**IMPLICATIONS AND LIMITATIONS OF SECURING AN INFINIBAND NETWORK**

THESIS

Lucas E. Mireles, Second Lieutenant, USAF

AFIT-ENG-MS-20-M-44

**DEPARTMENT OF THE AIR FORCE**
**AIR UNIVERSITY**

***AIR FORCE INSTITUTE OF TECHNOLOGY***

**Wright-Patterson Air Force Base, Ohio**

AFIT-ENG-MS-20-M-44

# IMPLICATIONS AND LIMITATIONS OF SECURING AN INFINIBAND NETWORK

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Engineering

Lucas E. Mireles, B.S.

Second Lieutenant, USAF

March 2020

AFIT-ENG-MS-20-M-44

IMPLICATIONS AND LIMITATIONS OF

SECURING AN INFINIBAND NETWORK

THESIS

Lucas E. Mireles, B.S.
Second Lieutenant, USAF

Committee Membership:

Scott R. Graham, Ph.D.
Chair

Patrick J. Sweeney, Ph.D., Lt. Col
Member

Stephen Dunlap, M.S.
Member

Matthew J. Dallmeyer, M.S.
Member

AFIT-ENG-MS-20-M-44

## Abstract

The InfiniBand Architecture is one of the leading network interconnects used in high performance computing, delivering very high bandwidth and low latency. As the popularity of InfiniBand increases, the possibility for new InfiniBand applications arise outside the domain of high performance computing, thereby creating the opportunity for new security risks. In this work, new security questions are considered and addressed. The study demonstrates that many 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 impose a significant performance loss for the network. The research concludes that not all network security practices used for Ethernet translate to InfiniBand as previously suggested and that an answer to meeting specific security requirements for an InfiniBand network might reside in hardware offload.

AFIT-ENG-MS-20-M-44

*This work is dedicated to my wife and family for their unfailing love and support.*

# Acknowledgements

Foremost, I would like to express my sincere gratitude to my advisor Dr. Scott Graham for guiding me throughout my graduate education with his expertise and immense knowledge. His enthusiasm and personal generosity made this research process invaluable and I could not have completed it without him.

I would also like to thank my professors and committee members Stephen Dunlap, Matthew Dallmeyer, and Lt Col Patrick Sweeney for mentoring me throughout this process and providing crucial feedback that not only improved my research, but improved me as a learner.

Finally, I must thank my wife for her continuous support, encouragement, and patience at all times. Throughout this entire process, you never once doubted me or let me doubt myself. This accomplishment would not have been possible without you.

Lucas E. Mireles

# Table of Contents

# List of Figures

# List of Tables

# List of Acronyms

**ADAS** Advanced Driver Assistance Systems

**ASIC** Application Specific Integrated Circuit

**CA** Channel Adapter

**CAN** Controller Area Network

**DMA** Direct Memory Access

**ECU** electronic control unit

**FPGA** Field Programmable Gate Array

**GID** Global ID

**GRH** Global Route Header

**GUID** Globally Unique ID

**HCA** Host Channel Adapter

**HDL** Hardware Description Language

**HPC** High Performance Computing

**IBA** InfiniBand Architecture

**IBTA** InfiniBand Trade Association

**ICRC** Invariant CRC

**IP** Internet Protocol

**IPoIB** IP over InfiniBand

**L_Key** Local Memory Key

**LID** local identifier

**LRH** Local Route Header

**LUT** Look up Table

**MAC** Media Access Control

**MST** Mellanox Software Tools

**NIST** National Institute of Standards and Technology

**OS** Operating System

**P_Key** Partition Key

**PCIe** Peripheral Component Interconnect Express

**Q_Key** Queue Pair Key

**QP** Queue Pair

**R_Key** Remote Memory Key

**RDMA** Remote Direct Memory Access

**RoCE** RDMA over Converged Ethernet

**SDK** Software Development Kit

**SDN** Software Defined Networking

**SM** Subnet Manager

**SOC** System on Chip

**TCA** Target Channel Adapter

**TLP** Transaction Layer Packet

**VCRC** Variant CRC

IMPLICATIONS AND LIMITATIONS OF
SECURING AN INFINIBAND NETWORK

# I. Introduction

## 1.1 Background and Motivation

The evolution of technology continues to demand an increase in computer processing power, which semiconductor manufacturers have continued to meet. However, industry-standard Input/Output (I/O) busses have not produced the levels of availability, reliability, performance, and scalability necessary to achieve the potential of this continual increase [3]. To overcome this hurdle, the InfiniBand Trade Association (IBTA) was founded, comprised of over 180 companies, to develop a new interconnect technology, the InfiniBand Architecture (IBA).

The IBA is a powerful interconnect architecture that is quickly becoming the standard for I/O connectivity in servers and High Performance Computing (HPC). In fact, 28% of the Top 500 Supercomputers use InfiniBand as their interconnect, accounting for over 35% of the total performance, and second only to Gigabit Ethernet as seen in Table 1 [4]. This is largely due to its ability to provide higher bandwidth and lower memory latency than its Ethernet competitor through a copy-avoidance architecture, to reduce CPU utilization. InfiniBand relies on point-to-point connections to accomplish data transfers and treats all I/O as a form of communication [3]. Its advanced capabilities provide extremely high bandwidth and very low latency communications between hosts and devices with little overhead, making it ideal to carry multiple traffic types including clustering, communications, storage, and management [5]. 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 continues to grow in all areas of computer communication [6].

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

**Table 1. Top 5 Supercomputer Interconnects.**

### 1.2 Problem Statement

Anticipating future deployment of InfiniBand networks outside the domain of HPC clusters, it is essential to explore and evaluate the security landscape of IBA. Though some research has been conducted on the security of InfiniBand [7, 8, 9, 10, 11], the experiments focused heavily on the protocol itself and did not consider other applications. The security features for the IBA were designed for its deployment in large data centers, and therefore did not consider possible risks outside this environment. As IBA deployments increase, it is inevitable that the IBA will soon be a target for malicious cyber attacks, and users should be aware of potential vulnerabilities and implications. This paper addresses some additional security questions not answered by previous work with the hope of finding a solution that will help mitigate potential cyber threats. This 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 their impact on network performance. An analysis of this work will help guide future research in securing an InfiniBand network.

## 1.3 Research Objectives

This research evaluates the implications of securing an InfiniBand network and explores potential solutions to accomplish this goal. Three case studies are performed to determine if traditional security practices for Ethernet networks could be implemented on InfiniBand. Additionally, three Mellanox adapters are explored to establish their security limitations. The research objectives for this work are outlined below:

- Present key concepts behind the IBA specification such as its communication model, software architecture, and it current security features.
- Share the motivation behind the creation of the IBA and how future applications can employ its advanced capabilities outside of the HPC environment.
- Understand the National Institute of Standards and Technology (NIST) Framework, its intention, and how it can be used to evaluate the security of an InfiniBand network.
- Set up and deploy an operational InfiniBand network.
- Develop a custom InfiniBand program that can be integrated with the deployed network.
- Integrate Ethernet network security systems on an InfiniBand network.
- Evaluate the network performance impact that network security systems have on InfiniBand networks.
- Understand the role of device drivers and how they interact with hardware devices to provide desired capabilities.
- Define what it means to secure an InfiniBand network, and determine what security capabilities are required.

- Identify the future hardware device that will be used to secure an InfiniBand network.

The questions that are to be answered by this research that accomplish the preceding objections are listed below:

- Is it feasible to secure an InfiniBand network with network security systems used on Ethernet networks?
- What are the network performance impacts associated with an implementation of a network security system on an InfiniBand network?
- Are there advantages to using a hardware offloaded security system as opposed to using traditional security system implemented within the kernel?
- Is there a hardware device compatible with the IBA that is capable of Protecting, Detecting, and Responding to potential cyber threats at line speed?

## 1.4 Organization

The organization of this thesis is outlined as follows. Chapter II introduces the IBA, the main concepts surrounding its implementation, and the components that allow its advanced network capabilities. It discusses the NIST Framework and the five core functions that will later be used to define securing an InfiniBand network. Additionally, it presents relevant technology, a possible example application of an InfiniBand network, and related research.

Chapter III is a collection of three case studies used to explore the difficulties and implications of securing an InfiniBand network. It describes the test bed setup, network configuration, and network security systems utilized to conduct the listed procedures of all three case studies. It finishes by presenting, analyzing, and discussing the results of each case study and concludes that a hardware offloaded security system may be the answer to secure an InfiniBand network.

Chapter IV explores three possible hardware devices that could be used to secure an InfiniBand network. It defines the desired security capabilities the solution must possess, assesses suitable technologies for device implementation, and presents the exploration approach used to examine all possible solutions. It compares all three devices' theoretical implementations of a hardware offloaded security system and concludes the Mellanox BlueField SmartNIC is the appropriate device for its implementation.

Finally, Chapter V summarizes the work that was accomplished and lists the major contributions to the area of InfiniBand security. It presents future work areas for this research that will improve upon the security and manageability of an InfiniBand network including Software Defined Networking (SDN) and machine learning approaches. In concludes by challenging the HPC and cyber communities to make securing InfiniBand a top priority.

# II. Background and Related Work

## 2.1 Overview

This chapter presents background information and knowledge about the IBA and discusses relevant technologies associated with this research effort. It begins by reviewing the NIST Cybersecurity Framework which will be used to help define the goal of securing an InfiniBand network. Next, it describes the IBA, the services that it provides, and the components that allow its operation. An overview of the software and hardware stack layers are presented and the relationship between them and the InfiniBand communication model is introduced. It describes three key technologies used in this experiment to help explore the security limitations of current InfiniBand networks. It concludes by summarizing current literature in the field of InfiniBand network security highlighting potential areas of interest that need to be further explored.

## 2.2 NIST Cybersecurity Framework

NIST was given the task of identifying and developing cybersecurity risk frameworks for critical infrastructure owners and operators by the Cybersecurity Enhancement Act of 2014 [12]. The act stated that NIST's framework must identify "a prioritized, flexible, repeatable, performance-based, and cost-effective approach, including information security measures and controls that may be voluntarily adopted by owners and operators of critical infrastructure to help them identify, assess, and manage cyber risks" [12]. The NIST framework offers a flexible way to address and mitigate the risks of cybersecurity by prioritizing and identifying required actions. The key reasons for selecting the NIST framework reside in its scalability. It can be used to help manage cyber risks for large, complex organizations or it can be used to manage cyber risks for specific critical services such as an interconnect architecture: IBA.

The NIST Framework consists of three main parts: Framework Implementation Tiers, Framework Profiles, and the Framework Core. Framework Implementation Tiers describe the degree of cybersecurity risk management that a particular organization is willing to practice. The implementation tiers set the tone for cybersecurity risk management within the organization. A Framework Profile characterizes the current or future desired state of an organization based on the alignment of its cybersecurity risk management practices and guidelines to the Framework Core. These profiles are conducted as self-assessments of an organization's cybersecurity risk management [12]. The Framework Core presents standards, guidelines, and practices that ease comprehension of cybersecurity activities and outcomes from the leadership level to the operations level [12]. The Framework Core will be the primary focus of this research because it deals directly with the implementation of cybersecurity risk management via its five Core Functions: Identify, Protect, Detect, Respond, Recover. These Core Functions act as the backbone of the Framework Core as they provide a high-level, strategic view of basic cybersecurity activities by organizing information, enabling risk management decisions, addressing threats, and improving from previous lessons learned [12]. The five Functions are defined below:

- **Identify** - Identifying possible cybersecurity risks that can affect an organization's resources, assets, data, and capabilities. It enables an organization to focus and prioritize its efforts of cybersecurity based on identified risks.

- **Protect** - The ability to limit or contain the impact of a potential cybersecurity event without affecting the organization's mission and services.

- **Detect** - Defines the appropriate actions to identify the occurrence of an anomalous cybersecurity event and enables its timely discovery.

- **Respond** - Defines the reactive activities after a cybersecurity event has been detected. It is a post-event activity that contains the impact of the detected cybersecurity event.

- **Recover** - Establishes the methods to restore the capabilities, assets, and data that were disturbed in a timely matter after an event has occurred.

### 2.3 The InfiniBand Architecture

The IBA is a network protocol architecture that is becoming the de facto standard for server I/O and server-to-server communications for large HPC clusters and Storage Area Networks. The IBA is comparable to the Ethernet network protocol, but designed to be implemented in data centers with HPC clusters and logically separated from the Internet [7]. The development and design of IBA was driven by the inability of industry standard I/O systems using traditional I/O buses 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 [3]. 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 [3, 1].

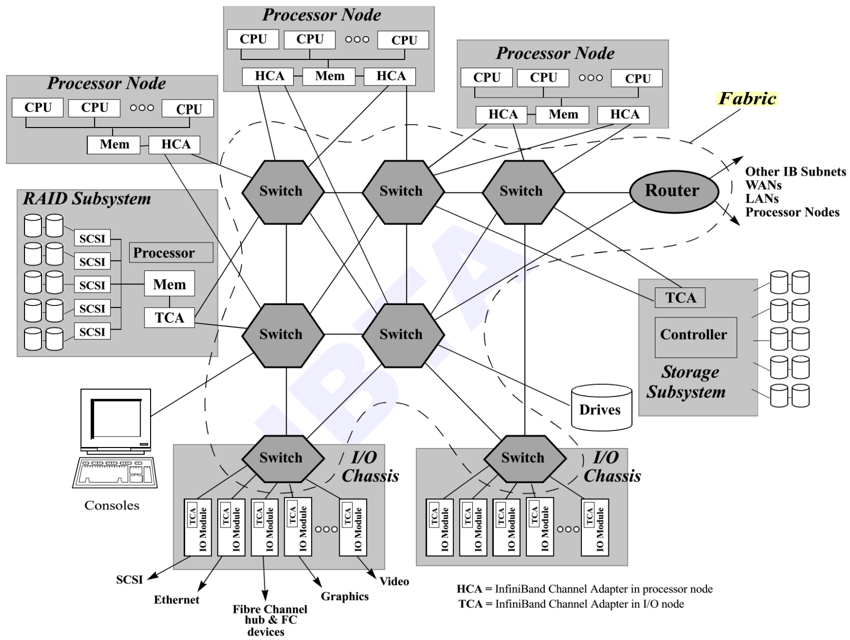


**Figure 1. IBA Storage Area Network with Fabric Highlighted [1]**

### 2.3.1 Infiniband Components

From a high level perspective, IBA is an interconnect for processors, I/O units, and routers which can all be considered endnodes. At its smallest, a complete IBA network can be an IBA subnet which is comprised of endnodes, switches, links, and a subnet manager [3]. 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.

