DPU implementation of a scalable and transparent security solution for numerous VPN connections

February 7, 2021

Students:
Mounir El Kirafi
11879106
Ilyas Rahimi
12330337

Supervisor:
Cedric Both
Abstract

With the tremendous amount of data in the modern era, there is a need for cost-effective and high speed data processing. A tool applicable in such a situation is the Data Processing Unit (DPU). This research looks at how the Bluefield SmartNIC, a DPU developed by NVIDIA Mellanox, can be used in such an environment. More specifically, we research how the DPU can enable a scalable and transparent security solution featuring Intrusion Detection Systems/Intrusion Prevention Systems (IDS/IPS), supporting numerous VPN connections. We have two main focuses here. The first is that the DPU is transparent to the OS it is connected to. This means that the DPU can process, filter and route all data to and from the host OS, reducing its load and increasing security. The second focus point is scalability. With the cryptographic acceleration and network offloading features of the card, it has the potential to create an efficient and scalable VPN networking solution with security monitoring. First however, we examine the current state of Next Generation Firewalls with regards to the current market leaders. Using the Gartner Magic Quadrant we identify these leaders. Then we look into their current capabilities and how they compare with other technologies. The next phase study is to implement those features with Bluefield SmartNIC, including our IDS/IPS of choice: Suricata. We then create a network simulating a real-world application where many VPN clients connect to the card. The card then has to filter using an IDS/IPS and route all data before it reaches the OS transparently. For testing this setup, we looked at the throughput of the card under different VPN related circumstances. We varied the amount of connections and the encryption types. For the testing of the IDS/IPS implementations, we ran attacks on the system and looked at the response from Suricata. Suricata uses Talos intelligence, including a collection of managed blocking rules, to secure the DPU among with a Security Information and Event Management (SIEM) system, GrayLog. We could see that it properly blocked flooding attacks. From this research, we have concluded that the Bluefield DPU is capable of creating a transparent and scalable security solution with regards to numerous VPN connections, and we have shown how.
Contents

1 Introduction 4
  1.1 Research Question ........................................... 5

2 Related work 5

3 Current state 5
  3.1 IDS/IPS ......................................................... 6
    3.1.1 Palo Alto Networks ........................................ 6
    3.1.2 Fortinet ................................................... 7
    3.1.3 Check Point Software Technologies .......................... 7
    3.1.4 Talos ..................................................... 7
  3.2 DPU solution .................................................... 8

4 Methodology 11
  4.1 Software & Hardware used ..................................... 11
  4.2 Network setup .................................................. 12
  4.3 IDS/IPS setup ................................................ 15
  4.4 Testing ......................................................... 15

5 Results 17
  5.1 VPN network .................................................. 17
  5.2 IDS/IPS implementation ....................................... 22

6 Discussion 23
  6.1 Conclusion .................................................... 24
  6.2 Future work ................................................... 25
  6.3 Ethical considerations ......................................... 25

A Extra results graphs 27
1 Introduction

Two key components of datacenter data processing are networking and security. There is a need for efficiently routed and processed packets, as well as the support for encapsulation and decapsulation for overlay technologies. Furthermore, the data that is delivered and sent should be known to be “safe”. This includes filtering from malicious data as well as networking attacks such as DDOS attacks. To combat this, we can make use of ACLs, firewalls and Intrusion Detection/Prevention Systems (IDS/IPS). These make it possible to create systems with secure VPN connections for example. However, with the large amounts of data that datacenters need to process, this can create a large overhead for networking and security.

Nowadays, datacenter architecture relies heavily on the CPU for the computation of various tasks. Because the CPU is a general-purpose hardware component, it can handle many different processing tasks. However, it cannot excel at any one task as dedicated hardware can. Although many CPU’s currently feature some sort of cryptography acceleration through the Intel Advanced Encryption Standard New Instructions (AES-NI), it is still not as efficient as dedicated hardware. On top of the lack of efficiency through generalization, there is also the environment where everything is implemented in software to run on the CPU. The way to counteract the limitations of the CPU was generally to add more cores. However, at some point, this is not viable anymore in terms of cost. The solution to this is Data Processing Units (DPUs) which can offload the networking and security management from the CPU by using specific hardware accelerators such as for cryptography and offloading network operations. The load on the CPU is then decreased, and the data is processed more efficiently. One other notable type of dedicated hardware should be mentioned here: ASICs. These "Application-Specific Integrated Circuits" are circuits very specifically made for a single purpose. This makes them among the most efficient methods for processing such data, but their specialization also makes them expensive and consequently less cost-effective in some areas.

