Network Working Group
Request for Comments: 5042
Category: Standards Track

J. Pinkerton
Microsoft Corporation
E. Deleganes
Self

October 2007

Direct Data Placement Protocol (DDP) / Remote Direct Memory Access Protocol (RDMAP) Security

Status of This Memo

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 those that can be mitigated by using secure communication channels across the network, attacks from Remote Peers, and attacks from Local Peers. Attack categories include spoofing, tampering, information disclosure, denial of service, and elevation of privilege.

Table of Contents

1. 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 Upper Layer Protocols (ULPs) utilizing RDMA and when implementing RDMA-aware NICs (RNICs). Note that for the purposes of this security analysis, an RNIC may implement RDMAP [RDMAP] and DDP [DDP], or just DDP. Also, a ULP may be an application or it may be a middleware library.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119. Additionally, the security terminology defined in [RFC4949] is used in this specification.

The document first develops an architectural model that is relevant for the security analysis. Section 2 details components, resources, and system properties that may be attacked. The document uses Local Peer to represent the RDMA/DDP protocol implementation on the local end of a Stream (implemented with a transport protocol, such as [RFC793] or [RFC4960]). The local Upper-Layer-Protocol (ULP) is used to represent the application or middle-ware layer above the Local Peer. The document does not attempt to differentiate between a Remote Peer and a Remote ULP (an RDMA/DDP protocol implementation on the remote end of a Stream versus the application on the remote end) for several reasons: often, the source of the attack is difficult to know for sure and, regardless of the source, the mitigations required of the Local Peer or local ULP are the same. Thus, the document generically refers to a Remote Peer rather than trying to further delineate the attacker.

The document then defines what resources a local ULP may share across Streams and what resources the local ULP may share with the Remote Peer across Streams in Section 3.

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.

This document defines a new term, "Partial Mutual Trust", to address this concept:

> Partial Mutual Trust - a collection of RDMAP/DDP Streams, which represent the local and remote end points of the Stream that are willing to assume that the Streams from the collection will not perform malicious attacks against any of the other Streams in the collection.

ULPs have explicit control of which collection of endpoints is in a Partial Mutual Trust collection through tools discussed in Appendix C, Partial Trust Taxonomy.

An untrusted peer relationship is appropriate when a ULP 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 ULP 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 that 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 4.

The rest of the document is focused on analyzing attacks and recommending specific mitigations to the attacks. Attacks are categorized into attacks mitigated by end-to-end security, attacks initiated by Remote Peers, and attacks initiated by Local Peers. For each attack, possible countermeasures are reviewed.

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

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

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

The appendices provide focused summaries of this specification. Appendix A, ULP Issues for RDDP Client/Server Protocols, focuses on implementers of traditional client/server protocols. Appendix B, Summary of RNIC and ULP Implementation Requirements, summarizes all normative requirements in this specification. Appendix C, Partial Trust Taxonomy, provides an abstract model for categorizing trust boundaries.

If an RDMAP/DDP protocol implementation uses the mitigations recommended in this document, that implementation should not exhibit additional security vulnerabilities above and beyond those of an implementation of the transport protocol (i.e., TCP or SCTP) and protocols beneath it (e.g., IP) without RDMAP/DDP.

2. 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 that 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 a ULP that sits above the RNIC Interface or from the network.

The intent here is to describe high level components and capabilities that 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 from those defined here. For example, the Privileged Resource Manager may be partially or completely encapsulated in the Privileged ULP. Regardless, it is expected that the security analysis of the potential threats and countermeasures still apply.

Note that the model below is derived from several specific RDMA implementations. A few of note are [VERBS-RDMAC], [VERBS-RDMAC-Overview], and [INFINIBAND].

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

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

Figure 1 - RDMA Security Model

2.1. Components

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

* RDMA Network Interface Controller 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 ULP is implementation dependent.

* Privileged ULP - See Section 1, Introduction, for a definition of Privileged ULP. The local host infrastructure can enable the Privileged ULP to map a Data Buffer directly from the RNIC Engine to the host through the RNIC Interface, but it does not allow the Privileged ULP to directly consume RNIC Engine resources.

* Non-Privileged ULP - See Section 1, Introduction, for a definition of Non-Privileged ULP.

A design goal of the DDP and RDMAP protocols is to allow, under constrained conditions, Non-Privileged ULP 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 document is to analyze this usage model for the enforcement that is required in the RNIC Engine to ensure that the system remains secure.

DDP provides two mechanisms for transferring data:

* Untagged Data Transfer - The incoming payload simply consumes the first buffer in a queue of buffers that are in the order specified by the receiving Peer (commonly referred to as the Receive Queue), and

* Tagged Data Transfer - The Peer transmitting the payload explicitly states which destination buffer is targeted, through use of an STag. STag-based transfers allow the receiving ULP to be indifferent to what order (or in what messages) the opposite Peer sent the data, or in what order packets are received.

Both data transfer mechanisms are also enabled through RDMAP, with additional control semantics. Typically, Tagged Data Transfer can be used for payload transfer, while Untagged Data Transfer is best used for control messages. However, each Upper Layer Protocol can determine the optimal use of Tagged and Untagged messages for itself. See [APPLICABILITY] for more information on application applicability for the two transfer mechanisms.

For DDP, the two forms correspond to Untagged and Tagged DDP Messages, respectively. For RDMAP, the two forms correspond to Send Type Messages and RDMA Messages (either RDMA Read or RDMA Write Messages), respectively.

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 ULP may require (after performing appropriate countermeasures).

* ULP Control Interface - A ULP 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 ULP launches an attack, it can prevent the attack from affecting other ULPs.

* Non-Privileged Data Transfer Interface - A Non-Privileged ULP uses this interface to initiate and 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 ULP is allowed to directly manipulate RNIC Engine mapping resources to map an STag to a ULP Data Buffer.

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 document.

2.2. Resources

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

2.2.1. 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 (see Section 2.2.3) interrelate. 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).

