Securing an InfiniBand Network and its Effect on Performance

Lucas Mireles, Scott Graham, Patrick Sweeney, Stephen Dunlap, Matthew Dallmeyer

Chapter 1

SECURING AN INFINIBAND NETWORK AND ITS EFFECT ON PERFORMANCE

Lucas Mireles, Scott Graham, Patrick Sweeney, Stephen Dunlap, Matthew Dallmeyer

Abstract The InfiniBand Architecture (IBA) is one of the leading interconnects used in High Performance Computing (HPC) today delivering very high bandwidth and low latency. As the popularity of InfiniBand increases, the possibility for new InfiniBand applications arise outside the realm of HPC, including the support of critical infrastructure, creating the opportunity for new security risks. In this work, new security questions are addressed that have formerly gone unanswered. This study demonstrates that common traffic analyzing tools cannot monitor or capture InfiniBand traffic transmitted between two hosts. Due to the kernel bypass nature of InfiniBand, many host-based network security systems cannot be executed on InfiniBand applications. Those that can, unfortunately, impose significant performance penalties for the network. This research concludes that network security practices used for Ethernet do not translate to InfiniBand as previously suggested and that the answer to securing an InfiniBand network might reside in hardware offload.

Keywords: InfiniBand Architecture Network Cyber Security IPsec

1. Introduction

The InfiniBand Architecture (IBA) is a powerful interconnect architecture that is quickly becoming the standard for I/O connectivity in servers and High Performance Computing (HPC) clusters. In fact, 28% of the Top 500 Supercomputers use InfiniBand as their interconnect which accounts for over 35% of the total performance, second only to Gigabit Ethernet seen in Table 1 [13]. This is largely due to its ability to provide higher bandwidth and lower memory latency than its Ethernet competitor. To reduce CPU utilization, Infiniband follows a copy-avoidance architecture. As the popularity of InfiniBand increases, it is expected that InfiniBand will be deployed in many applications beyond HPC clusters as the demand for high bandwidth and low latency continue to grow in all areas of computer communication [5].

Table 1. Top 5 Supercomputer Interconnects.

Interconnect:	**Count**	**Share (%)**
Gigabit Ethernet	259	51.8
InfiniBand	140	28
Omnipath	50	10
Custom Interconnect	45	9
Proprietary Network	5	1

Computer communication is essential in an information-based society. It is the backbone for the operations and services provided by all critical infrastructure sectors[14]. As technology continues to evolve, it is possible that InfiniBand will become the interconnect standard that provides the services and products that enable communication between all sectors in the US's critical infrastructure. It is therefore essential to explore and evaluate the security of an InfiniBand network in order to address the risk of potential cyber attacks. Though some research has been conducted on the security of InfiniBand [11, 4, 3, 15, 12], the experiments focused heavily on the protocol itself and did not consider other applications. This paper addresses some additional security questions not answered by previous works. In particular, the study will evaluate whether or not InfiniBand traffic can be monitored by common traffic analyzers used on Ethernet. It will also determine the effectiveness of network security systems on InfiniBand programs and network performance. An analysis of these case studies will help guide future research in securing an InfiniBand network.

2. Background

2.1 The InfiniBand Architecture

The IBA network protocol architecture is becoming the de facto standard for server I/O and server to server communications for large HPC clusters and Storage Area Networks. While comparable to Ethernet in some ways, the IBA was designed for data centers with HPC clusters and logically separated from the Internet [11]. The design and development of IBA was driven by the inability of existing protocols to provide sufficient network bandwidth and reduced memory latency to keep up with processing performance. IBA was able to improve I/O bandwidth by employing the following two characteristics: point-to-point connections (not bused) and channel semantics as messages [10]. In contrast to a bus architecture, the point to point connections allow for scaling of large switched networks along with fault isolation. Additionally, IBA communicates data and commands via messages instead of memory operations. To achieve this, the IBA has moved away from the traditional network topology and implements point to point switched I/O fabric that uses cascading switches as shown in Figure 1. This allows InfiniBand to explicitly treat I/O as a form of communication giving I/O units the same communications capabilities as any processor node [10, **?**].

2.1.1 Infiniband Components. From a high level perspective, IBA is an interconnect for processors, I/O units, and routers which can all be considered endnodes. The smallest complete IBA network is an IBA subnet which is comprised of endnodes, switches, links, and a subnet manager[10]. IBA subnets can be connected to other IBA subnets using a router. Furthermore, endnodes that are part of a subnet can be connected to multiple switches forming a switched fabric network.

Channel Adapters. Every end node that is a part of an IBA network must have a Channel Adapter (CA) which generates and consumes IBA packets[2], and is the focus of this study. A CA is defined as either an Host Channel Adapter (HCA) or as a Target Channel Adapter (TCA). The HCA provides a collection of features specified by IBA verbs whereas the TCA does not have a defined software interface. A CA is essentially a programmable Direct Memory Access (DMA) engine that can provide both local and remote DMA that crafts packets in hardware. All CAs communicate using Work Queues which consist of Send, Receive, and Completion Queues (discussed in further detail in the IBA Communication Model section). Each HCA is assigned a Globally Unique IDentifier (GUID) by the manufacture of the chip. Additionally, each