We wanted to research a scalable and transparent security solution for a large number of VPN connections. The DPU we used for this is the NVIDIA Mellanox Bluefield SmartNIC card [19], a programmable networking engine. Its feature combinations make it an interesting research candidate. The card has hardware accelerators for different types of computations. There are encryption accelerators for the calculations of AES, SHA, RSA, DH, ECC, and more. There is also transport offloading hardware for TCP, UDP, IP, VxLAN, and more, including an embedded hardware switch. A combination of hardware cryptographic acceleration and network offloading already makes it relatively unique and a great candidate. However, it has one more ability we are interested in. The possibility to have the card operate in a transparent manner towards the OS and outside world makes it useful in terms of security, efficiency and ease-of-use. In this mode all traffic is passed through and controlled by the card. This way all network management such as routing and hosting, decryption and encrypting VPN connections can be handled by the card. Moreover, an attacker has an extra layer to get through. All-in-all, in theory, this card decreases the hosts networking, cryptographic and management load while adding security benefits. That makes it a perfect solution for for low cost VPN concentrator where it is possible to add sophisticated security features above it, ideal for connections to public clouds or to encrypt connection between multiple client locations.

For this study, we have implemented a network setup featuring multiple machines creating a simulated real-life network. More details are in the methodology, but from a high-level we have a machine generating VPN clients connected to the card, which handles the networking and cryptography and passes the data through to OS without the OS ever having to know about or deal with the VPN connections. There is also an IDS/IPS system in place, with a dashboard for monitoring.
1.1 Research Question

The main research question is **How can a DPU be used to implement a scalable and transparent security solution for numerous VPN connections?** To answer this, we will be using the following sub questions:

- What networking and security features do network firewall market leaders use?
- How can we support a large number of VPN networks using a DPU?
- How can we implement an efficient networking monitoring and filtering system (IDS/IPS) which is transparent to the OS using a DPU?

2 Related work

In the current era the demand for hardware is enormous. Millions of users and billions of connections around the globe create a demand for the fastest and most secure connections possible. All these demands are fulfilled in modern datacenters.

As mentioned in the introduction, this requires a lot of data processing. And one of the ways to accelerate cryptographic operations is the mentioned AES-NI built into CPUs. Research done by D. Lacković and M. Tomić on the performance of AES-NI [13], indicates that it can increase throughput of IPSec by 62% and OpenVPN by 16% (two different VPN implementations). As noted in the paper, the small increase for OpenVPN suggests that the bottleneck lies elsewhere in the datapath. Further analysis of OpenVPN versus Strongswan IPSec shows a lack of scaling capabilities for OpenVPN. Because it is single-threaded, it is not easy to quickly scale without increasing the configurational complexity. Also, it means that multiple OpenVPN servers have to be used and assigned to different cores, which leads to load balancing concerns with regards to what cores get the most load. In this research [25] it has been found that a specifically designed co-processor for acceleration reduced the encryption time of a 1024-byte frame by 93%, in sigmaVPN. Although this is relative to the performance of the non-accelerated CPU, this does give an indication of acceleration performance.

Another security aspect is IDS/IPS. The two main implementations for us are Snort and Suricata. Suricata is a multi-threaded and adapted version based on Snort. As mentioned in the study since Suricata uses multi-Threading it drops less packages above 10Gbps, because in this study we will be using more and more connections and higher speeds is it a very important point. Consequently, according to various research it performs better, generally 10-15% as well as having a smaller memory footprint [23][27].

Research into the use of FPGA-based SmartNICs for acceleration found that there is a "3-5× improvement in terms of average and worst case latency as well as a sixfold increase in throughput when employing an FPGA-based SmartNIC" [14][8]. Although this is referring to the Azure SmartNIC, since it falls in the same category with similar hardware implementation we can extrapolate these findings to our case.

3 Current state

To get an understanding of NGFW Next Gen Firewall techniques, we looked at the market leader of the year 2020 according to Gardner Magic Quadrant [11], there were three players. Namely Palo Alto Networks, Fortinet and Check Point Software Technologies, as stated in the figure below. Not to forget Cisco is being a market leader for past years [22] but in 2020
it has moved from leaders to challengers. The following step is to understand their power and try to implement those techniques in the DPU.

**Figure 1. Magic Quadrant for Network Firewalls**

![Magic Quadrant for Network Firewalls](image)

Source: Gartner (November 2020)

**Figure 1: Market leaders Gartner Quadrant [magic quadrant]**

### 3.1 IDS/IPS

One common rule in all of the firewall market leaders is that they have huge intelligence networks to give the system admins the best security in the easiest way. Besides, they all want to analyze the whole infrastructure in the easiest way. In the next chapter you will see some unique features of all vendors.