2.2.2. Data Buffers

As mentioned previously, there are two different ways to expose a local ULP’s Data Buffers for data transfer: Untagged Data Transfer, where a buffer can be exposed for receiving RDMAP Send Type Messages (a.k.a. DDP Untagged Messages) on DDP Queue zero, or Tagged Data Transfer, where 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, for Tagged Data Transfer, a single logical Data Buffer is exposed with a single STag on a given Stream. 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 Privileged Resource Manager to ensure that no STag can be created that exposes memory that the consumer had no authority to expose.

A Data Buffer has specific access rights. The local ULP 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 would not be much point to the Advertisement). For RDMAP, when a ULP Advertises an STag, it can enable write-only, read-only, or both write and read access rights.

Similarly, some ULPs may wish to provide a single buffer with different access rights on a 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 ULP/Local Peer is allowed access.

2.2.3. Page Translation Tables

Page Translation Tables are the structures used by the RNIC to be able to access ULP 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 a ULP address space representation (e.g., a virtual address) 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.

2.2.4. Protection Domain (PD)

A Protection Domain (PD) is a local construct to the RDMA implementation, and never visible over the wire. Protection Domains are assigned to three of the resources of concern - Stream Context Memory, STags associated with Page Translation Table entries, and Data Buffers. A correct implementation of a Protection Domain requires that resources that belong to a given Protection Domain cannot be used on a resource belonging to another Protection Domain, because Protection Domain membership is checked by the RNIC prior to taking any action involving such a resource. Protection Domains are therefore used to ensure that an STag can only be used to access an associated Data Buffer on one or more Streams that are associated with the same Protection Domain as the specific STag.

If an implementation chooses not to 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 collection of Streams that have Partial Mutual Trust with each other.

Note that a ULP (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.

2.2.5. STag Namespace and Scope

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.

The scope of an STag is the set of DDP/RDMAP Streams on which the STag is valid. If an STag is valid on a particular DDP/RDMAP Stream, then that stream can modify the buffer, subject to the access rights that the stream has for the STag (see Section 2.2.2, Data Buffers, for additional information).

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.

2.2.6. Completion Queues

Completion Queues (CQ) are used in this document to conceptually represent how the RNIC Engine notifies the ULP about the completion of the transmission of data, or the completion of the reception of data through the Data Transfer Interface (specifically for Untagged Data Transfer; Tagged Data Transfer cannot cause a completion to occur). Because there could be many transmissions or receptions in flight at any one time, completions are modeled as a queue rather than as a single event. An implementation may also use the Completion Queue to notify the ULP of other activities; for example, the completion of a mapping of an STag to a specific ULP buffer. Completion Queues may be shared by a group of Streams, or may be designated to handle a specific Stream’s traffic. Limiting Completion Queue association to one, or a small number, of RDMAP/DDP Streams can prevent several forms of attacks by sharply limiting the scope of the attack’s effect.

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

2.2.7. 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 that 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 and/or ULPs can cause events to occur that 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 may be viewed simply as a very short queue.

2.2.8. 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. Some implementations may enable sharing of a single RDMA Read Request Queue across multiple Streams.

2.3. 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. As mentioned previously in Section 2.1, Components, there are two data transfer mechanisms to be examined, Untagged Data Transfer and Tagged Data Transfer.

2.3.1. Privileged Control Interface Semantics

Generically, the Privileged Control Interface controls the RNIC’s allocation, de-allocation, and initialization of RNIC global resources. This includes allocation and de-allocation 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 ULP resources for the Non-Privileged ULP (and possibly for the Privileged ULP 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).

2.3.2. 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, ULP data can be sent by one of the previously described data transfer mechanisms: Untagged Data Transfer or Tagged Data Transfer. Two RDMAP data transfer mechanisms are defined, one using Untagged Data Transfer (Send Type Messages), and one using Tagged Data Transfer (RDMA Read Responses and RDMA Writes). ULP data reception through RDMAP can be done by receiving Send Type Messages into buffers that have been posted on the Receive Queue or Shared Receive Queue. Thus, a Receive Queue or Shared Receive Queue can only be affected by Untagged Data Transfer. Data reception can also be done by receiving RDMA Write and RDMA Read Response Messages into buffers that have previously been exposed for external write access through Advertisement of an STag (i.e., Tagged Data Transfer). Additionally, to cause ULP data to be pulled (read) across the network, RDMAP uses an RDMA Read Request Message (which only contains RDMAP control information necessary to access the ULP buffer to be read), to cause an RDMA Read Response Message to be generated that contains the ULP data.

For DDP, transmitting data means sending DDP Tagged or Untagged Messages. For data reception, DDP 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 ULP of the completed work by placing completion information on the Completion Queue. For data reception, only an Untagged Data Transfer can cause completion information to be put in the Completion Queue.

2.3.3. 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.

2.3.4. 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 operations:

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). If there is no Page Translation Table, then attacks based on changing its contents or exhausting its resources are not possible.

Initialization of the contents of the Page Translation Table can be done by either the Privileged ULP or by the Privileged Resource Manager as a proxy for the Non-Privileged ULP. By definition, the Non-Privileged ULP is not trusted to directly manipulate the Page Translation Table. In general, the concern is that the Non-Privileged ULP 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 document. It may be allocated for a specific Data Buffer, or as a pooled resource to be consumed by potentially multiple Data Buffers, or be managed in some other way. This document attempts to abstract implementation dependent issues, and group them into 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 (i.e., Untagged Data Transfer), there is no wire visible mapping between an STag and the Data Buffer. Note that there may, in fact, be an STag that represents the buffer, if an implementation chooses to internally represent Untagged Data Buffer using STags. However, because the STag, by definition, is not visible on the wire, this is a local host, implementation-specific issue that should be analyzed in the context of a local host implementation-specific security analysis, and thus, is outside the scope of this document.

For a Tagged Data Buffer (i.e., Tagged Data Transfer), either the Privileged ULP or the Privileged Resource Manager acting on behalf of the Non-Privileged ULP 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 that 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 6.2.3, Multiple STags to Access the Same Buffer.