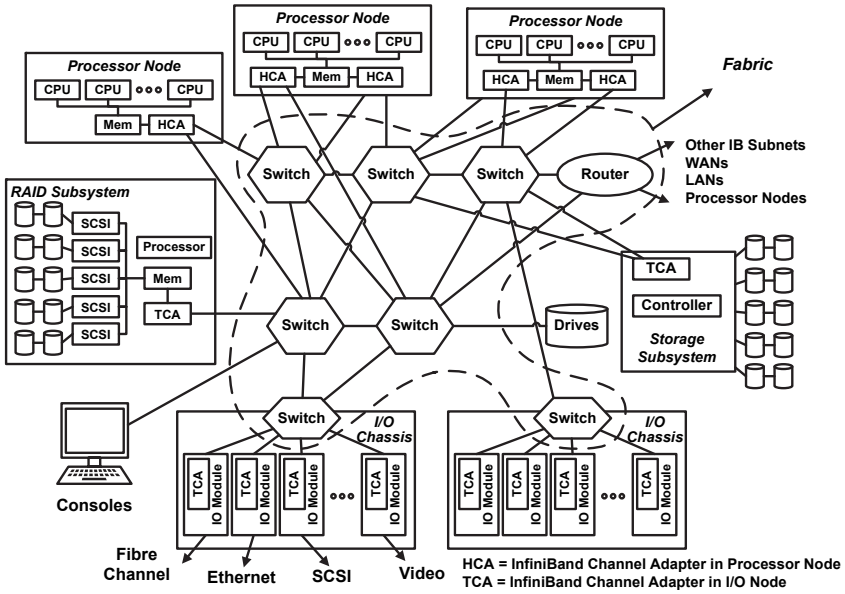

Figure 1. IBA network with Fabric Highlighted, adapted from [2].

of its ports is assigned a port GUID that identifies each port globally (within a subnet and between subnets).

Subnet Manager. InfiniBand's implementation of routing and forwarding are similar to the concept of Software Defined Networking (SDN) [7]. The routing and forwarding tables for IBA switches and routers are not decided on each device. Instead, the Subnet Manager (SM) is responsible for configuring and managing all switches, routers, and channel adapters that are part of a subnet [2, 7]. The SM actively communicates with each switch, CA, and router's Subnet Manager Agent to ensure all routing and forwarding tables are correct[11]. The IBA is designed to allow more than one SM on a subnet at a time for resiliency, i.e., there is only one active SM with the remaining SMs in a standby status. During subnet initialization, a polling algorithm is conducted using a state machine that allows all SMs to agree upon a single master SM based on highest priority.

Switch. Similar to Ethernet, where forwarding decisions are based upon MAC addresses, IBA switches make forwarding decisions based

upon Local Identifiers (LIDs). Every destination port within a subnet is assigned a LID by the SM. Destination LIDs represent a path by which a switch will forward the packet through. Every switch is configured with forwarding tables that include these paths for every LID within a subnet which is managed by the SM. Multiple paths to destinations may exist for redundancy and load sharing. It is important that the SM is configured properly to handle multiple paths when link failures occur or load sharing is desired. IBA supports both unicast (one to one) and multicast (one to many) functions allowing for Internet Protocol (IP) applications to function normally over InfiniBand fabric.

2.1.2 Software Architecture. To maintain independence of the host Operating System (OS) and processor, the InfiniBand Trade Association (IBTA) has produced a software architecture that is compatible with all major OSs [6]. InfiniBand's software architecture is comprised of kernel modules and protocols that exist solely in kernel space. Applications that function in user space need not be aware of the underlying IBA, allowing them to operate using InfiniBand just as they would Ethernet [6, 2]. A visual representation of the IBA software stack can be shown in Figure 2. InfiniBand's kernel space can be divided into three major layers: upper layer protocols, mid-layer core, and Host Channel Adapter (HCA) drivers [6].

The HCA driver's role in the IBA is no different than any other I/O device driver. The I/O drivers allow applications executing in user space to control the hardware by calling a set of character strings that identify the I/O protocol that the driver supports. These calls are then interpreted by the device driver and mapped to the specific device operation that is being called upon by the application [1]. Per the IBA specification, each HCA driver has its own specific driver that must be compatible with the mid-layer core kernel modules[2].

The kernel core modules located in the IBA's mid-layer serve many functions that allow access to multiple HCAs and provide a common set of shared services. Some of the most notable functions found in the mid-layer include management datagram interface (MAD), connection manager (CM) interface, and access to InfiniBand verbs. Infiniband verbs are an abstract description of operations that take place between the HCA and host[2]. The mid-layer core provides an interface to these functions for user application via InfiniBand's VPI Verbs API. This API enables users to directly craft packets in hardware using the functions/methods offered, bypassing the kernel completely, thus enabling the high bandwidth and low latency attributes associated with InfiniBand. Additionally, the mid-layer implements the necessary mechanisms that

