



Boston University
Electrical & Computer Engineering
EC463 Capstone Senior Design Project
First Semester Report

CyberTap

Submitted to:
Cybereason
John Hancock Tower
200 Clarendon St, Boston, MA 02116
(855) 695-8200
by



Team 2
CyberTap
Team Members:
Felipe Dale Figeman fdale@bu.edu
Alex Fatyga afatyga@bu.edu
Evan Lang evanlang@bu.edu
Noah Malhi malhin@bu.edu
Justin Morgan justinm@bu.edu

Submitted: December 10th, 2019

Table of Contents

Executive Summary.....	1
1.0 Introduction.....	2
2.0 Concept Development.....	4
3.0 System Description.....	6
4.0 First Semester Progress.....	8
5.0 Technical Plan.....	10
6.0 Budget Estimate.....	12
7.0 Attachments.....	13
7.1 Appendix 1 – Engineering Requirements.....	13
7.2 Appendix 2 – Gantt Chart.....	13
7.3 Appendix 3 – Technical References.....	14
7.4 Appendix 4 – Team Information Sheet.....	15
7.5 Appendix 5 – Technical Figures.....	16

Executive Summary

CyberTap
Group 2 – CyberTap

CyberTap is a high throughput hardware network tap designed for industrial control systems closed off from the internet. Other network monitoring solutions are software based and over utilize the CPU, which could theoretically result in packet loss. Packet loss is particularly undesirable in the context of the high data throughput involved in industrial control systems, especially when said systems directly rely on input data. CyberTap will be able to collect Operational Technology (OT) network packets, parse and generate metadata for all relevant network protocols of a system, and store the collected packets and generated metadata in storage. The product will be implemented on a Field Programmable Gate Array (FPGA) to utilize their ability to quickly process large data loads. The final deliverable will contain the following: A simulated high-data-volume OT network using 2 Raspberry Pis and a network switch, an FPGA that sniffs, parses, and outputs the packet data coming from the SPAN port of the switch, a solid state drive that contains the outputted metadata, and a web application for querying the drive and displaying the metadata.

1.0 Introduction

CyberTap's client, Cybereason, is a cybersecurity company focused on enterprise endpoint protection. Cybereason has no way of gaining visibility into or performing long term data collection of their customers' networks. This is where CyberTap comes in. They need CyberTap to be able to generate and store useful metadata for them to analyze with their cybersecurity tools. Therefore, the purpose of CyberTap is to generate metadata without any packet loss and format it for analysis.

The device will be utilized by the client primarily for Operational Technology (OT) networks and Industrial Control Systems (ICS). OT networks support infrastructures for utilities, manufacturing, and defense. They utilize both hardware and software to detect changes via monitoring of physical devices, such as sensors. An ICS is a type of control system used for operation and automation of industrial processes.

Figure 1: General Design



A general overview of CyberTap's system, going from the network switch through the FPGA to the SSD.

The general approach for the team was to utilize an FPGA, implement a packet sniffer to obtain the network activity flowing through a network switch, and create a packet parser to parse the incoming packets at high speeds. By doing this in hardware, the typical throughput and sniffing capabilities from a software implementation can be extended. The FPGA is a blank slate which allows for it to be used specifically for the tasks of packet sniffing and parsing. Since there is full control over the operation of the FPGA there are no other tasks that could interrupt the system, unlike what occurs with software on a CPU. Another key aspect of the FPGA is the sheer number of memory logic cells. This allows for the device to be scalable by enabling more protocol parsing modules to be added. This allows the FPGA module to be "hot swapped" into different network environments, containing the parsing modules needed for each specific network. For disk writes of the saved packets and generated data, the team will be transferring the data with the dedicated source HDMI port on the new FPGA through ethernet over HDMI. Using a motherboard attached to the FPGA initially seemed like the best plan due to its speed, but the complexity of the implementation would result in incompleteness of the project. Ethernet over HDMI allow for a high enough transfer rate as well as a simpler approach to designing the product data transfer interface.