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 that have security repercussions. This includes support for separation of mapping an STag versus mapping a set of Page Translation Table entries, and support for ULPs directly manipulating STag to Page Translation Table entry mappings (versus requiring access through the Privileged Resource Manager).

2.3.5. RNIC Data Transfer Interactions

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

For send operations, there is typically a queue that enables the ULP to post multiple operation requests 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 versus Untagged Data Buffers (i.e., Tagged Data Transfer vs. Untagged Data Transfer). For Untagged Data Transfer, if more than one Untagged Data Buffer can be posted by the ULP, the DDP specification requires that they be consumed in sequential order (the RDMAP specification also requires this). 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 consumption order guarantee between Streams.

For receive Tagged Data Transfer (i.e., Tagged Data Buffers, RDMA Write Buffers, or RDMA Read 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 Section 2.3.4 for interaction details).

3. 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 2.2, Resources. The advantage of not sharing resources between Streams is that it reduces the types of attacks that are possible. The disadvantage of not sharing resources is that ULPs might run out of resources. Thus, there can be a strong incentive for sharing resources, if the security issues associated with the sharing of resources can be mitigated.

It is assumed in this document 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 ULPs request resources from the Resource Manager (note that by definition both the Non-Privileged and the Privileged application might try to greedily consume resources, thus creating a potential Denial of Service (DOS) attack). 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 document. Note that for some systems the Privileged Resource Manager may be implemented within the Privileged ULP.

All Non-Privileged ULP interactions with the RNIC Engine that could affect other ULPs MUST be done using the Privileged Resource Manager as a proxy. All ULP resource allocation requests for scarce resources MUST also be done using a Privileged Resource Manager.

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

The Privileged Resource Manager MUST NOT assume that different Streams share Partial Mutual Trust unless there is a mechanism to ensure that the Streams do indeed share Partial Mutual Trust. This can be done in several ways, including explicit notification from the ULP that owns the Streams.

4. Attacker Capabilities

An attacker’s capabilities delimit the types of attacks that the 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. This requires at least one round-trip handshake to occur.

If the attacker is not the Remote Peer that created the initial connection, then the attacker’s capabilities can be segmented into send only capabilities or send and receive capabilities. Attacking with send only capabilities requires the attacker to first guess the current LLP Stream parameters before they can attack RNIC resources (e.g., TCP sequence number). If this class of attacker also has receive capabilities and the ability to pose as the receiver to the sender and the sender to the receiver, they are typically referred to as a "man-in-the-middle" attacker [RFC3552]. A man-in-the-middle attacker has a much wider ability to attack RNIC resources. The breadth of attack is essentially the same as that of an attacking Remote Peer (i.e., the Remote Peer that set up the initial LLP Stream).

5. Attacks That Can Be Mitigated with End-to-End Security

This section describes the RDMAP/DDP attacks where the only solution is to implement some form of end-to-end security. 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.

Some forms of attack involve modifying the RDMAP or DDP payload by a network-based attacker or involve monitoring the traffic to discover private information. An effective tool to ensure confidentiality is to encrypt the data stream through mechanisms, such as IPsec encryption. Additionally, authentication protocols, such as IPsec authentication, are an effective tool to ensure the remote entity is who they claim to be, as well as ensuring that the payload is unmodified as it traverses the network.

Note that connection setup and tear down 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/tear down 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 continue to apply (and are out of scope of this document). Thus, the analysis in this section focuses on attacks that are present, regardless of the LLP Stream type.

Tampering is any modification of the legitimate traffic (machine internal or network). Spoofing attack is a special case of tampering where the attacker falsifies an identity of the Remote Peer (identity can be an IP address, machine name, ULP level identity, etc.).

5.1. 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. This section analyzes the various types of spoofing attacks applicable to RDMAP and DDP.

5.1.1. Impersonation

A network-based attacker can impersonate a legal RDMAP/DDP Peer (by spoofing a legal IP address). This can either be done as a blind attack (see [RFC3552]) or by establishing an RDMAP/DDP Stream with the victim. Because an RDMAP/DDP Stream requires an LLP Stream to be fully initialized (e.g., for [RFC793], it is in the ESTABLISHED state), existing transport layer protection mechanisms against blind attacks remain in place.

For a blind attack to succeed, it requires the attacker to inject a valid transport layer segment (e.g., for TCP, it must match at least the 4-tuple as well as guess a sequence number within the window) while also guessing valid RDMAP or DDP parameters. There are many ways to attack the RDMAP/DDP protocol if the transport protocol is assumed to be vulnerable. For example, for Tagged Messages, this entails guessing the STag and TO values. If the attacker wishes to simply terminate the connection, it can do so by correctly guessing the transport and network layer values, and providing an invalid STag. Per the DDP specification, if an invalid STag is received, the Stream is torn down and the Remote Peer is notified with an error. If an attacker wishes to overwrite an Advertised Buffer, it must successfully guess the correct STag and TO. Given that the TO will often start at zero, this is straightforward. The value of the STag should be chosen at random, as discussed in Section 6.1.1, Using an STag on a Different Stream. For Untagged Messages, if the MSN is invalid then the connection may be torn down. If it is valid, then the receive buffers can be corrupted.

End-to-end authentication (e.g., IPsec or ULP authentication) provides protection against either the blind attack or the connected attack.

5.1.2. Stream Hijacking

Stream hijacking happens when a network-based attacker eavesdrops on the LLP connection through the Stream establishment phase, and waits until the authentication phase (if such a phase exists) is completed successfully. The attacker then spoofs the IP address and re-directs the Stream from the victim to its own machine. For example, an attacker can wait until an iSCSI authentication is completed successfully, and then hijack the iSCSI Stream.

The best protection against this form of attack is end-to-end integrity protection and authentication, such as IPsec, to prevent spoofing. Another option is to provide a physically segregated network for 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.

5.1.3. Man-in-the-Middle Attack

If a network-based attacker has the ability to delete or modify packets that 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 best protection against this form of attack is end-to-end integrity protection and authentication, such as IPsec, to prevent spoofing or tampering. If authentication and integrity protections are not used, then physical protection must be employed to prevent man-in-the-middle attacks.