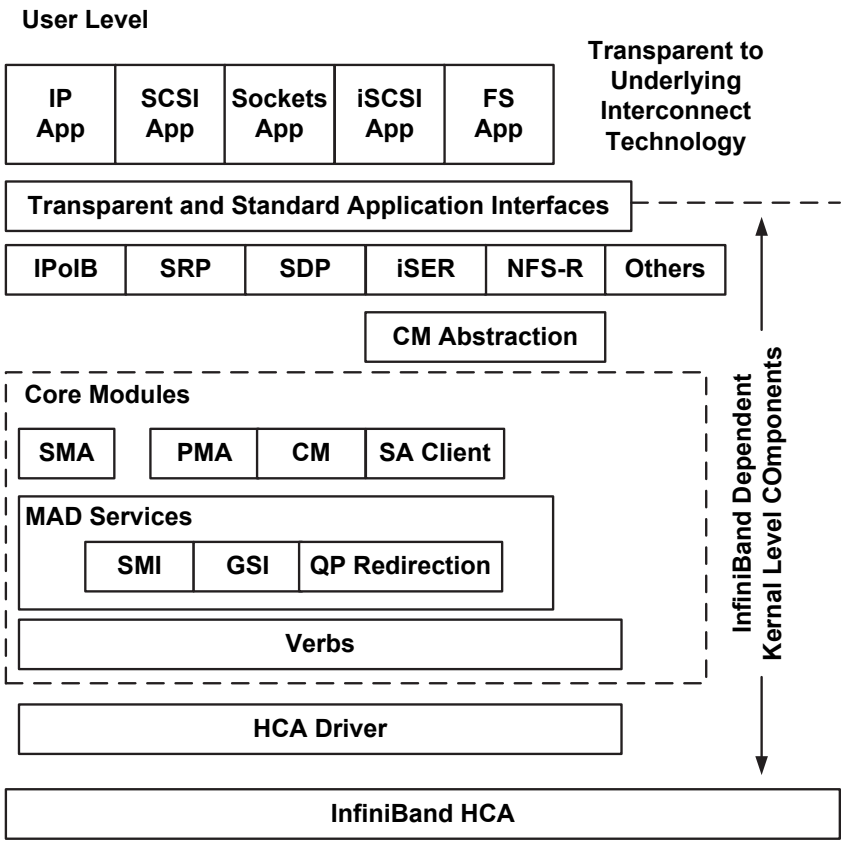

Figure 2. IBA Software Stack [6].

allows user applications to interact and have access to InfiniBand hardware[6].

The last layer of the kernel space to discuss is upper layer protocols. Upper layer protocols enable existing applications that employ standard data networking and file system access to operate over the IBA. Requiring no change to the applications, upper layer protocols allow the applications to benefit from the high bandwidth, low latency characteristics guaranteed by the IBA. Although there are many, this study will focus primarily on two upper layer protocols and how they compare to Remote Direct Memory Access (RDMA) operations.

2.1.3 IPoIB, RoCE, and RDMA. IP over InfiniBand (IPoIB) is an upper layer protocol that implements a network interface over the IBA. IPoIB encapsulates IP datagrams over an InfiniBand transport service [6]. This allows any application or kernel module that uses a standard Linux network interface to use IBA without modification. Applications running IPoIB will still traverse the TCP/IP call stack within the kernel.

One of the key capabilities provided by IBA is RDMA, which enables data to be transferred between two servers or between a server and storage without any involvement of the host processor. In traditional networks, applications request resources from the processor which in turn fulfills the request for the application. This requires significant processor overhead and leads to a large CPU utilization every time a request is made. With RDMA, the processor is only used to initialize the communication channel which allows the applications to directly communicate and share resources without processor involvement. RDMA devices allow applications to directly write and read to virtual memory. This provides low latency through stack bypass and copy avoidance, reduces CPU utilization, and provides high bandwidth utilization [9]. The combination of the IBA link layer and IBA software stack comprise the RDMA messaging service over InfiniBand.

In addition to the InfiniBand protocol, RDMA can be supported over Ethernet. This usage is referred to as RDMA over Converged Ethernet (RoCE). RoCE provides true RDMA semantics over Ethernet [9]. It is the most efficient low latency Ethernet solution today requiring far less CPU overhead than other RDMA solutions such as iWARP. Like the RDMA over InfiniBand, RoCE as well uses the InfiniBand Verbs to craft packets for its applications.

2.1.4 Communication Model. When an end user wants to communicate with another node on the network or queue up a series of requests that need to be completed by hardware, a work queue is created. Work queues are typically created in pairs and are used to hold the service requests that are made by consumers. These pairs are referred to as Queue Pairs (QP) and consist of a send queue and a receive queue. A send queue is used for send operations that hold data informing what information needs to be sent and from where. The receive queue is used for receive operations that inform the hardware where to place the data it is receiving from another consumer. After the HCA has executed the QP, a Completion Queue event is created that holds the information about the completion of a work queue that is eventually sent back to the host. QPs can be seen as the virtual interface that the consumer uses

to communicate with the hardware. IBA supports up to 2^{24} QPs per HCA. Each QP is independent of one another, providing isolation and protection from other QP operations being performed.

2.1.5 Current InfiniBand Security Features. InfiniBand can be viewed as a layer 2 protocol much like Ethernet. Thus, layer 3-7 application security mechanisms built on top of Ethernet will be implemented the same way with IBA [8]. Because of this, it is the developer's responsibility to implement application encryption, authentication, integrity, and authorization. This section will address IBA's claim to overcome known Ethernet vulnerabilities as well as advanced enforcement mechanisms that are implemented by IBA that claim to secure physical devices and resources. One such enforcement mechanism is the use of partitioning which provides private access to private devices, and allows access to shared resources [10]. To prevent unauthorized access to shared resources, a hardware mechanism called Partition Keys (P_Keys) are used. P_Keys enforce membership to a partition by requiring all QPs to be configured to the same partition in order to communicate. This requirement ensures that a P_Key is carried in every data packet guaranteeing no unauthorized access to shared resources [2]. Furthermore, partitions are controlled centrally from the SM preventing nodes from determining their own partitions. This further eliminates potential hacking and security holes because it entirely eliminates the ability for the host to manipulate what shared resources it can have access to [8].

Another feature that IBA claims to eliminate is the ability for an attacker to access unauthorized destinations, sniff unintended traffic, and impersonate other entities [8]. IBA is a switched fabric that does not allow traffic to arrive at an unwanted node. The SM implements specific switching tables that are strictly defined at every node and can only be updated by the SM. Because the switching tables are determined by the SM at a central location, the host cannot manipulate its own switching table which prevents traffic from arriving at unintended destinations. Furthermore, IBA's two transport services, reliable and unreliable, have security mechanisms that mitigate session hijacking and unauthorized access [8]. For unreliable communication, QPs are created to send and receive traffic. Within these QPs, a Queue Pair Key (Q_Key) is sent with the packet. Upon arrival at a destination, if the Q_Key that is sent with the packet does not match the Q_Key that the receiver has, then the packet is dropped. Likewise, reliable communication services use Q_Keys as well as sequences numbers and CRCs to ensure message security. If any of these mechanisms are wrong in a packet, it is reported to the fabric manager and all packets are dropped [8].

