Internet Draft
draft-ietf-rddp-security-03.txt
Category: Standards Track
Expires: February, 2005

James Pinkerton
Microsoft Corporation
Ellen Deleganes
Intel Corporation
Sara Bitan
Microsoft Corporation
August 2004

DDP/RDMAP Security

1 Status of this Memo

2 Abstract

This document analyzes security issues around implementation and use of the Direct Data Placement Protocol(DDP) and Remote Direct Memory Access Protocol (RDMAP). It first defines an architectural model for an RDMA Network Interface Card (RNIC), which can implement DDP or RDMAP and DDP. The document reviews various attacks against the resources defined in the architectural model and the countermeasures that can be used to protect the system. Attacks are grouped into spoofing, tampering, information disclosure, denial of service, and elevation of privilege. Finally, the document concludes with a summary of security services for DDP and RDMAP, such as IPSec.

Table of Contents

1 Status of this Memo 1
2 Abstract 1
2.1 Issues 3
2.2 Revision History 4
2.2.1 Changes from -02 to -03 version............................4
2.2.2 Changes from the -01 to the -02 version....................4
2.2.3 Changes from the -00 to -01 version........................4
3 Introduction 6
4 Architectural Model 8
4.1 Components 9
4.2 Resources 11
4.2.1 Stream Context Memory.....................................11
4.2.2 Data Buffers..11
4.2.3 Page Translation Tables...................................11
4.2.4 STag Namespace..12
4.2.5 Completion Queues...12
4.2.6 Asynchronous Event Queue..................................12
4.2.7 RDMA Read Request Queue...................................13
4.2.8 RNIC Interactions...13
4.2.8.1 Privileged Control Interface Semantics 13
4.2.8.2 Non-Privileged Data Interface Semantics 13
4.2.8.3 Privileged Data Interface Semantics 14
4.2.9 Initialization of RNIC Data Structures for Data Transfer..14
4.2.10 RNIC Data Transfer Interactions 15
5 Trust and Resource Sharing 17
6 Attacker Capabilities 18
7 Attacks and Countermeasures 19
7.1 Tools for Countermeasures 19
7.1.1 Protection Domain (PD)....................................19
7.1.2 Limiting STag Scope.......................................20
7.1.3 Access Rights...21
7.1.4 Limiting the Scope of the Completion Queue................21
7.1.5 Limiting the Scope of an Error............................21
7.2 Spoofing 21
7.2.1 Impersonation...22
7.2.2 Stream Hijacking..22
7.2.3 Man in the Middle Attack..................................22
7.2.4 Using an STag on a Different Stream.......................23
7.3 Tampering 24
7.3.1 Buffer Overrun - RDMA Write or Read Response..............24
7.3.2 Modifying a Buffer After Indication.......................25
7.3.3 Multiple STags to access the same buffer..................25
7.3.4 Network based modification of buffer content..............25
7.4 Information Disclosure 26
7.4.1 Probing memory outside of the buffer bounds...............26
7.4.2 Using RDMA Read to Access Stale Data......................26
7.4.3 Accessing a Buffer After the Transfer.....................26
7.4.4 Accessing Unintended Data With a Valid STag...............26
7.4.5 RDMA Read into an RDMA Write Buffer.......................27
7.4.6 Using Multiple STags Which Alias to the Same Buffer.......27

7.4.7 Remote Node Loading Firmware onto the RNIC................28
7.4.8 Controlling Access to PTT & STag Mapping..................28
7.4.9 Network based eavesdropping................................28
7.5 Denial of Service (DOS) 29
7.5.1 RNIC Resource Consumption..................................29
7.5.2 Resource Consumption By Active Applications...............30
7.5.2.1 Multiple Streams Sharing Receive Buffers 30
7.5.2.2 Local Peer Attacking a Shared CQ 31
7.5.2.3 Remote Peer Attacking a Shared CQ 32
7.5.2.4 Attacking the RDMA Read Request Queue 35
7.5.3 Resource Consumption by Idle Applications.................36
7.5.4 Exercise of non-optimal code paths........................36
7.5.5 Remote Invalidate an STag Shared on Multiple Streams......37
7.6 Elevation of Privilege 37
8 Security Services for RDMA and DDP 38
8.1 Introduction to Security Options 38
8.1.1 Introduction to IPsec......................................38
8.1.2 Introduction to SSL Limitations on RDMAP..................40
8.1.3 Applications Which Provide Security........................40
8.2 Requirements for IPsec Encapsulation of DDP 40
9 Security considerations 42
10 References 43
10.1 Normative References......................................43
10.2 Informative References....................................43
11 Appendix A: Implementing Client/Server Protocols 44
12 Appendix B: Summary Table of Attacks 47
13 Appendix C: Partial Trust Taxonomy 48
14 Author’s Addresses 50
15 Acknowledgments 51
16 Full Copyright Statement 52

Table of Figures

Figure 1 - RDMA Security Model......................................9

2.1 Issues

This section is temporary and will go away when all issues have been resolved.

Issue: Guidance for application protocols like NFS which implement security <TBD>......................................40

2.2 Revision History

2.2.1 Changes from -02 to -03 version

* ID changed from Informational to Standards Track. This caused previous RECOMMENDATIONS to be categorized into the categories of MUST, SHOULD, MAY, RECOMMENDED, and in one case, "recommended".

* Completed Appendix B: Summary of Attacks to provide a summary of implementation requirements for applications using RDDP and for RNICs in Appendix B: Summary of Attacks.

* Modified intro to better explain when concept of Partial Mutual Trust is useful.

* Misc minor changes from Tom Talpey's extensive review, including:

    * Send Queue/Receive Queue formally defined/used.

    * RI is gone, now use RNIC interface, RNIC, and Remote Invalidate.

    * Clarified attackers capabilities.

    * In many cases replaced "session" with "Stream".

    * Added definitions for equation variables in section 7.5.2.3.

* Changed section 8.2 to normative xref to IPS Security, plus comment on the value of end-to-end IPsec.

* Added clarifying example on STag invalidation (e.g. One-Shot STag discussion).

* Added clarifying text on why SSL is a bad idea.

* Normative statement on mitigation for Shared RQ.

2.2.2 Changes from the -01 to the -02 version

Minimal - some typos, deleted some text previously marked for deletion.

2.2.3 Changes from the -00 to -01 version

* Added two pages to the architectural model to describe the Asynchronous Event Queue, and the types of

interactions that can occur between the RNIC and the modules above it.

* Addressed Mike Krauses comments submitted on 12/8/2003

* Moved "Trust Models" from the body of the document to an appendix. Removed references to it throughout the document (including use of "partial trust". Document now assumes Remote Peer is untrusted. Thus the key issue is whether local resources are shared, and what the resource is.

* Misc cleanup throughout the document.

* The Summary of Attacks at the end of the document is now an Appendix. It also now provides a summary. Cleared change bars because became unreadable. Also shortened section names for attacks to fit in table.

* Added a new concept of "Partial Mutual Trust" between a collection of Streams to better characterize a set of attacks in a client/server environment.

* Filled in Security Services for RDMA and DDP section (almost all is new, except IPsec overview).

* Globally tried to change "connection" to "Stream". In some cases it can be either a connection or stream.

3 Introduction

RDMA enables new levels of flexibility when communicating between two parties compared to current conventional networking practice (e.g. a stream-based model or datagram model). This flexibility brings new security issues that must be carefully understood when designing application protocols utilizing RDMA and when implementing RDMA-aware NICs (RNICs). Note that for the purposes of this security analysis, an RNIC may implement RDMAP and DDP, or just DDP.

The specification first develops an architectural model that is relevant for the security analysis - it details components, resources, and system properties that may be attacked in Section 4.

It then defines what resources a ULP may share locally across Streams and what resources the ULP may share with the Remote Peer across Streams in Section 5. Intentional sharing of resources between multiple Streams may imply some level of trust between the Streams. However, some types of resource sharing have unmitigated security attacks which would mandate not sharing a specific type of resource unless there is some level of trust between the Streams sharing resources. Partial Mutual Trust is defined to address this concept:

> Partial Mutual Trust - a collection of RDMAP/DDP Streams, which represent the local and remote end points of the Stream, are willing to assume that the Streams from the collection will not perform malicious attacks against any of the Streams in the collection. Applications have explicit control of which collection of endpoints is in the collection through tools discussed in Section 7.1 Tools for Countermeasures on page 19.

An untrusted peer relationship is appropriate when an application wishes to ensure that it will be robust and uncompromised even in the face of a deliberate attack by its peer. For example, a single application that concurrently supports multiple unrelated Streams (e.g. a server) would presumably treat each of its peers as an untrusted peer. For a collection of Streams which share Partial Mutual Trust, the assumption is that any Stream not in the collection is untrusted. For the untrusted peer, a brief list of capabilities is enumerated in Section 6.

The rest of the specification is focused on analyzing attacks. First, the tools for mitigating attacks are listed (Section 7.1), and then a series of attacks on components, resources, or system properties is enumerated in the rest of Section 7. For each attack, possible countermeasures are reviewed. If all recommended mitigations are in place the implemented usage models, the

RDMAP/DDP protocol can be shown to not expose any new security vulnerabilities.

Applications within a host are divided into two categories - Privileged and Non-Privileged. Both application types can send and receive data and request resources. The key differences between the two are:

The Privileged Application is trusted by the system to not maliciously attack the operating environment, but it is not trusted to optimize resource allocation globally. For example, the Privileged Application could be a kernel application, thus the kernel presumably has in some way vetted the application before allowing it to execute.

A Non-Privileged Application’s capabilities are a logical sub-set of the Privileged Application’s. It is assumed by the local system that a Non-Privileged Application is untrusted. All Non-Privileged Application interactions with the RNIC Engine that could affect other applications need to be done through a trusted intermediary that can verify the Non-Privileged Application requests.

4 Architectural Model

This section describes an RDMA architectural reference model that is used as security issues are examined. It introduces the components of the model, the resources that can be attacked, the types of interactions possible between components and resources, and the system properties which must be preserved.