Because the connection/Stream itself is established by the LLP, some LLPs are more exposed to man-in-the-middle attack than 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.

5.2. Tampering - 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 5.1.3, Man-in-the-Middle Attack.

5.3. Information Disclosure - Network-Based Eavesdropping

An attacker that is able to eavesdrop on the network can read the content of all read and write accesses to a Peer’s buffers. To prevent information disclosure, the read/written data must be encrypted. See also Section 5.1.3, Man-in-the-Middle Attack. The encryption can be done either by the ULP, or by a protocol that can provide security services to RDMAP and DDP (e.g., IPsec).

5.4. Specific Requirements for Security Services

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 depends on whether the authentication is machine level authentication (such as IPsec), or ULP authentication.

5.4.1. Introduction to Security Options

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

1. Session confidentiality - Protects against eavesdropping (Section 5.3).

2. Per-packet data source authentication - Protects against the following spoofing attacks: network-based impersonation (Section 5.1.1) and Stream hijacking (Section 5.1.2).

3. Per-packet integrity - Protects against tampering done by network-based modification of buffer content (Section 5.2) and when combined with authentication, also protects against man-in-the-middle attacks (Section 5.1.3).

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).

IPsec is a protocol suite that 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. Please see those RFCs for a complete description of the respective protocols.

IPsec is 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.

5.4.2. TLS Is Inappropriate for DDP/RDMAP Security

TLS [RFC4346] provides Stream authentication, integrity and confidentiality for TCP based ULPs. TLS supports one-way (server only) or mutual certificates based authentication.

If TLS is layered underneath RDMAP, TLS’s connection orientation makes TLS inappropriate for DDP/RDMA security. 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 TLS is used to protect DDP/RDMAP traffic, then TCP must gather all out-of-order packets before TLS can decrypt them. Only after this is done can RDMAP/DDP place them into the ULP buffer. Thus, one of the primary features of DDP/RDMAP - enabling implementations to have a flow-through architecture with little to no buffering - cannot be achieved if TLS is used to protect the data stream.

If TLS is layered on top of RDMAP or DDP, TLS 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 place the data into the wrong buffer, thus effectively corrupting the data stream.

For these reasons, it is not RECOMMENDED that TLS be layered on top of RDMAP or DDP.

5.4.3. DTLS and RDDP

DTLS [DTLS] provides security services for datagram protocols, including unreliable datagram protocols. These services include anti-replay based on a mechanism adapted from IPsec that is intended to operate on packets as they are received from the network. For these and other reasons, DTLS is best applied to RDDP by employing DTLS beneath TCP, yielding a layering of RDDP over TCP over DTLS over UDP/IP. Such a layering inserts DTLS at roughly the same level in the protocol stack as IPsec, making DTLS's security services an alternative to IPsec's services from an RDDP standpoint.

For RDDP, IPsec is the better choice for a security framework, and hence is mandatory-to-implement (as specified elsewhere in this document). An important contributing factor to the specification of IPsec rather than DTLS is that the non-RDDP versions of two initial adopters of RDDP (iSCSI [iSCSI][iSER] and NFSv4 [NFSv4][NFSv4.1]) are compatible with IPsec but neither of these protocols currently uses either TLS or DTLS. For the specific case of iSCSI, IPsec is the basis for mandatory-to-implement security services [RFC3723]. Therefore, this document and the RDDP protocol specifications contain mandatory implementation requirements for IPsec rather than for DTLS.

5.4.4. ULPs That Provide Security

ULPs that provide integrated security but wish to leverage lower-layer protocol security, should be aware of security concerns around correlating a specific channel's security mechanisms to the authentication performed by the ULP. See [NFSv4CHANNEL] for additional information on a promising approach called "channel binding". From [NFSv4CHANNEL]:

> "The concept of channel bindings allows applications to prove that the end-points of two secure channels at different network layers are the same by binding authentication at one channel to the session protection at the other channel. The use of channel bindings allows applications to delegate session protection to lower layers, which may significantly improve performance for some applications."

5.4.5. 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 not to replicate this effort, an RNIC implementation MUST follow the requirements defined in RFC 3723, Section 2.3 and Section 5, including the associated normative references for those sections. Note that this means that support for IPSEC ESP mode is normative.

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 a 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.

The IPsec requirements for RDDP are based on the version of IPsec specified in RFC 2401 [RFC2401] and related RFCs, as profiled by RFC 3723 [RFC3723], despite the existence of a newer version of IPsec specified in RFC 4301 [RFC4301] and related RFCs. One of the important early applications of the RDDP protocols is their use with iSCSI [iSER]; RDDP’s IPsec requirements follow those of IPsec in order to facilitate that usage by allowing a common profile of IPsec to be used with iSCSI and the RDDP protocols. In the future, RFC 3723 may be updated to the newer version of IPsec; the IPsec security requirements of any such update should apply uniformly to iSCSI and the RDDP protocols.

6. Attacks from Remote Peers

This section describes remote attacks that are possible against the RDMA system defined in Figure 1 - RDMA Security Model and the RNIC Engine resources defined in Section 2.2. 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.

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

6.1. Spoofing

This section analyzes the various types of spoofing attacks applicable to RDMAP and DDP. Spoofing attacks can be launched by the Remote Peer or by a network-based attacker. For countermeasures against a network-based attacker, see Section 5, Attacks That Can Be Mitigated with End-to-End Security.

6.1.1. 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 an 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 either perform 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 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). For some ULPs, using an STag on multiple Streams within the same Protection Domain could be desired behavior. For other ULPs, attempting to use an STag on a different Stream could be considered an attack. Since this varies by ULP, a ULP 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 is 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 ULP 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 RDMAP Stream is 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 MUST be limited to a single Stream.

An RNIC MUST ensure that a specific Stream in a specific Protection Domain cannot access an STag in a different Protection Domain.

An RNIC MUST ensure that, if an STag is limited in scope to a single Stream, no other Stream can use the STag.