By using this approach, the needs of the client can be adequately met. Their first request was for a hardware device with the ability to sniff a network without high loss of data. By using the FPGA it will allow for a lossless system that should be able to run under high stress networks. Additionally, Cybereason will later be able to adapt the device to their various needs as CyberTap will be easily scalable. The basic parsing and generation will be completed in the FPGA and different protocols can be added to the logic as per need basis. Lastly, by writing to memory with SD, this satisfies their requirement for a long-term solution. Cybereason needs this long-term solution so they may perform long-term data analysis; the group's method of implementing memory transfers to SSD will ensure no packets will be loss.

In a software design unlike CyberTap's, CPU interrupts often occur resulting in some packets being lost. CyberTap's approach sets itself apart from other solutions as it is hardware-based using an FPGA and therefore, extremely fast. By the implementation of the FGPA's internal memory transfer flow, no packets will be lost from input, to parsing, to generation of data, to writing to SSD. Buffers between these different portions will be operating in a producer/consumer manner which will stabilize the flow of data protecting the threat of over-feeding components and overflowing. CyberTap will also include an external SSD to provide long term data storage. This provides a history of data that can be utilized for analysis that Wireshark does not provide.

2.0 Concept Development

Until recently, cybersecurity was an afterthought for companies and their technological infrastructure. Much progress has been made since in enterprise environments with many companies realizing the importance of keeping their networks secure as the machines in these environments employ the typical internet protocols (TCP/IP) in their communications.

Industrial settings, on the other hand, are typically closed off from the internet. Few machines are able to connect outside of their Local Area Network (LAN). The rest, such as microcontrollers on the industrial equipment, communicate within the network with special Industrial Control System (ICS) protocols, some of which can be unique to a specific manufacturer.

Network sniffing and network protocol analysis in an Industrial Control System (ICS) network enables organizations to collect and analyze the traffic sent between devices, allowing them to identify and track down potentially malicious activities. Conventional network sniffing methods and protocol analyzers are slow as they are mostly software based, failing to offer sufficient analysis of the network protocols used in an ICS environment. In an industrial setting this results in limited visibility into potential cyber attacks. CyberTap's client seeks to achieve this without the latency associated with traditional methods. In order to meet that requirement much of the workload will be offloaded to an FPGA.

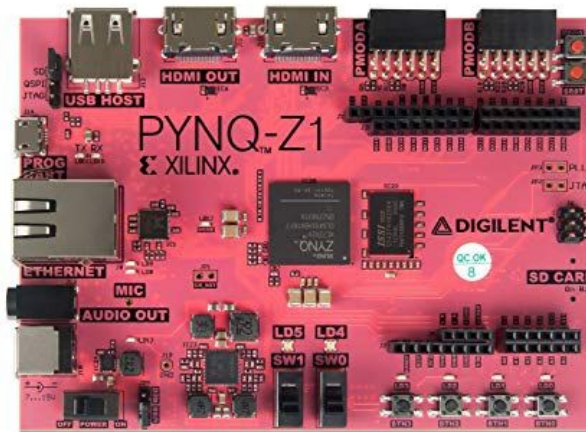
By performing ICS network protocol analysis on hardware, the latency from traditional software methods is massively reduced without impacting the capabilities of the network sensor. Such a solution would allow for organizations to easily keep their systems secure without impacting the efficiency of their already established infrastructure.

From the start, the group planned to use an FPGA and started to research adequate boards immediately. However, it was difficult to get started on the project without a board. Therefore, the group was provided with a Xilinx Nexys A7 board from the ECE Department. The board was capable of running the Microblaze IP (intellectual property) necessary for network interaction. Similarly, it possessed basic I/O features needed for the FPGA to interact with the external web app. However, the results of lab testing highlighted that the Nexys A7 had inadequate I/O speeds via UART for displaying intercepted packets. Lab testing using UART resulted in almost 5% of packets dropping or containing corrupted payloads, making UART an insufficient interim solution until SSD offloading is implemented.

The group was also informed of an available NVIDIA Jetson TX2 development board that could be provided free of charge. This development board is a GPU which the group does not have thorough experience with. The board is capable of writing directly to SSD at more than 1 GB/s. Even though a GPU can parallelize operations by utilizing large amounts of threads in numerous GPU blocks, the reason this option was not chosen is due to the latency of memory transfers. The implementation of a GPU is typically done in a CUDA file, and how it operates is that there is both CPU code and GPU code. The packets would be received in the CPU code and have to be passed to the GPU via `cudamemcpy`. The latency from copying memory may result in the loss of packets because of the rate of memory copies that would need to be done.