The last mechanism for security that IBA implements is memory protection. Because IBA uses RDMA which can raise security issues as nodes directly access another nodes virtual memory, it must implement a mechanism to limit the memory region nodes have access to. The IBA accomplishes this by issuing an L_Key and an R_Key with every RDMA communication. The L_Key defines the local region of memory that the specific QP has access to. The R_Key is passed to a remote node. When the remote node wants to execute an RDMA operation, it passes the R_Key that it was given to validate the remote node's right to access the destination's memory. This security mechanism cannot be disabled and changed and ensures memory protection on all IBA devices. [2] [8].

2.2 Related Work in IBA Security

Research into the security of IBA began shortly after the formation of the InfiniBand Trade Association (IBTA) in 1999.

Early studies discovered significant vulnerabilities within the IBA and presented possible solutions to mitigate them.

Additional research evaluated the implementation of an InfiniBand network rather than the architecture itself, presenting potential vulnerabilities that were not found in previous studies.

2.2.1 Insights into IBA vulnerabilities. The authors of [4] and [3] identified IBA security gaps and suggested that with moderate effort, the associated vulnerabilities could be exploited. The work found that there were two major vulnerabilities present in regards to authentication. First, IBA's use of partitioning keys to prevent illegal traffic on a network does not fully mitigate the risk of an attack. All partitioning keys used in the network are sent plain-text over the network, allowing easy access to the attacker. The solution for this risk is two key management/distribution methods: Partition level and QP level. The partition level key management scheme ensures that all forms of communication inside of a partition are done using the same shared secret key. Because the QP is the smallest communication entity, the QP level key management scheme attempts to guarantee integrity and confidentiality within a partition by implementing a form of temporary session keys between QPs. Second, the research provided another method for authentication using the Invariant CRC (ICRC). The ICRC is normally used as an end-to-end error detection method, however, the research proposes using it as an authentication tag to further harden IBA's security for two reasons. First is that the ICRC does not change from end to end, and second is that it does not require changing the IBA packet format. This study concluded that implementation of these two authentication

methods to mitigate security vulnerabilities strengthened IBA's security without hindering the performance of the network.

2.2.2 IBA GUID Spoofing. Ethernet MAC spoofing is trivially accomplished and allows for simple attacks which have been used for many years. Similar to an Ethernet MAC address, IBA uses a GUID to uniquely specify an HCA. In order to solve the MAC spoofing issue [8], IBA packets are crafted in hardware, with the GUID residing in firmware and only changeable via reprogramming the HCA (by flashing the firmware). However, [15] successfully exploited an InfiniBand network through GUID spoofing. After detailing the attack, the author also suggested a GUID spoofing mitigation approach, which relies on a monitoring system to capture an initial link state configuration. After system startup, the monitoring system sends alerts to an administrator whenever link state changes occur and LID-GUID matches change because these two changes are necessary for the attack to be successful.

2.2.3 Security Analysis of InfiniBand Protocol Implementation. In [12], one of the newest studies done on the security of IBA, the research sought to determine new potential vulnerabilities of the protocol's implementation which they claim is still missing in literature. The research performed a static code analysis as well as a dynamic analysis of the protocol's implementation. The static code analysis employed multiple tools listed in the study that inspected all lines of code that defined IBA and identified potentially vulnerable functions. The dynamic analysis was performed via "fuzz" testing in which inputs are carefully crafted and the output response is monitored for known vulnerabilities. The study concluded that there was no significant security vulnerabilities in the protocol itself, however, three functions were potentially vulnerable that they recommend be replaced.

2.2.4 A Framework for Cyber Vulnerability Assessments of Infiniband Networks. A cyber vulnerability assessment was conducted on the IBA network to determine the possible cyber vulnerabilities that may be present for IBA in [11] and concluded that some cybersecurity aspects of InfiniBand have yet to be thoroughly investigated. The InfiniBand Architecture was designed as a data center technology, logically separated from the Internet, rendering defensive mechanisms such as packet encryption unnecessary. To date, nefarious actors do not appear to have taken a significant interest in InfiniBand, but that is likely to change as as the technology proliferates. This paper considers the security implications of InfiniBand features and proposes key

elements that would be useful in a technical Cyber Vulnerability Assessment[11]. The results from the Cyber Vulnerability Assessment suggest a few potential tools and mitigation techniques that could be adapted by the IBA to include hardware and software cyber tools. The most interesting of the proposed solutions was the idea of moving towards a SDN approach in fabric management. As mentioned previously in [15], a proposed method of preventing GUID spoofing from occurring in an IBA network would be to implement a monitoring system. A way into which this might be implemented might be to use an SDN approach as proposed by [11]. The Cyber Vulnerability Assessment concluded that although cyber security was not a high priority when developing the IBA, it is inherently resistant to many cyber attacks.

3. Case Study Setups

The goals of this research are to explore how to secure an InfiniBand Network and to analyze the potential effects that a security implementation might have on the network/architecture. In particular, this study will examine the difficulties of implementing well-known network security systems on multiple types of InfiniBand applications that demonstrate its security limitations. Additionally, effects on network bandwidth will be examined to determine the performance implications of securing InfiniBand and how alternate methods may have to be used to achieve the desired speeds associated with this architecture.

