5 Security Threats and Requirements

There are two sets of security requirements that need to be considered, those of the applications that use WS-RM and those of the WS-RM protocol itself.

In general, we cannot make 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. The sole exception to this is the creation of additional denial of service attack points. For example, a particular WS-RM implementation may implement the Reliable Messaging Destination (RMD) as an independent SOAP-processing node. Once an application begins using this WS-RM implementation it becomes vulnerable to attacks against the machine(s) and services that support the RMD.

The primary security requirement of the WS-RM protocol is to protect the WS-RM semantics and protocol invariants against various threats. The following sections detail some of these threats.

5.1 Integrity Threats

In general, any mechanism which allows an attacker to alter the information in a Sequence Traffic Message or Sequence Lifecycle Message, 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, the WS-RM specification states:

> The RM Source MUST assign each message within a Sequence a message number beginning at 1 and increasing by exactly 1 for each subsequent message. These numbers MUST be assigned in the same order in which messages are sent by the Application Source.

If an attacker is able to swap `<wsrm:Sequence>` headers on messages in transit between the RMS and RMD then they have undermined the implementation's ability to guarantee this invariant. 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.

5.2 Resource Consumption Threats

The creation of a Sequence with an RMD consumes various resources on the systems used to implement that RMD. These resources include network connections, database handles, database tables, etc. This behavior can be exploited to conduct denial of service attacks against an RMD. For example, a simple attack is to repeatedly send `<wsrm: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.

5.3 Sequence Spoofing Threats

Sequence spoofing is a class of threats in which the attacker uses its knowledge of the `<wsrm:Identifier>` for a particular Sequence to forge Sequence Lifecycle or Traffic Messages. For example the attacker creates a fake `<wsrm: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 `<wsrm:MessageNumber>` for their target Sequence.

In general any Sequence Lifecycle Message, RM Protocol Header Block, or sequence-correlated SOAP fault (e.g. `<wsrm: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 `<wsrm:SequenceAcknowledgement>` header into a message that it sends to the “AcksTo” EPR of an RMS.

5.3.1 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 `<wsrm:Sequence>` headers into those messages.

The following scenario provides an example:

1. An RMS and RMD create a Sequence with ID “urn:uuid:72dfcac0-3d09-11da-8cd6-0800200c9a66”.
2. The RMS transmits messages 1-10 under this sequence.
3. An attacker gains knowledge of the above sequence ID and message numbers and transmits messages 11-19 under “urn:uuid:72dfcac0...” to the RMD using fake sequence headers.

At this point one or both of the following could occur:

- The RMS transmits the “legitmate” message number 11 under “urn:uuid:72dfcac0...”. This message is silently ignored by the RMD since it considers message number 11 to have already been received. The result is that the “real” message number 11 is not received by the RMD and the RMS has no way of determining this fact.
- The RMS receives acknowledgments for messages 12-19 thus causing it to send a `<wsrm:InvalidAcknowledgement>` to the RMD (see Section 4.4 of [WS-RM]) and close the sequence.

Note that “sequence hijacking” should not be equated with “secure session hijacking”. Although a Sequence may be bound to a secure session in order to counter the threats described in this section, applications MUST NOT rely on sequence 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 peer (no Sequence to correlate) which contradicts requirement (1) in Section 1.1, Composition Requirements.

5.4 Message Correlation Threats

The `<wsrm:CreateSequenceRefused>`, `<wsrm:UnknownSequence>`, and `<wsrm:WSRMRequired>` fault messages are not correlated to a Sequence. Instead they are correlated to a particular request message. In the asynchronous case these fault messages are correlated using the WS-Addressing [WS-Addressing] defined `<wsa:RelatesTo>` header that references the `<wsa:MessageID>` of the request message. An attacker with knowledge of the message ID and the `<wsa:FaultTo>` EPR can use this information to undermine an RMS' ability to create or use a Sequence.

For example, an attacker with the ability to snoop the traffic between an RMS and an RMD would be able to observe the `<wsrm:CreateSequence>` message. Using the information in this message (specifically the `<wsa:MessageID>` and `<wsa:FaultTo>` headers) the attacker could create a phony `<wsrm:CreateSequenceRefused>` message and transmit it to the RMS' FaultTo endpoint before the RMD could process the request and respond.

Although this document refers to some of the threats involving the mis-use of WS-Addressing information it does not derive any security requirements to address these threats. The threats related to the use of WS-Addressing are felt to be both more general and more difficult to defend against than those that are strictly related to the use of WS-RM and are therefore outside the scope of these profiles.

5.5 Detailed Security Requirements

The following are specific security requirements of the WS-RM protocol:

1. The Sequence Lifecycle Messages should be integrity protected while in transit between the RMS and RMD.
2. The RM Protocol Header Blocks should be integrity protected while in transit between the RMS and RMD.
3. The `<wsrm:Sequence>` RM Protocol Header Block should be bound to the message body to which it applies.
4. SOAP headers that effect the semantics of the WS-RM message elements (such as a WS-Addressing defined `<wsa:ReplyTo>` header attached to a `<wsrm:CreateSequence>` message) should be bound to those messages.
5. It should be possible for an RMD to perform a check to ensure that the entity that issued a `<wsrm:CreateSequence>` message is entitled to create Sequences with the target RMD.
6. The RMS and RMD should be able to perform “Sequence ownership checks” against an entity that issues or responds with a `<wsrm:CreateSequenceResponse>`, `<wsrm:CloseSequence>`, `<wsrm:CloseSequenceResponse>`, `<wsrm:TerminateSequence>`, or `<wsrm:TerminateSequenceResponse>` message. A “Sequence ownership check” is a test to determine if the identity of particular entity (e.g. the creator of a `<wsrm:TerminateSequence>` message) is the same as the identity of the entity that “owns” (or “is permitted to operate on”) a particular Sequence. As we will see below, the determination of who (or what) owns a Sequence varies between profiles.
7. The RMS and RMD should be able to perform Sequence ownership checks against an entity that inserts an RM Protocol Header Block into a SOAP message.
8. The RMS and RMD should be able to perform Sequence ownership checks against an entity that issues or responds with the `<wsrm:SequenceTerminated>`, `<wsrm:InvalidAcknowledgement>`, `<wsrm:MessageNumberRollover>`, or `<wsrm:SequenceClosed>` fault messages.