A PCIe board was heavily considered but the time it would take to develop it would exceed two semesters. This type of board was considered due to its ability to connect with a motherboard. The PCIe would allow for faster memory transfers between the FPGA and the motherboard, which would have a high enough throughput to not drop or lose packets. On the motherboard memory could be written to an SSD using direct memory access (DMA). Even though this board would satisfy CyberTap's need, to actually implement the DMA access would take an enormous portion of time. Reads and Writes would have to be scheduled and numerous buffers within the FPGA and motherboard would need to be initialized to maintain a flow of steady data. This added logic and scheduling is doable but not within the timeline for completion of the project. Another downside of one of these boards is the price. The cheapest PCIe FPGA board that was researched is around \$650. This is more than half the budget spent and if there were any issues then the group would be stuck with the board.

Figure 2. PYNQ-Z1 FPGA.



The PYNQ-Z1 board, chosen due to its mix of cheap upfront cost, versatile I/O, and innovative tool suite.

It was finally decided that any new board chosen must satisfy the client's requirements while still being reasonably within budget. The major requirement that needed to be met was the ability to perform 1Gb/s Ethernet connections. Increased I/O functionality was also greatly considered, as the USB 2.0, UART, and GPIO pins did not possess adequate transfer speeds for external packet information storage. After comparing Xilinx and Intel FPGA boards, it was once again decided that Xilinx type boards and tools provided better functionality for the client's usage needs compared to Intel offerings. Specifically, the Zynq-7000 based Xilinx boards offered the best mix of board functionality, low initial cost, and quality tools. After searching the large number of Zynq-7000 board offerings, the PYNQ FPGA was decided as the best option. This board offered a unique tool suite that enabled the group to do HDL coding with Python rather than Verilog, a language many of the team have experience with. That combined with its 1 GB/s Ethernet, added HDMI I/O, and reasonable price (~\$200) made it the most attractive option for the team.

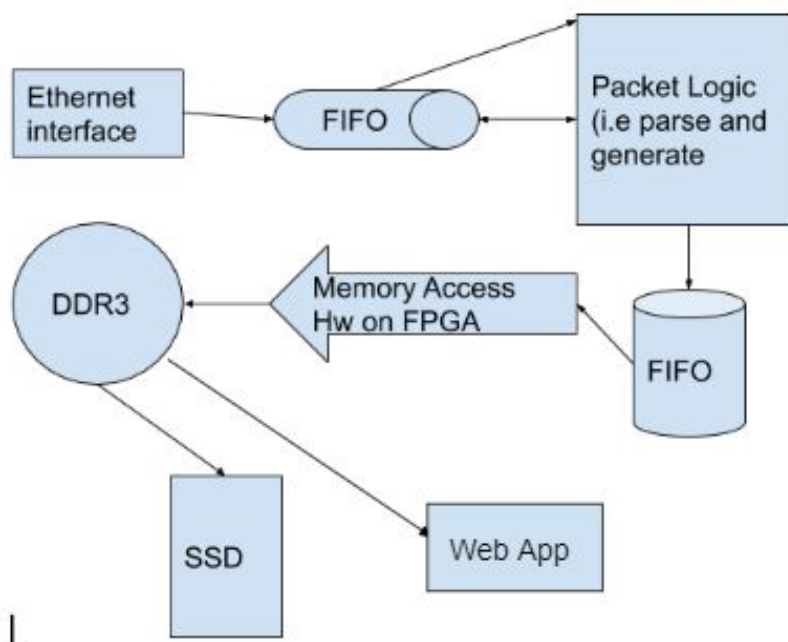
3.0 System Description

The hardware flow of the device first begins with an ethernet connection. The ethernet interface will connect the FPGA to a network switch's SPAN port. Through port mirroring, all of the network packets going through the switch will be duplicated and sent to the FPGA. When the packets are received by the FPGA they will be sent to a First in First Out (FIFO) data structure. The FIFO will be used as a buffer for the device to regulate the different speeds of information entering the FPGA and maintain a consistent flow. This is necessary as the usual devices in an ICS network send data on regular intervals, and so the amount of data generated in an interval may exceed our SSD IOPS, and so this allows for a smoothing out of the disk operations reducing the odds of not backing up a packet.