3.1 Testbed Setup

The IBA allows for Ethernet and InfiniBand protocols to coexist on the same network and device without changing the application software. This is largely possible due to the InfiniBand Verb construct that allows the application to communicate directly with the hardware that crafts the traffic. Because of this, this research will include studies that use both the InfiniBand and Ethernet interconnect protocols. Mellanox was chosen as the primary hardware supplier as they are the largest producer of InfiniBand technology solutions and services, some of which include Ethernet support. The main network configuration that will be used in this research is found below:

3.1.1 Ethernet 10GbE with Connect-X 5 Adapter. The network configuration used in this study is shown in Figure 3. This configuration is comprised of two host machines, each with a Connect-X 5 adapter.

The two Connect-X 5s will be connected "back-to-back" via a 10GbE Active Optical Cable (AOC). For this example program, no switch is required, and the Ethernet protocol will be used for interconnect traffic.

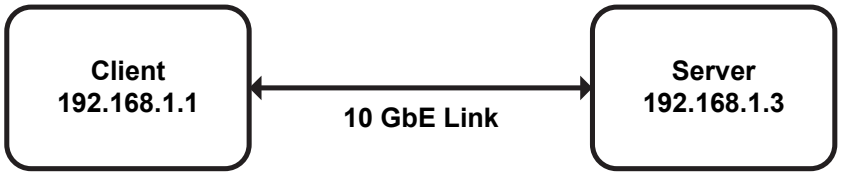

Figure 3. Network Diagram of Ethernet 10GbE with Connect-X 5 Adapter.

3.1.2 IPsec Configuration. The following configuration is used for all tests that implement IPsec:

- Encryption algorithm: AES-GCM 128/256-bit key, and 128-bit ICV
- IPsec operation mode: Transport mode
- IPsec protocol: ESP
- IP version: IPv4

IProute2 is a user application that controls TCP/IP network flows which is necessary to implement the above IPsec configuration. IProute2 will be used for this entire study.

3.1.3 Kernel Bypass and Network Security System Implementation. Kernel bypass is a key feature that allows for low latency and high bandwidth communication over InfiniBand networks. A program written with InfiniBand Verbs will bypass the host machine's kernel, irrespective of the use of RDMA or raw Ethernet packets. A custom kernel bypass program was written to demonstrate the potential effects that a security implementation might have on an InfiniBand network. A typical application written with InfiniBand verbs implements the following procedure:

1 Get the InfiniBand device list.

2 Open the requested device.

3 Query the device capabilities;

4 Allocate a Protection Domain to contain your resources.

5 Register a memory region.

6 Create a Completion Queue.

7 Create a Queue Pair.

8 Bring up a Queue Pair.

9 Post work requests and poll for completion.

10 Cleanup.

For this study, the program demonstrates kernel bypass via a raw Ethernet Client/Server model. In this model the client will send prebuilt TCP/IP packets to a server. The server will receive any packets destined to its MAC address. Although this program is sending raw Ethernet packets, this simply describes the type of QP that will be established between the two devices and will still abide by the IBA specification as it is written with InfiniBand verbs.

This program will be used in two of the three following case studies. Case Study 1 demonstrates the ability to monitor traffic which bypasses the kernel on a host machine. Case Study 2 explores the implications of bypassing the kernel with network security systems in place, specifically IPsec. Case Study 3 examines the performance impact of executing IPsec on a program that does not bypass the kernel.

3.2 Case Study 1: Traffic Monitoring

The first case study illustrates the ability of an InfiniBand application to bypass a kernel by executing the previously mentioned Client/Server program. Wireshark and Tcpdump are used to explore the possibility of monitoring InfiniBand traffic with common network monitoring tools. Both applications are packet analyzers that use the library *libpcap* to sniff packets entering a host machine. Libpcap is an API that allows applications to capture and analyze link layer packets traversing the kernel. Because its implementation occurs in the kernel, and the Raw Ethernet Client/Server program is written to illustrate bypassing the kernel, another tool will be used to capture traffic that does bypass the kernel: Mellanox's Offloaded Traffic Sniffer. The Offloaded Traffic Sniffer is defined in the MLNX_OFED and uses the standard capabilities of the utility Ethtool to capture packets in hardware. These packets can then be analyzed in packet analyzer programs such as tcpdump. The confirmation of a kernel bypass occurs if packets sent with the program can only be captured with the Offloaded Traffic Sniffer.

This case study uses the Ethernet 10GbE with Connect-X 5 Adapter network configuration and will be conducted with two tests. The first

test will use Tcpdump in an attempt to capture the TCP/IP packets being transmitted. The second test will enable the Offloaded Traffic Sniffer to determine if the packets have bypassed the kernel. The Raw Ethernet Client/Server program has two executables: Receiver and Sender. The Receiver program represents the server and the Sender represents the client. During execution, both programs will report each successful message transmission to the user by polling the Completion Queue. If the program was not successful in bypassing the kernel, the TCP/IP packets will be captured by Tcpdump and can be opened and analyzed in Wireshark without utilizing hardware offload. These two tests (with and without offloaded traffic sniffer) follow the following steps:

3.2.1 Without Offloaded Traffic Sniffer.

1 Configure both the server and client host machines to enable IPoIB. This allows QPs to be established based on IP addresses rather than GUIDs.

2 Start Receiver on the server (192.168.1.3). Receiver must be run as root to create QPs.

3 Initiate Tcpdump on the server and specify the appropriate interface to capture packets on.

4 Run Sender on the client (192.168.1.1) to send pre-formatted TCP/IP packets to the receiver.

5 After 10 seconds of capture, terminate tcpdump and save packets to a .pcap file.

6 Terminate both programs on server and client machines.

3.2.2 With Offloaded Traffic Sniffer Enabled.

1 Enable the Offloaded Traffic Sniffer by entering the below command where "enp9s0f0" is the desired interface:

ethtool –set-priv-flags enp9s0f0 sniffer on

2 Repeat steps 2-6 from the previous experiment.

If the Raw Ethernet Client/Server program bypassed the kernel successfully, the pre-formatted TCP/IP packets that were sent will only be captured with the Offloaded Traffic Sniffer enabled.