#### 3.1.1 Palo Alto Networks

Palo Alto Networks is been Market leader for pas few years, some of their unique technologies are; within the application command center, all applications activities, user activities, IP bytes send and received and regions are nicely categorized. There is even a possibility to categorize applications. A system admin can allow certain applications within specific category to be allowed and block all of the other applications. A sophisticated URL filtering watches over the users. If an URL is registered as a legitimate financial website, even though it is not being recognized as a malicious category, it still marks the website as not
safe because it is not a proved URL. After one month if the website is legitimate the security team will automatically mark it as safe if not proven malicious. [21]

3.1.2 Fortinet

The current flagship of FourtiGate is the 4400F with following amazing specs. [9]

- 2 High availability ports
- 100 QSFP ports
- 1.2 TB/s firewall performance
- 75Gbps threat protection
- 86Gbps SSL inspections
- 310 Gbps VPN

It has integrated advanced thread protection, combined with FortiGuard lab which had inspected thousands of applications. Which offers a great protection against malware and viruses. It has a very powerful dashboard allowing a system admin to create a own perspective of the infrastructure. Besides categorizing the users and applications the admin is been provided with hardware widgets to be ensued of all the activities on the important machines. Among the all known techniques. such as no latency SSL offloading/inspection. [10]

3.1.3 Check Point Software Technologies

The Flagship is 64000 Quantum Security Gateway [7]

- 180 Gbps Full threat Prevention with Sand Blas Zero-Day
- 408 Gbps NGFW
- 880 Gbps Firewall Bandwidth
- 10,40,100 GBE Interfaces Speed

The unique features of CPST are Zero-day protection which keeps track of users, devices and application. All of them have to gain trust. Check point has granted the highest level of security effectiveness 98.5%, with no stability, reliability or evasion problems. Providing drill-down capabilities into events for easy troubleshooting. [18]

For obtaining security of these depths, these vendors are having huge security intelligence departments. The smartest way is to find a way to implement these intelligence on to the blue card. The best way to implement security intelligence of these depths is to be able to use these intelligence on the card rather than trying to create a security IDS/IPS implementation in such a short period.

3.1.4 Talos

This bring us to Talos, which is the intelligence branch of Cisco, that can be implemented in Snort or Suricata. Talos is having very advance security intelligence.

- IP and Domain Reputation center
It has a real-time threat detection network. Where the data is made up of daily security intelligence across millions of deployed web, email, firewall and IPS appliance. Where security team detects and correlates threats in real time using the largest threat detection network to ensure security of whole network infrastructure. All this security intelligence can be implemented with Talos rules. Talos intelligence keep track of all the at a global scale. Where it tracks that data in to different chunks of send and received, malware, emails, spam. On top of that having the track of recent outbreaks. Having this level of security intelligence on top of the hardware acceleration makes the Nvidia NIC the perfect IDS/IPS implementation. Where it is possible to keep intrusions outside the door and by implementing an analyzing tool the data can be monitored and made more secure. Above that with the help of a SIEM system it is possible to obtain the security level as mentioned above and to be able of having a clear overview of the whole infrastructure.

3.2 DPU solution

![Mellanox Bluefield DPU Layout](image)

The DPU for our intended workload is the Mellanox Bluefield DPU, also called the Bluefield SmartNIC due to the combination of networking capabilities as well as filtering and routing capabilities through the Arm system. In figure, we can see a hardware overview of the card. A list of features enable the application we are looking for. First off, the card has 16 64-bit Armv8 A72 cores and 16GB of ECC DDR4 RAM. This enables it to have a separate system on the card for data processing. Alongside the CPU cores for processing, the card also has hardware Public Key Acceleration and a "true" Random Number Generator (RNG). The hardware Public Key Acceleration enables the acceleration of various Public Key Infrastructure (PKI) algorithms such as RSA, DH, DSA, ECC and more. Further acceleration for hashing and private key cryptography is provided through an extended Arm A64, A32 and T32 instruction set for AES and SHA algorithms.

The ConnectX-5 network controller is responsible for the network flow. On the networking side of hardware acceleration, it also has transport offloads. These are for example TCP/UDP/IP stateless offloading, Remote Direct Memory Access (RDMA) over Converged Ethernet (RoCE), VxLAN en- and decapsulation and more. The hardware embedded switch can also
offload traffic flow shaping such as Open vSwitch. Finally, for feeding the card with the required data, it has high-speed data connections on either end. There are two 25 Gbps SFP28 ethernet ports for connectivity with the outside world. For connection with a host, the card has a PCIe Gen3.0/4.0 16x interface. It should be noted that there are different models of the card with higher and lower speed network ports.