The data structure will receive the packets and hold them until the next step in the process will request more data. Since the size of the FIFO can be dynamically grown, there should not be any form of packet loss from this. From the FIFO, the packets will be sent to a packet parsing module. This module is responsible for deciphering the different protocols and then generating metadata on them. The metadata will include things such as length, payload, protocol type, and more. To make sure this piece of logic is not overloaded, the FIFO will allow it to request more data when it is ready. After parsing the packets and generating the metadata, the resulting metadata is sent to another FIFO buffer. This buffer serves a similar purpose as before. However, this time it will direct data to be written to memory. The generating data will be stored temporarily on-chip on DDR3 and from there written to a Solid State Drive (SSD) by the SD port. A male-to-male SD adapter will be used to connect the FPGA to the SSD. On the new board, ethernet over HDMI out serves as another, potentially more viable, option for moving the generated metadata towards the SSD.

For viewing data, the web application will provide a real time look into the network activity. The web application writes to a TingoDB database and allows users to search through their packets in real-time. From the FPGA, the website will be hosted. The new development board that has been ordered, the PYNQ, has web hosting capabilities. The data that will be displayed on the web application will be retrieved from on-chip DDR3. This will allow for a soft real-time analysis of the network data stream. If progress moves faster than estimated, adding web-application access to long term storage, SSD, will be added. This stretch goal will allow a user to access network data through the web app from the SSD from the past.

Figure 3: Block Diagram.



Hardware Diagram for the Network Tap. The general flow is from ethernet packet input to parsing and generation of metadata to memory. FIFO memory structures are used as buffers as a form of producer/consumer.

Microblaze is a Xilinx I.P. microprocessor that runs in the FPGA and is capable of operating many of the boards important features required for this project. Specifically, Microblaze is capable of running the memory interface, direct memory access, and Ethernet controller of the board. These components represent the most vital parts of the FPGA needed for packet sniffing and external packet storage. This processor will function as the ‘brain’ of the CyberTap device. While it does not yet have the ability to parse network packets or store information on external drives, the Microblaze base provides a modular and easily modified foundation for the rest of the project. This Microblaze processor also includes an MII to RMI interface, allowing the lower level ethernet controller on the Nexys A7 to properly interact with the network.

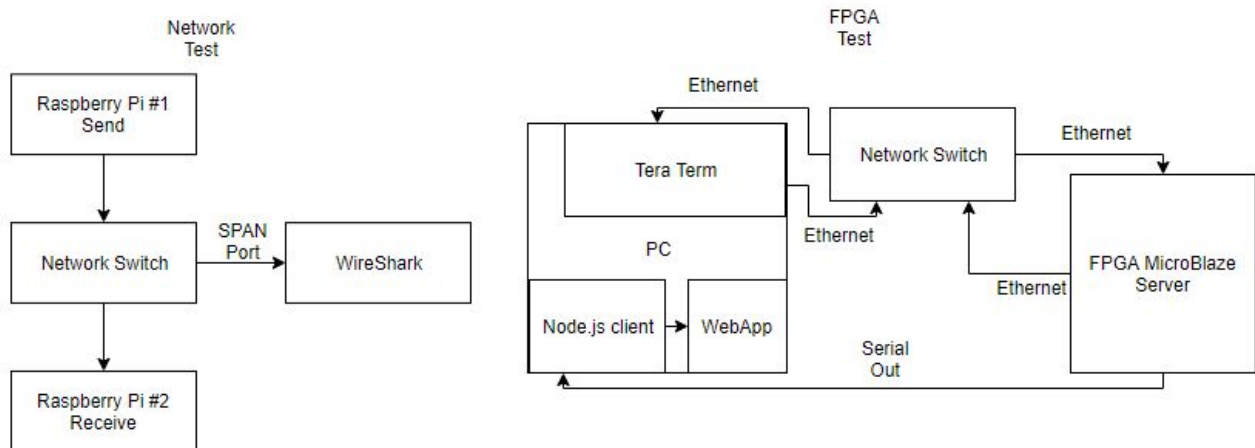
In the web application, the backend server takes in the packet information while noting the exact time the packet came in. Using socket.io, the back end sends the packet information and timestamp to the front end. Furthermore, the back end adds the packet information to a Tingodb database. Tingodb will be an intermediate database service as until the database service with the best and most consistent speed is determined. By both sending the packet information to web client and adding to the database, the web client receives information much faster instead of waiting until it’s written to the database. The web client takes in the packet information and time and adds the information into the table. The web application also allows a user to search packets by source IP, destination IP and protocol at the same time. This was completed by taking in the user inputs and filtering out rows accordingly. The searching is completed in real time for user ease of use.