3.3 Case Study 2: Implementation of a Network Security System with InfiniBand Verbs

This case study will examine the impact of implementing a network security system on an InfiniBand program. For traditional Ethernet networks, there are many types of network security systems that can monitor and control network traffic on the host itself. Below is a list of commonly used systems and a short description of each:

- **Firewalls:** Establish a barrier between trusted and untrusted networks by monitoring and controlling packets entering and leaving the host
- **Host-Based Intrusion Detection Systems:** Signature based applications that monitor network traffic at the host as well as dynamically monitoring a system state
- **Deep Packet Inspection:** A method of filtering network traffic by examining the contents of the payload rather than the headers of the traffic
- **Secure Network Protocols:** Protocols that are used to secure data in transit to prevent unauthorized users/programs access (IPsec, SSL, and SFTP are all examples)

A similarity between all of the mentioned security systems is that they are all commonly implemented as kernel modules that enforce security policies based upon information found within the kernel. How can these systems enforce security on packets that bypasses the kernel all together? This case study will answer this question.

This study explores the implications of implementing IPsec on an InfiniBand application. IPsec will be the focus of this study because of its ability to secure communications between and within a network. Because of this, IPsec proves to be valuable in securing computer communications between critical infrastructure sectors. As mentioned previously, IPsec is a secure network protocol that provides authentication, integrity, and confidentiality between two IP devices and is implemented as a kernel module. Based on the IPsec security policy configured by the user, the IPsec module is forwarded packets based on the source and destination IP addresses and encrypts the packets using a specified algorithm in the TCP/IP stack kernel layer.

In particular, this case study will determine whether or not IPsec can be executed on an InfiniBand program sending TCP/IP packets. For this experiment, the Ethernet 10GbE with Connect-X 5 Adapter

network configuration will be used. The InfiniBand program that will be studied is the Raw Ethernet Client/Server program. Additionally, this case study assumes that the Offloaded Traffic Sniffer must be enabled to capture packets for InfiniBand programs. The steps to conduct this test are listed below:

1 Configure both the server and client host machines to enable IPoIB. This allows QPs to be established based on IP addresses rather than GUIDs.

2 Start the Receiver on the server (192.168.1.3). Receiver must be run as root to create QPs.

3 Initiate Tcpdump on the server with the Offloaded Traffic Sniffer enabled and specify the appropriate interface to capture packets on.

4 Run Sender on the client (192.168.1.1) to send pre-formatted TCP/IP packet to the receiver.

5 After 10 seconds of capture, terminate tcpdump and save packets to a .pcap file.

6 Terminate both programs on server and client machines.

7 Implement the IPsec configuration described previously using the IProute2 utility

8 Repeat Steps 2-6

9 Compare the .pcap files to determine if IPsec was executed.

If IPsec was executed on the InfiniBand program, the second set of packets captured will be in the form of Encapsulated Security Packets (ESP), and packet examination will reveal the encryption. This demonstrates a successful execution of IPsec because the TCP/IP packets are now encrypted with the configured algorithm.

3.4 Case Study 3: Performance of Software-Based Security

The third case study examines the effects of implementing a security system on an application that does not bypass the kernel. As mentioned earlier, IPoIB encapsulates TCP/IP packets after they have traversed the TCP/IP stack in the kernel. Thus, a program that uses IPoIB does not bypass the kernel and will allow IPsec to be executed on its packets.

The program that will be used to evaluate the performance of IPsec on an InfiniBand network is Iperf, a network performance application that tests the maximum throughput a device can handle. Iperf was selected as a the demo application for this test because it replicates the client/server model and sends TCP/IP packets like the Raw Ethernet Client/Server program. Using a 10GbE cable, Iperf will produce a bandwidth slightly under 10Gbps.

Because IPsec's high computing power requirement can limit network throughput performance, it is essential to measure bandwidth with and without IPsec. A total of 10 tests will be run in random order: five with Ipsec and five without Ipsec. Each test will record 300 samples. Each sample is the average bandwidth of a one second interval. The steps to perform each test are:

1 Reboot both Server and Client machines.

2 Configure IPoIB on both server and client with the correct IP configurations.

3 (If using IPsec) Implement IPsec according to the configuration in Experiment Setup

4 Execute Iperf on the server (192.168.1.3) specifying the server's IP address.

5 Run Iperf on the client (192.168.1.1) specifying the client's IP address, the server's IP address, and transmission time.

6 Capture 310 samples, and discard the first 10 to account for ramp-up.

7 Terminate both server and client Iperf programs once specified time interval has been reached.

The results of this case study were analyzed to determine the performance effect of executing IPsec on an InfiniBand network.

4. Results

This section presents the results of the three Case Studies that were performed and discusses how these findings effect the overall security of an InfiniBand network.

4.1 Case Study 1: Results

This study was designed to explore the security implications of bypassing the kernel with an InfiniBand program. In particular, it deter-

mined the ability to monitor kernel bypass traffic with common network traffic analyzers. The first test of this study used the network analyzer tcpdump to attempt to capture packets on the server machine.

The first test revealed that no packets were captured on the specified interface using tcpdump/Wireshark with the Offloaded Traffic Sniffer disabled. The test successfully registered message completions back to the server and client sides of the program indicating successful packet transmission. This strongly suggests that the InfiniBand program did indeed bypass the kernel completely because messages were successfully transmitted yet were not captured in the kernel (recall that tcpdump uses libpcap which is implemented as a kernel module). Therefore, monitoring InfiniBand traffic must be executed outside of the host machine's kernel. (The second test in this study attempts to do exactly that by enabling the Offloaded Traffic Sniffer.)