#### 2.3.1.1 Channel Adapters

There are many components that comprise an IBA network but this study will focus on the Channel Adapter (CA). Every end node that is a part of an IBA network must have a CA as they are the devices in the network that generate and consume IBA packets [1]. A CA is defined as either an Host Channel Adapter (HCA) or as a Target

Channel Adapter (TCA). The HCA provides the consumer a collection of features that are 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 Section 2.3.4). Each HCA is assigned a Globally Unique ID (GUID) by the manufacture of the chip. Additionally, each of its ports is assigned a port GUID that identifies it globally (within a subnet and between subnets).

#### 2.3.1.2 Subnet Manager

Another important component of the IBA is the Subnet Manager (SM). InfiniBand's implementation of routing and forwarding are similar to the concept of SDN [13]. The routing and forwarding tables for IBA switches and routers are not decided on each device. Instead, the SM is responsible for configuring and managing all switches, routers, and channel adapters that are part of a subnet [1, 13]. The SM actively communicates with each switch, CA, and router's Subnet Manager Agent to ensure all routing and forwarding tables are correct [7]. The IBA is designed to allow more than one SM on a subnet at a time for resiliency, i.e., there is only one master 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. The state machine can be found in Figure 2.

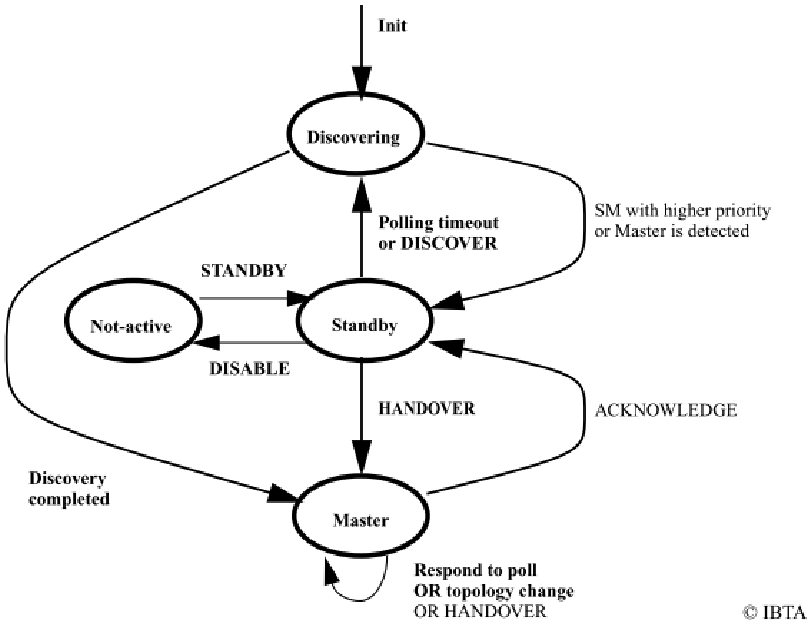


**Figure 2. State Machine for Initialization of Subnet Manager [1]**

#### 2.3.1.3 Switch

As in other network protocols such as Ethernet, the switch is responsible for forwarding data from one port to another based upon addresses at the data link layer. Similar to Ethernet, where forwarding decisions are based upon Media Access Control (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 through which a switch will forward a packet. Every switch is configured with forwarding tables that include these paths for every LID within a subnet. Multiple paths to and from destinations can exist: 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 an InfiniBand fabric.

### 2.3.2 Software Architecture

To maintain independence of the host Operating System (OS) and processor, the IBTA has produced a software architecture that is compatible with all major OSs [2]. 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 [2, 1]. A visual representation of the IBA software stack can be shown in Figure 3. InfiniBand's kernel space can be divided into three major layers: upper layer protocols, mid-layer core, and HCA drivers [2].

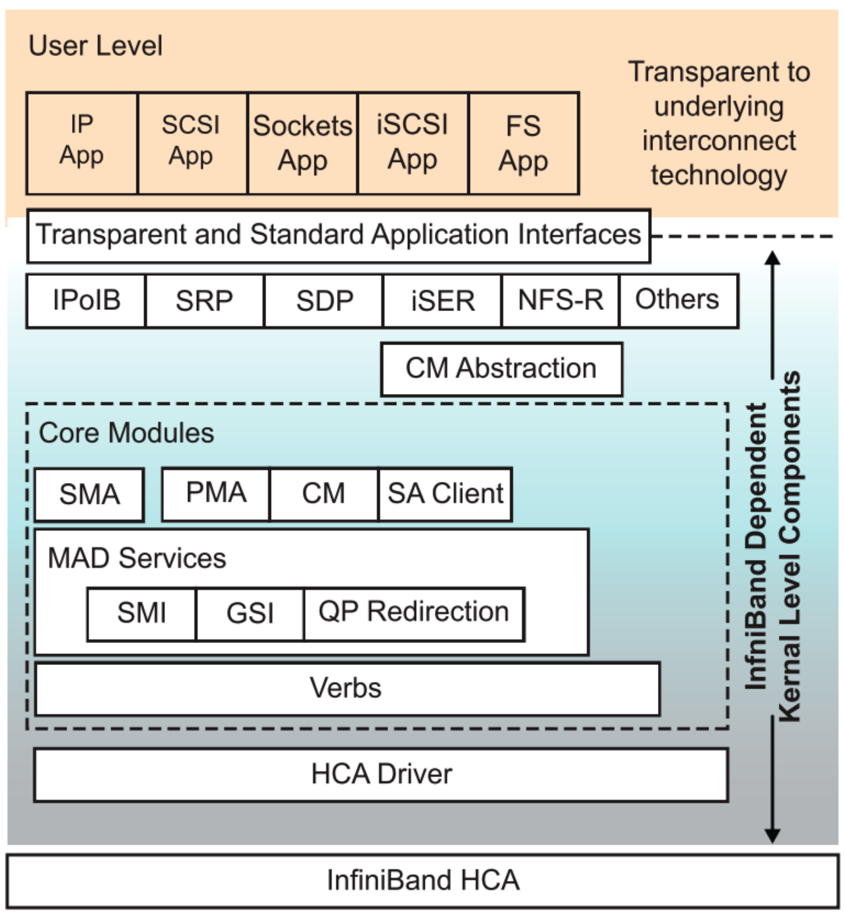


**Figure 3. IBA Software Stack [2]**

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 [14]. Per the IBA specification, each HCA driver has its own specific driver that must be compatible with the mid-layer core kernel modules [1].

The kernel 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 (MAD) interface, 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 [1]. 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 allow user applications to interact and have access to InfiniBand hardware [2].

The last layer of the kernel space to discuss is the upper layer protocols. Upper layer protocols enable existing applications that employ standard data networking and file system access to operate over the IBA [2]. 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–IP over InfiniBand (IPoIB) and RDMA over Converged Ethernet (RoCE)– and how they compare to Remote Direct Memory Access (RDMA) operations.

#### 2.3.2.1 IPoIB, RoCE, and RDMA

IPoIB is an upper layer protocol that implements a network interface over the IBA. IPoIB encapsulates IP datagrams over an InfiniBand transport service [2]. 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 [15]. 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 RoCE. RoCE provides true RDMA semantics over Ethernet [15]. It is the most efficient low latency Ethernet solution today requiring far less CPU overhead than other RDMA solutions such as iWARP[15]. Like the RDMA over InfiniBand, RoCE as well uses the InfiniBand Verbs to craft packets for its applications.

### 2.3.3 IBA Stack Layers

Much like Ethernet and other interconnect protocols, IBA is a stack based communication architecture that is comprised of the physical, link, network, and transport layers of the 7-layer OSI network model. The protocol at each layer is completely independent of the others; yet, the IBA operations at a layer are dependent on the service of the layer below and provide a service to the layer above. The layers of the IBA architecture are shown in Figure 4. A brief introduction of each layer is outlined below:

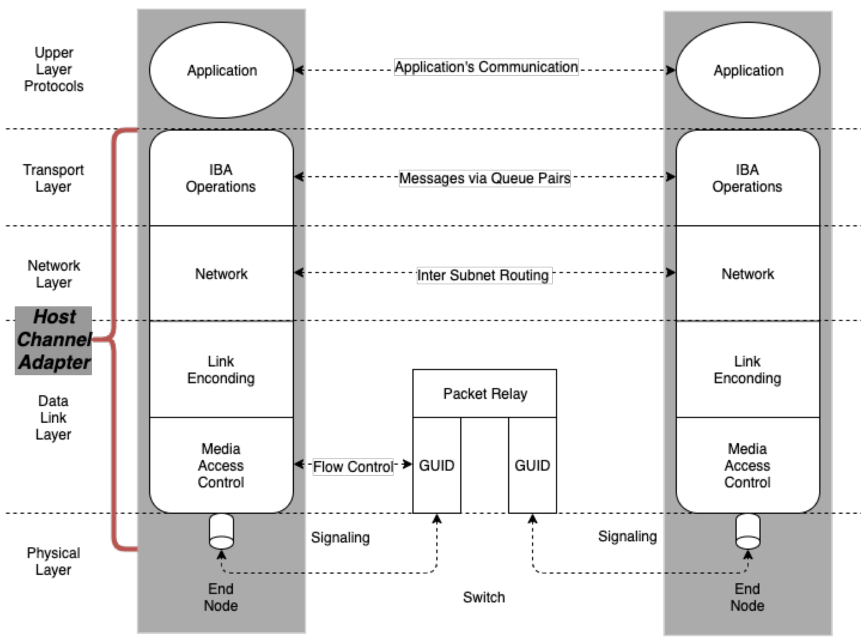


**Figure 4. InfiniBand Architecture Stack Layers**

- **Physical Layer.** The physical layer is responsible for establishing a physical link, informing the link layer of the current mechanical/thermal status of the physical link, and informing the link layer whether it is up/down. It specifies the IBA's signaling protocol by defining proper symbol encoding, alignment of framing symbols, and a synchronization method used by valid packets [1].

The physical layer establishes the bit rates, media, connectors, and signaling techniques that are to be used within an InfiniBand network.

- **Link Layer.** IBA's link layer specifies packet format, addressing within a subnet, flow control, and error detection. There are two packet types specified by the IBA: link management and data packets. Link management packets are used to carry control information that help configure link width, data rates, flow control management, and link integrity. Although these packets are commonly used, this study will focus primarily on data packets. Data packets carry out IBA operations.

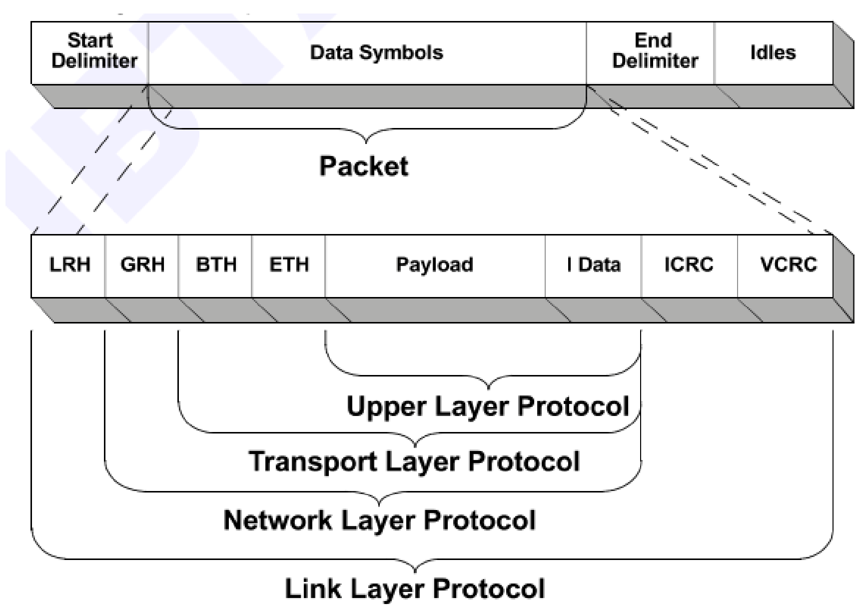


**Figure 5. IBA Data Packet Format** [1]

As seen in Figure 5, each data packet contains multiple headers. The data link layer is responsible for creating the Local Route Header (LRH) which identifies the source and destination ports that switches will use to forward packets. To accomplish this, the source port places both the source and destination LIDs within the LRH so that the switches can properly forward packets to their destination.

Flow control is the process of managing data transmission between two nodes to ensure the sender does not overwhelm the receiver. The IBA handles this process at the link layer using a credit based method [1]. Credits indicate the number of data packets that the receiver can accept per Virtual Lane and are sent periodically from the receiver. If the receiver indicates that it has no more room for packets, the transmitter discontinues transmission until the receiver has room.

The last service that is provided by the link layer is error detection. The link layer is responsible for detecting physical errors, receiver errors, and transmission errors. To accomplish this, the link layer implements two forms of CRCs: Invariant CRCs (ICRCs) and Variant CRCs (VCRCs). The ICRC covers all fields within a packet that do not change during the packet's lifetime. The VCRC covers all fields within the packet including those that do change. Together, these CRCs allow routers and switches to change necessary fields while still maintaining end-to-end data integrity enabling error detection [1].