4.0 First Semester Progress

In the first semester, the group worked primarily on device input. The main goal was to be able to connect the FPGA to a network switch and be able to receive packets. A Switched Port Analyzer (SPAN) was used on the switch to mirror the traffic flowing through the other ports. This allowed CyberTap to be connected via ethernet to the SPAN and allow for the full monitoring of the traffic being sent through the switch.

Additionally, a NodeJS web application was written using Javascript on the back end and HTML/CSS with some Javascript on the front end. The web server is running on local host for security. It allows users to search through the packets and adds the more recent packets to a database for further backup.

Figure 4. Testing Setup.



These two diagrams show the two tests run on testing day. The diagram on the left is the Network Test involving Wireshark, the network switch and the Raspberry Pis. The diagram on the right is the FPGA test which used the FPGA, Tera Term, NodeJS Server and MicroBlaze server.

By the first prototype day, the group had 2 tests that displayed progress. The first test involved testing the end to end functionality of the current networking set up with the FPGA Microblaze echo server and Tera Term. The packet payload was then echoed back to the source IPv4 address (Tera Term) and also sent to the serial port COM6 (UART). The serial port was then read by the NodeJS server which displayed incoming data in the web app. After completing this test, the group confirmed their belief that the throughput provided by UART would be too slow to transmit the packet payload. At a speed of 1 packet per second, the group still had some partial packet loss of 4.5%.

The second part of testing involved creating a mock network by having one Raspberry Pi send data to another Pi via TCP protocol. The two Raspberry Pi's were connected to a network switch as a proxy between the two. Using a SPAN on the network switch, monitoring of the packets could be done using Wireshark. This test provided a proof of concept for CyberTap as well as a testing server for future use. Being able to view packets being sent with Wireshark set a baseline for what the hardware device needs to accomplish. Completing this test server allows for future experimentation by using various network protocols such as DNP3, that CyberTap will be able to parse and generate metadata on.

After testing, the FPGA proved it could only grab packets addressed to its specific IP address and port. This prevents the FPGA from accepting SPAN port traffic, rendering the network tap test impossible. Additionally, once the FPGA is able to receive the traffic the group will be able to increase the realism of the tests by not having a set wait time between packets, or at most a small delay in the millisecond range. Therefore, the group focused on packet sniffing and connecting the two tests together.

To complete packet sniffing, the group utilized the provided IP blocks and modified the files responsible for low network layer processing so that the packets would no longer be rejected and discarded. This required understanding of the networking interface driver and close inspection of the functions called throughout execution. As this is run on hardware, the group does not have the normal ability to step through program execution using gdb, and so must rely on the group's understanding of low level systems to find the relevant functions. The functions that had to be found and modified were `xemac_liteif_input` in `xemac_liteif.cm`, `netif_input` in `netif.c`, `ethernet_input` in `ethernet.c`, and `ip4_input` in `ip4.c`.

At the end of the semester, packet sniffing was completed. By utilizing SDK software files on top of the hardware design, CyberTap was able to view all packets sent through the ethernet cable. As the cable is connected to the SPAN port of the switch, these packets are those of the other devices on the switch. However, this functionality has been successfully tested when connected to devices other than the switch.

Currently, the board alerts when it has received an ARP or an IPv4 packet, and displays the source MAC address in the case of ARP packets. While IP packets are received and alerted on, due to the fact that some packets can be fragmented and reconstructed out of order, displaying the contents of these packets is still not implemented.

When connected to the SPAN port of the switch, the FPGA can view the source and destination IP's of the two Raspberry Pis through the echo server that was set up. The echo server that was used in the first prototype testing was altered to keep rather than throw away the mirrored packets.

5.0 Technical Plan

After the work completed in the first semester, the group has proposed the following plan to complete the solution:

Task 1. New Board Setup.