Figure 1 shows the components comprising the architecture and the interfaces where potential security attacks could be launched. External attacks can be injected into the system from an application that sits above the RNIC Interface or from the network.

The intent here is to describe high level components and capabilities which affect threat analysis, and not focus on specific implementation options. Also note that the architectural model is an abstraction, and an actual implementation may choose to subdivide its components along different boundary lines than defined here. For example, the Privileged Resource Manager may be partially or completely encapsulated in the Privileged Application. Regardless, it is expected that the security analysis of the potential threats and countermeasures still apply.

```
          +-------------+
          |  Privileged |
          |  Resource   |
 Admin<-+>|  Manager    |     App Control Interface
        | |             |<------+-------------------+
        | +-------------+       |                   |
        |       ^               v                   v
        |       |         +-------------+   +-----------------+
        |---------------->| Privileged  |   |  Non-Privileged |
                |         | Application |   |  Application    |
                |         +-------------+   +-----------------+
                |               ^                   ^
                |Privileged     |Privileged         |Non-Privileged
                |Control        |Data               |Data
                |Interface      |Interface          |Interface
RNIC            |               |                   |
Interface       v               v                   v
=================================================================

              +--------------------------------------+
              |                                      |
              |               RNIC Engine            | <-- Firmware
              |                                      |
              +--------------------------------------+
                                ^
                                |
                                v
                             Internet
```

Figure 1 - RDMA Security Model

4.1 Components

The components shown in Figure 1 - RDMA Security Model are:

* RNIC Engine (RNIC) - the component that implements the RDMA protocol and/or DDP protocol.

* Privileged Resource Manager - the component responsible for managing and allocating resources associated with the RNIC Engine. The Resource Manager does not send or receive data. Note that whether the Resource Manager is an independent component, part of the RNIC, or part of the application is implementation dependent. If a specific implementation does not wish to address security issues resolved by the Resource Manager, there may in fact be no resource manager at all.

* Privileged Application - See Section 3 Introduction for a definition of Privileged Application. The local host infrastructure can enable the Privileged Application to

map a data buffer directly from the RNIC Engine to the host through the RNIC Interface, but it does not allow the Privileged Application to directly consume RNIC Engine resources.

* Non-Privileged Application - See Section 3 Introduction for a definition of Non-Privileged Application. All Non-Privileged Application interactions with the RNIC Engine that could affect other applications MUST be done using the Privileged Resource Manager as a proxy.

A design goal of the DDP and RDMAP protocols is to allow, under constrained conditions, Non-Privileged applications to send and receive data directly to/from the RDMA Engine without Privileged Resource Manager intervention - while ensuring that the host remains secure. Thus, one of the primary goals of this paper is to analyze this usage model for the enforcement that is required in the RNIC Engine to ensure the system remains secure.

The host interfaces that could be exercised include:

* Privileged Control Interface - A Privileged Resource Manager uses the RNIC Interface to allocate and manage RNIC Engine resources, control the state within the RNIC Engine, and monitor various events from the RNIC Engine. It also uses this interface to act as a proxy for some operations that a Non-Privileged Application may require (after performing appropriate countermeasures).

* Application Control Interface - An application uses this interface to the Privileged Resource Manager to allocate RNIC Engine resources. The Privileged Resource Manager implements countermeasures to ensure that if the Non-Privileged Application launches an attack it can prevent the attack from affecting other applications.

* Non-Privileged Data Transfer Interface - A Non-Privileged Application uses this interface to initiate and to check the status of data transfer operations.

* Privileged Data Transfer Interface - A superset of the functionality provided by the Non-Privileged Data Transfer Interface. The application is allowed to directly manipulate RNIC Engine mapping resources to map an STag to an application data buffer.

* Figure 1 also shows the ability to load new firmware in the RNIC Engine. Not all RNICs will support this, but it is shown for completeness and is also reviewed under potential attacks.

If Internet control messages, such as ICMP, ARP, RIPv4, etc. are processed by the RNIC Engine, the threat analyses for those protocols is also applicable, but outside the scope of this paper.

4.2 Resources

This section describes the primary resources in the RNIC Engine that could be affected if under attack. For RDMAP, all of the defined resources apply. For DDP, all of the resources except the RDMA Read Queue apply.

4.2.1 Stream Context Memory

The state information for each Stream is maintained in memory, which could be located in a number of places - on the NIC, inside RAM attached to the NIC, in host memory, or in any combination of the three, depending on the implementation.

Stream Context Memory includes state associated with Data Buffers. For Tagged Buffers, this includes how STag names, Data Buffers, and Page Translation Tables inter-relate. It also includes the list of Untagged Data Buffers posted for reception of Untagged Messages (commonly called the Receive Queue), and a list of operations to perform to send data (commonly called the Send Queue).

4.2.2 Data Buffers

There are two different ways to expose a data buffer; a buffer can be exposed for receiving RDMAP Send Type Messages (a.k.a. DDP Untagged Messages) on DDP Queue zero or the buffer can be exposed for remote access through STags (a.k.a. DDP Tagged Messages). This distinction is important because the attacks and the countermeasures used to protect against the attack are different depending on the method for exposing the buffer to the network.

For the purposes of the security discussion, a single logical Data Buffer is exposed with a single STag. Actual implementations may support scatter/gather capabilities to enable multiple physical data buffers to be accessed with a single STag, but from a threat analysis perspective it is assumed that a single STag enables access to a single logical Data Buffer.

In any event, it is the responsibility of the RNIC to ensure that no STag can be created that exposes memory that the consumer had no authority to expose.

4.2.3 Page Translation Tables

Page Translation Tables are the structures used by the RNIC to be able to access application memory for data transfer operations.

Even though these structures are called "Page" Translation Tables, they may not reference a page at all - conceptually they are used to map an application address space representation of a buffer to the physical addresses that are used by the RNIC Engine to move data. If on a specific system a mapping is not used, then a subset of the attacks examined may be appropriate. Note that the Page Translation Table may or may not be a shared resource.

4.2.4 STag Namespace

The DDP specification defines a 32-bit namespace for the STag. Implementations may vary in terms of the actual number of STags that are supported. In any case, this is a bounded resource that can come under attack. Depending upon STag namespace allocation algorithms, the actual name space to attack may be significantly less than 2^{32}.

4.2.5 Completion Queues

Completion Queues are used in this specification to conceptually represent how the RNIC Engine notifies the Application about the completion of the transmission of data, or the completion of the reception of data through the Data Transfer Interface. Because there could be many transmissions or receptions in flight at any one time, completions are modeled as a queue rather than a single event. An implementation may also use the Completion Queue to notify the application of other activities, for example, the completion of a mapping of an STag to a specific application buffer. Completion Queues may be shared by a group of Streams, or may be designated to handle a specific Stream's traffic.

Some implementations may allow this queue to be manipulated directly by both Non-Privileged and Privileged applications.

4.2.6 Asynchronous Event Queue

The Asynchronous Event Queue is a queue from the RNIC to the Privileged Resource Manager of bounded size. It is used by the RNIC to notify the host of various events which might require management action, including protocol violations, Stream state changes, local operation errors, low water marks on receive queues, and possibly other events.

The Asynchronous Event Queue is a resource that can be attacked because Remote or Local Peers can cause events to occur which have the potential of overflowing the queue.

Note that an implementation is at liberty to implement the functions of the Asynchronous Event Queue in a variety of ways, including multiple queues or even simple callbacks. All vulnerabilities identified are intended to apply regardless of the implementation of the Asynchronous Event Queue. For example, a callback function is simply a very short queue.

4.2.7 RDMA Read Request Queue

The RDMA Read Request Queue is the memory that holds state information for one or more RDMA Read Request Messages that have arrived, but for which the RDMA Read Response Messages have not yet been completely sent. Because potentially more than one RDMA Read Request can be outstanding at one time, the memory is modeled as a queue of bounded size.

4.2.8 RNIC Interactions

With RNIC resources and interfaces defined, it is now possible to examine the interactions supported by the generic RNIC functional interfaces through each of the 3 interfaces - Privileged Control Interface, Privileged Data Interface, and Non-Privileged Data Interface.

4.2.8.1 Privileged Control Interface Semantics

Generically, the Privileged Control Interface controls the RNIC's allocation, deallocation, and initialization of RNIC global resources. This includes allocation and deallocation of Stream Context Memory, Page Translation Tables, STag names, Completion Queues, RDMA Read Request Queues, and Asynchronous Event Queues.

The Privileged Control Interface is also typically used for managing Non-Privileged Application resources for the Non-Privileged Application (and possibly for the Privileged Application as well). This includes initialization and removal of Page Translation Table resources, and managing RNIC events (possibly managing all events for the Asynchronous Event Queue).

4.2.8.2 Non-Privileged Data Interface Semantics

The Non-Privileged Data Interface enables data transfer (transmit and receive) but does not allow initialization of the Page Translation Table resources. However, once the Page Translation Table resources have been initialized, the interface may enable a specific STag mapping to be enabled and disabled by directly communicating with the RNIC, or create an STag mapping for a buffer that has been previously initialized in the RNIC.

For RDMAP, transmitting data means sending RDMAP Send Type Messages, RDMA Read Requests, and RDMA Writes. For data reception, for RDMAP it can receive Send Type Messages into buffers that have been posted on the Receive Queue or Shared Receive Queue. It can also receive RDMA Write and RDMA Read Response Messages into buffers that have previously been exposed for external write access through advertisement of an STag.

For DDP, transmitting data means sending DDP Tagged or Untagged Messages. For data reception, for DDP it can receive Untagged Messages into buffers that have been posted on the Receive Queue or Shared Receive Queue. It can also receive Tagged DDP Messages into buffers that have previously been exposed for external write access through advertisement of an STag.

Completion of data transmission or reception generally entails informing the application of the completed work by placing completion information on the Completion Queue.

4.2.8.3 Privileged Data Interface Semantics

The Privileged Data Interface semantics are a superset of the Non-Privileged Data Transfer semantics. The interface can do everything defined in the prior section, as well as create/destroy buffer to STag mappings directly. This generally entails initialization or clearing of Page Translation Table state in the RNIC.