Figure 3: Functional overview of the Bluefield DPU

In figure 3 we can see the way the card is meant to be used in a functional overview of the card. The card is connected with a host over PCIe. This host can see the card as a network interface and pass data to it. The DPU receives this data and passes it through its hardware embedded switch. Here the data can go straight out to the ethernet ports, or first pass to the Arm system for processing. Here it is important to introduce the two modes the DPU can operate in.
First off, we have the separated mode (figure 4). Here the host and the Arm system act as separate entities. They communicate with each other and through the underlying ConnectX-5 networking controller. In the SmartNIC mode however, the host can only communicate to the outside world through the ARM system of the card, instead of bypassing it. Using virtual interfaces, all data is routed through the Arm system, filtered/routed and then sent back out through the ethernet port. In this mode, the DPU is "transparent" to the host, the host only sees a "normal" NIC. Meanwhile, as can be seen in figure 5, the data is routed through the Arm system and through various filtering and routing layers.

This transparent mode has an ease-of-use and a security purpose. First of all, it is possible to have an administrator take care of all networking related operations on the card itself, rather than the host having to worry about it. For example, as in our case, VPN connections can be terminated on the card rather than the host. This means that the host only sees an IP to send unencrypted data to (less CPU load). The data is passed through the Arm system. Then a VPN connection can be managed from there essentially taking all the crypto...
and networking load and administration off of the host. The same goes for incoming data. It is automatically decrypted and routed to the host without the host knowing about it or having to worry about it. In terms of security, it is a benefit due to the fact that there is an added (transparent) layer of security between the outside world and the host. An attacker could for example think that they accessed the host while in reality they still are a layer down, in the Arm system.

4 Methodology

To answer our second and third research questions - "How can we support a large number of VPN networks using a DPU?" and "How can we implement an efficient networking monitoring and filtering system (IDS/IPS) which is transparent to the OS using a DPU?" - we had to create an environment suitable for testing the capabilities of this card with regards to high-speed networking and security monitoring. The networking structure and routing implementation as well as the IDS/IPS implementation will be outlined here. But first we will list the relevant hardware and software.

4.1 Software & Hardware used

Main machines used (Bluefield host for the DPU to connect to, the DPU card, a VPN machine to host VPN clients, a Windows VM to host a GrayLog interface and a Ubuntu VM to host the GrayLog server):

- Bluefield Host
  - CPU - i5-7500 @ 3.8GHz (4 cores)
  - Memory - 3706 MiB
  - OS - CentOS Linux 7 x86_64
- VPN1 Host
  - CPU - i7-7700 @ 4.2GHz (8 cores)
  - Memory - 15778 MiB
  - OS - CentOS Linux 7 x86_64
  - Network card - 40Gbps ConnectX5 network card
- Bluefield card
  - CPU - Armv8 A72 @ ?GHz (16 cores)
  - Memory - 16188 MiB
  - OS - Ubuntu 18.04.5 LTS aarch64
- Windows VM
  - CPU - Intel Xeon D-1537 @ 1.70GHz (1 vcore)
  - Memory - 3906 MiB
  - OS - Windows 10 Pro
- Graylog VM
  - CPU - Intel Xeon D-1537 @ 1.70GHz (2 vcores)
  - Memory - 3945 MiB
OS - Ubuntu 18.04.5 LTS x86_64

And finally we also used a SN2100B 16-port 100Gbps switch.

In terms of software used on the Bluefield card:

OpenVPN:
OpenVPN 2.4.4 aarch64-unknown-linux-gnu [SSL (OpenSSL)] [LZ0] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] built on May 14 2019
library versions: OpenSSL 1.1.1i 8 Dec 2020, LZO 2.08

OpenSSL:
OpenSSL 1.1.1i 8 Dec 2020 (compiled from source)

iperf3:
iperf 3.9 (compiled from source)

Suricata:
Suricata 6.0.1

GrayLog:
graylog-server 4.0.1-1

Elasticsearch:
Elasticsearch 7.10.0

MongoDB:
Mongod 4.0.21

4.2 Network setup

Since our research pertains to the use of a large amount of VPN connections to the Bluefield card and eventually the Bluefield Host, we essentially needed three machines: A machine to host VPN clients, the Bluefield card itself and a machine to act as the Bluefield Host where the card can be plugged in to. In figure 11 we can see these three from left to right as "VPN1", "Bluefield Card" (BC) and "Bluefield Host" (BH).