Setting up the new board will include familiarizing the group with its functionality and completing simple programs to get a handle on how to implement packet parsing. This task will be completed during winter break and will be important to complete before packet parsing. It is preliminary packet parsing that will help to speed up the implementation of packet parsing. Hardware design specifications need to be fully understood prior to receiving the board to allow for faster progress. By having prior research there will be less time wasted while waiting for the board to arrive.

On completion, the group shall have general knowledge on how to operate the board. The board will be setup to have the similar capabilities of the previous prototype. The hardware design on which the software will run on, should be established.

Lead: Evan; Assisting: Noah

Task 2. Packet Parsing.

Packet parsing shall be implemented first after winter break and will take several weeks. Packets that were received from the ethernet interface would be put into a hardware First-in-First-Out data structure on the FPGA. From this structure data needs to be pulled and parsed by this logic to differentiate various packets.

On completion, packets stored in the hardware FIFO buffer will be consumed by the parsing logic. All data will be parsed and no packets will be dropped. Various protocols such as TCP, Modbus, UDP, and ARP will be distinguishable. From the protocols, data will be procured which includes length, type, and payload.

Lead: Evan

- Milestone 1: Parse source, destination, and content of a TCP packet via hardware.
 - Assisting: Felipe, Noah
- Milestone 2: Integrate single packet parse into FPGA network.
 - Assisting: Alex, Felipe
- Milestone 3: Parse source, destination, and content of Modbus packet.
 - Assisting: Justin
- Milestone 4: Enable detection of TCP vs. Modbus.
 - Assisting: Felipe
- Milestone 5: Integrate detection, and parsing into FPGA network.
 - Assisting: Justin

Task 3. SSD I/O.

SSD I/O shall create a backup of the network activity that users will be able to sort through. This task will begin with the SD writing data from the FPGA and leading to the SSD being written to and finally, being able to query from the SSD. By having this done, it accomplishes one of the key requirements of the client to have a long term memory solution.

On completion, data should be written from the on-chip FIFO buffer to DDR3. From the temporary on-chip storage, data shall be written through the SD port to the SSD. No data should be lost from this no matter the rate of flow of data. By having the FIFOs act in a producer/consumer method, the transfers should conclude steadily. Data stored in SSD should be stored in a readable manner.

Lead: Noah; Assisting: Justin

- Milestone 1: Be able to write parsed data to on chip DDR3
- Milestone 1: Write data from DDR3 to an SD card on the FPGA
- Milestone 2: Write data from SD to SSD via an SD male-male adapter
- Milestone 3: Be able to read/query from SSD for data output reasons

Task 4. Output to Web Application.

The web application and database have already been set up and now are in need of the packet information to come in. Outputting to the web application from the FPGA shall include receiving an output from the FPGA and then connecting it to the web app and into the database.

On completion, the web application shall be hosted off of the PYNQ FPGA. The application will be easily readable and accessible by the board's IP address on the local network. Data shown should be updated on refresh of the page. The latency between when the data is sent to when it is displayed on the web page should be less than five seconds. The metadata on the web application will be able to be separated by protocol type for easier analysis.

Lead: Alex; Assisting: Noah

- Milestone 1: Receive packet information from FPGA
- Milestone 2: Receive packets into the web application
- Milestone 3: Add packets to database
- Milestone 4: Improve database and Web Application for speed

6.0 Budget Estimate.

Item	Description	Cost	Quantity
1	Raspberry Pi 3	\$38	2
2	PYNQ-Z1 FPGA	\$199.99	1
3	SD to SD adapter	\$7.99	1
4	tp-link Pro Network Switch	\$30	1
5	SSD	\$60	1
6	Nexys A7 FPGA*	\$250	2
7	tp-link Desktop Switch*	\$15.99	1
8	Ethernet Cables*	\$4	3

* denotes an item donated by Boston University College of Engineering

The group was provided with 2 Nexys A7 FPGAs and a network switch by the University. These boards served as the original foundation of CyberTap's system. However, lab testing results showed 5% packet loss, which was considered unacceptable. It was then decided a new FPGA board would need to be purchased that satisfied performance requirements. The FPGA is easily the most expensive component of the system so picking the right FPGA was vital to the group's success and budget. After thorough consideration, the group decided on the PYNQ-Z1 because of its functionality and pricing. This pricing fits extremely well within the budget without compromising the functionality. It also includes an advanced tool suite that will streamline future work on HDL module, with its Python HDL suite.