- **Network Layer.** The network layer specifies the protocol for routing packets between IBA subnets. Routing packets between IBA subnets is handled by IBA routers (not discussed because they are not within the scope of this study). The routers use the Global Route Header (GRH) to identify the source and destination Global IDs (GIDs) of the packet. The GID is a combination of the unique subnet prefix and the ports' GUIDs. The source places the GID in the GRH and the LID of the next router in the LRH. As the packet traverses different subnets, the routers modify the contents of the GRH and LRH in order for the packets to reach the final destination. However, the GID is never replaced and is protected by the ICRC because it is never changed during the lifetime of the packet. The last router along the path replaces the LRH with the destination LID [1].

- **Transport Layer.** In the typical Ethernet OSI model, the transport layer ensures logical end-to-end communication between processes. With IBA, the transport layer ensures the packet is delivered to the correct Queue Pair (QP) and instructs the QP how to process the data [1]. Additionally, the transport layer is responsible for segmentation of packets that exceed the MTU and the re-combination of received packets. Upon creation, a QP is associated with one of five IBA transport services or one of two non-IBA protocol encapsulation protocols. These transport services determine the degree of reliability and the means by which the QP communicates its data. For reliable services, the receiver sends either an ACK or a NAK to the sender to notify whether the packet was received or not. Unreliable services do not use acknowledgement messages rather generate sequence numbers. The sequence numbers are used to detect out-of-order and missing packets enabling the responder to perform local recovery processes.

  **IBA Transport Services**:

  - Reliable Connection
  - Reliable Datagram
  - Extended Reliable Connection
  - Unreliable Datagram
  - Unreliable Connection

  **Non-IBA protocol Encapsulation Services**:

– Raw IPv6 Datagram

– Raw Ethertype Datagram

### 2.3.4 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 QPs 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 where from. 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 [1]. Each QP is independent of one another which provides isolation and protection from other QP operations being performed.

### 2.3.5 Current 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 [16]. 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 [3]. To prevent unauthorized access to shared resources, a hardware mechanism called Partition Keys (P_Keys) is 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 [1]. Furthermore, partitions are controlled centrally from the SM preventing nodes from determining their own partitions. This further reduces potential hacking and security holes because it entirely eliminates the ability for the host to manipulate what shared resources it can have access to [16].

Additional claims are that the IBA eliminates an attacker's ability to access unauthorized destinations, sniff unintended traffic, and impersonate other entities [16]. 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 [16]. 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 SM and all packets are dropped [16].

The last mechanism for security that IBA implements is memory protection. Because IBA uses RDMA, which can raise security issues as node's 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 a Local Memory Key (L_Key) and a Remote Memory Key (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 or changed in software, ensuring memory protection on all IBA devices. [1, 16].

## 2.4 Example Application: Vehicle Networks and ADAS

Advancements in vehicle hardware and processing technology continue to promote new innovations in vehicle networks. The history of modern vehicle networks began when new electronic sensors and vehicle applications were implemented as stand alone electronic control units (ECUs) [17]. This led to very complex systems utilizing different network protocols that did not allow for subsystems to communicate with one another. Eventually, point-to-point communication links were implemented between individual ECUs, enabling more functionality through data sharing [17]. As expected, this solution proved to be very inefficient as the complexity of wiring and links increased exponentially as the number of ECUs increased. The solution to this problem led to the first modern vehicle network: the Controller Area Network (CAN) bus.

The CAN bus is a standard that allows ECUs and other vehicle applications to communicate with one another on a shared bus. The CAN bus is used to transmit the majority of all intra-vehicle communication. Specifically, it is utilized to communicate powertrain and body control information within the respective domains as well as being the standard to retrieve On-Board Diagnostic information about the vehicle [18]. Although the bus network solves the issue of connectivity between ECUs, increasing the number of ECUs and applications that are connected to the bus creates limitations to the amount of bandwidth available. Previously in the automotive industry, a vehicle's network bandwidth consumption was not a concern due to the sensors' low data rates used in control applications [17]. However, the vehicle bandwidth requirement has now become a major concern due to Advanced Driver Assistance Systems (ADAS) and the movement towards autonomous vehicles. In fact, ABI Research Vice President stated that "The emergence of drive-by-wire, the explosion of in-vehicle sensors for ADAS and automated driving, and the adoption of connected infotainment, poses new challenges for in-vehicle networking technologies in terms of cost, bandwidth, cable harness weight, and complexity" [19]. ADAS and autonomous vehicles require a multitude of sensors, including high-resolution cameras, radars, ultrasonic sensors, and LiDARs. Because these technologies require significantly more bandwidth than typical control traffic, an alternate to the CAN bus must be found as it can no longer keep up with the timing and bandwidth requirements of future technologies [18].

Recent developments in the automotive industry suggest that Ethernet may be the new standard for intra-vehicle communication. Ethernet is a network protocol that defines how nodes communicate with one another within a wired local area network, perhaps a vehicle in this case. One of Ethernet's advantages over the CAN bus is the increased bandwidth that it has to offer. In the paper [17], a raw bandwidth requirement calculation was made for an uncompressed 1280x960 pixel resolution camera stream at 30 frames/s. They concluded that for the transmission of this video stream, a vehicle network would need to be able to support 884.74 Mbps. This calculated requirement far exceeds the bandwidth limits of the CAN bus (1Mbps) and is only one of the many sensors needed for driver assistance/autonomy. With Gigabit Ethernet, this requirement can be met.

The Mobileye EyeQ processor is one example of Gigabit Ethernet implemented in vehicle networks. It is the leading processor for ADAS and autonomous vehicles, used in over 15 million vehicles sold as of 2017 [20]. The latest Mobileye EyeQ is the EyeQ5. The EyeQ5 has dedicated 40Gbps Ethernet to support its sensor interfaces including high-resolution cameras, radars, and LiDARs [21]. The EyeQ5 can support additional sensors via PCIe and Gigabit Ethernet Ports with 18Gbps of additional bandwidth. Clearly, the bandwidth requirement for future ADAS and autonomous vehicles has far surpassed the limits of the CAN bus and all other vehicle network protocols. As more innovative technology surfaces and sensors begin capturing more and more data to be processed, these bandwidth requirements will only continue to grow. At some point, Gigabit Ethernet may no longer be able to meet the growing demand of bandwidth and timing constraints, perhaps leading to the adoption of IBA as a new standard for vehicle networks, one of the many applications outside the domain of HPC that InfiniBand may be deployed.

### 2.5 Relevant Technologies

Although this research is focused primarily around the IBA and the services it provides, it is important to discuss relevant technologies used in this work. Specifically, Field Programmable Gate Arrays (FPGAs), the Peripheral Component Interconnect Express (PCIe) bus, and device drivers are presented and discussed in the following sections:

### 2.5.1 Field Programmable Gate Array

FPGAs are semiconductor devices that can be reconfigured and reprogrammed for desired application or functionality requirements after manufacturing. FPGAs have become the standard for digital design and implementation of integrated circuits due to their unique architecture comprised of programmable logic units, configurable interconnects, and logic gates [22].

As mentioned before, FPGAs contain an array of programmable logic blocks and programmable interconnects enabling a multitude of digital designs to be implemented on the same device. The term “programmable” indicates the ability to reprogram the functionality of an FPGA after chip fabrication, differentiating FPGAs from Application Specific Integrated Circuits (ASICs) [23]. Programmable logic blocks are the fundamental component of FPGA architecture that are comprised of Look up Tables (LUTs), multiplexers, and memory elements which provide the user configurable logic gates. LUTs are customizable pieces of hardware that store an array of values defined by the programmer. The multiplexer selects the appropriate value stored in the LUT to use as the output of the logic block. This array of logic blocks is “wired” together by programmable interconnects. These interconnects enable the implementation of a variety of circuit topologies by allowing the user to change the connections and routing between logic blocks and other I/O blocks (e.g. memory) [23]. Hardware Description Languages (HDLs) are used by developers to create and configure integrated circuits to be programmed onto the FPGAs. HDLs are computer languages used to describe the digital logic that forms the desired circuit to be placed on chip.

FPGAs are known largely for their ability to accomplish specific tasks much more efficiently than traditional CPUs. Even though their clock rates are at a much slower rate than CPUs (hundreds of Mhz compared to Ghz), FPGAs rely on the parallel nature and optimality of their architecture and resources to accomplish tasks [22].

FPGAs optimize primitive resources and operations spatially as opposed to sequentially like CPUs. This spatial organization permits parallelism which allows faster processing, less instruction overhead, and more active computations within the same area as compared to that of a CPU [22]. Additionally, the low level nature of FPGAs allow bitwise operations that are inherently faster computationally than CPUs. Although FPGAs are very efficient in applications that allow parallelism and high clock to data ratios, their use is not always optimal. Specific examples include complex calculations and floating point math. Nevertheless, FPGAs are implemented in a key application relevant to this research: cyber security. Current FPGA applications in cyber security deal primarily with cryptography and digital key exchanges due to CPU speed limitations. However, as network protocols continue to increase in speed, FPGAs will eventually be implemented in other cyber security applications dealing with network traffic.

### 2.5.2 Peripheral Component Interconnect Express

The PCIe is a high performance, high bandwidth, general purpose interconnect implemented as a computer extension card standard for high performance devices [24]. Its design is an improvement over the previous bus standards of PCI and PCI-X yet is still compatible with both. The PCIe is a switched lane architecture that utilizes lanes to communicate packets over point-to-point connections as opposed to a shared bus[22]. It accomplishes this feature by taking advantage of recent advances in point-to-point communication, switch-based technologies, and packetized protocols [24]. A lane is characterized as a set of differential signal pairs: one for transmission and another for reception. Each differential signal pair in a lane is a dedicated, unidirectional, serial, point-to-point connection. To scale bandwidth across multiple lanes, the PCIe specification allows for x1, x2, x4, x8 , x16, and x32 lane widths.

Currently, Gen 4 PCIe, the standard interface on all hardware used in this study, provides an effective 16.0 Gigabits/second/lane/direction of raw bandwidth. To put this in perspective, a Mellanox Host Channel Adapter Card with a PCIe 4.0 x16 interface provides a raw bandwidth rate of 256 Gigabits/second/direction over the PCIe interface. However, due to necessary overheard and other system design trade-offs, the effective performance is lower than specified raw data rate.

As previously mentioned, the PCIe is implemented as a switched lane architecture via point-to-point connections as opposed to a true shared bus architecture. Previous interconnect standards were implemented as true shared bus architectures meaning each peripheral device connected could “listen” to each packet transmission. Conversely, the PCIe is implemented much like a switch-based network. Communication on the PCIe is conducted via Transaction Layer Packets (TLPs). A TLP is a packet generated by the PCIe protocol to convey a request or completion at the transaction level [24]. Although the PCIe specification does not use MAC addresses to route packets, TLPs contain the geographic location in the I/O address space for the destination device. PCIe switches use the address provided by the TLP to properly route the packet to the correct I/O address space. PCIe switches connect two or more PCIe ports which allow TLPs to be forwarded. This discussion of the PCIe illustrates the complexity of a commonly overlooked technology.

### 2.5.3 Linux Device Drivers

The host machines used in this research use Linux as an operating system. Thus, this section deals specifically with Linux device drivers. Device drivers are software programs that enable users to control and operate hardware devices connected to a host machine. They hide the low level operations to and from the device from the user through well-defined internal programming interfaces [14]. Device drivers operate in the kernel space and can access built-in kernel functions to communicate with peripheral devices. Device drivers can either be compiled into the kernel statically or can be loaded at run-time as kernel modules. It is standard practice to implement device drivers as kernel modules as opposed to static device drivers to provide the user flexibility to add or remove functionality while the system is running. After a device driver is loaded into the kernel, device files are created into user space to represent the loaded kernel module. They can then be used to interact with the device through system calls such as "getchar" and "fread" [22]. This ability establishes device files as the interface between user applications and device drivers.

All peripheral devices utilized in this research use PCIe as the interface to the host machine. Thus, it is necessary to discuss PCIe device drivers and how they function. Upon system boot, all PCIe devices are automatically configured by mapping the devices' memory and I/O regions to the processor's address space. This process is performed by the kernel and is unique to PCIe devices due to their requirement for configuration registers [14]. All PCIe devices, whether a graphics card or an FPGA with a custom image, must abide by the PCIe specification for the host's kernel to recognize them as a device. To identify a particular PCIe device in the system, three configuration registers are used: vendor ID, device ID, and class. These three registers are then used by the device driver to "look up" the device when the module is loaded.

It is important to note that many device drivers are created by device manufacturers specific to the hardware's capabilities. In other words, the device drivers cannot be manipulated to change the hardware's capabilities. In contrast, PCIe FPGA devices can. As mentioned earlier, FPGAs are re-programmable pieces of hardware that perform different operations depending on the loaded image. When implemented as a PCIe device, each image burned onto the FPGA requires an updated device driver that provides access to the different functionality provided by the image and abides by the PCIe standards. This discussion of device drivers and their relationship with hardware helps guide the exploration of potential solutions for securing an InfiniBand network in this work.

## 2.6 Related Work in IBA Security

Research into the security of IBA began shortly after the formation of the 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.

### 2.6.1 Insights into IBA vulnerabilities

The authors of [8] and [9] identified IBA security gaps and suggested that with moderate effort, the associated vulnerabilities could be exploited. The work found two major authentication vulnerabilities. First, IBA's partitioning keys are sent in plain-text over the network, and therefore do not fully mitigate the risk of illegal traffic on a network. 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 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 as it traverses network hops, 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.

### 2.6.2 IBA GUID Spoofing

Ethernet MAC spoofing is trivially accomplished and allows for simple attacks that 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 [16], IBA packets are crafted in hardware, with the GUID residing in firmware and only changeable via reprogramming the HCA (by flashing the firmware). However, [10] 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.

### 2.6.3 Security Analysis of InfiniBand Protocol Implementation

In [11], 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 were no significant security vulnerabilities in the protocol itself, however, three functions were potentially vulnerable that they recommend be replaced.

### 2.6.4 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 [7] 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 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 [7]. 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 an SDN approach in fabric management. As mentioned previously in [10], a proposed method of preventing GUID spoofing from occurring in an IBA network would be to implement a monitoring system. A way in which this might be implemented is to use an SDN approach as proposed by [7]. 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.