First, as seen in figure 6 and 7, a networking setup was created between VPN1 - Bluefield Card - Bluefield Host switch. For this we first had a physical connection between VPN1 and the BC - the eth1 interface on the VPN1 machine was its physical motherboard ethernet port and p0 on the BC was
its physical SPF28 ethernet port. Next, connecting the BC and BH, are 4 virtual interfaces, two on each side representing the PCIe connection. pf0hpf on the BC side connects with eth2 on the BH side and pf1hpf connects with eth3. For routing purposes the physical connection (VPN1 - BC) was put on a different subnet (eth1 - 10.0.0.1, p0 - 10.0.0.2) than the virtual interface PCIe connection (pf0hpf - 10.0.1.1, eth2 - 10.0.1.2). Let us call these subnets subnet A (eth1-p0) and subnet B (pf0hpf-eth2, pf1hpf-eth3).

Figure 8: OpenVPN setup

Next, we added the VPN aspect. We had several choices in terms of VPN protocols, like IPsec, SSL/TLS, PPTP. We ended up using OpenVPN with SSL/TLS due to the ease of use and lesser configurational complexity compared with IPsec. An OpenVPN server was installed on the BC. Here multiple servers are created with a separate configuration file for each.

Figure 9: OpenVPN server configuration template

Figure 9 shows a configurational template for an OpenVPN server. This template is edited with a script to automatically generate \( n \) servers with unique IPs and ports each. The first two lines are for setting the port of the server, as well as the IP of the server and
the client which connects to it. If we want multiple clients connecting to the server, we would change this into an ifconfig-pool setting which has a pool of IPs to be assigned to clients. Next we have several lines indicating that a tunneling device should be used as an interface, as well as that the route to subnet B should be pushed to the client. This means that packets to subnet B should go through the VPN tunnel to reach it, where the card can route it further.

The next interesting block of configurational lines starts at compress lz4-v2. This block contains configurations that affect the throughput of the VPN connection, either through optimizations or through a different (or none at all) cipher. First of all, we use the compression algorithm lz4-v2. This is an improved version of the previously in OpenVPN widely-used lz4. It features no overhead when a packet is uncompressable [12]. Next, we can change what cipher to use. A smaller key size will result in less computation and consequently, in theory, more throughput. Because we are looking at the cryptographic capabilities of the card, we ran tests with varying ciphers, such as AES-256-CBC, AES-128-CBC and no encryption.

Now we have the most important performance influencing configuration in our testing: the MTU. Due to the fact that in the IP header the length of the packet is specified in a 16 bit value, in theory it could be $2^{16}$ bits, or 65535 bits. However, a problem arises when a link in your network does not support this MTU size: the packets get fragmented (split up) to be able to “fit through weakest link”. With the ConnectX network controller, this maximum value seems to be 9978 [5]. This poses a limitation on the networking throughput capabilities. Because the links we are using (40G from VPN1 to the card) are capable of much more, we would ideally want to use the maximum MTU possible ($2^{16}$). It appears that when using a VPN tunnel, the MTU can be increased beyond the 9978 of the card, although that value is only present between the client and the server on the virtual interfaces. In between the packets still get fragmented. However, as can later be seen in the results, increasing the MTU beyond 9978 seems to have significant performance benefits. This is why we use this value in the configuration file.

To further increase performance we have also disabled fragmentation with fragment 0, as well as removing any limit on the packet size set by OpenVPN with mssfix 0. There are other optimizations we tried but did not affect the throughput in a way significant enough to notice. A commonly used one is setting the sending and receive buffer to the maximum value by using sndbuf 0 and rcvbuf 0. This would eliminate any buffer bottlenecks. Also, the flag fast-io is an experimental I/O optimization by reducing blocking during the writing of data. Although this can lead to a CPU performance benefit of 5-10% on some system, it had no effect on ours.

Finally, for authentication we use a single secret static key generated on the server and shared with the client using openvpn --genkey --secret static.key. In the client configuration we can see it as well as the previously mentioned optimizations being used:

```
# remote server IP + port placeholder
dev tun
# ifconfig client IP and server IP placeholder
# secret static.key
compress lz4-v2
cipher AES-256-CBC
mssfix 0
mss mtu 65535
fragment 0
mssfix 0
```

Figure 10: OpenVPN client configuration template
Here we also use a template to automatically generate clients to connect to a server. The remote server IP is always 10.0.0.2, but the port used is dependent on the server. The client and server IP is the same as generated on the server side of the according server.