CyberTap is designed to be plugged into the SPAN port of a network switch within an ICS environment in order to "tap" the network by backing up all incoming/outgoing packets through the switch. In order to configure a SPAN port on a network switch, the switch must be managed and be capable of port mirroring, which allows the user to copy traffic going through the switch ports and send it through a specified mirroring port. The switch donated by the College of Engineering is an unmanaged switch, so the group purchased the mirroring-enabled Pro variety of the original switch.

7.0 Attachments

7.1 Appendix 1 – Engineering Requirements

Team #2

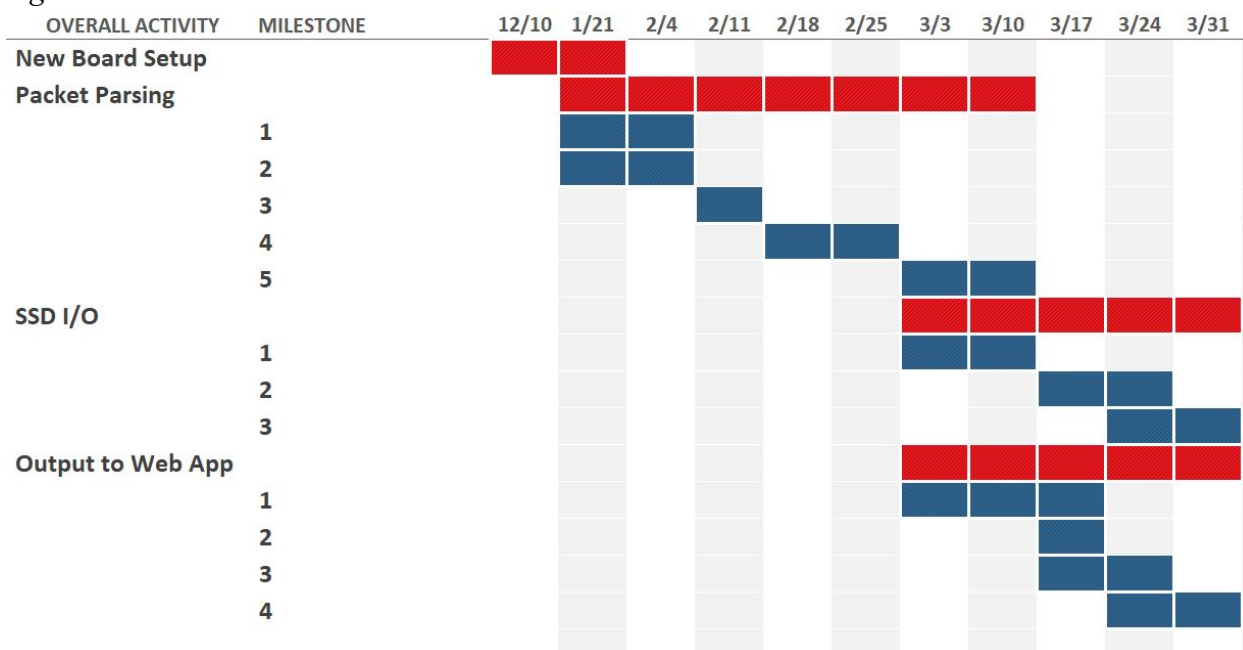
Team Name: CyberTap

Project Name: CyberTap

Requirement	Value, range, tolerance, units
Packet Loss through FPGA	0%
Packet Loss To SSD	0%
Packet Loss to Web App	0%
High Throughput/Stress Test	Network flow up to 1 Gigabit ethernet
Soft Real Time Web App	< 5 sec latency of memory packet transfer to display
Long-term Use	Continuous network flow >5 days
Ability to sniff common OT Protocols	Modbus, DNP3, TCP, IP, Bluetooth, ARP
Hot Pluggable	Able to plug into any span port and begin data processing and storage

7.2 Appendix 2 – Gantt Chart

Figure 5. Gantt Chart.



This chart shows the group's plan from December 10th and onwards. The focus will be on completing packet parsing and then being able to backup the packets for future use.