4.2.9 Initialization of RNIC Data Structures for Data Transfer

Initialization of the mapping between an STag and a Data Buffer can be viewed in the abstract as two separate opertions:

a. Initialization of the allocated Page Translation Table entries with the location of the Data Buffer, and

b. Initialization of a mapping from an allocated STag name to a set of Page Translation Table entry(s) or partial-entries.

Note that an implementation may not have a Page Translation Table (i.e. it may support a direct mapping between an STag and a Data Buffer). In this case threats and mitigations associated with the Page Translation Table are not relevant.

Initialization of the contents of the Page Translation Table can be done by either the Privileged Application or by the Privileged Resource Manager as a proxy for the Non-Privileged Application. By definition the Non-Privileged Application is not trusted to directly manipulate the Page Translation Table. In general the concern is that the Non-Privileged application may try to maliciously initialize the Page Translation Table to access a buffer for which it does not have permission.

The exact resource allocation algorithm for the Page Translation Table is outside the scope of this specification. It may be allocated for a specific Data Buffer, or be allocated as a pooled resource to be consumed by potentially multiple Data Buffers, or be managed in some other way. This paper attempts to abstract implementation dependent issues, and focus on higher level security issues such as resource starvation and sharing of resources between Streams.

The next issue is how an STag name is associated with a Data Buffer. For the case of an Untagged Data Buffer, there is no wire visible mapping between an STag and the Data Buffer. Note that there may, in fact, be an STag which represents the buffer. However, because the STag by definition is not visible on the wire, this is a local host specific issue which should be analyzed in the context of local host implementation specific security analysis, and thus is outside the scope of this paper.

For a Tagged Data Buffer, either the Privileged Application, the Non-Privileged Application, or the Privileged Resource Manager acting on behalf of the Non-Privileged Resource Manager may initialize a mapping from an STag to a Page Translation Table, or may have the ability to simply enable/disable an existing STag to Page Translation Table mapping. There may also be multiple STag names which map to a specific group of Page Translation Table entries (or sub-entries). Specific security issues with this level of flexibility are examined in Section 7.3.3 Multiple STags to access the same buffer on page 25.

There are a variety of implementation options for initialization of Page Translation Table entries and mapping an STag to a group of Page Translation Table entries which have security repercussions. This includes support for separation of Mapping an STag verses mapping a set of Page Translation Table entries, and support for Applications directly manipulating STag to Page Translation Table entry mappings (verses requiring access through the Privileged Resource Manager).

4.2.10 RNIC Data Transfer Interactions

RNIC Data Transfer operations can be subdivided into send operations and receive operations.

For send operations, there is typically a queue that enables the Application to post multiple operations to send data (referred to as the Send Queue). Depending upon the implementation, Data Buffers used in the operations may or may not have Page Translation Table entries associated with them, and may or may not have STags associated with them. Because this is a local host specific implementation issue rather than a protocol issue, the security analysis of threats and mitigations is left to the host implementation.

Receive operations are different for Tagged Data Buffers verses Untagged Data Buffers. If more than one Untagged Data Buffer can be posted by the Application, the DDP specification requires that they be consumed in sequential order. Thus the most general implementation is that there is a sequential queue of receive

Untagged Data Buffers (Receive Queue). Some implementations may also support sharing of the sequential queue between multiple Streams. In this case defining "sequential" becomes non-trivial - in general the buffers for a single stream are consumed from the queue in the order that they were placed on the queue, but there is no order guarantee between streams.

For receive Tagged Data Buffers, at some time prior to data transfer, the mapping of the STag to specific Page Translation Table entries (if present) and the mapping from the Page Translation Table entries to the Data Buffer must have been initialized (see the prior section for interaction details).

5 Trust and Resource Sharing

It is assumed that in general the Local and Remote Peer are untrusted, and thus attacks by either should have mitigations in place.

A separate, but related issue is resource sharing between multiple streams. If local resources are not shared, the resources are dedicated on a per Stream basis. Resources are defined in Section 4.2 - Resources on page 10. The advantage of not sharing resources between Streams is that it reduces the types of attacks that are possible. The disadvantage is that applications might run out of resources.

It is assumed in this paper that the component that implements the mechanism to control sharing of the RNIC Engine resources is the Privileged Resource Manager. The RNIC Engine exposes its resources through the RNIC Interface to the Privileged Resource Manager. All Privileged and Non-Privileged applications request resources from the Resource Manager. The Resource Manager implements resource management policies to ensure fair access to resources. The Resource Manager should be designed to take into account security attacks detailed in this specification. Note that for some systems the Privileged Resource Manager may be implemented within the Privileged Application.

The sharing of resources across Streams should be under the control of the application, both in terms of the trust model the application wishes to operate under, as well as the level of resource sharing the application wishes to give Local Peer processes. For more discussion on types of trust models which combine partial trust and sharing of resources, see Appendix C: Partial Trust Taxonomy on page 48.

6 Attacker Capabilities

An attacker's capabilities delimit the types of attacks that attacker is able to launch. RDMAP and DDP require that the initial LLP Stream (and connection) be set up prior to transferring RDMAP/DDP Messages. Attackers with send only capabilities must first guess the current LLP Stream parameters before they can attack RNIC resources (e.g. TCP sequence number). Attackers with both send and receive capabilities have presumably setup a valid LLP Stream, and thus have a wider ability to attack RNIC resources.

7 Attacks and Countermeasures

This section describes the attacks that are possible against the RDMA system defined in Figure 1 - RDMA Security Model and the RNIC Engine resources defined in Section 4.2. The analysis includes a detailed description of each attack, what is being attacked, and a description of the countermeasures that can be taken to thwart the attack.

Note that connection setup and teardown is presumed to be done in stream mode (i.e. no RDMA encapsulation of the payload), so there are no new attacks related to connection setup/teardown beyond what is already present in the LLP (e.g. TCP or SCTP). Note, however, that RDMAP/DDP parameters may be exchanged in stream mode, and if they are corrupted by an attacker unintended consequences will result. Therefore, any existing mitigations for LLP Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, or Elevation of Privilege continues to apply (and is out of scope of this document). Thus the analysis in this section focuses on attacks that are present regardless of the LLP Stream type.

The attacks are classified into five categories: Spoofing, Tampering, Information Disclosure, Denail of Service (DoS) attacks, and Elevation of Privileges. Tampering is any modification of the legitimate traffic (machine internal or network). Spoofing attack is a special case of tempering; where the attacker falsifies an identity of the Remote Peer (identity can be an IP address, machine name, ULP level identity etc.).

7.1 Tools for Countermeasures

The tools described in this section are the primary mechanisms that can be used to provide countermeasures to potential attacks.

7.1.1 Protection Domain (PD)

Protection Domains are associated with two of the resources of concern, Stream Context Memory and STags associated with Page Translation Table entries and data buffers. Protection Domains are used mainly to ensure that an STag can only be used to access the associated data buffer through Streams in the same Protection Domain as that STag.

If an implementation chooses to not share resources between Streams, it is recommended that each Stream be associated with its own, unique Protection Domain. If an implementation chooses to allow resource sharing, it is recommended that Protection Domain be limited to the number of Streams that have Partial Mutual Trust.

Note that an application (either Privileged or Non-Privileged) can potentially have multiple Protection Domains. This could be used, for example, to ensure that multiple clients of a server do not have the ability to corrupt each other. The server would allocate a Protection Domain per client to ensure that resources covered by the Protection Domain could not be used by another (untrusted) client.

7.1.2 Limiting STag Scope

The key to protecting a local data buffer is to limit the scope of its STag to the level appropriate for the Streams which share Partial Mutual Trust. The scope of the STag can be measured in multiple ways.

* Number of Connections and/or Streams on which the STag is valid. One way to limit the scope of the STag is to limit the connections and/or Streams that are allowed to use the STag. As noted in the previous section, use of Protection Domains appropriately can limit the scope of the STag. The analysis presented in this document assumes two mechanisms for limiting the scope of Streams for which the STag is valid:

    * Protection Domain scope. The STag is valid if used on any Stream within a specific Protection Domain, and is invalid if used on any Stream that is not a member of the Protection Domain.

    * Single Stream scope. The STag is valid on a single Stream, regardless of what the Stream association is to a Protection Domain. If used on any other Stream, it is invalid.

* Limit the time an STag is valid. By Invalidating an Advertised STag (e.g., revoking remote access to the buffers described by an STag when done with the transfer), an entire class of attacks can be eliminated.

* Limit the buffer the STag can reference. Limiting the scope of an STag access to *just* the intended application buffers to be exposed is critical to prevent certain forms of attacks.

* Allocating and/or advertising STag numbers in an unpredictable way. If STags are allocated/advertised using an algorithm which makes it hard for the attacker to guess which STag(s) are currently in use, it makes it more difficult for an attacker to guess the correct value. As stated in the RDMAP specification [RDMAP], an invalid STag will cause the RDMAP Stream to be terminated. For the case of [DDP], at a minimum it must

signal an error to the ULP, and commonly this will cause the DDP stream to be terminated.

7.1.3 Access Rights

Access Rights associated with a specific Advertised STag or RDMAP/DDP Stream provide another mechanism for applications to limit the attack capabilities of the Remote Peer. The Local Peer can control whether a data buffer is exposed for local only, or local and remote access, and assign specific access privileges (read, write, read and write) on a per stream basis.

For DDP, when an STag is advertised, the Remote Peer is presumably given write access rights to the data (otherwise there was not much point to the advertisement). For RDMAP, when an application advertises an STag, it can enable write-only, read-only, or both write and read access rights.

Similarly, some applications may wish to provide a single buffer with different access rights on a per-Stream or per-Stream basis. For example, some Streams may have read-only access, some may have remote read and write access, while on other Streams only the Local Peer is allowed access.

7.1.4 Limiting the Scope of the Completion Queue

Completions associated with sending and receiving data, or setting up buffers for sending and receiving data, could be accumulated in a shared Completion Queue for a group of RDMAP/DDP Streams, or a specific RDMAP/DDP Stream could have a dedicated Completion Queue. Limiting Completion Queue association to one, or a small number of RDMAP/DDP Streams can prevent several forms of Denial of Service attacks.