An additional issue may be unintended sharing of STags (i.e., a bug in the ULP) or a bug in the Remote Peer that 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, and also ensure that STags are reused at as slow a rate as possible. Any allocation method that would lead to intentional or unintentional reuse of an STag by the peer should be avoided (e.g., a method that always starts with a given STag and monotonically increases it for each new allocation, or a method that always uses the same STag for each operation).

6.2. 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 from a Remote Peer are outlined in the sections that follow. For countermeasures against a network-based attacker, see Section 5, Attacks That Can Be Mitigated with End-to-End Security.

6.2.1. 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 ULP has a bug.

The countermeasure for this type of attack must be in the RNIC implementation, leveraging the STag. When the local ULP specifies to the RNIC the base address and the umber 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 that 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.

6.2.2. 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 or local ULP (by a variety of means) 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 that 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 ULP 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 ULP can do this either 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 Remote Invalidate capability. If the Remote Peer did not invalidate the STag, the local ULP then explicitly revokes the STag remote access rights. (See Section 6.4.5, Remote Invalidate an STag Shared on Multiple Streams for a definition of Remote Invalidate.)

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

An RNIC MUST ensure that network packets using the STag for a previously Advertised Buffer can no longer modify the buffer after the ULP revokes remote access rights for the specific STag.

6.2.3. Multiple STags to Access the Same Buffer

See Section 6.3.6 Using Multiple STags That Alias to the Same Buffer, for this analysis.

6.3. 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.

6.3.1. Probing Memory Outside of the Buffer Bounds

This is essentially the same attack as described in Section 6.2.1, Buffer Overrun - RDMA Write or Read Response, except that an RDMA Read Request is used to mount the attack. The same countermeasure applies.

6.3.2. Using RDMA Read to Access Stale Data

If a buffer is being used for some combination of reads and writes (either remote or local), and is exposed to a Remote Peer with at least remote read access rights before it is initialized with the correct data, there is a potential race condition where the Remote Peer can view the prior contents of the buffer. This becomes a security issue if the prior contents of the buffer were not intended to be shared with the Remote Peer.

To eliminate this race condition, the local ULP 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). This ensures that the Remote Peer cannot access the buffer until the stale data has been removed.

6.3.3. Accessing a Buffer after the Transfer

If the Remote Peer has remote read access to a buffer and, by some mechanism, tells the local ULP that the transfer has been completed, but the local ULP 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 6.2.2, Modifying a Buffer after Indication. The same countermeasures apply. In addition, the local ULP SHOULD grant remote read access rights only for the amount of time needed to retrieve the data.

6.3.4. Accessing Unintended Data with a Valid STag

If the ULP 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 ULP SHOULD set the base and bounds of the buffer when the STag is initialized to expose only the data to be retrieved.

6.3.5. 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 ULP data, or data from a transfer on an unrelated Stream, the Remote Peer could retrieve the data through an RDMA Read operation. Note that an RNIC implementation is not required to support STags that have both read and write access.

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 a ULP 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. Note that this requirement is not meant to restrict the use of zero-length RDMA Reads. Zero-length RDMA Reads do not expose ULP data. Because they are intended to be used as a mechanism to ensure that all RDMA Writes have been received, and do not even require a valid STag, their use is permitted even if a buffer has only been enabled for write access.

6.3.6. Using Multiple STags That Alias to the Same Buffer

Multiple STags that 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 ULP has complete control over information disclosure. Thus, a server that 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, a ULP with multiple Remote Peers that do not share Partial Mutual Trust MUST NOT grant 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.

6.4. 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 ULP environments require communication with untrusted Remote Peers. If the Remote Peer can be authenticated or the ULP payload 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 (e.g., a 3-message transport layer handshake has occurred), 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.

6.4.1. RNIC Resource Consumption

This section covers attacks that fall into the general category of a local ULP attempting to unfairly allocate scarce (i.e., bounded) RNIC resources. The local ULP 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 due to attacks by currently active local ULPs 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 ULP 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 (with associated resources) 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 that 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 ULP (for a bounded, reasonable number of local ULPs). This analysis further assumes that the Resource Manager implements policies to ensure that untrusted local ULPs are not able to consume all 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 ULP is acting on the Remote Peer’s behalf.

6.4.2. Resource Consumption by Idle ULPs

The simplest form of a DOS attack, given a fixed amount of resources, is for the Remote Peer to create an RDMAP Stream to a Local Peer, request dedicated resources, and 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 at the Local Peer.

A general countermeasure for this style of attack is to monitor active RDMAP Streams and, if resources are getting low, to 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 6.4.1 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 ULP 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.

6.4.3. Resource Consumption by Active ULPs

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 6.4.1, RNIC Resource Consumption, so they are not included here.

6.4.3.1. Multiple Streams Sharing Receive Buffers

The Remote Peer can attempt to consume more than its fair share of receive Data Buffers (i.e., Untagged Buffers for DDP or Send Type Messages for RDMAP) 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 ULP allowed, resulting in no buffers being 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, and 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 locally 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 ULP if the number of buffers in the receive queue is less than a threshold.

If all the following conditions are true, then the Local Peer or local ULP 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 or local ULP 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 ULP buffer (i.e., the Untagged Buffer for DDP or the buffer consumed by a Send Type Message for RDMAP) as a reassembly buffer. In this case, the Remote Peer can consume a significant number of ULP buffers, but never send enough data to enable the ULP buffer to be completed to the ULP.

An effective countermeasure is to create a high-water notification that alerts the ULP 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 or local ULP can apply countermeasures to detect and prevent the attack.

6.4.3.2. Remote or Local Peer Attacking a Shared CQ

For an overview of the shared CQ attack model, see Section 7.1.

The Remote Peer can attack a shared 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 cause the local ULP to reserve a specified number of CQ entries, possibly leaving insufficient entries for other Streams that are sharing the CQ.