After enabling the Offloaded Traffic Sniffer, the InfiniBand Client/Server program was run again. Unlike before, the .pcap file recorded by tcpdump did capture packets. A screenshot of the first five packets analyzed in Wireshark can be found below in Figure 4.

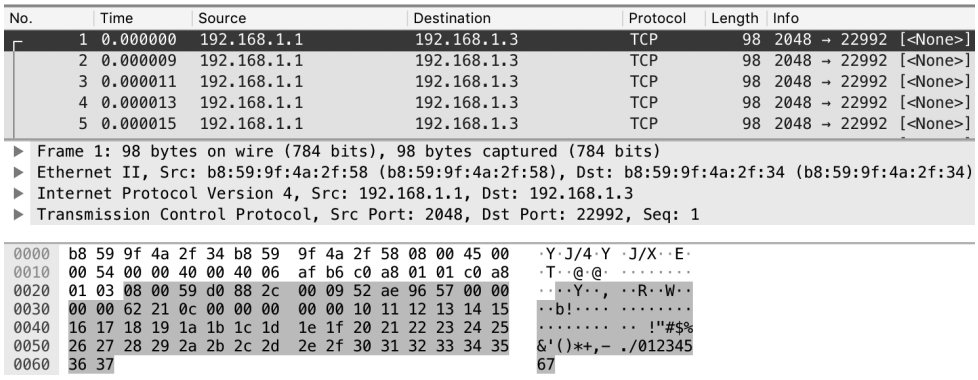

Figure 4. Wireshark Analysis of Captured InfiniBand Packets.

As seen in Figure 4, the exact pre-formated TCP/IP packets created by the InfiniBand Client/Server program are captured. This result has two implications. First, InfiniBand programs can successfully send Ethernet TCP/IP packets without traversing the TCP/IP stack in the kernel suggesting a potential vulnerability with security applications that are executed in the kernel. Second, monitoring InfiniBand traffic is possible

with the assistance of the Offloaded Traffic Sniffer implying the need for hardware implementation of traffic monitoring.

4.2 Case Study 2: Results

Case Study 2's intent was to determine whether or not IPsec could be implemented on an InfiniBand program sending TCP/IP packets. IPsec is executed within the IP layer of the kernel stack when a TCP/IP packet is formed. Thus the question is, if TCP/IP packets are created by an InfiniBand program, can IPsec be implemented on those packets securing that channel?

The first half of the experiment executes the InfiniBand Client/Server program without IPsec just as in Case Study 1. Accordingly, the results from the first half of Case Study 2 are identical to the ones in Figure 4 from Case Study 1, illustrating the successful transmission of TCP/IP packets. The highlighted data section of the packet is sent in plaintext and is not encrypted. The second half of the experiment runs the InfiniBand Client/Server program again with IPsec implemented. These results can be found in Figure 5.

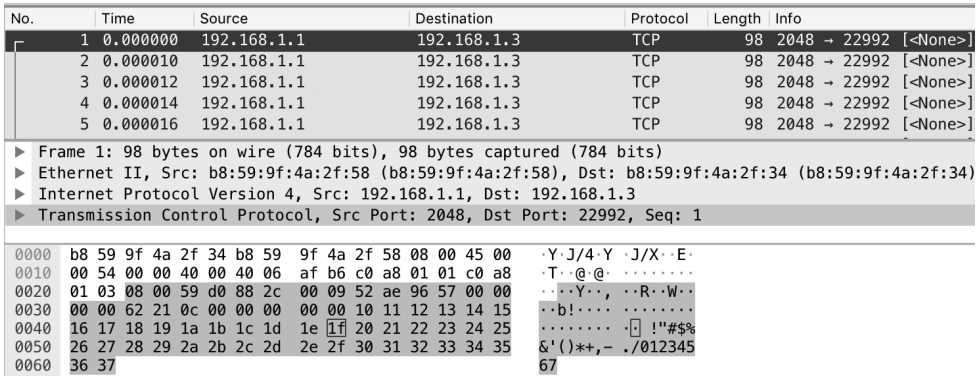

Figure 5. Wireshark Analysis of InfiniBand Packets with IPsec.

The results reveal that IPsec was not executed on the InfiniBand Program. If IPsec was executed correctly, the protocol of the captured packets would no longer be Transport Control Protocol (TCP) but would be Encapsulated Security Payload (ESP), and the payload of the packets would be encrypted using the AES-GCM algorithm and would no longer be human readable. This Case Study can conclude that because the IPsec is executed in the kernel stack, and the InfiniBand Client/Server

Table 2. Case Study 3 results.

Test:	Mean(Gbps)	Max(Gbps)	Min(Gbps)	Std Dev
IPerf no IPsec	8.636	9.40	6.30	0.247
IPerf with IPsec	2.359	2.65	1.93	0.112

Program bypasses the kernel, IPsec cannot be implemented on programs that use InfiniBand verbs. Thus, a need for a different solution outside of software exists. Furthermore, this experiment suggests that many other security systems that are implemented in the kernel cannot be implemented on InfiniBand programs either.

4.3 Case Study 3: Results

After determining that IPsec could not be implemented on InfiniBand program written with InfiniBand verbs to bypass the kernel, the next logical step was to find a program that would allow IPsec execution and evaluate its effect on the performance of the network. Unlike the InfiniBand Client/Server program, Iperf creates TCP/IP packets and thus traverses the kernel stack. Because the implementation of IPsec takes place within the kernel, IPsec can be executed on the packets being transmitted by Iperf.