7.1.5 Limiting the Scope of an Error

To prevent a variety of attacks, it is important that an RDMAP/DDP implementation be robust in the face of errors. If an error on a specific Stream can cause other unrelated Streams to fail, then a broad class of attacks are enabled against the implementation.

For example, an error on a specific RDMAP stream should not cause the RNIC to stop processing incoming packets, or corrupt a receive queue for an unrelated stream.

7.2 Spoofing

Spoofing attacks can be launched by the Remote Peer, or by a network based attacker. A network based spoofing attack applies to all Remote Peers.

Because the RDMAP Stream requires an LLP Stream in the ESTABLISHED state, certain types of traditional forms of wire attacks do not apply -- an end-to-end handshake must have occurred to establish the RDMAP Stream. So, the only form of spoofing that applies is one when a remote node can both send and receive packets. Yet even with this limitation the Stream is still exposed to the following spoofing attacks.

7.2.1 Impersonation

A network based attacker can impersonate a legal RDMA/DDP peer (by spoofing a legal IP address), and establish an RDMA/DDP Stream with the victim. End to end authentication (i.e. IPsec, SSL or ULP authentication) provides protection against this attack. For additional information see Section 8, Security Services for RDMA and DDP, on page 38.

7.2.2 Stream Hijacking

Stream hijacking happens when a network based attacker follows the Stream establishment phase, and waits until the authentication phase (if such a phase exists) is completed successfully. He can then spoof the IP address and re-direct the Stream from the victim to its own machine. For example, an attacker can wait until an iSCSI authentication is completed successfully, and hijack the iSCSI Stream.

The best protection against this form of attack is end-to-end integrity protection and authentication, such as IPsec (see Section 8, Security Services for RDMA and DDP, on page 38), to prevent spoofing. Another option is to provide physical security. Discussion of physical security is out of scope for this document.

Because the connection and/or Stream itself is established by the LLP, some LLPs are more difficult to hijack than others. Please see the relevant LLP documentation on security issues around connection and/or Stream hijacking.

7.2.3 Man in the Middle Attack

If a network based attacker has the ability to delete, inject replay, or modify packets which will still be accepted by the LLP (e.g., TCP sequence number is correct) then the Stream can be exposed to a man in the middle attack. One style of attack is for the man-in-the-middle to send Tagged Messages (either RDMAP or DDP). If it can discover a buffer that has been exposed for STag enabled access, then the man-in-the-middle can use an RDMA Read operation to read the contents of the associated data buffer, perform an RDMA Write Operation to modify the contents of the associated data buffer, or invalidate the STag to disable further access to the buffer. The only countermeasure for this form of attack is to either secure the RDMAP/DDP Stream (i.e. integrity protect) or attempt to provide physical security to prevent man-in-the-middle type attacks.

The best protection against this form of attack is end-to-end integrity protection and authentication, such as IPsec (see Section 8 Security Services for RDMA and DDP on page 38), to prevent spoofing or tampering. If Stream or session level authentication and integrity protection are not used, then a man-in-the-middle attack can occur, enabling spoofing and tampering.

Because the connection/Stream itself is established by the LLP, some LLPs are more exposed to man-in-the-middle attack then others. Please see the relevant LLP documentation on security issues around connection and/or Stream hijacking.

Another approach is to restrict access to only the local subnet/link, and provide some mechanism to limit access, such as physical security or 802.1.x. This model is an extremely limited deployment scenario, and will not be further examined here.

7.2.4 Using an STag on a Different Stream

One style of attack from the Remote Peer is for it to attempt to use STag values that it is not authorized to use. Note that if the Remote Peer sends an invalid STag to the Local Peer, per the DDP and RDMAP specifications, the Stream must be torn down. Thus the threat exists if a STag has been enabled for Remote Access on one Stream and a Remote Peer is able to use it on an unrelated Stream. If the attack is successful, the attacker could potentially be able to perform either RDMA Read Operations to read the contents of the associated data buffer, perform RDMA Write Operations to modify the contents of the associated data buffer, or to Invalidate the STag to disable further access to the buffer.

An attempt by a Remote Peer to access a buffer with an STag on a different Stream in the same Protection Domain may or may not be an attack depending on whether resource sharing is intended (i.e. whether the Streams shared Partial Mutual Trust or not). For some applications using an STag on multiple Streams within the same Protection Domain could be desired behavior. For other applications attempting to use an STag on a different Stream could be considered to be an attack. Since this varies by application, an application typically would need to be able to control the scope of the STag.

In the case where an implementation does not share resources between Streams (including STags), this attack can be defeated by assigning each Stream to a different Protection Domain. Before allowing remote access to the buffer, the Protection Domain of the Stream where the access attempt was made is matched against the Protection Domain of the STag. If the Protection Domains do not match, access to the buffer is denied, an error is generated, and the RDMAP Stream associated with the attacking Stream should be terminated.

For implementations that share resources between multiple Streams, it may not be practical to separate each Stream into its own Protection Domain. In this case, the application can still limit the scope of any of the STags to a single Stream (if it is enabling it for remote access). If the STag scope has been limited to a single Stream, any attempt to use that STag on a different Stream will result in an error, and the RDMA Stream should be terminated.

Thus for implementations that do not share STags between Streams, each Stream MUST either be in a separate Protection Domain or the scope of an STag be limited to a single Stream.

An additional issue may be unintended sharing of STags (i.e. a bug in the application) or a bug in the Remote Peer which causes an off-by-one STag to be used. For additional protection, an implementation SHOULD allocate STags in such a fashion that it is difficult to predict the next allocated STag number. Allocation methods which deterministically allocate the next STag should be avoided (e.g. a method which always starts with STag equal to one and monotonically increases it for each new allocation, or a method which always uses the same STag for each operation).

7.3 Tampering

A Remote Peer or a network based attacker can attempt to tamper with the contents of data buffers on a Local Peer that have been enabled for remote write access. The types of tampering attacks that are possible are outlined in the sections that follow.

7.3.1 Buffer Overrun - RDMA Write or Read Response

This attack is an attempt by the Remote Peer to perform an RDMA Write or RDMA Read Response to memory outside of the valid length range of the data buffer enabled for remote write access. This attack can occur even when no resources are shared across Streams. This issue can also arise if the application has a bug.

The countermeasure for this type of attack must be in the RNIC implementation, using the STag. When the Local Peer specifies to the RNIC the base address and the number of bytes in the buffer that it wishes to make accessible, the RNIC must ensure that the base and bounds check are applied to any access to the buffer referenced by the STag before the STag is enabled for access. When an RDMA data transfer operation (which includes an STag) arrives on a Stream, a base and bounds byte granularity access check must be performed to ensure the operation accesses only memory locations within the buffer described by that STag.

Thus an RNIC implementation MUST ensure that a Remote Peer is not able to access memory outside of the buffer specified when the STag was enabled for remote access.

7.3.2 Modifying a Buffer After Indication

This attack can occur if a Remote Peer attempts to modify the contents of an STag referenced buffer by performing an RDMA Write or an RDMA Read Response after the Remote Peer has indicated to the Local Peer that the STag data buffer contents are ready for use. This attack can occur even when no resources are shared across Streams. Note that a bug in a Remote Peer, or network based tampering, could also result in this problem.

For example, assume the STag referenced buffer contains ULP control information as well as ULP payload, and the ULP sequence of operation is to first validate the control information and then perform operations on the control information. If the Remote Peer can perform an additional RDMA Write or RDMA Read Response (thus changing the buffer) after the validity checks have been completed but before the control data is operated on, the Remote Peer could force the ULP down operational paths that were never intended.

The Local Peer can protect itself from this type of attack by revoking remote access when the original data transfer has completed and before it validates the contents of the buffer. The Local Peer can either do this by explicitly revoking remote access rights for the STag when the Remote Peer indicates the operation has completed, or by checking to make sure the Remote Peer Invalidated the STag through the RDMAP Invalidate capability, and if it did not, the Local Peer then explicitly revokes the STag remote access rights.

The Local Peer SHOULD follow the above procedure to protect the buffer before it validates the contents of the buffer (or uses the buffer in any way).

7.3.3 Multiple STags to access the same buffer

See section 7.4.6 on page 27 for this analysis.

7.3.4 Network based modification of buffer content

This is actually a man in the middle attack - but only on the content of the buffer, as opposed to the man in the middle attack presented above, where both the signaling and content can be modified. See Section 7.2.3 Man in the Middle Attack on page 22.

7.4 Information Disclosure

The main potential source for information disclosure is through a local buffer that has been enabled for remote access. If the buffer can be probed by a Remote Peer on another Stream, then there is potential for information disclosure.

The potential attacks that could result in unintended information disclosure and countermeasures are detailed in the following sections.

7.4.1 Probing memory outside of the buffer bounds

This is essentially the same attack as described in Section 7.3.1, except an RDMA Read Request is used to mount the attack. The same countermeasure applies.

7.4.2 Using RDMA Read to Access Stale Data

If a buffer is being used for a combination of reads and writes (either remote or local), and is exposed to the Remote Peer with at least remote read access rights, the Remote Peer may be able to examine the contents of the buffer before they are initialized with the correct data. In this situation, whatever contents were present in the buffer before the buffer is initialized can be viewed by the Remote Peer, if the Remote Peer performs an RDMA Read.

Because of this, the Local Peer SHOULD ensure that no stale data is contained in the buffer before remote read access rights are granted (this can be done by zeroing the contents of the memory, for example).

7.4.3 Accessing a Buffer After the Transfer

If the Remote Peer has remote read access to a buffer, and by some mechanism tells the Local Peer that the transfer has been completed, but the Local Peer does not disable remote access to the buffer before modifying the data, it is possible for the Remote Peer to retrieve the new data.

This is similar to the attack defined in Section 7.3.2 Modifying a Buffer After Indicati on page 25. The same countermeasures apply. In addition, the Local Peer SHOULD grant remote read access rights only for the amount of time needed to retrieve the data.