* If the Remote Peer, Local Peer, or local ULP (or any combination) 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 if there is a trusted intermediary, such as a Privileged Resource Manager. Unfortunately, it is often unrealistic not to 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 2.1, Components, for a definition of a 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’s receive queue(s) do not run out of receive buffers (if they do, then this is a different attack, documented in Section 6.4.3.1 Multiple Streams Sharing Receive Buffers), 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 the completions generated) or for another Stream not to 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 the Streams associated with the CQ are in the same or 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 needs to be resized, resizing the CQ is not required to succeed in all cases, 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 ULP(s) will post as many resources to the various queues as the size of the queue enables. If the local ULP(s) 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 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 that can cause a completion that will be posted on a specific Receive Queue.

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

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

If the local ULP must be able to completely fill the queues, or cannot 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 that
                can cause a completion that can ever be posted
                on a specific Receive Queue.

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

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

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

If the ULP is sharing a CQ across multiple Streams that do not share Partial Mutual Trust, then the ULP MUST implement a mechanism to ensure that the Completion Queue does 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 ULP can be trusted not to post more than MaxPostedOnEachRQ, MaxPostedOnEachSRQ, and MaxPostedOnEachSQ, then the first formula applies. If the ULP cannot be trusted to obey the limit, then the second formula applies.

6.4.3.3. Attacking the RDMA Read Request Queue

The RDMA Read Request Queue can be attacked 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 no longer functional (e.g., torn down). One approach to solving the shared RDMA Read Request Queue would be to create thresholds, similar to those described in Section 6.4.3.1, Multiple Streams Sharing Receive Buffers. A simpler approach is to not share RDMA Read Request Queue resources among Streams or to enforce hard limits of consumption per Stream. Thus, RDMA Read Request Queue resource consumption MUST be controlled by the Privileged Resource Manager such that RDMAP/DDP Streams that 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, but 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 not to 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 that the prior paragraph’s recommendation is implemented). Thus, an RNIC implementation SHOULD provide a mechanism to cap the number of outstanding RDMA Read Requests. The configuration of this limit is outside the scope of this document.

6.4.4. Exercise of Non-Optimal Code Paths

Another form of a 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 that do not cause the Stream to be torn down.

If an RNIC implementation contains "slow paths" that 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 that is exercising data paths, which consume a disproportionate amount of resources.

6.4.5. 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 remotely 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 remotely invalidate the STag prematurely), the ULP MUST NOT enable an STag that would be valid across multiple Streams.

6.4.6. Remote Peer Attacking an Unshared CQ

The Remote Peer can attack an unshared CQ if the Local Peer does not size the CQ correctly. For example, if the Local Peer enables the CQ to handle completions of received buffers, and the receive buffer queue is longer than the Completion Queue, then an overflow can potentially occur. The effect on the attacker’s Stream is catastrophic. However, if an RNIC does not have the proper protections in place, then an attack to overflow the CQ can also cause corruption and/or termination of an unrelated Stream. Thus, an RNIC MUST ensure that if a CQ overflows, any Streams that do not use the CQ MUST remain unaffected.

6.5. 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 ULP is able to elevate its privilege level to a Privileged ULP, 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 ULP 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 is thus outside the scope of this document.

7. Attacks from Local Peers

This section describes local attacks that are possible against the RDMA system defined in Figure 1 - RDMA Security Model and the RNIC Engine resources defined in Section 2.2.

7.1. Local ULP Attacking a Shared CQ

DOS attacks against a Shared Completion Queue (CQ - see Section 2.2.6, Completion Queues) can be caused by either the local ULP 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 ULP such that no Completion Queue entries are available.

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

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

For these reasons, an RNIC MUST NOT enable sharing a CQ across ULPs that do not share Partial Mutual Trust.

7.2. Local Peer Attacking the RDMA Read Request Queue

If RDMA Read Request Queue resources are pooled across multiple Streams, one attack is if the local ULP attempts to unfairly allocate RDMA Read Request Queue resources for its Streams. For example, a local ULP attempts to allocate all available resources on a specific RDMA Read Request Queue for its Streams, thereby denying the resource to ULPs sharing the RDMA Read Request Queue. The same type of argument applies even if the RDMA Read Request is not shared, but a local ULP attempts to allocate all the RNIC’s resources 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 ULP from allocating more than its fair share of resources.

7.3. Local ULP Attacking the PTT and STag Mapping

If a Non-Privileged ULP 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 ULP could point the Page Translation Table at an unrelated Stream’s or ULP’s buffers and, thereby, be able to gain access to information of the unrelated Stream/ULP.

As discussed in Section 2, Architectural Model, introduction of a Privileged Resource Manager to arbitrate the mapping requests is an effective countermeasure. This enables the Privileged Resource Manager to ensure that a local ULP can only initialize the Page Translation Table (PTT) to point to its own buffers.

Thus, if Non-Privileged ULPs are supported, the Privileged Resource Manager MUST verify that the Non-Privileged ULP has the right to access a specific Data Buffer before allowing an STag for which the ULP 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.

8. Security considerations

Please see Sections 5, Attacks That Can be Mitigated with End-to-End Security; Section 6, Attacks from Remote Peers; and Section 7, Attacks from Local Peers, for a detailed analysis of attacks and normative countermeasures to mitigate the attacks.

Additionally, the appendices provide a summary of the security requirements for specific audiences. Appendix A, ULP Issues for RDDP Client/Server Protocols, provides a summary of implementation issues and requirements for applications that implement a traditional client/server style of interaction. It provides additional insight and applicability of the normative text in Sections 5, 6, and 7. Appendix B, Summary of RNIC and ULP Implementation Requirements, provides a convenient summary of normative requirements for implementers.

9. IANA Considerations

IANA considerations are not addressed by this document. Any IANA considerations resulting from the use of DDP or RDMA must be addressed in the relevant standards.

10. References

10.1. Normative References

[DDP] Shah, H., Pinkerton, J., Recio, R., and P. Culley, "Direct Data Placement over Reliable Transports", RFC 5041, October 2007.

[RDMAP] Recio, R., Culley, P., Garcia, D., and J. Hilland, "A Remote Direct Memory Access Protocol Specification", RFC 5040, October 2007.