### 2.6.5 An FPGA implementation for a high-speed optical link with a PCIe interface

This study [22] sought a solution to overcome performance bottlenecks in Ethernet and InfiniBand based networks to achieve speedup for multi-node and multi-GPU computing platforms. The solution was to implement an optical fiber high speed interface between two devices using FPGAs. The FPGA acted as the physical interface between the fiber optic link and the computer via the PCIe. Additionally, a Linux device driver was used that enables applications on the host computer to interact with the optical link used during the experiment. The study was able to successfully transmit and receive messages at over 8.5 Gbit/s which exceeded the previous works in this area. This study relates to this thesis study due to its use of FPGAs, a high speed fiber link, and its detailed explanations of driver implementations. It lays out the groundwork for creating a new network interface other than Ethernet and Infiniband which allows for greater insight into how the IBA is employed and how it can be altered to add security features.

## 2.7 Summary

This chapter began by providing a brief description of the NIST framework and highlighted the core functions that ultimately guide the direction of this research. The chapter then continued to describe the basics of the IBA, its functionality, and the security mechanisms implemented to ensure security. Relevant technologies to the research were briefly discussed as well as a possible application for InfiniBand: Vehicle Networks. By mentioning the possible application of InfiniBand in Vehicle Networks, it becomes obvious that a further investigation into the IBA is needed. IBA was designed for HPC use and therefore its security was scoped to examine common applications that are used in HPCs. Thus, with new potential InfiniBand applications, the security landscape must be expanded as new vulnerabilities could be present.

# III. Infiniband Case Studies

## 3.1 Objective

Past research into the security of InfiniBand occurred within the HPC domain and focused primarily on its architecture. Thus, the current security landscape of future InfiniBand applications remains unknown. To obtain this knowledge, three case studies are performed. The goal of these case studies is 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 work will examine the difficulties of implementing well-known network security systems on multiple types of InfiniBand applications to demonstrate potential 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.

### 3.1.1 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 supported by the InfiniBand Verb construct that allows the application to communicate directly with the hardware that crafts the traffic. This research therefore includes 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:

**Ethernet 10GbE with Connect-X 5 Adapter.** The network configuration used in this study is shown in Figure 6. This configuration is comprised of two host machines, each with a Connect-X 5 adapter. The two Connect-X 5s are 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.

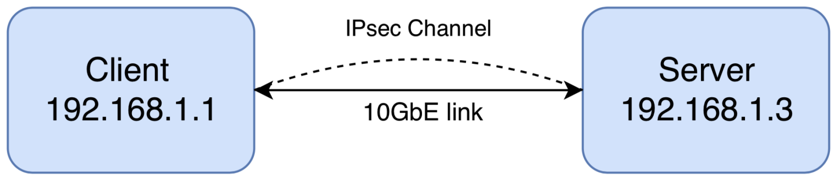


**Figure 6. Network Diagram of Ethernet 10GbE with Connect-X 5 Adapter.**

**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 is used for this entire study.

**Kernel Bypass 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, whether using RDMA or raw Ethernet packets. A typical application written with InfiniBand verbs implements the following procedure:

1. Get the 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, a custom kernel bypass program was written to demonstrate the potential effects that a security implementation might have on an InfiniBand network via a raw Ethernet client/server model. In this model the client sends prebuilt TCP/IP packets to a server. The server receives 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.

This program is used in two of the three following case studies as shown in Table 2. Case Study 1 demonstrates the ability to monitor traffic that 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 implementing IPsec on a program that does not bypass the kernel.

| Case Study: | InfiniBand Client/Server | iPerf |
|---|---|---|
| 1: Traffic Monitoring | X | |
| 2: IPsec Implementation | X | |
| 3: Network Performance | | X |

**Table 2. Overview of Programs used in Case Studies.**

### 3.1.2 Case Study 1: Traffic Monitoring

The first case study examines 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 or exiting 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 should 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. The first test uses Tcpdump in an attempt to capture the TCP/IP packets being transmitted. The second test enables 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 report each successful message transmission to the user by polling the Completion

Queue. If the program is not successful in bypassing the kernel, the TCP/IP packets are captured by tcpdump and can be opened and analyzed in Wireshark without utilizing hardware offload. These two tests (with and without offloaded traffic sniffer) execute the following steps:

#### 3.1.2.1 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.

#### 3.1.2.2 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 would only be captured with the Offloaded Traffic Sniffer enabled.

### 3.1.3 Case Study 2: Implementation of a Network Security System on InfiniBand Verbs.

This case study examines the impact of implementing a network security system on an InfiniBand program. For traditional Ethernet networks, there are many network security systems capable of monitoring and controlling 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 and anomaly based applications that monitor network traffic at the host as well as dynamically monitoring a system state
- **Deep Packet Inspection:** A method of 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 access (IPsec, SSL, and SFTP are all examples)

A similarity between the previously listed security systems is that they are commonly implemented as kernel modules that enforce security policies based upon information found within the kernel. This raises the question, “How can these systems enforce security on packets that bypass the kernel?”

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 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 receives packets from an application 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 is used. The InfiniBand program that is 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 would be in the form of Encapsulated Security Packets (ESP), and packet examination would reveal the encryption. This demonstrates a successful execution of IPsec because the TCP/IP packets are now encrypted with the configured algorithm.

### 3.1.4 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 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 the demo application for this test because it replicates the client/server model and sends TCP/IP packets in the same manner as the Raw Ethernet Client/Server program. Using a 10GbE cable, iPerf produces 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 are run in random order: five with Ipsec and five without Ipsec. Each test records 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 analyzes to determine the performance effect of executing IPsec on an InfiniBand network.

## 3.2 Results

This section presents the results of the three Case Studies that were performed along with the implications of each on overall network security in an InfiniBand network.

### 3.2.1 Case Study 1: Results

This study was designed to explore the security implications of bypassing the kernel with an InfiniBand program. In particular, this study examined the effect bypassing the kernel had on the ability to monitor InfiniBand traffic. The first test of this study used the network analyzer tcpdump to attempt to capture packets on the server machine.

Although the test successfully registered message completions back to the server and client sides of the program, the test resulted in exactly zero packets captured on the specified interface using tcpdump/Wireshark with the Offloaded Traffic Sniffer disabled. This indicates 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). This implies that, to be successful, any effort to monitor InfiniBand traffic must be executed outside of the host machine's kernel.

The second test in this study follows naturally from the first, and seeks to monitor InfiniBand traffic outside the host machine's kernel. After enabling the Offloaded Traffic Sniffer, the InfiniBand Client/Server program was executed again. This time, the .pcap file recorded by tcpdump did capture packets. A screenshot of the first five packets analyzed in Wireshark can be found in Figure 7.

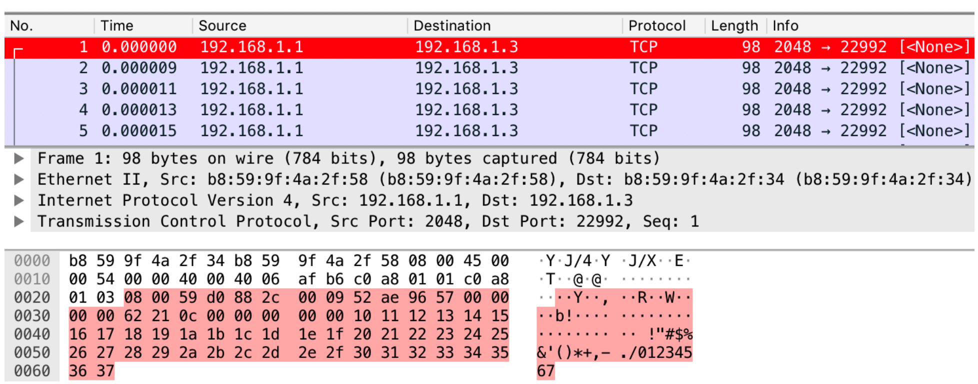


**Figure 7. Wireshark Analysis of Captured InfiniBand Packets.**

As seen in Figure 7, the exact pre-formated TCP/IP packets created by the Infini-

Band 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 limitation 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.

### 3.2.2 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 7 from Case Study 1, illustrating the successful transmission of TCP/IP packets. The highlighted data section of the packet is sent in plain-text 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 8.

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 demonstrates that because the IPsec is executed in the kernel stack, and the InfiniBand Client/Server

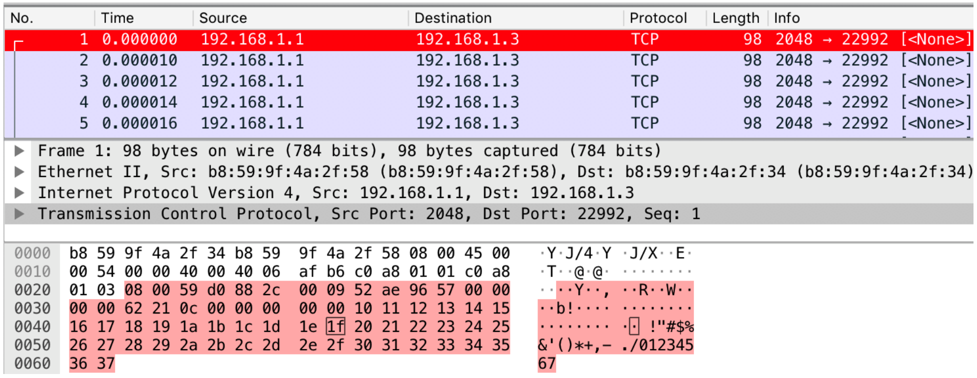


**Figure 8. Wireshark Analysis of InfiniBand Packets with IPsec.**

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.

### 3.2.3 Case Study 3: Results

After determining that IPsec could not be implemented on an InfiniBand program written with InfiniBand verbs to bypass the kernel, the next logical step is 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 is a TCP/IP application that requires the use of the Upper Layer Protocol IPoIB. Because it requires IPoIB, the generation of iPerf's TCP/IP packets occurs within the kernel stack. Because the implementation of IPsec takes place within the kernel, IPsec can be executed on the packets being transmitted by iPerf.

| 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 |

**Table 3. Case Study 3 results.**

Based on the results in Table 3, 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 that limit network performance. A possible solution may reside in hardware. Offloading IPsec processes to hardware may reduce 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. Together, these case studies demonstrate that if a traditional 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.

### 3.3 Conclusion

This chapter explored the implications and limitations of securing an InfiniBand network with traditional Ethernet practices by conducting three case studies. 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. The three case studies concluded 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 capable of overcoming the hurdles found in this work: a hardware offloaded security system.

# IV. Hardware Security Solutions

## 4.1 Objective

The results of the previous case studies motivate the need to tailor network security systems specifically for InfiniBand when traditional Ethernet practices are not sufficient. This chapter provides an assessment of hardware devices that have the potential to secure an InfiniBand network via a hardware offloaded security system. The desired security requirements are discussed and determined based on the NIST Framework. A procedure is presented that describes the exploration approach taken for each selected device. A description of all hardware devices as well as their theoretical implementations are presented and compared against one another. Existing device security features and limitations are evaluated to find potential InfiniBand security solutions. The implications of these findings are then analyzed based on the defined requirements to guide future research in an InfiniBand hardware offloaded security system.

## 4.2 Possible Technology

ASIC devices offer extremely high performance combined with low power consumption providing the advanced capabilities needed for the IBA. When considering possible solutions for an offloaded security system, it is intuitive to consider ASIC devices as all InfiniBand HCAs are implemented as ASIC devices. However, the flexibility of an ASIC device is limited due to the pre-defined functions of the device that prevent certain work offloads (such as a security system) from being implemented [25]. Because of this, similar technologies such as FPGAs and System on Chips (SOCs) are considered as they offer advantages in ease of programming and flexibility while providing similar performance. Two FPGA devices are explored which offer high performance, open programmability, and can theoretically implement any type of functionality within the constraints of the available gates [25]. Despite its advantages, FPGAs are notoriously difficult to program as they use low level HDLs rather than high level software programming languages. HDLs require the user to describe the structure and behaviour of digital circuits synthesized as hardware whereas software programming languages abstract low level implementation to describe sequences of logical and mathematical expressions executed by the CPU. Thus, a second technology is used to explore a hardware offloaded security system: SOC. SOCs, programmable with common high level programming languages, offer the highest flexibility with similar performance. This accessible programming model eases the development of customized applications.

### 4.3 Requirements

The NIST Cybersecurity Framework provides five fundamental functions that help guide a successful and holistic cybersecurity program [12]. Thus, these functions are used to determine the capabilities required to secure an InfiniBand network. The three functions taken from the NIST Framework to define what it means to secure an InfiniBand network are Protect, Detect, and Respond. The omitted functions, Identify and Recover, are out of scope of this effort, as the focus of this research relates to a network security system, not an entire cybersecurity program. These two functions provide a high-level, strategic view of an organization's management of cybersecurity risk and cannot be confined to an individual security system.

**Protect** The Protect Function is used as a guideline to limit and/or contain the impact of potential cybersecurity events. Whether these events are malicious or anomalous, it is essential that a hardware device is able to Protect InfiniBand network traffic against them. In particular, the cybersecurity outcome of the Protect Function that we will focus on is Data Security [12]. Data Security protects the confidentiality, integrity, and authenticity of all information transmitted and received. Thus, a device that is capable of securing an InfiniBand network against malicious cyber events must protect the confidentiality, integrity, and authenticity of network traffic.

**Detect** The Detect Function provides the development and implementation of required steps to identify cybersecurity events in a timely manner. A timely discovery of cybersecurity events ensures the potential impact of the threat is understood so that appropriate measures are taken. In order to Detect cybersecurity events on an InfiniBand network, a device must be able to inspect/monitor network traffic both to and from the host machine. A device with this capability provides the necessary security desired to Detect malicious and anomalous events on an InfiniBand network.