7.4.4 Accessing Unintended Data With a Valid STag

If the Local Peer enables remote access to a buffer using an STag that references the entire buffer, but intends only a portion of the buffer to be accessed, it is possible for the Remote Peer to access the other parts of the buffer anyway.

To prevent this attack, the Local Peer MUST set the base and bounds of the buffer when the STag is initialized to expose only the data to be retrieved.

7.4.5 RDMA Read into an RDMA Write Buffer

One form of disclosure can occur if the access rights on the buffer enabled remote read, when only remote write access was intended. If the buffer contained application data, or data from a transfer on an unrelated Stream, the Remote Peer could retrieve the data through an RDMA Read operation.

The most obvious countermeasure for this attack is to not grant remote read access if the buffer is intended to be write-only. Then the Remote Peer would not be able to retrieve data associated with the buffer. An attempt to do so would result in an error and the RDMAP Stream associated with the Stream would be terminated.

Thus if an application only intends a buffer to be exposed for remote write access, it MUST set the access rights to the buffer to only enable remote write access.

7.4.6 Using Multiple STags Which Alias to the Same Buffer

Multiple STags which alias to the same buffer at the same time can result in unintentional information disclosure if the STags are used by different, mutually untrusted, Remote Peers. This model applies specifically to client/server communication, where the server is communicating with multiple clients, each of which do not mutually trust each other.

If only read access is enabled, then the Local Peer has complete control over information disclosure. Thus a server which intended to expose the same data (i.e. buffer) to multiple clients by using multiple STags to the same buffer creates no new security issues beyond what has already been described in this document. Note that if the server did not intend to expose the same data to the clients, it should use separate buffers for each client (and separate STags).

When one STag has remote read access enabled and a different STag has remote write access enabled to the same buffer, it is possible for one Remote Peer to view the contents that have been written by another Remote Peer.

If both STags have remote write access enabled and the two Remote Peers do not mutually trust each other, it is possible for one

Remote Peer to overwrite the contents that have been written by the other Remote Peer.

Thus multiple Remote Peers which do not share Partial Mutual Trust MUST NOT be granted write access to the same buffer through different STags. A buffer should be exposed to only one untrusted Remote Peer at a time to ensure that no information disclosure or information tampering occurs between peers.

7.4.7 Remote Node Loading Firmware onto the RNIC

If the Remote Peer can cause firmware to be loaded onto the RNIC, there is an opportunity for information disclosure. See Elevation of Privilege in Section 7.6 for this analysis.

7.4.8 Controlling Access to PTT & STag Mapping

If a Non-Privileged application is able to directly manipulate the RNIC Page Translation Tables (which translate from an STag to a host address), it is possible that the Non-Privileged application could point the Page Translation Table at an unrelated application’s buffers and thereby be able to gain access to information in the unrelated application.

As discussed in Section 4 Architectural Model on page 8, introduction of a Privileged Resource Manager to arbitrate the mapping requests is an effective countermeasure. This enables the Privileged Resource Manager to ensure an application can only initialize the Page Translation Table (PTT)to point to its own buffers.

Thus if Non-Privileged applications are supported, the Privileged Resource Manager MUST verify that the Non-Privileged application has the right to access a specific Data Buffer before allowing an STag for which the application has access rights to be associated with a specific Data Buffer. This can be done when the Page Translation Table is initialized to access the Data Buffer or when the STag is initialized to point to a group of Page Translation Table entries, or both.

7.4.9 Network based eavesdropping

An attacker that is able to eavesdrop on the network can read the content of all read and write access to the peer’s buffers. To prevent information disclosure, the read/written data must be encrypted. See also Section 7.2.3 Man in the Middle Attack on page 22. The encryption can be done either by the ULP, or by a protocol that provides security services to the LLP (e.g. IPsec or SSL). Refer to section 8 for discussion of security services for DDP/RDMA.

7.5 Denial of Service (DOS)

A DOS attack is one of the primary security risks of RDMAP. This is because RNIC resources are valuable and scarce, and many application environments require communication with untrusted Remote Peers. If the remote application can be authenticated or encrypted, clearly, the DOS profile can be reduced. For the purposes of this analysis, it is assumed that the RNIC must be able to operate in untrusted environments, which are open to DOS style attacks.

Denial of service attacks against RNIC resources are not the typical unknown party spraying packets at a random host (such as a TCP SYN attack). Because the connection/Stream must be fully established, the attacker must be able to both send and receive messages over that connection/Stream, or be able to guess a valid packet on an existing RDMAP Stream.

This section outlines the potential attacks and the countermeasures available for dealing with each attack.

7.5.1 RNIC Resource Consumption

This section covers attacks that fall into the general category of a Local Peer attempting to unfairly allocate scarce (i.e. bounded) RNIC resources. The Local Peer may be attempting to allocate resources on its own behalf, or on behalf of a Remote Peer. Resources that fall into this category include: Protection Domains, Stream Context Memory, Translation and Protection Tables, and STag namespace. These can be attacks by currently active Local Peers or ones that allocated resources earlier, but are now idle.

This type of attack can occur regardless of whether resources are shared across Streams.

The allocation of all scarce resources MUST be placed under the control of a Privileged Resource Manager. This allows the Privileged Resource Manager to:

* prevent a Local Peer from allocating more than its fair share of resources.

* detect if a Remote Peer is attempting to launch a DOS attack by attempting to create an excessive number of Streams and take corrective action (such as refusing the request or applying network layer filters against the Remote Peer).

This analysis assumes that the Resource Manager is responsible for handing out Protection Domains, and RNIC implementations will provide enough Protection Domains to allow the Resource Manager to be able to assign a unique Protection Domain for each unrelated, untrusted Local Peer (for a bounded, reasonable number of Local Peers). This analysis further assumes that the Resource Manager implements policies to ensure that untrusted Local Peers are not able to consume all of the Protection Domains through a DOS attack. Note that Protection Domain consumption cannot result from a DOS attack launched by a Remote Peer, unless a Local Peer is acting on the Remote Peer’s behalf.

7.5.2 Resource Consumption By Active Applications

This section describes DOS attacks from Local and Remote Peers that are actively exchanging messages. Attacks on each RDMA NIC resource are examined and specific countermeasures are identified. Note that attacks on Stream Context Memory, Page Translation Tables, and STag namespace are covered in Section 7.5.1 RNIC Resource Consumption, so are not included here.

7.5.2.1 Multiple Streams Sharing Receive Buffers

The Remote Peer can attempt to consume more than its fair share of receive data buffers (Untagged DDP buffers or for RDMAP buffers consumed with Send Type Messages) if receive buffers are shared across multiple Streams.

If resources are not shared across multiple Streams, then this attack is not possible because the Remote Peer will not be able to consume more buffers than were allocated to the Stream. The worst case scenario is that the Remote Peer can consume more receive buffers than the Local Peer allowed, resulting in no buffers to be available, which could cause the Remote Peer’s Stream to the Local Peer to be torn down, and all allocated resources to be released.

If local receive data buffers are shared among multiple Streams, then the Remote Peer can attempt to consume more than its fair share of the receive buffers, causing a different Stream to be short of receive buffers, thus possibly causing the other Stream to be torn down. For example, if the Remote Peer sent enough one byte Untagged Messages, they might be able to consume all local shared receive queue resources with little effort on their part.

One method the Local Peer could use is to recognize that a Remote Peer is attempting to use more than its fair share of resources and terminate the Stream (causing the allocated resources to be released). However, if the Local Peer is sufficiently slow, it may be possible for the Remote Peer to still mount a denial of service attack. One countermeasure that can protect against this attack is implementing a low-water notification. The low-water notification alerts the application if the number of buffers in the receive queue is less than a threshold.

If all of the following conditions are true, then the Local Peer can size the amount of local receive buffers posted on the receive queue to ensure a DOS attack can be stopped.

* a low-water notification is enabled, and

* the Local Peer is able to bound the amount of time that it takes to replenish receive buffers, and

* the Local Peer maintains statistics to determine which Remote Peer is consuming buffers.

The above conditions enable the low-water notification to arrive before resources are depleted and thus the Local Peer can take corrective action (e.g., terminate the Stream of the attacking Remote Peer).

A different, but similar attack is if the Remote Peer sends a significant number of out-of-order packets and the RNIC has the ability to use the application buffer as a reassembly buffer. In this case the Remote Peer can consume a significant number of application buffers, but never send enough data to enable the application buffer to be completed to the application.

An effective countermeasure is to create a high-water notification which alerts the application if there is more than a specified number of receive buffers "in process" (partially consumed, but not completed). The notification is generated when more than the specified number of buffers are in process simultaneously on a specific Stream (i.e., packets have started to arrive for the buffer, but the buffer has not yet been delivered to the ULP).

A different countermeasure is for the RNIC Engine to provide the capability to limit the Remote Peer's ability to consume receive buffers on a per Stream basis. Unfortunately this requires a large amount of state to be tracked in each RNIC on a per Stream basis.

Thus, if an RNIC Engine provides the ability to share receive buffers across multiple Streams, the combination of the RNIC Engine and the Privileged Resource Manager MUST be able to detect if the Remote Peer is attempting to consume more than its fair share of resources so that the Local Peer can apply countermeasures to detect and prevent the attack.

7.5.2.2 Local Peer Attacking a Shared CQ

DOS attacks against a Shared Completion Queue (CQ) can be caused by either the Local Peer or the Remote Peer if either attempts to cause more completions than its fair share of the number of entries, thus potentially starving another unrelated Stream such that no Completion Queue entries are available.

A Completion Queue entry can potentially be consumed by a completion from the Send Queue or a Receive Queue completion. In the former, the attacker is the Local Peer. In the later, the attacker is the Remote Peer.

A form of attack can occur where the Local Peers can consume resources on the CQ. A Local Peer that is slow to free resources on the CQ by not reaping the completion status quickly enough could stall all other Local Peers attempting to use that CQ.

One of two countermeasures can be used to avoid this kind of attack. The first is to only share a CQ between Streams that share Partial Mutual Trust (i.e. Streams within the same Protection Domain). The other is to use a trusted Local Peer to act as a third party to free resources on the CQ and place the status in intermediate storage until the untrusted Local Peer reaps the status information. For these reasons, an RNIC MUST NOT enable sharing a CQ across Streams that belong to different Protection Domains. Addtionally, an application SHOULD NOT share a CQ between Streams which do not share Partial Mutual Trust.