[RFC2401] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC2402] Kent, S. and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[RFC2406] Kent, S. and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[RFC3723] Aboba, B., Tseng, J., Walker, J., Rangan, V., and F. Travostino, "Securing Block Storage Protocols over IP", RFC 3723, April 2004.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

10.2. Informative References

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.

[APPLICABILITY] Bestler, C. and L. Coene, "Applicability of Remote Direct Memory Access Protocol (RDMA) and Direct Data Placement (DDP)", RFC 5045, October 2007.

[NFSv4CHANNEL] Williams, N., "On the Use of Channel Bindings to Secure Channels", Work in Progress, July 2004.

[VERBS-RDMAC] "RDMA Protocol Verbs Specification", RDMA Consortium standard, April 2003, <http://www.rdmaconsortium.org/home/draft-hilland-iwarp-verbs-v1.0-RDMAC.pdf>.

[VERBS-RDMAC-Overview]
"RDMA enabled NIC (RNIC) Verbs Overview", slide presentation by Renato Recio, April 2003, <http://www.rdmaconsortium.org/home/RNIC_Verbs_Overview2.pdf>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

[INFINIBAND] "InfiniBand Architecture Specification Volume 1", release 1.2, InfiniBand Trade Association standard, <http://www.infinibandta.org/specs>. Verbs are documented in chapter 11.

[DTLS] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.

[iSCSI] Satran, J., Meth, K., Sapuntzakis, C., Chadalapaka, M., and E. Zeidner, "Internet Small Computer Systems Interface (iSCSI)", RFC 3720, April 2004.

[iSER] Ko, M., Chadalapaka, M., Hufferd, J., Elzur, U., Shah, H., and P. Thaler, "Internet Small Computer System Interface (iSCSI) Extensions for Remote Direct Memory Access (RDMA)", RFC 5046, October 2007.

[NFSv4] Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and D. Noveck, "Network File System (NFS) version 4 Protocol", RFC 3530, April 2003.

[NFSv4.1] Shepler, S., Ed., Eisler, M., Ed., and D. Noveck, Ed., "NFSv4 Minor Version 1", Work in Progress, September 2007.

Appendix A: ULP Issues for RDDP Client/Server Protocols

This section is a normative appendix to the document that is focused on client/server ULP implementation requirements to ensure a secure server implementation.

The prior sections outlined specific attacks and their countermeasures. This section summarizes the attacks and countermeasures that have been defined in the prior section, which are applicable to creation of a secure ULP (e.g., application) server. A ULP server is defined as a ULP that must be able to communicate with many clients that do not necessarily have a trust relationship with each other, and to ensure that each client cannot 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. Note that this section assumes a compliant RNIC and Privileged Resource Manager implementation - thus, it focuses specifically on ULP server (e.g., application) implementation issues.

All of the prior section’s details on attacks and countermeasures apply to the server; thus, requirements that are repeated in this section use non-normative "must", "should", and "may". In some cases, normative SHOULD statements for the ULP from the main body of this document are made MUST statements for the ULP server because the operating conditions can be refined to make the motives for a SHOULD inapplicable. If a prior SHOULD is changed to a MUST in this section, it is explicitly noted and it uses uppercase normative statements.

The following list summarizes the relevant attacks that clients can mount on the shared server by re-stating the previous normative statements to be client/server specific. Note that each client/server ULP may employ explicit RDMA Operations (RDMA Read, RDMA Write) in differing fashions. Therefore, where appropriate, "Local ULP", "Local Peer", and "Remote Peer" are used in place of "server" or "client", in order to retain full generality of each requirement.

* Spoofing

  * Sections 5.1.1 to 5.1.3. For protection against many forms of spoofing attacks, enable IPsec.

  * Section 6.1.1, Using an STag on a Different Stream. To ensure that one client cannot access another client’s data via use of the other client’s STag, the server ULP must either scope an STag to a single Stream or use a unique 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 among the associated Streams (see Section 5.4.4, ULPs That Provide Security, for additional issues). To prevent unintended sharing of STags within the associated Streams, a server ULP should use STags in such a fashion that it is difficult to predict the next allocated STag number.

* Tampering

  * 6.2.2 Modifying a Buffer after Indication. Before the local ULP operates on a buffer that was written by the Remote Peer using an RDMA Write or RDMA Read, the local ULP MUST ensure the buffer can no longer be modified by invalidating the STag for remote access (note that this is stronger than the SHOULD in Section 6.2.2). This can be done either 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. If the Remote Peer did not invalidate the STag, the local ULP then explicitly revokes the STag remote access rights.

* Information Disclosure

  * 6.3.2, Using RDMA Read to Access Stale Data. In a general purpose server environment, there is no compelling rationale not to require a buffer to be initialized before remote read is enabled (and an enormous downside of unintentionally sharing data). Thus, a local ULP MUST (this is stronger than the SHOULD in Section 6.3.2) ensure that no stale data is contained in a buffer before remote read access rights are granted to a Remote Peer (this can be done by zeroing the contents of the memory, for example).

  * 6.3.3, Accessing a Buffer after the Transfer. This mitigation is already covered by Section 6.2.2 (above).

  * 6.3.4, Accessing Unintended Data with a Valid STag. The ULP must set the base and bounds of the buffer when the STag is initialized to expose only the data to be retrieved.

  * 6.3.5, RDMA Read into an RDMA Write Buffer. If a peer 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.

    * 6.3.6, Using Multiple STags That Alias to the Same Buffer. The requirement in Section 6.1.1 (above) mitigates this attack. A server buffer is exposed to only one client at a time to ensure that no information disclosure or information tampering occurs between peers.

    * 5.3, Network-Based Eavesdropping. Confidentiality services should be enabled by the ULP if this threat is a concern.

* Denial of Service

    * 6.4.3.1, Multiple Streams Sharing Receive Buffers. ULP memory footprint size can be important for some server ULPs. If a server ULP is expecting significant network traffic from multiple clients, using a receive buffer queue per Stream where there is a large number of Streams can consume substantial amounts of memory. Thus, a receive queue that can be shared by multiple Streams is attractive.