**Respond** After detecting the occurrence of a cybersecurity event, the Respond Function defines the appropriate activities required to contain the potential cybersecurity incident. For securing an InfiniBand network, the desired response is Mitigation. By mitigating the detected cybersecurity event, the expansion of the event and its associated effects are prevented, resolving the incident [12]. For a security device to mitigate an anomaly, it must have the ability to filter, stop, or manipulate network traffic to contain/mitigate the newly identified vulnerabilities. This capability allows for the offloaded security system to take the appropriate actions.

**Inline at Line Rate** In addition to providing the three security requirements derived from the NIST Framework, securing an InfiniBand network must be done inline at line rate. For this research, *line rate* indicates that the network security system does not limit the rate at which bits are transmitted. InfiniBand is utilized primarily for its high speed, low latency capabilities. Thus, an inline implementation not at line rate would conflict with the desired performance provided by the IBA. A security device able to Protect, Detect, and Respond to cybersecurity events inline, at line speed provides efficient use of the CPU resources and can be implemented into a network application without affecting the network performance.

## 4.4 Exploration Approach

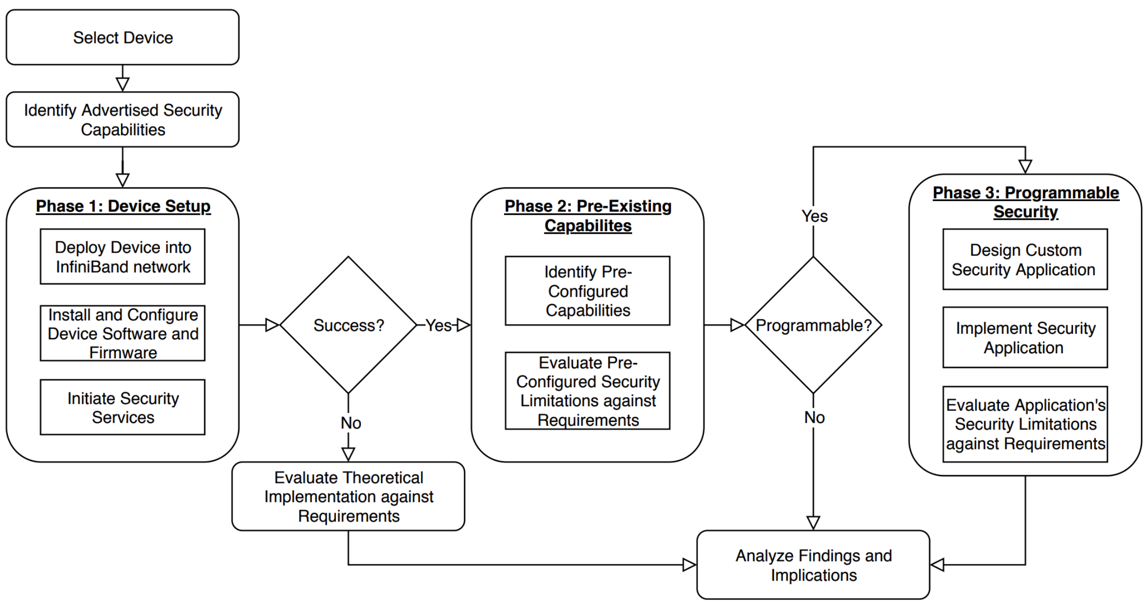


**Figure 9. Security Device Exploration Method**

For each selected device, a procedure similar to the one in Figure 9 is conducted and is broken down into three distinct phases. Before Phase 1 begins, a hardware device is selected and the available security capabilities advertised by its manufacturer are identified. Three hardware devices manufactured by Mellanox were selected for this research. Phase 1 describes the Device Setup and begins by deploying the selected device in a suitable environment and configuration within an InfiniBand network. If the product is properly supported by the vendor, the correct and updated software/firmware are then installed and configured enabling the initialization of the security services provided. If the security services can not be initialized successfully, further exploration into the device is halted and the device's theoretical implementation is evaluated against the security requirements instead. Successful initialization of the services leads to Phase 2 of the exploration approach. The device's pre-existing security capabilities are identified and then exercised. The device's security capability is then evaluated on its ability to meet the requirements for securing an InfiniBand network. If the device is not programmable, the findings and implications of the device are analyzed and exploration is halted as its security limitations have been reached. If the device is programmable, the security limitations have not yet been reached and the exploration approach transitions to Phase 3, which begins with the design of a custom security application. The custom security application is developed to meet the Protect, Detect, and Respond requirements of securing an InfiniBand network. The *inline at line rate* requirement is not considered during development as it relies on the hardware to provide this capability. Once developed, the security application is implemented and its security limitations are evaluated. The findings and implications associated with the selected device are analyzed and the exploration is complete.

## 4.5 Hardware Accelerated Security Protocol

The first hardware device examined is the Mellanox Innova IPsec Adapter. This device was selected because it comes pre-configured to offload the established IPsec security protocol to hardware. This device uses the Connect-X 4 HCA to provide the InfiniBand network capabilities and a bump-in-the-wire FPGA to offload the IPsec security protocol. This device was developed to combat the growing concern for security and privacy in large data centers. In fact, Mellanox states, "Growing concerns over traffic interception, as well as the collection and use of unencrypted information, have kindled a global desire for privacy protection. This has led to a massive increase in the use of encryption to protect data-in-motion and data-at-rest in the data center" [26]. Encryption is progressing as the standard in cloud-based applications as it provides both confidentiality and integrity to the data being shared. Unfortunately, encryption is very CPU intensive. Oftentimes, encryption requires more CPU resources than the actual application, which limits overall network performance. The results from Case Study 3 confirm this. Additionally, software/CPU based encryption cannot scale to meet the growing demand and speed needed to process the data [26]. Implementing an offloaded security system on the Innova IPsec has the potential to overcome this hurdle.

The Innova IPsec Adapter is capable of offloading the computationally intensive encryption and authentication tasks from the CPU to its FPGA-based AES-GCM cryptographic engines [27]. By doing so, the Innova IPsec Adapter eases network bottlenecks by freeing valuable CPU resources which allows the execution of the application to be the focus of the processor. The architecture used to accomplish this task is very unique as the FPGA offloads the IPsec protocol via a "bump-in-the-wire" architecture. Other IPsec acceleration devices are built using look-aside architectures comprised of a single CPU and a PCIe hardware encryption accelerator[26]. Packets are moved from the CPU to the accelerator for encryption/decryption, then back to the CPU to either be received by an application or transmitted on the network. This is neither simple nor efficient as the CPU is now responsible for multiple flows of network traffic, consuming unneeded CPU resources, and limiting overall network performance. In contrast, the Mellanox Innova IPsec's bump-in-the-wire architecture is efficient and simple as the packets being transmitted/received are encrypted/decrypted as the packets pass through the network card. As seen in Figure 10, the encryption and decryption of the IPsec packets occurs inline with the network flow allowing the ConnectX-4 HCA to provide the InfiniBand network services with the IPsec encrypted packets This unique feature in addition to protecting the confidentiality, integrity and authenticity of InfiniBand traffic is why the Mellanox Innova IPsec was chosen as one of the devices to be evaluated as a potential solution to securing an InfiniBand network [27].

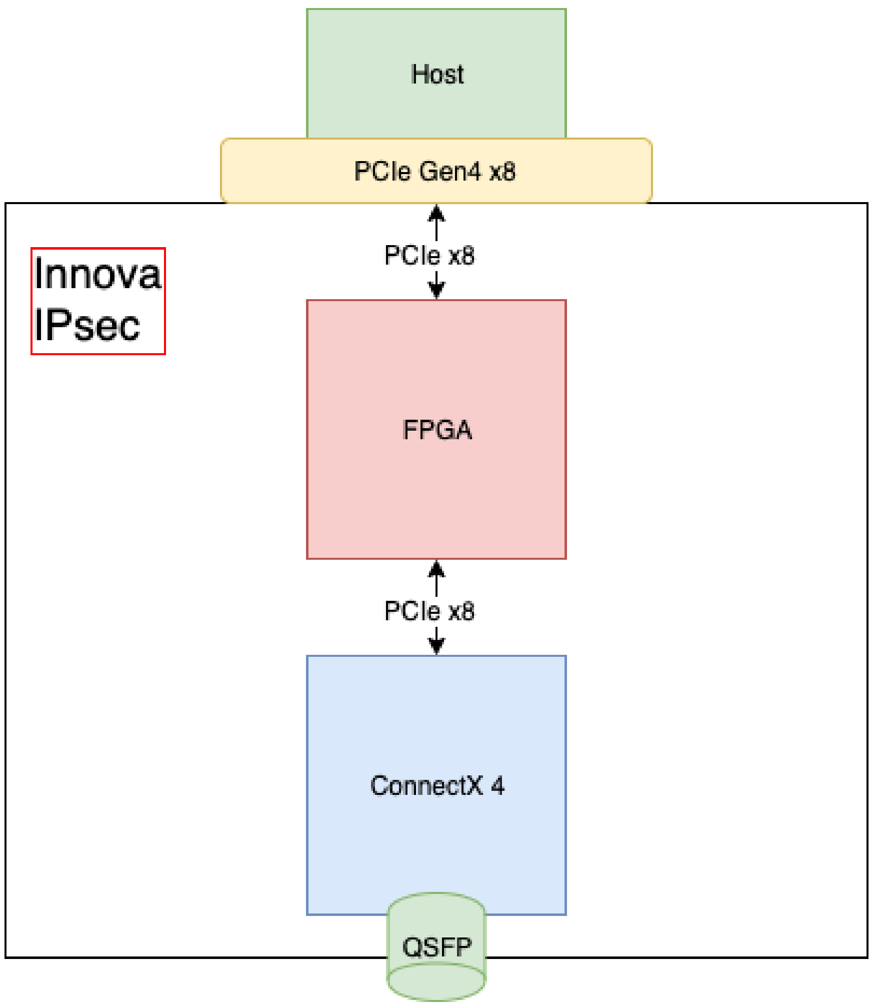


**Figure 10. Mellanox Innova IPsec Adapter**

### 4.5.1 Procedure

This section explores security capabilities and limitations of the Mellanox Innova IPsec adapter.

Like standard Mellanox HCA devices, the Innova IPsec is a PCIe device. Thus, the first step in this exploration was to deploy the device into an operational InfiniBand network and ensure the Mellanox drivers were correctly installed on the host machine. The proprietary device drivers for the Innova IPsec allow user interaction to the unique functionality available both by the Connect-X 4 and FPGA. After the hardware and drivers were installed, the appropriate FPGA and HCA firmware images are loaded. The FPGA image contains the digital logic that implements the IPsec cryptographic engines provided by Mellanox. The firmware image enables the Connect-X 4 to communicate and interact with the FPGA. To accomplish the *burn and load* of images, Mellanox's Mellanox Software Tools (MST) service will be used. Mellanox documentation refers to *burn* as the act of writing a firmware/binary image to a flash device. Furthermore, the documentation refers to *load* as the selection of an image from the flash device to be used by the targeted device. All Mellanox devices (HCAs, FPGAs, switches, etc.) have associated flash devices to which firmware and binary images are burned. Once burned onto the flash device, the firmware or binary image are loaded from flash onto the device. The MST service is a set of management and debug software tools that allow users to interact with InfiniBand devices. In particular, the MST flint tool is used to accomplish the burn of the firmware image onto the Connect-X 4 HCA and the mlx_fpga tool is used to burn FPGA image. The burn of the FPGA image is carried out first.

When initiating the MST service on the host machine and querying for InfiniBand devices, only the Connect-X 4 appeared, not the FPGA. The MST service was then restarted using an additional argument *–with_fpga*. By doing so, the MST service uses an additional driver to interact with the FPGA on the adapter: the FPGA_tools driver. Essentially, this additional device driver is utilized to interact with the FPGA on the Innova IPsec adapter and is discussed further in this chapter. After restarting the MST services with the *–with_fpga* argument and querying devices, the FPGA device was still not appearing. The reasoning for this was that the Innova IPsec was in a recovery mode as the factory image was not properly loaded preventing normal access to the FPGA. To overcome this, the MST service was once again restarted with the *–with_fpga_fw_access* argument. This argument allowed the MST service to load the Factory Image to the FPGA via firmware rather than the FPGA_tools driver. After the successful restart of the MST service with the *–with_fpga_fw_access* argument, the FPGA device was now successfully populating when queried. As mentioned previously, the Innova IPsec was in a recovery mode and needed the Factory Image loaded onto the FPGA. Using the mlx_fpga tool, the Factory Image was now loaded onto the FPGA. After restarting the MST service with the *–with_fpga* argument, the ability to read/write to the FPGA on the Innova IPsec was available. To enable the IPsec offload, the provided IPsec FPGA image is burned and then loaded. The standard burn of the IPsec image uses RDMA rather than I2C as the means to burn FPGA images as it is much more efficient. However, due to the initial recovery mode of the FPGA, the RDMA functionality was not available and I2C was utilized to accomplish the burn. The burn of the IPsec image took roughly three hours, but was successfully loaded to the FPGA offloading the IPsec protocol.

To ensure successful loading of the IPsec protocol to the FPGA, a secure connection similar to that in Case Study 3 is utilized. The difference between the two connections is a custom version of Iproute2 provided by Mellanox. This version of Iproute2 exposes new flags to the user that provide the option to offload IPsec security associations to the Innova IPsec. Now when opening an IPsec secure connection, the following flags will be used to specify the desired offload device and the direction:

$$offload\ dev\ <device>\ dir\ in/out$$

After creating an IPsec secure connection, the crypto offload parameters were checked to verify that the IPsec protocol was indeed offloaded to the Innova IPsec adapter. The crypto offload parameters were checked by querying the state of the ip xfrm utility (the utility that Iproute2 uses to open an IPsec connection). These parameters indicate the state of the offload device. Upon inspection, the parameters were not present. This indicated that the encryption/decryption of the IPsec protocol is not offloaded and is still being performed in the kernel.