7.5.2.3 Remote Peer Attacking a Shared CQ

For an overview of the Shared CQ attack model, see Section 7.5.2.2.

The Remote Peer can attack a CQ by consuming more than its fair share of CQ entries by using one of the following methods:

* The ULP protocol allows the Remote Peer to reserve a specified number of CQ entries, possibly leaving insufficient entries for other Streams that are sharing the CQ.

* If the Remote Peer or Local Peer (or both) can attack the CQ by overwhelming the CQ with completions, then completion processing on other Streams sharing that Completion Queue can be affected (e.g. the Completion Queue overflows and stops functioning).

The first method of attack can be avoided if the ULP does not allow a Remote Peer to reserve CQ entries or there is a trusted intermediary such as a Privileged Resource Manager. Unfortunately it is often unrealistic to not allow a Remote Peer to reserve CQ entries - particularly if the number of completion entries is dependent on other ULP negotiated parameters, such as the amount of buffering required by the ULP. Thus an implementation MUST implement a Privileged Resource Manager to control the allocation of CQ entries. See Section 4.1 Components on page 9 for a definition of Privileged Resource Manager.

One way that a Local or Remote Peer can attempt to overwhelm a CQ with completions is by sending minimum length RDMAP/DDP Messages to cause as many completions (receive completions for the Remote Peer, send completions for the Local Peer) per second as possible. If it is the Remote Peer attacking, and we assume that the Local Peer does not run out of receive buffers (if they do, then this is a different attack, documented in Section 7.5.2.1 Multiple Streams Sharing Receive Buffers on page 30), then it might be possible for the Remote Peer to consume more than its fair share of Completion Queue entries. Depending upon the CQ implementation, this could either cause the CQ to overflow (if it is not large enough to handle all of the completions generated) or for another Stream to not be able to generate CQ entries (if the RNIC had flow control on generation of CQ entries into the CQ). In either case, the CQ will stop functioning correctly and any Streams expecting completions on the CQ will stop functioning.

This attack can occur regardless of whether all of the Streams associated with the CQ are in the same Protection Domain or are in different Protection Domains - the key issue is that the number of Completion Queue entries is less than the number of all outstanding operations that can cause a completion.

The Local Peer can protect itself from this type of attack using either of the following methods:

* Size the CQ to the appropriate level, as specified below (note that if the CQ currently exists, and it needs to be resized, resizing the CQ can fail, so the CQ resize should be done before sizing the Send Queue and Receive Queue on the Stream), OR

* Grant fewer resources than the Remote Peer requested (not supplying the number of Receive Data Buffers requested).

The proper sizing of the CQ is dependent on whether the Local Peer will post as many resources to the various queues as the size of the queue enables or not. If the Local Peer can be trusted to post a number of resources that is smaller than the size of the specific resource’s queue, then a correctly sized CQ means that the CQ is large enough to hold completion status for all of the outstanding Data Buffers (both send and receive buffers), or:

```
CQ_MIN_SIZE = SUM(MaxPostedOnEachRQ)
            + SUM(MaxPostedOnEachSRQ)
            + SUM(MaxPostedOnEachSQ)
```

Where:

MaxPostedOnEachRQ = the maximum number of requests which can cause a completion that will be posted on a specific Receive Queue.

MaxPostedOnEachSRQ = the maximum number of requests which can cause a completion that will be posted on a specific Shared Receive Queue.

MaxPostedOnEachSQ = the maximum number of requests which can cause a completion that will be posted on a specific Send Queue.

If the local peer must be able to completely fill the queues, or can not be trusted to observe a limit smaller than the queues, then the CQ must be sized to accommodate the maximum number of operations that it is possible to post at any one time. Thus the equation becomes:

```
CQ_MIN_SIZE = SUM(SizeOfEachRQ)
            + SUM(SizeOfEachSRQ)
            + SUM(SizeOfEachSQ)
```

Where:

SizeOfEachRQ = the maximum number of requests which can cause a completion that can ever be posted on a specific Receive Queue.

SizeOfEachSRQ = the maximum number of requests which can cause a completion that can ever be posted on a specific Shared Receive Queue.

SizeOfEachSQ = the maximum number of requests which can cause a completion that can ever be posted on a specific Send Queue.

Where MaxPosted*OnEach*Q and SizeOfEach*Q varies on a per Stream or per Shared Receive Queue basis.

The Local Peer MUST implement a mechanism to ensure that the Completion Queue can not overflow. Note that it is possible to share CQs even if the Remote Peers accessing the CQs are untrusted if either of the above two formulas are implemented. If the Local Peer can be trusted to not post more than MaxPostedOnEachRQ, MaxPostedOnEachSRQ, and MaxPostedOnEachSQ, then the first formula applies. If the Local Peer can not be trusted to obey the limit, then the second formula applies.

7.5.2.4 Attacking the RDMA Read Request Queue

If RDMA Read Request Queue resources are pooled across multiple Streams, one attack is if the Local Peer attempts to unfairly allocate RDMA Read Request Queue resources for its Streams. For example, the Local Peer attempts to allocate all available resources on a specific RDMA Read Request Queue for its Streams, thereby denying the resource to applications sharing the RDMA Read Request Queue. The same type of argument applies even if the RDMA Read Request is not shared - but a Local Peer attempts to allocate all of the RNICs resource when the queue is created.

Thus access to interfaces that allocate RDMA Read Request Queue entries MUST be restricted to a trusted Local Peer, such as a Privileged Resource Manager. The Privileged Resource Manager SHOULD prevent a Local Peer from allocating more than its fair share of resources.

Another form of attack is if the Remote Peer sends more RDMA Read Requests than the depth of the RDMA Read Request Queue at the Local Peer. If the RDMA Read Request Queue is a shared resource, this could corrupt the queue. If the queue is not shared, then the worst case is that the current Stream is disabled. One approach to solving the shared RDMA Read Request Queue would be to create thresholds, similar to those described in Section 7.5.2.1 Multiple Streams Sharing Receive Buffers on page 30. A simpler approach is to not share RDMA Read Request Queue resources amoung Streams or enforce hard limits of consumption per Stream. Thus RDMA Read Request Queue resource consumption MUST be controlled such that RDMAP/DDP Streams which do not share Partial Mutual Trust do not share RDMA Read Request Queue resources.

If the issue is a bug in the Remote Peer's implementation, and not a malicious attack, the issue can be solved by requiring the Remote Peer's RNIC to throttle RDMA Read Requests. By properly configuring the Stream at the Remote Peer through a trusted agent, the RNIC can be made to not transmit RDMA Read Requests that exceed the depth of the RDMA Read Request Queue at the Local Peer. If the Stream is correctly configured, and if the Remote Peer submits more requests than the Local Peer's RDMA Read Request Queue can handle, the requests would be queued at the Remote Peer's RNIC until previous requests complete. If the Remote Peer's Stream is not configured correctly, the RDMAP Stream is terminated when more RDMA Read Requests arrive at the Local Peer than the Local Peer can handle (assuming the prior paragraph's recommendation is implemented). Thus an RNIC implementation MUST provide a mechanism to cap the number of outstanding RDMA Read Requests.

7.5.3 Resource Consumption by Idle Applications

The simplest form of a DOS attack given a fixed amount of resources is for the Remote Peer to create a RDMAP Stream to a Local Peer, and request dedicated resources then do no actual work. This allows the Remote Peer to be very light weight (i.e. only negotiate resources, but do no data transfer) and consumes a disproportionate amount of resources in the server.

A general countermeasure for this style of attack is to monitor active RDMAP Streams and if resources are getting low, reap the resources from RDMAP Streams that are not transferring data and possibly terminate the Stream. This would presumably be under administrative control.

Refer to Section 7.5.1 for the analysis and countermeasures for this style of attack on the following RNIC resources: Stream Context Memory, Page Translation Tables and STag namespace.

Note that some RNIC resources are not at risk of this type of attack from a Remote Peer because an attack requires the Remote Peer to send messages in order to consume the resource. Receive Data Buffers, Completion Queue, and RDMA Read Request Queue resources are examples. These resources are, however, at risk from a Local Peer that attempts to allocate resources, then goes idle. This could also be created if the ULP negotiates the resource levels with the Remote Peer, which causes the Local Peer to consume resources, however the Remote Peer never sends data to consume them. The general countermeasure described in this section can be used to free resources allocated by an idle Local Peer.

7.5.4 Exercise of non-optimal code paths

Another form of DOS attack is to attempt to exercise data paths that can consume a disproportionate amount of resources. An example might be if error cases are handled on a "slow path" (consuming either host or RNIC computational resources), and an attacker generates excessive numbers of errors in an attempt to consume these resources. Note that for most RDMAP or DDP errors, the attacking Stream will simply be torn down. Thus for this form of attack to be effective, the Remote Peer needs to exercise data paths which do not cause the Stream to be torn down.

If an RNIC implementation contains "slow paths" which do not result in the tear down of the Stream, it is recommended that an implementation provide the ability to detect the above condition and allow an administrator to act, including potentially administratively tearing down the RDMAP Stream associated with the Stream exercising data paths consuming a disproportionate amount of resources.

7.5.5 Remote Invalidate an STag Shared on Multiple Streams

If a Local Peer has enabled an STag for remote access, the Remote Peer could attempt to remote invalidate the STag by using the RDMAP Send with Invalidate or Send with SE and Invalidate Message. If the STag is only valid on the current Stream, then the only side effect is that the Remote Peer can no longer use the STag; thus there are no security issues.

If the STag is valid across multiple Streams, then the Remote Peer can prevent other Streams from using that STag by using the remote invalidate functionality.

Thus if RDDP Streams do not share Partial Mutual Trust (i.e. the Remote Peer may attempt to invalidate the STag prematurely), the application MUST NOT allow an STag to be valid across multiple Streams.