Based on the results found in Table 2, it is evident that IPsec implementation drastically reduces the network performance. The average bandwidth with IPsec implementation has been reduced to 27.3% of the original. A key reason for this is that IPsec, along with many other network security systems, requires significant resources and CPU utilization to accomplish the cryptographic tasks associated with IPsec. This limits the overall network performance as the CPU has to be dedicated to the cryptographic tasks instead of the application. A possible solution to combat CPU intensive tasks may reside in hardware. Offloading IPsec processes to hardware may free up CPU utilization, speed up encryption algorithms, and increase the network's bandwidth. Additionally, the use of hardware may solve other issues found in the previous case studies. The research has currently discovered that if a security system is to be executed on an InfiniBand network, the application must traverse the kernel. This thwarts the performance benefits of InfiniBand entirely. We contend that a security hardware offload may be able to overcome both of these challenges.

5. Conclusion

The three case studies in this paper attempt to explore the implications and limitations of securing an InfiniBand network. Contributions made by [11] suggested that the IBA is inherently resistant to tampering and attack. However, IBA relies not only on its architecture for security, but relies on the application developer for security at higher layers [8]. Because of this, many applications developed for InfiniBand are built on the same principles and premises as they would be on an Ethernet network. Therefore, it is reasonable to explore the implications of implementing network security systems designed for Ethernet on an InfiniBand network in an attempt to secure it.

Case Study 1 demonstrated that InfiniBand traffic cannot be monitored or captured with traditional network analysis tools due to hardware packet generation that bypasses the kernel completely. Case Study 2 illustrated the impact of bypassing the kernel, suggesting that any network security system implemented in software (specifically the operating system) will be ineffective when used with an InfiniBand program using IBA verbs. Case Study 3 revealed the detrimental performance impact that using a network security system on InfiniBand would have. These case studies motivate the need for a network security system specifically targeted for InfiniBand: a hardware security offload. The tests in this study demonstrated that a hardware device was needed in order to monitor and capture InfiniBand traffic due to the nature of the InfiniBand Verbs. Traditional applications that reside in the host's kernel were unable to inspect the InfiniBand traffic. The test further revealed the significant performance impact of a host based security system running on an InfiniBand network due to the required CPU utilization. This challenge can be overcome by offloading the security system onto hardware thereby reducing CPU utilization and improving network performance. This study indicates that traditional network security practices used for Ethernet networks cannot be directly translated to InfiniBand networks urging the creation of a new class of security system: a hardware offloaded security system.

5.0.1 Disclaimer:. The views expressed in this paper are those of the authors, and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government. This document has been approved for public release; distribution unlimited, case #88ABW-2019-6098.

References

[1] J. Corbet, A. Rubini and G. Kroah-Hartman, *Linux Device Drivers*, O'Reilly Media, Sebastopol, California, 2005.

[2] InfiniBand Trade Association, InfiniBand Architecture Specification Volume 1 Release 1.3, Beaverton, Oregon, USA (`cw.infinibandta.org/document/dl/7859`), 2015.

[3] M. Lee, E. Kim, and M. Yousif, Security enhancement in InfiniBand architecture, *Proceedings of the Nineteenth IEEE International Parallel and Distributed Processing Symposium*, 2005.

[4] M. Lee and E. Kim, A comprehensive framework for enhancing security in InfiniBand architecture, *IEEE Transactions on Parallel and Distributed Systems*, vol. 18(10), pp. 1393–1406, 2007.

[5] Mellanox Technologies, Introduction to InfiniBand, Sunnyvale, California, USA (`www.mellanox.com/pdf/whitepapers/IB_Intro_WP_190.pdf`), 2003.

[6] Mellanox Technologies, InfiniBand Software and Protocols Enable Seamless Off-the-shelf Applications Deployment, Sunnyvale, California, USA (`www.mellanox.com/pdf/whitepapers/WP_2007_IB_Software_and_Protocols.pdf`), 2007.

[7] Mellanox Technologies, InfiniBand: The Production SDN, Sunnyvale, California, USA (`www.mellanox.com/related-docs/whitepapers/WP_InfiniBand_Production_SDN.pdf`), 2012.

[8] Mellanox Technologies, Security in Mellanox Technologies InfiniBand Fabrics, Sunnyvale, California, USA (`www.mellanox.com/related-docs/whitepapers/WP_Secuirty_In_InfiniBand_Fabrics_Final.pdf`), 2012.

[9] Mellanox Technologies, RDMA Aware Networks Programming User Manual, Sunnyvale, California, USA (`www.mellanox.com/related-docs/prod_software/RDMA_Aware_Programming_user_manual.pdf`), 2015.

[10] G. Pfister, An Introduction to the InfiniBand Architecture, in *High Performance Mass Storage and Parallel I/O*, H. Jin, T. Cortes and R. Buyya (Eds.), Wiley-IEEE Press, Hoboken, NJ, pp. 617–632, 2001.

[11] D. Schmitt, S. Graham, P. Sweeney and R. Mills, Vulnerability assessment of InfiniBand networking, in *Critical Infrastructure Protection XIII*, J. Staggs and S. Shenoi (Eds.), Springer, Cham, Switzerland, pp. 179–205, 2019.

[12] K. Subedi, D. Dasgupta and B. Chen, Security analysis on InfiniBand protocol implementations, *Proceedings of the IEEE Symposium Series on Computational Intelligence*, 2016.

[13] TOP500, List Statistics, Sinsheim, Germany (`www.top500.org/lists/2019/11/highs/`), 2019.

[14] U.S. Department of Homeland Security, Communications Sector-Specific Plan An Annex to the NIPP 2013, Washington DC (`www.hsdl.org/?view&did=796518`), 2015.

[15] A. Warren, InfiniBand Fabric and Userland Attacks, SANS Institute, North Bethesda, Maryland, USA (`www.sans.org/reading-room/whitepapers/incident/paper/34012`), 2012.