the establishment of the Security Context. These claims can then be used to determine if the RM Source is authorized to create a Sequence with the RM Destination.

This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring an RM node's Sequence peer to be equivalent to their security context session peer. This allows the authorization decisions described in section 5.1.3.2 to be based on the identity of the message's security context rather than on any authentication claims that may have been established during security context initiation. Note that other methods of using WS-SecurityConversation to implement the countermeasures (such as associating specific authentication claims to a Sequence) are possible but not covered by this document.

As with transport security, the requisite equivalence of a security context peer and with a Sequence peer limits the lifetime of a Sequence to the lifetime of the protecting security context. Unlike transport security, the association between a Sequence and its protecting security context cannot always be established implicitly at Sequence creation time. This is due to the fact that the `CreateSequence` and `CreateSequenceResponse` messages may be signed by more than one security context.

Copyright © OASIS Open 2005. All Rights Reserved.

Issues specific to the life-cycle management of WS-SecurityConversation security contexts (such as amending or renewing contexts) are outside the scope of this specification.

Copyright © OASIS Open 2005. All Rights Reserved.