7.6 Elevation of Privilege

The RDMAP/DDP Security Architecture explicitly differentiates between three levels of privilege - Non-Privileged, Privileged, and the Privileged Resource Manager. If a Non-Privileged Application is able to elevate its privilege level to a Privileged Application, then mapping a physical address list to an STag can provide local and remote access to any physical address location on the node. If a Privileged Mode Application is able to promote itself to be a Resource Manager, then it is possible for it to perform denial of service type attacks where substantial amounts of local resources could be consumed.

In general, elevation of privilege is a local implementation specific issue and thus outside the scope of this specification.

There is one issue worth noting, however. If the RNIC implementation, by some insecure mechanism (or implementation defect), can enable a Remote Peer or un-trusted Local Peer to load firmware into the RNIC Engine, it is possible to use the RNIC to attack the host. Thus, an implementation MUST NOT enable firmware to be loaded on the RNIC Engine directly from a Remote Peer, unless the Remote Peer is properly authenticated (by a mechanism outside the scope of this specification. The mechanism presumably entails authenticating that the remote application has the right to perform the update), and the update is done via a secure protocol, such as IPsec (See Section 8 Security Services for RDMA and DDP on page 38). Further, an implementation MUST NOT allow a Non-Privileged Local Peer to update firmware in the RNIC Engine.

8 Security Services for RDMA and DDP

RDMA and DDP are used to control, read and write data buffers over IP networks. Therefore, the control and the data packets of these protocols are vulnerable to the spoofing, tampering and information disclosure attacks listed in Section 7.

Generally speaking, Stream confidentiality protects against eavesdropping. Stream and/or session authentication and integrity protection is a counter measurement against various spoofing and tampering attacks. The effectiveness of authentication and integrity against a specific attack, depend on whether the authentication is machine level authentication (as the one provided by IPsec and SSL), or ULP authentication.

8.1 Introduction to Security Options

The following security services can be applied to an RDMAP/DDP Stream:

1. Session confidentiality - protects against eavesdropping (section 7.4.9).

2. Per-packet data source authentication - protects against the following spoofing attacks: network based impersonation (section 7.2.1), Stream hijacking (section 7.2.2), and man in the middle (section 7.2.3).

3. Per-packet integrity - protects against tampering done by network based modification of buffer content (section 7.3.4)

4. Packet sequencing - protects against replay attacks, which is a special case of the above tampering attack.

If an RDMAP/DDP Stream may be subject to impersonation attacks, or Stream hijacking attacks, it is recommended that the Stream be authenticated, integrity protected, and protected from replay attacks; it MAY use confidentiality protection to protect from eavesdropping (in case the RDMAP/DDP Stream traverses a public network).

Both IPsec and SSL are capable of providing the above security services for IP and TCP traffic respectively. ULP protocols are able to provide only part of the above security services. The next sections describe the different security options.

8.1.1 Introduction to IPsec

IPsec is a protocol suite which is used to secure communication at the network layer between two peers. The IPsec protocol suite is specified within the IP Security Architecture [RFC2401], IKE [RFC2409], IPsec Authentication Header (AH) [RFC2402] and IPsec

Encapsulating Security Payload (ESP) [RFC2406] documents. IKE is the key management protocol while AH and ESP are used to protect IP traffic.

An IPsec SA is a one-way security association, uniquely identified by the 3-tuple: Security Parameter Index (SPI), protocol (ESP/AH) and destination IP address. The parameters for an IPsec security association are typically established by a key management protocol. These include the encapsulation mode, encapsulation type, session keys and SPI values.

IKE is a two phase negotiation protocol based on the modular exchange of messages defined by ISAKMP [RFC2408],and the IP Security Domain of Interpretation (DOI) [RFC2407]. IKE has two phases, and accomplishes the following functions:

1. Protected cipher suite and options negotiation - using keyed MACs and encryption and anti-replay mechanisms.

2. Master key generation - via Diffie-Hellman calculations.

3. Authentication of end-points (usually machine level authentication).

4. IPsec SA management (selector negotiation, options negotiation, create, delete, and rekeying).

Items 1 through 3 are accomplished in IKE Phase 1, while item 4 is handled in IKE Phase 2.

IKE phase 1 defines four authentication methods; three of them require both sides to have certified signature or encryption public keys; the forth require the side to exchange out-of-band a secret random string - called pre-shared-secret (PSS).

An IKE Phase 2 negotiation is performed to establish both an inbound and an outbound IPsec SA. The traffic to be protected by an IPsec SA is determined by a selector which has been proposed by the IKE initiator and accepted by the IKE Responder. The IPsec SA selector can be a "filter" or traffic classifier, defined as the 5-tuple: <Source IP address, Destination IP address, transport protocol (e.g. UDP/SCTP/TCP), Source port, Destination port>. The successful establishment of a IKE Phase-2 SA results in the creation of two uni-directional IPsec SAs fully qualified by the tuple <Protocol (ESP/AH), destination address, SPI>.

The session keys for each IPsec SA are derived from a master key, typically via a MODP Diffie-Hellman computation. Rekeying of an existing IPsec SA pair is accomplished by creating two new IPsec SAs, making them active, and then optionally deleting the older IPsec SA pair. Typically the new outbound SA is used immediately, and the old inbound SA is left active to receive packets for some locally defined time, perhaps 30 seconds or 1 minute. Optionally, rekeying can use Diffie-Helman for keying material generation.

8.1.2 Introduction to SSL Limitations on RDMAP

SSL and TLS [RFC 2246] provide Stream authentication, integrity and confidentiality for TCP based applications. SSL supports one-way (server only) or mutual certificates based authentication.

There are at least two limitations that make SSL underneath RDMAP less appropriate then IPsec for DDP/RDMA security:

1. The maximum length supported by the TLS record layer protocol is 2^14 bytes - longer packets must be fragmented (as a comparison, the maximal length of an IPsec packet is determined by the maximum length of an IP packet).

2. SSL is a connection oriented protocol. If a stream cipher or block cipher in CBC mode is used for bulk encryption, then a packet can be decrypted only after all the packets preceding it have already arrived. If SSL is used to protect DDP/RDMA traffic, then SSL must gather all out-of-order packets before RDMAP/DDP can place them into the ULP buffer, which might cause a significant decrease in its efficiency.

If SSL is layered on top of RDMAP or DDP, SSL does not protect the RDMAP and/or DDP headers. Thus a man-in-the-middle attack can still occur by modifying the RDMAP/DDP header to incorrectly place the data into the wrong buffer, thus effectively corrupting the data stream.

8.1.3 Applications Which Provide Security

Issue: Guidance for application protocols like NFS which implement security <TBD>.

8.2 Requirements for IPsec Encapsulation of DDP

The IP Storage working group has spent significant time and effort to define the normative IPSec requirements for IP Storage [RFC3723]. Portions of that specification are applicable to a wide variety of protocols, including the RDDP protocol suite. In order to not replicate this effort, an RNIC implementation MUST follow the requirements defined in RFC3723 Section 2.3 and Section 5, including the associated normative references for those sections.

Additionally, since IPsec acceleration hardware may only be able to handle a limited number of active IKE Phase 2 SAs, Phase 2 delete messages may be sent for idle SAs, as a means of keeping the number of active Phase 2 SAs to a minimum. The receipt of an

IKE Phase 2 delete message MUST NOT be interpreted as a reason for tearing down an DDP/RDMA Stream. Rather, it is preferable to leave the Stream up, and if additional traffic is sent on it, to bring up another IKE Phase 2 SA to protect it. This avoids the potential for continually bringing Streams up and down.

Note that there are serious security issues if IPSec is not implemented end-to-end. For example, if IPSec is implemented as a tunnel in the middle of the network, any hosts between the peer and the IPSec tunneling device can freely attack the unprotected Stream.

9 Security considerations

This entire specification is focused on security considerations.

10 References

10.1 Normative References

[RFC2828] Shirley, R., "Internet Security Glossary", FYI 36, RFC 2828, May 2000.

[DDP] Shah, H., J. Pinkerton, R.Recio, and P. Culley, "Direct Data Placement over Reliable Transports", Internet-Draft draft-ietf-rddp-ddp-01.txt, February 2003.

[RDMAP] Recio, R., P. Culley, D. Garcia, J. Hilland, "An RDMA Protocol Specification", Internet-Draft draft-ietf-rddp-rdmap-01.txt, February 2003.

[RFC3723] Aboba B., et al, "Securing Block Storage Protocols over IP", Internet draft (work in progress), RFC3723, April 2004.

[SCTP] R. Stewart et al., "Stream Control Transmission Protocol", RFC 2960, October 2000.

[TCP] Postel, J., "Transmission Control Protocol - DARPA Internet Program Protocol Specification", RFC 793, September 1981.

10.2 Informative References

[IPv6-Trust] Nikander, P., J.Kempf, E. Nordmark, "IPv6 Neighbor Discovery trust modelsTrust Models and threats", Internet-Draft draft-ietf-send-psreq-01.txt, January 2003.

11 Appendix A: Implementing Client/Server Protocols

<TBD: This section has not been updated to reflect the new normative focus of this specification. It will be updated in the next version.>

The prior sections outlined specific attacks and their countermeasures. This section summarizes the attacks and countermeasures defined in the prior section which are applicable to creation of a secure application server. An application server is defined as an application which must be able to communicate with many clients which do not trust each other and ensure that each client can not attack another client through server interactions. Further, the server may wish to use multiple Streams to communicate with a specific client, and those Streams may share mutual trust.

All of the prior section's details on attacks and countermeasures to protect a single Stream apply to the server. This section focuses on security issues where multiple clients are talking with a single server, and what mitigations the server application must have in place to ensure robust operation.

The following list summarizes the relevent attacks that clients can mount on the shared server, by re-stating the previous normative statements to be client/server specific:

* General Requirements

    * Section 4.1 Components on page 9. To ensure Non-Privileged applications running on the server can not create a DOS attack on each other, all Non-Privileged Application interactions with the RNIC Engine that could affect other applications MUST be done using the Privileged Resource Manager as a proxy.

* Spoofing

    * For protection against many forms of spoofing attacks, enable IPSec.