The initial thought to overcome this challenge resided with the firmware of the Connect-X 4. The current firmware image for the Connect-X 4 on the Innova IPsec was not the one included with the IPsec FPGA image. This is potentially because the IPsec protocol was not successfully offloaded to the FPGA and why the RDMA burn could not be utilized to load the IPsec image. The firmware provides the HCA the appropriate set of instructions on how to communicate and interact with other hardware devices so the incorrect firmware image would prevent the HCA from communicating with the "bump-in-the-wire" FPGA. Using MST flint (the Mellanox tool used to update and burn firmware on HCAs), the Connect-X 4 was flashed with the provided firmware image in the IPsec bundle. In the event that the new firmware image enabled the RDMA burn functionality, the burn of the IPsec image to the FPGA was re-attempted. Despite the new firmware image, the RDMA functionality was still not available and the IPsec burn accomplished with I2C. After the burn and load of the IPsec image onto the FPGA, the loaded image was queried to verify its success. Previous queries before this displayed that the User image was loaded; instead, the current image now said “Factory Failover Image”. Despite this error message, an IPsec secure connection was opened to offload the encryption/decryption IPsec protocol. The state of the offload device was again inspected, and the crypto offload parameters were still not found indicating the IPsec protocol was not offloaded from kernel space. Because the initialization of security services were unsuccessful, further exploration was halted and the theoretical implementation will be examined instead.

### 4.5.2 Findings and Implications

The exploration of the Mellanox Innova IPsec proved unsuccessful in offloading the IPsec encryption and authentication tasks. This appears to be a result of compatibility issues between the IPsec FPGA image and the Connect-X 4 firmware image. Each time a new firmware or FPGA image was loaded, a factory failover error was displayed indicating a corruption with one of the images. Due to the discontinuation of the product, additional FPGA/firmware images that might work with the current device were not available. Although the lack of vendor support prevented successful exploration of the capabilities of the Innova IPsec adapter, the theoretical implementation merits discussion.

The key attribute of the Innova IPsec is its ability to protect network traffic at line rate via encryption and authentication. Encrypting network traffic protects the confidentiality, integrity, and authenticity of all data that passes through the adapter. By doing so, the Innova IPsec can secure internal networks from unauthorized access and eavesdropping replacing/supplementing the need for perimeter security [26]. This is a key implication as it allows the deployment of an InfiniBand network outside the typical HPC environment while still providing security. Furthermore, the Innova IPsec provides this capability at line rate. As shown in Case Study 3, IPsec execution in the CPU drastically impacts the network performance of an InfiniBand network reducing the overall throughput to 27.3% of the original. To combat this, the Innova

IPsec offloads the AES-GCM encryption/decryption and authentication algorithms necessary for IPsec to ease the burden of the CPU improving network performance. In fact, a test performed by Mellanox demonstrated that the Innova IPsec could achieve nearly the same network throughput as a Connect-X 4 would without encryption [26]. This accomplishment is ground breaking in terms of securing an InfiniBand network as it combines the advanced network capabilities of the IBA while protecting all network traffic from potential threats at line rate. Although these attributes meet the *Protect* and *inline* requirements for securing an InfiniBand network, further exploration is needed.

### 4.6 Programmable SmartNIC via FPGA

The next device explored is the Mellanox Innova-2 Flex. The Innova-2 Flex is a programmable SmartNic that combines the network capabilities of the Connect-X 5 HCA with a fully programmable "state-of-the-art" FPGA that can be dedicated to user application logic [28]. The FPGA is a Xilinx KU15P FPGA with 520K LUTs, 70Mb of internal RAM and 1970 DSP blocks. It allows the implementation of top-of-the-line offload acceleration engines delivering over 100Gb/s of throughput capable of meeting the most demanding offload tasks [28]. This device was chosen because it e. Gilad Shainer, vice president of Marketing at Mellanox Technologies stated that "the Innova-2 product line brings new levels of acceleration to Mellanox intelligent interconnect solutions ... equip[ing] our customers with new capabilities to develop their own innovative ideas, whether related to security, big-data analytics, deep learning training and inferencing, cloud and other applications. The solution allows our customers to achieve unprecedented performance and flexibility for the most demanding market needs" [29]. Inspired by its predecessor the Innova IPsec, the Innova-2 Flex offers flexible usage models through the use of both "bump-in-the-wire" and "look-aside" architectures. As seen in Figure 11, the FPGA is connected to the host via an embedded PCIe switch allowing the FPGA to be visible by the host as a PCIe device. In theory, the embedded switch can be utilized in such a way to provide "bump-in-the-wire" and "look-aside" architectures permitting more than just data encryption/decryption.

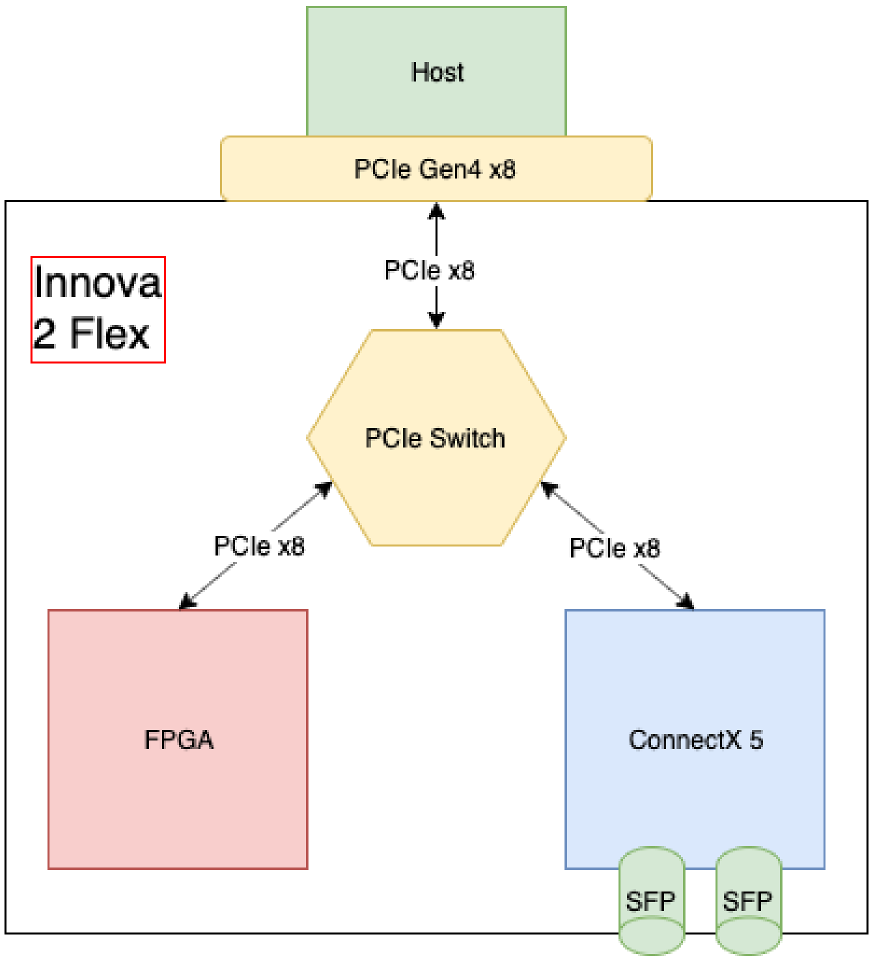


**Figure 11. Mellanox Innova 2 Flex**

In addition to being fully programmable, this device is also available with pre-programmed security encryption offloads such as IPsec and TLS/SSL [29]. The availability of pre-programmed applications combined with the potential to create custom applications is a defining characteristic for the selection of this device as it allows the seamless combination of encryption offloads with custom made applications. The first device examined, the Innova IPsec, is not fully programmable, thereby limiting the security capabilities to only *protecting* traffic inline at line rate as a custom FPGA image containing additional capabilities to secure an InfiniBand network cannot be developed. By dedicating all FPGA resources solely to the user's application, the Innova-2 Flex has the ability to implement security in more ways than software encryption/decryption, including network-distributed Denial-of Service (DDoS) protection, traffic monitoring and more [28]. The limitations of these security capabilities are examined in the subsequent section.

### 4.6.1 Procedure

The exploration of the Innova-2 Flex occurred in two distinct areas according to the Exploration Approach: the Innova-2 Flex Application (Pre-existing Capabilities) and a custom application that interacted with the FPGA (Programmable Security). The Innova-2 Flex application was originally explored to understand how the device drivers interacted with the FPGA and HCA and to identify and evaluate its pre-configured capabilities. The second area of exploration dealt with communication to and from the on board FPGA by developing a security application based on the defined security requirements.

#### 4.6.1.1 Pre-existing Capabilities: Innova-2 Flex Application

Included with the Innova-2 Flex Bundle is the Innova-2 Flex Application. The application is used to perform FPGA and board diagnostics, burn custom user images to the FPGA, and determine whether the Innova-2 Flex or User image will be loaded onto the FPGA. For the Innova-2 Flex Application section, the Innova-2 Flex image is loaded onto the FPGA as it is needed for diagnostic capabilities. The application uses two device drivers to interact with the board which create device files that serve as the actual interface for applications to access the hardware's functionality: the *bope* device file and the I2C device file.

The *bope* device file allows access to the Innova-2 Flex Image's diagnostics capabilities via the PCIe bus. The Innova-2 Flex's diagnostic capabilities are as follows:

- **PCIe Test**: exercises the PCIe interface between the host and the FPGA.

- **DDR Stress Test**: data, which can be either 1s, 0s or pseudo-random, is written to incremental addresses until every DDR address is written to. The test then reads back the sequence and compares it to the expected sequence. The test continues until terminated by the user.

- **Single Test**: writes data all over the DDR space and validates that data is written properly.

- **Query FPGA Version**: reads the Innova-2 Flex Image FPGA version and presents it to the user.

To determine how the device drivers interact with the Innova-2 Flex Image and what pre-existing capabilities exist on it, the code for the Innova Flex application was manipulated. Doing so demonstrates the ability to read/write to/from the Innova-2 Flex image using the application and the device driver without changing the FPGA image. By changing the application code based on the device files generated by the drivers, the successful alteration of the diagnostic tests enabled reads and writes to the addresses on the FPGA and the DDR memory. Although this demonstrated the knowledge of how the device drivers interacted with the Innova-2 Flex application, the information gained from this was minuscule. Further exploration into the *bope* driver indicated the diagnostic tests for the Innova Flex Image burned onto the FPGA were the only pre-existing capabilities available. Thus, the only finding from the exploration of the *bope* driver was an insightful understanding of the interaction between device drivers and host based applications as there were no pre-existing security capabilities.

The I2C device file is the interface to the FPGA_tools driver which provides I2C communication with both the Connect-X 5 and FPGA on board. The Innova Flex Application uses this device file to read the thermal status of both the FPGA and the Connect-X 5. Additionally, the application controls the fan speed and the power consumption of the FPGA via the I2C device file. Because the Innova-2 Flex claims to come pre-configured for IPsec and TLS/SSL security offloads, the FPGA_tools device driver was exmained to see what potential capabilities existed and how to potentially implement the pre-configured offloads.

Upon examination of the FPGA_tools driver, it was discovered the header of the device driver includes a file named "ipsec.c". This finding was promising as it suggests that an IPsec offload was possible via this device driver. After further investigation, it was determined that the FPGA_tools driver used by the Innova Flex Application was identical to the device driver for the Innova IPsec adapter. The utility Iproute2 used this device driver to offload network traffic to the FPGA on the Innova IPsec adapter and communicate with the HCA. This is important because it demonstrates the ability for a device driver to control how the FPGA communicates between an HCA. Communicating between a host and an FPGA and communicating between a host and an HCA via the PCIe bus is trivial; however, communication through an FPGA to an HCA via the PCIe bus (bump-in-the-wire) is not. A possible solution to accomplish this feature theoretically resides in the multitude of InfiniBand device drivers that interact with the FPGA_tools driver. However, an attempt to utilize the FPGA_tools driver to communicate back and forth between the FPGA and HCA via the Innova-2 Flex would require an FPGA image with the exact functionality of the Innova IPsec but configured for this specific FPGA on the Innova-2 flex. In an attempt to obtain an IPsec image for the Innova-2 flex that would hopefully shed light on the "bump-in-the-wire" communication, the vendor, Mellanox, was contacted.

Mellanox confirmed that there is no FPGA image pre-configured for an IPsec offload for the Innova-2 Flex. Additionally, they reported that Innova-2 Flex is not a "bump-in-the-wire" architecture and any FPGA logic would have to be implemented as a look-aside application. Otherwise, to achieve a "bump-in-the-wire" architecture, the Innova-2 Flex would require the embedded PCIe switch to be used by the FPGA to allow network traffic to flow to the FPGA in line. Thus, the creation of custom FPGA logic implementing a security capability would require the FPGA to control the flow of network traffic both to and from the HCA. This would require a new device driver as the FPGA_tools driver does not possess this ability. Additionally, this device driver would have to be compatible with all other InfiniBand device drivers. Due to the difficulty of that effort, this was not attempted within the scope of this research and it was concluded that, in its current configuration, the Innova-2 Flex cannot secure an InfiniBand network inline.

#### 4.6.1.2 Programmable Security: Xillybus

Despite the fact that the Innova-2 Flex cannot be used to secure an InfiniBand network inline, examining possible security applications implemented with a "look-aside" architecture were still explored. The first step to create custom FPGA logic capable of protecting, detecting, and responding to anomalous cyber events on an Infiniband network is to control the flow of network traffic. In order to avoid reinventing the wheel in terms of FPGA/PCIe communication, an existing solution was used: Xillybus. Xillybus is an efficient DMA-based solution for data transport between an FPGA and a Linux host via the PCIe bus [30]. It provides the user the ability to customize FPGA application logic without having to completely recreate device drivers to communicate with the hardware. Both the FPGA application logic and the host-based application interact with one another using common interfaces. The custom FPGA application logic uses FIFOs to communicate with the Xillybus FPGA logic and the host application performs file I/O operations on pipe-like device files [30].

Figure 12 is a simplified block diagram of Xillybus illustrating the flow of data between the host and the FPGA security logic. The Xillybus IP Core communicates with the security logic via Application FIFOs, initiating data transfers to and from the security logic when the FIFOs are ready [30]. The Xillybus IP Core uses the Xilinx PCIe Integrated Block IP Core as its high bandwidth serial interconnect between it and the host over the PCIe bus.