For each created client and server, a tunnel (tun0, tun1, tun2 etc) is created on each side. These tunnels link to each other over OpenVPN. To be able to route packets from VPN1 to BH, packets need to know how to get there. As mentioned, the OpenVPN server pushes the route to the client telling it that the way to get to subnet B is through the VPN tunnel. Then, from there, the BC can route it further to the BH host. However, packets returning from the BH to VPN1 do not know how to reach it. Because of this an iptables masquerading has to be added to tell the BH that the packets are originating from the B subnet such that they can be routed back to the BC and from there routed back to VPN1 over the tunnel:

```bash
1. iptables -t nat -A POSTROUTING -o pf0hpf -j MASQUERADE
2. iptables -t nat -A POSTROUTING -o p0 -j MASQUERADE
```

Figure 11: Final network overview

### 4.3 IDS/IPS setup

For IDS/IPS implementation, snort and Suricata are the preferable IDS/IPS solutions. Each having specific benefit, for specific use cases. As the paper refers to how rules are released and the usage of snort on network based IDS and host based IDS, makes Snort a preferable IDS/IPS solution[6]. Besides usage of Talos intelligence makes snort a preferable choice. Also the impressive 10 Gbps with no latency in 2006, however according to [24] it is still possible VRT rules with Suricata. The deciding factors between the two is that Suricata uses multi threading and above that it also supports file extraction. So the preferable choice for current use case is Suricata.

### 4.4 Testing

For testing our setup, we have two different parts: the networking part and the IDS/IPS part. To test the networking part, we ran multiple performance tests using the iperf3 tool. Each time an iperf3 server was started at either the BH or BC using

```bash
1. for i in {0..n}; do
2. iperf3 -s -p $(5201+$i)) -D;
3. done
```

to have the servers run in the the background, with n being the amount of servers to start. Then, on the client side:
for i in {0..n}; do
    nohup iperf3 -c 10.8.0.($i*2) -t 60 -i 1 -B 10.8.0.($i*2 + 1) -p $(( $(5201 + $i)) -Z --timestamps >> /tmp/$(i)_$i.txt &
done

This would start $n$ clients, connect with each corresponding server and its port (originating from the client IP) and output the log to a user-specified filename appended with the client number.

<table>
<thead>
<tr>
<th>Connections</th>
<th>Options</th>
<th>Graphs</th>
<th>CPU</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MTU 1500, AES-256-CBC</td>
<td>Throughput</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>MTU 9000, AES-256-CBC</td>
<td>Throughput</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>MTU 65535, AES-256-CBC</td>
<td>Throughput</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>MTU 65535, no encryption</td>
<td>Throughput</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>AES-256-CBC</td>
<td>Throughput, CPU, MEM</td>
<td>All cores, %</td>
<td>Usage %</td>
</tr>
<tr>
<td>8</td>
<td>AES-128-CBC</td>
<td>Throughput, CPU, MEM</td>
<td>All cores, %</td>
<td>Usage %</td>
</tr>
<tr>
<td>8</td>
<td>No encryption</td>
<td>Throughput, CPU, MEM</td>
<td>All cores, %</td>
<td>Usage %</td>
</tr>
<tr>
<td>16</td>
<td>AES-256-CBC</td>
<td>Throughput, CPU, MEM</td>
<td>All cores, %</td>
<td>Usage %</td>
</tr>
<tr>
<td>16</td>
<td>AES-128-CBC</td>
<td>Throughput, CPU, MEM</td>
<td>All cores, %</td>
<td>Usage %</td>
</tr>
<tr>
<td>16</td>
<td>No encryption</td>
<td>Throughput, CPU, MEM</td>
<td>All cores, %</td>
<td>Usage %</td>
</tr>
</tbody>
</table>

Figure 12: All tests ran

We ran multiple different tests each time varying the amount of connections or the encryption type. First though, we did a separate test to show the difference the MTU value makes. A single iperf3 client and server was started and the MTU value was increased from 1500 to 9000 to 65535 and finally 65535 with no encryption as the previous tests were all with AES-256-CBC encryption. For the throughput tests with varying connections we did three tests each with 8 and 16 VPN client connections, and changing between AES-256-CBC encryption, AES-128-CBC encryption and no encryption at all.

For the testing of the IDS/IPS implementation, we used the hping3 tool to generate a UDP flooding attack. This originates from the VPN1 machine and is target at the p0 interface of the BC. If implemented correctly, the IDS/IPS should pick this up and block it. Then we should be able to see a log entry for the attack in the suricata log as well as the GrayLog log and dashboard.
5 Results

5.1 VPN network

In figure 13 we can see the test with a single connection and multiple MTU values. We can see that as the MTU value increases on both sides, the throughput does as well. We go from an average 50 Mbps with MTU 1500 to 250 Mbps with MTU 9000 to around 1200 Mbps with the maximum MTU of 65535. This is an increase of 2400% in comparison with the lowest tested MTU of 1500. Finally, we also tested the effect of disabling encryption to see the throughput effect and there is a slight increase in throughput.