    * Section 7.2.4 Using an STag on a Different Stream on page 23. To ensure that one client can not access another client's data via use of the other client's STag, the server MUST either scope an STag to a single Stream or use a Protection Domain per client. If a single client has multiple streams that share Partial Mutual Trust, then the STag can be shared between the associated Streams by using a single Protection Domain amoung the associated Streams. To prevent unintended sharing of STags within the associated Streams, an implementation SHOULD allocate

STags in such a fashion that it is difficult to predict the next allocated STag number.

* Tampering

  * 7.3.1 Buffer Overrun - RDMA Write or Read Response on page 24. To ensure a client can not intentionally or accidentally cause a buffer overrun, an RNIC implementation MUST ensure that a Remote Peer is not able to access memory outside of the buffer specified when the STag was enabled for remote access.

  * 7.3.3 Multiple STags to access the same buffer on page 25. See the following bullet's discussion of Section 7.4.6.

* Information Disclosure

  * 7.4.2 Using RDMA Read to Access Stale Data on page 26. A server SHOULD ensure that no stale data is contained in a buffer before remote read access rights are granted to a client (this can be done by zeroing the contents of the memory, for example).

  * 7.4.5 RDMA Read into an RDMA Write Buffer on page 27. It is RECOMMENDED that if a server only intends a buffer to be exposed for remote write access, it set the access rights to the buffer to only enable remote write access.

  * 7.4.6 Using Multiple STags Which Alias to the Same Buffer on page 27. It is RECOMMENDED that separate clients not be granted write access to the same buffer through different STags. A buffer should be exposed to only one client at a time to ensure that no information disclosure or information tampering occurs between peers.

* Denial of Service

  * 7.5.1 RNIC Resource Consumption on page 29. It is RECOMMENDED that the server place the allocation of all scarce resources be placed under the control of a Privileged Resource Manager.

  * 7.5.2.1 Multiple Streams Sharing Receive Buffers on page 30. If an RNIC Engine provides the ability to share receive buffers across multiple Streams, it is RECOMMENDED that it enable the server to detect if the client is attempting to consume more than its fair share of resources so that the server can apply countermeasures to detect and prevent the attack.

* 7.5.2.2 Local Peer Attacking a Shared CQ on page 31. Sharing a CQ across Streams that belong to different Protection Domains is NOT RECOMMENDED.

* 7.5.2.3 Remote Peer Attacking a Shared CQ on page 32. If a server allows the client to influence CQ entry resource allocation, then it is RECOMMENDED that the CQ be isolated to Streams within a single Protection Domain (i.e. streams that share Partial Mutual Trust).

  It is RECOMMENDED that the Local Peer implement a mechanism to ensure that the Completion Queue can not overflow.

* 7.5.2.4 Attacking the RDMA Read Request Queue on page 35. It is RECOMMENDED that access to interfaces that allocate RDMA Read Request Queue entries be restricted to a trusted Local Peer, such as a Privileged Resource Manager.

  It is RECOMMENDED that RDMA Read Request Queue resource consumption be controlled such that RDMAP/DDP Streams which do not share Partial Mutual Trust do not share RDMA Read Request Queue resources.

* 7.5.3 Resource Consumption by Idle Applications on page 36. Refer to Section 7.5.1.

* 7.5.5 Remote Invalidate an STag Shared on Multiple Streams on page 37. If DDP/RDMAP Streams do not share Partial Mutual Trust (i.e. the client may attempt to invalidate the STag prematurely), it is NOT RECOMMENDED that the server allow an STag to be valid across multiple Streams.

12 Appendix B: Summary Table of Attacks

Below is a summary of implementation requirements for the RNIC:

* 7.3.1 Buffer Overrun - RDMA Write or Read Response
* 7.4.8 Controlling Access to PTT & STag Mapping
* 7.5.1 RNIC Resource Consumption
* 7.5.2.1 Multiple Streams Sharing Receive Buffers
* 7.5.2.2 Local Peer Attacking a Shared CQ
* 7.5.2.3 Remote Peer Attacking a Shared CQ
* 7.5.2.4 Attacking the RDMA Read Request Queue
* 7.5.4 Exercise of non-optimal code paths
* 7.6 Elevation of Privilege

Below is a summary of implementation requirements for the application above the RNIC:

* 7.2.4 Using an STag on a Different Stream
* 7.3.2 Modifying a Buffer After Indication
* 7.4.2 Using RDMA Read to Access Stale Data
* 7.4.3 Accessing a Buffer After the Transfer
* 7.4.4 Accessing Unintended Data With a Valid STag
* 7.4.5 RDMA Read into an RDMA Write Buffer
* 7.4.6 Using Multiple STags Which Alias to the Same Buffer
* 7.5.2.2 Local Peer Attacking a Shared CQ
* 7.5.5 Remote Invalidate an STag Shared on Multiple Streams

13 Appendix C: Partial Trust Taxonomy

Partial Trust is defined as when one party is willing to assume that another party will refrain from a specific attack or set of attacks, the parties are said to be in a state of Partial Trust. Note that the partially trusted peer may attempt a different set of attacks. This may be appropriate for many applications where any adverse effects of the betrayal is easily confined and does not place other clients or applications at risk.

The Trust Models described in this section have three primary distinguishing characteristics. The Trust Model refers to a Local Peer and Remote Peer, which are the local and remote application instances communicating via RDMA/DDP.

* Local Resource Sharing (yes/no) - When local resources are shared, they are shared across a grouping of RDMAP/DDP Streams. If local resources are not shared, the resources are dedicated on a per Stream basis. Resources are defined in Section 4.2 - Resources on page 11. The advantage of not sharing resources between Streams is that it reduces the types of attacks that are possible. The disadvantage is that applications might run out of resources.

* Local Partial Trust (yes/no) - Local Partial Trust is determined based on whether the local grouping of RDMAP/DDP Streams (which typically equates to one application or group of applications) mutually trust each other to not perform a specific set of attacks.

* Remote Partial Trust (yes/no) - The Remote Partial Trust level is determined based on whether the Local Peer of a specific RDMAP/DDP Stream partially trusts the Remote Peer of the Stream (see the definition of Partial Trust in Section 3 Introduction).

Not all of the combinations of the trust characteristics are expected to be used by applications. This paper specifically analyzes five application Trust Models that are expected to be in common use. The Trust Models are as follows:

* NS-NT - Non-Shared Local Resources, no Local Trust, no Remote Trust - typically a server application that wants to run in the safest mode possible. All attack mitigations are in place to ensure robust operation.

* NS-RT - Non-Shared Local Resources, no Local Trust, Remote Partial Trust - typically a peer-to-peer application, which has, by some method outside of the scope of this specification, authenticated the Remote Peer. Note that unless some form of key based

authentication is used on a per RDMA/DDP Stream basis, it may not be possible be possible for man-in-the-middle attacks to occur. See section 8, Security Services for RDMA and DDP on page 38.

* S-NT - Shared Local Resources, no Local Trust, no Remote Trust - typically a server application that runs in an untrusted environment where the amount of resources required is either too large or too dynamic to dedicate for each RDMAP/DDP Stream.

* S-LT - Shared Local Resources, Local Partial Trust, no Remote Trust - typically an application, which provides a session layer and uses multiple Streams, to provide additional throughput or fail-over capabilities. All of the Streams within the local application partially trust each other, but do not trust the Remote Peer. This trust model may be appropriate for embedded environments.

* S-T - Shared Local Resources, Local Partial Trust, Remote Partial Trust - typically a distributed application, such as a distributed database application or a High Performance Computer (HPC) application, which is intended to run on a cluster. Due to extreme resource and performance requirements, the application typically authenticates with all of its peers and then runs in a highly trusted environment. The application peers are all in a single application fault domain and depend on one another to be well-behaved when accessing data structures. If a trusted Remote Peer has an implementation defect that results in poor behavior, the entire application could be corrupted.

Models NS-NT and S-NT above are typical for Internet networking - neither Local Peers nor the Remote Peer is trusted. Sometimes optimizations can be done that enable sharing of Page Translation Tables across multiple Local Peers, thus Model S-LT can be advantageous. Model S-T is typically used when resource scaling across a large parallel application makes it infeasible to use any other model. Resource scaling issues can either be due to performance around scaling or because there simply are not enough resources. Model NS-RT is probably the least likely model to be used, but is presented for completeness.

14 Author's Addresses

James Pinkerton
Microsoft Corporation
One Microsoft Way
Redmond, WA. 98052 USA
Phone: +1 (425) 705-5442
Email: jpink@windows.microsoft.com

Ellen Deleganes
Intel Corporation
MS JF5-355
2111 NE 25th Ave.
Hillsboro, OR 97124 USA
Phone: +1 (503) 712-4173
Email: ellen.m.deleganes@intel.com

Sara Bitan
Microsoft Corporation
Email: sarab@microsoft.com

15 Acknowledgments

Allyn Romanow
Cisco Systems
170 W Tasman Drive
San Jose, CA 95134 USA
Phone: +1 408 525 8836
Email: allyn@cisco.com

Catherine Meadows
Naval Research Laboratory
Code 5543
Washington, DC 20375
Email: meadows@itd.nrl.navy.mil

Patricia Thaler
Agilent Technologies, Inc.
1101 Creekside Ridge Drive, #100
M/S-RG10
Roseville, CA 95678
Phone: +1-916-788-5662
email: pat_thaler@agilent.com

James Livingston
NEC Solutions (America), Inc.
7525 166th Ave. N.E., Suite D210
Redmond, WA 98052-7811
Phone: +1 (425) 897-2033
Email: james.livingston@necsam.com

John Carrier
Adaptec, Inc.
691 S. Milpitas Blvd.
Milpitas, CA 95035 USA
Phone: +1 (360) 378-8526
Email: john_carrier@adaptec.com

Caitlin Bestler
Email: cait@asomi.com

Bernard Aboba
Microsoft Corporation
One Microsoft Way
Redmond, WA. 98052 USA
Phone: +1 (425) 706-6606
Email: bernarda@windows.microsoft.com

16 Full Copyright Statement

Funding for the RFC Editor function is currently provided by the Internet Society.