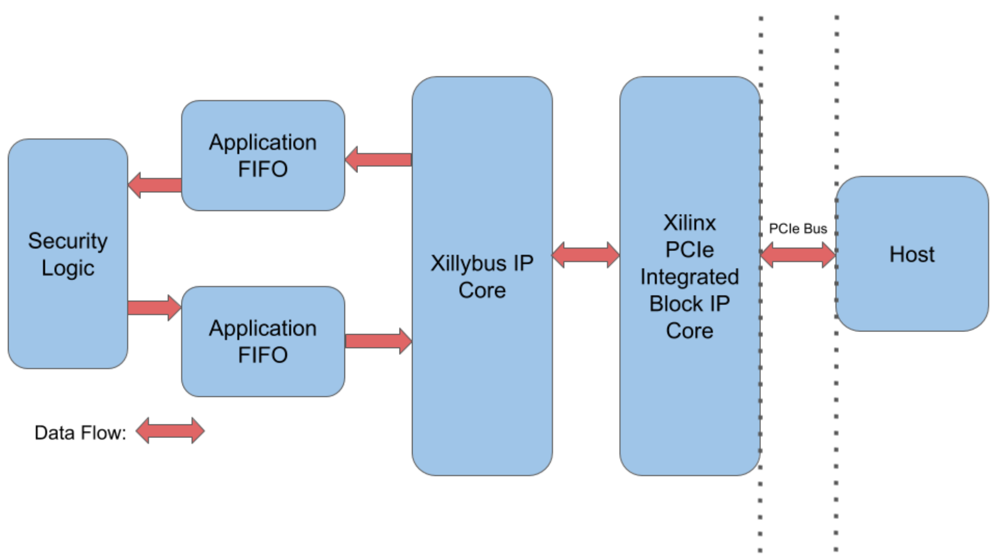


**Figure 12. FPGA Block Diagram for Xillybus**

The device driver for Xillybus produces device files pipes that are written to and read from much like the pipes used in TCP/IP streams [30]. This allows for high-rate data transfers and simplifies programming host based applications. Because we are attempting to model the continuous flow of InfiniBand network traffic, the example application run on the host implements a RAM FIFO for continuous data streams. The program is comprised of two threads, read and write, that continuously stream data to and from the FIFOs on the FPGA. This application was chosen because it would allow a constant flow of data to be sent to the security logic on the FPGA to mimic the high throughput traffic produced by InfiniBand networks.

In an ideal situation, the security logic on the FPGA would be able to accomplish all three of the desired security capabilities: Protect, Detect, and Respond. As a proof of concept however, the security logic for this study is designed to manipulate network data at the bit level and at line speed. Having the ability to manipulate network traffic demonstrates the Innova-2 Flex's potential ability to implement a hardware offloaded security system that can accomplish protecting, detecting, and responding to anomalous cyber events. The example security logic created to demonstrate this, and executed on the FPGA monitors all network traffic in byte sized segments. When a specific hex value of a byte is detected (hex value x"0101"), the logic flips the least significant bit of the byte and continues the transfer. After the completion of this FPGA security logic, a bit-stream was created and burned onto the Innova-2 Flex. Using the example application provided by Xillybus on the host computer, data was successfully written to and read from the FPGA. Additionally, every time the hex value x"0101" was written, the security logic successfully detected the byte and flipped the least significant bit returning the hex value x"0100". The execution of this security application demonstrates the potential to monitor, detect, and manipulate network traffic with the Innova-2 Flex.

Although the security logic was successfully implemented on the FPGA, it still lacked the ability to control the flow of InfiniBand traffic to and from the HCA. As previously discussed, Mellanox informed us that a "bump-in-the-wire" architecture was not achievable with the Innova-2 Flex. Thus, routing any kind of network traffic from host, to FPGA, to HCA (or vice-versa), was impossible. Because of this significant limitation, further exploration of this device was halted after the completion of Phase 3 of the Exploration Approach.

### 4.6.2 Findings and Implications

The exploration into the capabilities of the Innova-2 Flex revealed the ability to monitor, detect, and manipulate network traffic. The ability to customize and burn a user image onto the device was the key advantage over the Innova IPsec as the Innova IPsec was limited to encryption/decryption and authentication of network traffic. The implementation of the security logic onto the FPGA illustrated the security potential the device has. The custom logic was able to monitor inputs, detect specific hex values, and manipulate the bits at line speed. This demonstrated the potential the Innova-2 Flex possesses to Detect, Protect, and Respond to anomalous network activity. The issue however resides with the nature of the architecture of the device. The previous Innova IPsec adapter was a "bump-in-the-wire" architecture which allowed all traffic traversing the HCA to also traverse the FPGA enabling encryption/decryption and authentication to occur inline. Conversely, the Innova-2 Flex does not allow this. Any type of offloaded security system implemented on the FPGA would require the InfiniBand traffic to travel from host, to offloaded security system on the FPGA, then back to the host. This look-aside architecture is not efficient as it would consume large CPU resources and would impact overall network performance. Although the Innova-2 Flex has the capabilities to Protect, Detect, and Respond to anomalous activity on an InfiniBand network, its failure to meet the inline requirement negates the device's attribute of securing an InfiniBand network. Thus, an alternative device will be discussed in the following section that is implemented as a SOC.

## 4.7 Programmable SmartNIC via System on Chip

The last device that considered is the Mellanox BlueField SmartNIC. The BlueField is an intelligent programmable networking engine implemented as a SOC. It has the ability to accelerate security, networking, and storage workloads via SOC offload enabling a more efficient use of CPU resources [31]. This allows the CPU to focus on performing the application tasks rather than processing networking and security tasks. A block diagram of the BlueField SmartNIC can be seen in Figure 13.

The BlueField I/O Processing Unit combines the advanced networking capabilities of the Connect-X 5 with an array of Arm A72 multicore processors into a single SOC. The BlueField incorporates the Arm software ecosystem by offloading a x86 software stack onto the SOC enabling the ability to develop advanced offloaded applications directly on chip [31]. Although the BlueField was developed with many applications in mind, security is the primary discussion. By combining hardware security accelerators with embedded software, the BlueField provides an ideal environment for proprietary security applications. Mellanox states that the "BlueField builds security into the DNA of the data center and enables **prevention**, **detection** and **response** to potential threats **in real time**" [31]. Based on this statement that the BlueField enables the prevention (protection), detection, and response to potential threats in real time (at line speed), it meets our requirements for securing an InfiniBand network.

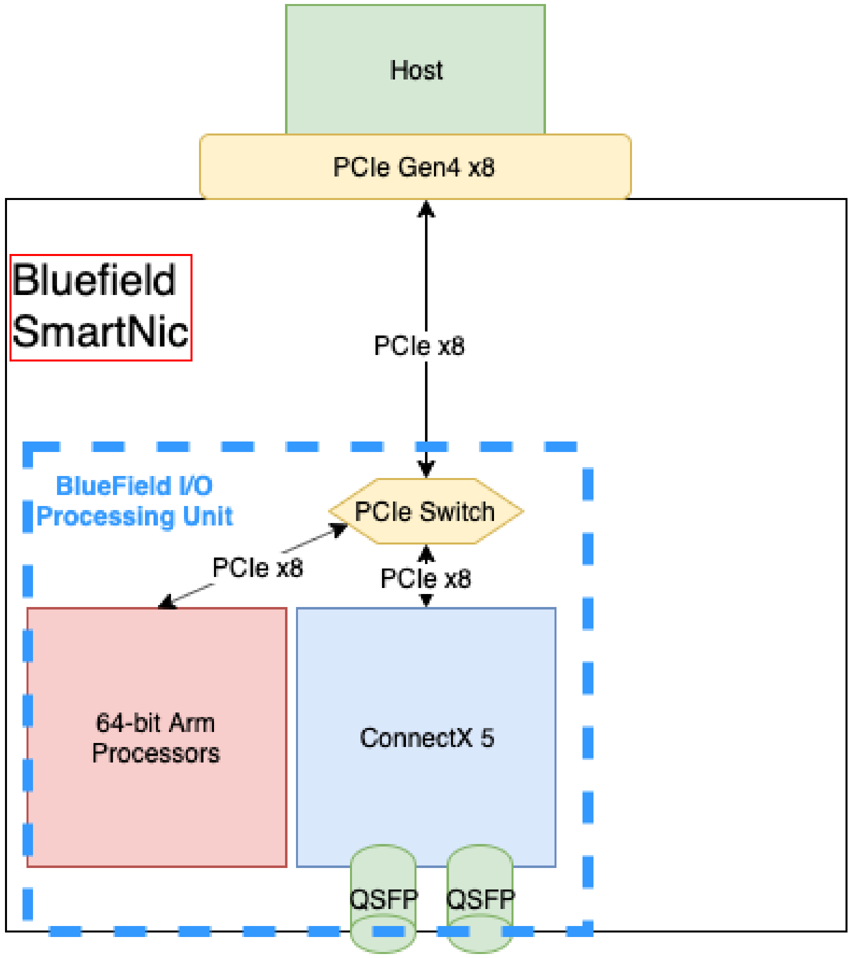


**Figure 13. Mellanox BlueField SmartNic**

### 4.7.1 Future Implications

After working with both the Innova IPsec and the Innova-2 Flex, it was apparent that the direction of offloading a security system for an InfiniBand resided with the BlueField SmartNIC. Unfortunately, this realization was made very late into the exploration of devices and delayed the acquisition of the BlueField preventing a comprehensive examination of the device. Despite this, Mellanox was generous enough to discuss potential future security applications involving the BlueField and its successor the BlueField-2. Thus, the approach to exploring the Bluefield is solely based on its theoretical implementation. Future research with the Bluefield should follow the exploration approach outlined in Figure 9. Based on the suggested goal and the nature of this research, Mellanox has suggested the use of their Security SDK to detect cyber security anomalies in an attempt to secure an InfiniBand network. The Security Software Development Kit (SDK) executes Deep Packet Inspection (DPI) to implement Application Recognition and begins at Layer 3. The Application Recognition uses unique application signatures (regular expression based) to validate traffic and also validates the connections. The DPI engines come with their own parsers removing the need to write new parsers for different signatures. The SDK allows users to write their own signatures in JSON format based on various fields within a particular protocol. These signatures are then compiled using a provided Mellanox compiler and used by the BlueField for DPI. The beta version of the Security SDK has only been implemented using the Ethernet protocol. A future implication of this product would be its employment in an InfiniBand environment securing both IPoIB and RDMA channels. Despite not being able to implement the Security SDK on the BlueField, our research has demonstrated the potential of the BlueField to Protect, Detect, and Respond to anomalous cyber activity at wire speed, ultimately meeting the requirements to secure an InfiniBand network.

### 4.8 Summary

This chapter defined the requirements needed to secure an InfiniBand network, described three possible hardware solutions for implementing an offloaded security system, and explored the security limitations of all three devices. The research discovered that the Innova IPsec was theoretically capable of protecting InfiniBand network traffic at line speed but due to compatibility issues between the firmware and FPGA images, implementation of the device was unsuccessful. The Innova-2 Flex proved to possess the ability to Protect, Detect, and potentially Respond to anomalous cyber events as custom security logic was successfully loaded onto the FPGA on the device. However, the inability for the security logic to be placed inline with the network traffic ruled out the device as an InfiniBand security solution. The last device assessed was the BlueField SmartNIC which uses a SOC based approach to offload custom security applications. Although not executed on the device itself, the Mellanox Security SDK enables the prevention, detection, and response to potential threats in real time meeting all requirements necessary to secure an InfiniBand network. A summary of the devices' capabilities can be found in Table 4.

| **Device:** | **Protect** | **Detect** | **Respond** | **Inline at Line Rate** |
|---|---|---|---|---|
| Innova IPsec Adapter | X | | | X |
| Innova-2 Flex Adapter | X | X | X | |
| BlueField SmartNIC | X | X | X | X |

**Table 4. Hardware Device Capabilities Summary**

# V. Conclusion

## 5.1 Overview

This chapter summarizes the research conducted on an InfiniBand network's security including the deployment and exploration of numerous types of security systems and devices. It reiterates the motivation behind securing an InfiniBand network through a hardware offloaded security system rather than with traditional Ethernet security practices. It discusses the proposed hardware device which will be used to help guide future development of a security system designed specifically for InfiniBand. The chapter closes by discussing the significance of the research performed as well as future work that needs to be conducted to ensure the security of an InfiniBand network.

## 5.2 Summary

This research focused on the security of the IBA and the implications associated with attempting to secure an InfiniBand network outside of its typical HPC environment. It describes how the IBA functions as an advanced interconnect technology and discusses current security features that the IBA possesses. A cybersecurity framework was also introduced to survey possible security capabilities that are desired when securing an InfiniBand network. An example application of InfiniBand in Vehicle Networks was discussed to demonstrate the need for security outside of HPC cluster environments. Previous research involving the IBA and security vulnerabilities was presented, exemplifying the need for further research in this area.

The potential effects of securing an Infiniband network were discussed and analyzed in the form of three case studies. An InfiniBand Client/Server application was also created to illustrate the kernel bypass feature which enables low latency and high bandwidth communication over InfiniBand networks. This program was used in two of the three case studies. Case Study 1 explored the ability to detect and monitor InfiniBand traffic that bypasses the kernel. Case Study 2 implemented a network security system on an InfiniBand network and analyzed the security limitations of it when attempting to secure InfiniBand traffic. Case Study 3 examined the performance impact IPsec produces when executed on a non kernel bypass application.

Case Study 1 identified that monitoring InfiniBand traffic must take place outside of the host machine's kernel. Typical network traffic analyzers used on Ethernet networks were implemented in an attempt to capture TCP/IP packets sent with the InfiniBand Client/Server program. These applications were unsuccessful in capturing the network packets suggesting that network traffic that bypasses a host machine's kernel cannot be monitored by applications that reside within the kernel. Thus, a proprietary hardware device capable of capturing kernel bypass traffic in hardware was employed. This device proved to be successful in capturing the packets produced by the custom application. This case study proved that monitoring and detecting InfiniBand traffic is possible with a proprietary hardware device and implies the need for hardware implementation of traffic monitoring.