      However, because of the attacks outlined in this section, sharing a single receive queue between multiple clients must only be done if a mechanism is in place to ensure that one client cannot consume receive buffers in excess of its limits, as defined by each ULP. For multiple Streams within a single client ULP (which presumably shared Partial Mutual Trust), this added overhead may be avoided.

    * 7.1 Local ULP Attacking a Shared CQ. The normative RNIC mitigations require that the RNIC not enable sharing of a CQ if the local ULPs do not share Partial Mutual Trust. Thus, while the ULP is not allowed to enable this feature in an unsafe mode, if the two local ULPs share Partial Mutual Trust, they must behave in the following manner:

      1) The sizing of the completion queue is based on the size of the receive queue and send queues, as documented in 6.4.3.2, Remote or Local Peer Attacking a Shared CQ.

      2) The local ULP ensures that CQ entries are reaped frequently enough to adhere to Section 6.4.3.2’s rules.

    * 6.4.3.2, Remote or Local Peer Attacking a Shared CQ. There are two mitigations specified in this section - one requires a worst-case size of the CQ, and can be implemented entirely within the Privileged Resource Manager. The second approach requires cooperation with the local ULP server (not to post too many buffers), and enables a smaller CQ to be used.

In some server environments, partial trust of the server ULP (but not the clients) is acceptable; thus, the smaller CQ fully mitigates the remote attacker. In other environments, the local server ULP could also contain untrusted elements that can attack the local machine (or have bugs). In those environments, the worst-case size of the CQ must be used.

* 6.4.3.3, Attacking the RDMA Read Request Queue. The section requires a server's Privileged Resource Manager not to allow sharing of RDMA Read Request Queues across multiple Streams that do not share Partial Mutual Trust for a ULP that performs RDMA Read operations to server buffers. However, because the server ULP knows which of its Streams best share Partial Mutual Trust, this requirement can be reflected back to the ULP. The ULP (i.e., server) requirement, in this case, is that it MUST NOT allow RDMA Read Request Queues to be shared between ULPs that do not have Partial Mutual Trust.

* 6.4.5, Remote Invalidate an STag Shared on Multiple Streams. This mitigation is already covered by Section 6.2.2 (above).

Appendix B: Summary of RNIC and ULP Implementation Requirements

This appendix is informative.

Below is a summary of implementation requirements for the RNIC:

* 3 Trust and Resource Sharing
* 5.4.5 Requirements for IPsec Encapsulation of DDP
* 6.1.1 Using an STag on a Different Stream
* 6.2.1 Buffer Overrun - RDMA Write or Read Response
* 6.2.2 Modifying a Buffer after Indication
* 6.4.1 RNIC Resource Consumption
* 6.4.3.1 Multiple Streams Sharing Receive Buffers
* 6.4.3.2 Remote or Local Peer Attacking a Shared CQ
* 6.4.3.3 Attacking the RDMA Read Request Queue
* 6.4.6 Remote Peer Attacking an Unshared CQ
* 6.5 Elevation of Privilege 39

* 7.1 Local ULP Attacking a Shared CQ

* 7.3 Local ULP Attacking the PTT and STag Mapping

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

* 5.3 Information Disclosure - Network-Based Eavesdropping

* 6.1.1 Using an STag on a Different Stream

* 6.2.2 Modifying a Buffer after Indication

* 6.3.2 Using RDMA Read to Access Stale Data

* 6.3.3 Accessing a Buffer after the Transfer

* 6.3.4 Accessing Unintended Data with a Valid STag

* 6.3.5 RDMA Read into an RDMA Write Buffer

* 6.3.6 Using Multiple STags That Alias to the Same Buffer

* 6.4.5 Remote Invalidate an STag Shared on Multiple Streams

Appendix C: Partial Trust Taxonomy

This appendix is informative.

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 ULPs where any adverse effects of the betrayal is easily confined and does not place other clients or ULPs at risk.

The Trust Models described in this section have three primary distinguishing characteristics. The Trust Model refers to a local ULP and Remote Peer, which are intended to be the local and remote ULP 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 2.2, Resources. The advantage of not sharing resources between Streams is that it reduces the types of attacks that are possible. The disadvantage is that ULPs 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 ULP or group of ULPs) mutually trust each other not to perform a specific set of attacks.

* Remote Partial Trust (yes/no) - The Remote Partial Trust level is determined based on whether the local ULP of a specific RDMAP/DDP Stream partially trusts the Remote Peer of the Stream (see the definition of Partial Trust in Section 1, Introduction).

Not all the combinations of the trust characteristics are expected to be used by ULPs. This document specifically analyzes five ULP 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 ULP 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 ULP that has, by some method outside of the scope of this document, 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 for man-in-the-middle attacks to occur.

* S-NT - Shared Local Resources, no Local Trust, no Remote Trust; typically, a server ULP 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, a ULP that provides a session layer and uses multiple Streams, to provides additional throughput or fail-over capabilities. All the Streams within the local ULP 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 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 the local ULP nor the Remote Peer is trusted. Sometimes, optimizations can be done that enable sharing of Page Translation Tables across multiple local ULPs; thus, Model S-LT can be advantageous. Model S-T is typically used when resource scaling across a large parallel ULP 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.

Acknowledgments

Sara Bitan
Microsoft Corporation
EMail: sarab@microsoft.com

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 USA
EMail: meadows@itd.nrl.navy.mil

Patricia Thaler
Agilent Technologies, Inc.
1101 Creekside Ridge Drive, #100
M/S-RG10
Roseville, CA 95678 USA
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 USA
Phone: +1 (425) 897-2033
EMail: james.livingston@necsam.com

John Carrier
Cray Inc.
411 First Avenue S, Suite 600
Seattle, WA 98104-2860
Phone: 206-701-2090
EMail: carrier@cray.com

Caitlin Bestler
Broadcom
49 Discovery
Irvine, CA 92618
EMail: cait@asomi.com

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

Authors' Addresses

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

Ellen Deleganes
Self
P.O. Box 9245
Brooks, OR 97305
Phone: (503) 642-3950
EMail: deleganes@yahoo.com

Full Copyright Statement

Intellectual Property