In figure 14 we can see the memory and CPU usage per core during the 1 VPN 1500 MTU test. We can clearly see that there is always a single core at 100% utilization. A similar image can be seen in appendix A for the other 1 VPN tests.
Figure 15: 8 VPN connections with no encryption

Figure 16: 8 VPN connections with AES-128-CBC encryption
In figures 15, 16 and 17 we can see the tests performed with 8 VPN/iperf3 connections and different encryption methods. Although the effect is less pronounced, the average throughput keeps dropping whenever the cipher used becomes more processing intensive. We go from an average throughput of 286 Mbps with no encryption, to 278 Mbps with 128 bit encryption to 253 Mbps with 256 bit encryption. That is a 3% and 12% decrease in throughput respectively.

In figure 18 we can see the memory and CPU usage per core during the 8 VPN with multiple different ciphers. There are some irregularities in terms of CPU usage consistencies.
These happened repeatedly are we do not know why. There seems to be a link between the peaks and valleys of the 128 bit AES test’s CPU usage versus throughput, but the other tests don’t match. Here we did not include all cores in the figure due to them all hovering around the average, making the image unnecessarily cluttered. We can observe that the memory usage does increase slightly with every increase in cryptographic load.

Figure 19: 16 VPN connections with no encryption
Next, here in figures 19, 20 and 21 we can see the tests performed with 16 VPN and iperf3 connections with different encryption ciphers used. Here we have the most interesting result, where the encryption type does not change the throughput in any significant way. Any differences are within the margin of error. All three tests hover around 140 Mbps.
Finally, in figure 22 we can see the memory and CPU usage per core during the 16 VPN test with multiple different ciphers. The CPU usage for all encryption types remains around 35%. The memory usage is also stable around 27%.

5.2 IDS/IPS implementation

For this paper Suricata was installed on the BC. After initial configuration of rules and the Home net, Suricata was ready. To test the IDS/IPS implementation a test is carried out from the VPN1 box. With the following command a DDos simulation attack is carried out.

```
$ hping3 -i u20 -S -p 80 -c 50000 10.0.0.2
```

To test if Suricata has effectively blocked the incoming attack, the fast.log file is being monitored.

```
02/02/2021 -16:07:29.992612 [**] [1:2400011:2795] ET DROP Spamhaus DROP Listed
  Traffic Inbound group 12 [**] [Classification: Misc Attack] [Priority → : 2] { TCP }
2
122.8.184.180:41705 -> 10.0.0.2:8
```

```
02/02/2021 -16:07:29.979155 [**] [1:2400036:2795] ET DROP Spamhaus DROP Listed
  Traffic Inbound group 37 [**] [Classification: Misc Attack] [Priority → : 2] { TCP }
2
206.143.177.150:40144 -> 10.0.0.2:8
```

```
02/02/2021 -16:07:30.066783 [**] [1:2400028:2795] ET DROP Spamhaus DROP Listed
  Traffic Inbound group 29 [**] [Classification: Misc Attack] [Priority → : 2] { TCP }
2
195.207.120.114:50189 -> 10.0.0.2:8
```

For this study the aim is to analyze and visualize the attacks or the malicious attempts, so that the system admin can take steps to improve the security, or to create user awareness. For this step a SIEM system needed to be implemented. To do so, a Graylog is installed with the IP: 172.30.230.111 and a windows server as a Graylog GUI with the IP: 172.30.230.112 listening to Graylog on port 9000. A filebeat agent is installed on BH to transfer logs to Graylog server.
There is also a dashboard presenting the possibility of a DDoS attack. As presented in the figure below, the dashboard makes it possible to see an incoming DDoS attack happening, since it has a trend indication which will change color from green, and indicate the number of extra incoming logs in comparing with the older logging trend.

6 Discussion

During this research, there were different moments and topics where we ran into a problem or made a specific choice relevant enough to discuss. First of all, and the biggest, is being unable to fully use the card with all of it’s acceleration capabilities. Over the course of the project we were constantly trying to get some kind of cryptographic acceleration working. Specifically for the use in OpenVPN, which in turn uses OpenSSL. This means that OpenSSL needs
to recognize the engine. For this we tried many different things like getting the cryptodev
device (“a device that allows access to Linux kernel cryptographic drivers” [15]) working. This didn’t work. We also tried installing the engine from the Mellanox PKA (public key acceleration program) [16], but the engine couldn’t be loaded into OpenSSL. There seems to be a lack of (public) documentation from both the developers and previous users, to the point that we did not have enough time to look into this enough to get it working. With the differences in throughput depending on the encryption, we believe that this being enabled would have had a positive impact.