Case Study 2 highlighted that common network security systems used on Ethernet networks cannot secure InfiniBand traffic. This case study attempted to encrypt and authenticate TCP/IP packets transmitted by the InfiniBand Client/Server program using the IPsec protocol. Despite its implementation, all packets captured and analyzed remained as plaintext TCP/IP packets. These results illustrate the inability for the IPsec protocol to execute on InfiniBand traffic. Case Study 2 concludes that because IPsec is executed in the TCP/IP stack kernel layer, and the InfiniBand Client/Server Program bypasses the kernel, IPsec cannot protect programs that use InfiniBand verbs. It also suggests that other network security systems executed in the kernel will be unsuccessful and proposes the need for a new type of security system.

Case Study 3 examined the effects of implementing a network security system on InfiniBand traffic that did traverse the kernel stack. After determining that IPsec could not be implemented on the InfiniBand Client/Server program, an alternate program was chosen that did allow IPsec to execute. This program was designed to measure the maximum throughput a device could handle making it ideal for this case study. The results of this study demonstrated that IPsec reduced the overall throughput of an InfiniBand network to 27.3% of the original. Case Study 3 determined the potential security provided by IPsec is not worth the performance impact and suggests that a security hardware offload is the solution to securing an InfiniBand network.

The desired capabilities for a hardware offloaded network security system designed for an InfiniBand network were defined. This research determined that such a system needed to Protect, Detect, and Respond to anomalous cyber events and do so inline and at line rate. To evaluate whether or not potential hardware devices could meet the requirements to secure an InfiniBand network, a methodology was developed. The Exploration Approach was comprised of three distinct phases to evaluate a given device's security limitations. The technology of potential hardware devices were compared among one another to determine the best possible solutions based on their associated strengths and weaknesses. This work selected three hardware devices to explore that had the potential to secure an InfiniBand network.

The first device selected was the Mellanox Innova IPsec adapter. Recall that the Innova IPsec was chosen because it offloads the IPsec protocol onto a "bump-in-the-wire" FPGA allowing encryption/decryption of network traffic to occur inline and line speed. This device would protect the confidentiality, integrity, and authenticity of all network traffic without impacting network performance. The exploration conducted on the device proved unsuccessful as the firmware and FPGA images needed to execute the IPsec offload were confirmed to be incompatible for this purpose. Still, this research concluded that if implemented successfully, the Innova IPsec would be able to *Protect* an InfiniBand network against anomalous cyber activity at line rate. Nevertheless, the device would not be able to fully secure an InfiniBand network as it does not offer the Detect and Respond capabilities.

The second device selected was the Mellanox Innova 2 Flex adapter. The Innova 2 Flex was chosen because it offered an open-programmable platform which could be used to create a wide range of possible security applications improving upon its predecessor the Innova IPsec. The exploration into this device occurred in two parts. The results from the first study indicated that implementing a "bump-in-the-wire" architecture with the on-board FPGA was infeasible thus securing InfiniBand network traffic inline was not achievable. The second study implemented security logic onto the FPGA and was able to successfully demonstrate the ability to monitor, detect, and manipulate network traffic. This work revealed that the Innova 2 Flex did possess the ability to *Protect*, *Detect*, and *Respond* but lacked the capability to do so inline and at line rate.

The last device explored in this research was the Mellanox BlueField SmartNIC. The BlueField provides an ideal environment for security applications by accelerating security, networking, and storage workloads via SOC offload. A delayed acquisition of the device forced this exploration to become theoretical as a network security system was never attempted to be implemented. The proposed network security system to be implemented was Mellanox's Security SDK which would enable the prevention, detection, and response to potential cyber threats in real time via DPI and Application Recognition. This research concludes the future of securing an InfiniBand network resides with the BlueField SmartNIC.

## 5.3 Research Contributions

This research has made a number of contributions to the areas of InfiniBand network security and hardware network security systems. The inability to monitor/-capture kernel bypass traffic with traditional network traffic analyzers demonstrated the need for alternative hardware devices targeted for InfiniBand traffic. Additionally, host-based network security systems used to control and monitor Ethernet networks cannot be implemented on InfiniBand networks as many of these security systems are executed in the hosts' kernel. This contribution suggests a new security system designed specifically for InfiniBand network traffic is needed. Securing an InfiniBand network was defined as the Protection, Detection, and Response to potential cyber threats derived from the NIST Framework. The comparison among the three hardware devices made it evident the BlueField SmartNIC has the potential to secure an InfiniBand network because it can Protect, Detect, and Respond to potential cyber threats in real time. Thus, future development of an offloaded hardware security system should reside with it.

## 5.4 Future Work

Given the rapid development of the IBA and the ever changing cyber threat landscape, there are additional areas that need to be explored and developed. Listed below are select topics that would expand the scope of this research:

- **Security SDK**: As mentioned in Chapter 4, the Security SDK provided by Mellanox was never implemented onto the BlueField SmartNIC. Future research should not only deploy the Security SDK, but evaluate the effectiveness of the Application Recognition capability to determine the security limitations. Additionally, the Security SDK should attempt to secure all types of InfiniBand channels including RDMA, IPoIB, and RoCE to expand the scope of this research across all potential protocols utilized within an InfiniBand network.

- **Software Defined Network Approach**: An SDN approach to securing an InfiniBand network is an area worth exploring due to the SDN nature of the IBA. Forwarding and routing tables are already controlled centrally by a SM thus a centralized network security system would allow an easy transition. Additionally, a centralized network security system would not need its own network protocol as it could use the current InfiniBand control plane protocol simplifying a potential implementation. Future research into an SDN approach could bring security advancements to the control plane of future InfiniBand networks.

- **Machine Learning Security**: The rate at which network traffic flows via InfiniBand is ground breaking. Thus, managing the vast workloads produced by an InfiniBand network to perform security risk assessment on is overwhelming. The use of machine learning could potentially ease this burden and by identifying major risks found within an InfiniBand network to help prioritize security resources [32]. Specifically, Machine Learning could be utilized to implement the GUID spoofing mitigation approach described in [10]. Machine Learning could be used to monitor link state configurations and respond appropriately to anomalous activity detected.

## 5.5 Conclusion

This research demonstrated that network security practices used on traditional Ethernet networks do not translate to InfiniBand networks as previously suggested and that a hardware network security system was needed in order to secure an InfiniBand network. It defined the desired security capabilities of such a system as the

Protection, Detection, and Response to potential cyber threats at line rate which guided the selection of the BlueField SmartNIC as the appropriate device to implement the system with. It is obvious that not all implications of securing an InfiniBand network have been explored as the true potential of the IBA is yet to be determined. As the popularity of InfiniBand continues to grow outside the HPC domain, securing InfiniBand networks should be at the highest of priority for both the HPC and cyber security communities to protect against the ever changing cyber threat landscape.

# Bibliography

1. IBTA, "InfiniBand TM Architecture Specification Volume 1 Release 1.3," 2015.

2. Mellanox Technologies, "InfiniBand Software and Protocols Enable Seamless Off-the-shelf Applications Deployment," *White paper, Mellanox*, no. December, pp. 1–8, 2007.

3. G. F. Pfister, "An Introduction to the Infiniband Architecture," *Emerging Technologies and Future Trends*, vol. Part IX, p. Chapter 42.

4. "List Statistics," 2019. [Online]. Available: https://www.top500.org/lists/2019/11/highs/

5. IBTA, "About InfiniBand." [Online]. Available: https://www.infinibandta.org/about-infiniband/

6. Mellanox Technologies, "Introduction to InfiniBand," *Technical Report*, pp. 1–20, 2003.

7. D. Schmitt, S. Graham, P. Sweeney, and R. Mills, "A Cyber Vulnerability Assessment of Infiniband Networking," Tech. Rep.

8. M. Lee and E. J. Kim, "A comprehensive framework for enhancing security in InfiniBand architecture," *IEEE Transactions on Parallel and Distributed Systems*, 2007.

9. M. Lee, E. J. Kim, and M. Yousif, "Security enhancement in infiniBand architecture," in *Proceedings - 19th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2005*, 2005.

10. A. Warren, "InfiniBand Fabric and Userland Attacks," SANS Institute, Tech. Rep., 2012.

11. K. P. Subedi, D. Dasgupta, and B. Chen, "Security analysis on InfiniBand protocol implementations," in *2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016*. Institute of Electrical and Electronics Engineers Inc., feb 2017.

12. National Institute of Standards and Technology, "NIST CyberSecurity Framwork." [Online]. Available: https://www.nist.gov/cyberframework

13. Mellanox, "InfiniBand: The Production SDN," Mellanox Technologies, Sunnyvale, CA, Tech. Rep., 2012. [Online]. Available: http://www.mellanox.com/related-docs/whitepapers/WP{_}InfiniBand{_}Production{_}SDN.pdf

14. J. Corbet, A. Rubini, and G. Kroah-Hartman, “Linux Device Drivers,” Tech. Rep., 2005.

15. Mellanox, “RDMA Aware Networks Programming User Manual,” Tech. Rep., 2013. [Online]. Available: www.mellanox.com

16. Mellanox Technologies, “Security in Mellanox Technologies InfiniBand Fabrics,” *White paper*, p. 7, 2012.

17. S. Tuohy, M. Glavin, C. Hughes, E. Jones, M. Trivedi, and L. Kilmartin, “Intra-Vehicle Networks: A Review,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 534–545, 2015.

18. W. Zeng, M. A. Khalid, and S. Chowdhury, “In-vehicle networks outlook: Achievements and challenges,” *IEEE Communications Surveys and Tutorials*, vol. 18, no. 3, pp. 1552–1571, 2016.

19. R. Boagey, “Ethernet : the fast track to the connected car,” *Automotive World Ltd*, pp. 36–38, 2014.

20. The Economist, “Mobileye and Intel Join Forces,” 2017. [Online]. Available: https://www.economist.com/business/2017/03/16/mobileye-and-intel-join-forces

21. “The Evolution of EyeQ,” 2018. [Online]. Available: https://www.mobileye.com/our-technology/evolution-eyeq-chip/

22. E. Kadric, N. Manjikian, and Z. Zilic, “An FPGA implementation for a high-speed optical link with a PCIe interface,” in *International System on Chip Conference*, 2012.

23. I. Kuon, R. Tessier, and J. Rose, “FPGA Architecture: Survey and Challenges,” *Foundations and Trends® in Electronic Design Automation*, vol. 2, no. 2, pp. 135–253, may 2008.

24. Pci-Sig, “PCI Express Base Specification Revision 5.0,” pp. 1–704, 2019. [Online]. Available: http://www.pcisig.com/specifications/pciexpress/base3/

25. K. Deierling, “What Is a SmartNIC?” 2018.

26. Mellanox Technologies, “Mellanox Innova ™ IPsec : Achieve Groundbreaking Security for VPN , Data Privacy & Data-in-Motion , while Reducing Total Cost of Ownership ( TCO ),” pp. 1–5, 2018.

27. ——, “Mellanox Innova ™ IPsec Adapter Card,” 2017.

28. ——, “Mellanox Innova $^{\text{TM}}$ -2 Flex Open Programmable SmartNIC,” 2018.

29. ——, “Mellanox Announces Innova-2 FPGA-Based Programmable Adapter Family to Power Next Generation of Cloud, Security, Big Data and Deep Learning Platforms,” 2017.

30. E. Billauer, “Xillybus on a Linux Host.”

31. Mellanox Technologies, “BlueField SmartNIC,” 2019.

32. P. Efstathopoulos, “Cloud Security is Overwhelming. AI and Machine Learning Can Help,” NortonLifeLock Research Group, Tech. Rep., 2019.

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704–0188*



**1. REPORT DATE** *(DD–MM–YYYY)*
26–03–2020

**2. REPORT TYPE**
Master's Thesis

**3. DATES COVERED** *(From — To)*
Sept 2018 — Mar 2020

**4. TITLE AND SUBTITLE**
Implications and Limitations of Securing an InfiniBand Network

**5a. CONTRACT NUMBER**

**5b. GRANT NUMBER**

**5c. PROGRAM ELEMENT NUMBER**

**5d. PROJECT NUMBER**
19G230

**5e. TASK NUMBER**

**5f. WORK UNIT NUMBER**

**6. AUTHOR(S)**
Mireles, Lucas, E., 2d Lt, USAF

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Air Force Institute of Technology
Graduate School of Engineering an Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

**8. PERFORMING ORGANIZATION REPORT NUMBER**
AFIT-ENG-MS-20-M-044

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
Air Force Research Laboratory
2241 Avionics Circle
WPAFB OH 45433-7765
Attn: Steven Stokes
COMM 937-528-8035
Email: steven.stokes@us.af.mil

**10. SPONSOR/MONITOR'S ACRONYM(S)**
AFRL/RYWA

**11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

**12. DISTRIBUTION / AVAILABILITY STATEMENT**
DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**

The InfiniBand Architecture is one of the leading network interconnects used in high performance computing, delivering very high bandwidth and low latency. As the popularity of InfiniBand increases, the possibility for new InfiniBand applications arise outside the domain of high performance computing, thereby creating the opportunity for new security risks. In this work, new security questions are considered and addressed. The study demonstrates that many 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 impose a significant performance loss for the network. The research concludes that not all network security practices used for Ethernet translate to InfiniBand as previously suggested and that an answer to meeting specific security requirements for an InfiniBand network might reside in hardware offload.

**15. SUBJECT TERMS**
InfiniBand Architecture, Network Cyber Security, IPsec

| 16. SECURITY CLASSIFICATION OF: a. REPORT | b. ABSTRACT | c. THIS PAGE | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| U | U | U | UU | 94 | Dr. Scott Graham, AFIT/ENG |

**19b. TELEPHONE NUMBER** *(include area code)*
(937) 255-6565 x4581; scott.graham@afit.edu