Next, changing the MTU had effects that were unexpected. When changing the MTU of the virtual interfaces of the VPN client and server to a value past the maximum MTU of the card (9978), the throughput kept increasing. It is logical to assume that the weakest link should be the determining factor. This means that the packets should automatically get fragmented at 9978 bytes regardless of an MTU value beyond that and the performance should be capped there, but it was not. In fact it increased significantly more than expected. It would be interesting to look into what exactly is happening with the packets with regards to fragmentation.

In the related work we discussed the performance of IPSec vs OpenVPN. Here some work indicated that IPSec is more scalable and efficient. We originally also had a choice in using OpenVPN or IPSec. However, although IPSec could have greater performance, we chose for OpenVPN simply due to the ease of configuration. The time limit of the project combined with the work that had to be done made us come to the conclusion that using something that is easier to work with would eventually benefit the research because of more work done. Finally, we would like to mention the Talos intelligence has an annual pricing where the customer gets 24/7 support and live updates of newest intelligence. The free version of Talos provides one month older signature and security.

6.1 Conclusion

In this study, we have researched the implementation of the Mellanox Bluefield DPU in a scalable and transparent security solution including a network with VPN connections. Consequently, we have answered our research question

**How can a DPU be used to implement a scalable and transparent security solution for numerous VPN connections?**

with our three sub questions. First off we researched the current state of networking and firewalling with regards to the market leaders. The security branches of market leaders vendors are having very advanced firewall, malware, and monitoring widgets. They are having huge network of advanced security teams working to get the most advanced and secure intelligence. So implementing security at that levels is almost impossible in the timeline of this study. So the smartest choice would be to use such intelligence instead of trying accomplish such lev of security. Talos security branch of Cisco is the perfect outcome.

Next, we looked at support for a large number of VPN connections. In the end we were able to create a network which sustains multiple VPN connections at once with relatively high speeds. In our results in the test with 16 VPN connections and 256 bit AES encryption we were able to get speeds of 140 Mbps per client. It should be noted that in other tests we have been able to get 300 Mbps in the same setting, although we were unable to reproduce this result in time to add in the results. This gives enough headroom to split into many more VPN connections. This is supported by the CPU performance graphs seen in figures [18] and [22]. There is enough room in terms of CPU utilization left for more connections. However, there are still some bottlenecks such as the lack of cryptographic acceleration and various oddities which limited the true potential of the card. The networking setup created also accounts for easy scalability by simply creating more clients or servers and connecting.
them with each other using the scripts.

Using Talos intelligence within Suricata, and visualizing and analyzing the logs and rules makes it a decent alternative. It makes it possible to select or analyze, users, applications or traffic, besides the possibility to create an alert. SIEM also provides the possibility to look back in older logs and make the environment more secure. The whole system stack gives a system admin the perfect tool to keep whole infrastructure secure.

All in all we can conclude that the Bluefield DPU is capable of creating a scalable and transparent security solution, in theory capable of handling numerous VPN connections.

6.2 Future work
As noted in the results and conclusion, the main hurdles was the lack of enabled cryptographic acceleration for both the extended Arm instruction set and the PKI acceleration. It would increase performance for the VPN connection throughput. As a consequence of the disabled cryptographic acceleration, we were also unable to perform tests to what extent this would benefit in terms of VPN connections. There is a new version of the Bluefield DPU, the Bluefield DPU-2. This promises to have better performance, hardware acceleration and software support. Ideally, these tests and the setup should be repeated on that platform with a better ability to enable the hardware offloading required to obtain the results we hoped to get. Finally, an expansion on the IDS/IPS visualization would be beneficial. A dashboard system with possible alerts would give a better overview of the situation, instead of a list of logs.

6.3 Ethical considerations
During this research, all the tests were carried out in a test environment. No private information is used or shared.

References
[1] URL: https://docs.mellanox.com/display/bluefieldsniceth/Supported+Interfaces
[2] URL: https://docs.mellanox.com/display/BlueFieldSWv25111213/Functional+Diagram


[17] Modes of Operation. URL: https://docs.mellanox.com/display/BlueFieldSWv22011000/Modes+of+Operation


[26] Vulnerability Information. URL: https://talosintelligence.com/reputation

A Extra results graphs

Figure 25: per-core CPU and Memory usage for a 1 VPN 9000 MTU test

Figure 26: per-core CPU and Memory usage for a 1 VPN 65535 MTU test
Figure 27: per-core CPU and Memory usage for a 1 VPN 65535 MTU no encryption test