7.3 Appendix 3 – Technical References

(n.d.). Retrieved from <https://ieeexplore.ieee.org/document/5518537>.

Adaptable. Intelligent. (n.d.). Retrieved from <https://www.xilinx.com/>.

Arty - Getting Started with Microblaze Servers. (n.d.). Retrieved from <https://reference.digilentinc.com/learn/programmable-logic/tutorials/arty-getting-started-with-microblaze-servers/start>.

DNP3 Tutorial Part 4: Understanding DNP3 Message Structure. (n.d.). Retrieved from <https://www.dpstele.com/dnp3/tutorial-understanding-message-structure.php>.

Modbus. (2019, December 6). Retrieved from <https://en.wikipedia.org/wiki/Modbus>.

Modbus Organization. (n.d.). Retrieved from <http://www.modbus.org/specs.php>.

Python productivity for Zynq. (n.d.). Retrieved from <http://www.pynq.io/>.

Rouse, M., Gerwig, K., Rouse, M., & Rouse, M. (n.d.). What is TCP/IP and How Does It Work? Retrieved from <https://searchnetworking.techtarget.com/definition/TCP-IP>.

Socket Programming in C/C . (2019, May 31). Retrieved from <https://www.geeksforgeeks.org/socket-programming-cc/>.

What is ICS Security? (2018, December 5). Retrieved from <https://digitalguardian.com/blog/what-ics-security>.

where FPGAs are fun. (n.d.). Retrieved from <https://www.fpga4fun.com/>.

7.4 Appendix 4 – Team Information Sheet

In the summer of 2019, Felipe Dale Figeman asked his employers at Cybereason if they had an idea for a year long project. Thus, CyberTap was born. The group formed and agreed to build a hardware based network tap.

Felipe Dale Figeman. Email: fdale@bu.edu

Felipe is a Computer Engineering major that has taken classes such as Embedded Systems, Cloud Computing, and Operating Systems. His role is focused on packet sniffing and being able to connect the Pis and FPGA so the FPGA may receive the packets to parse.

Alex Fatyga. Email: afatyga@bu.edu

Alex is a Computer Engineering major that has taken classes such as Operating Systems, Smart and Connected Systems, and CS Software Engineering. Her role is focused on the front end and receiving packet information into the front end. She will also be working on hosting of the web application from the PYNQ board.

Evan Lang. Email: evanlang@bu.edu

Evan is a Computer Engineering major that has taken classes such as Digital VLSI Design, Electronics and Computer Architecture. Evan created, generated, and tested the Microblaze processor used to operate the ethernet controller and run the server on the FPGA. He used Vivado SDK to create and test the echo server used as a template for the project. He will be taking the main role in packet parsing next semester.

Noah Malhi. Email: malhin@bu.edu

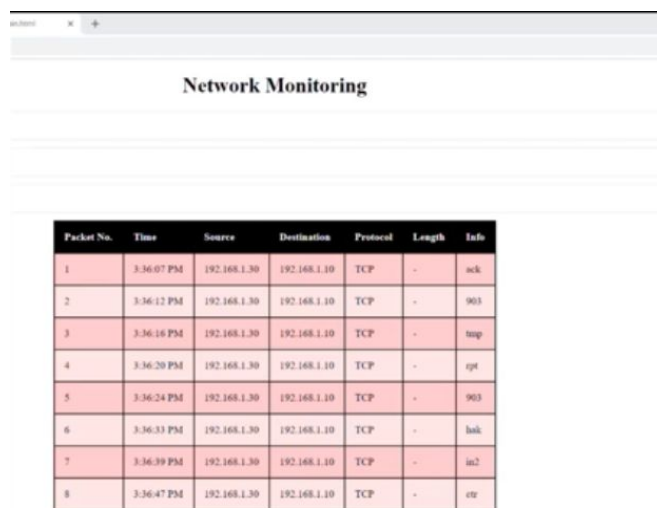
Noah is a Computer Engineering major that has taken classes such as Embedded Systems, Operating Systems, and High Performance Programming. Noah is focused on simulating the network activity with the Raspberry Pis and writing the packet information to SD and SSD. Noah has also assisted in the first semester's efforts to sniffing the packets on the Nexys A7 from the network switch.

Justin Morgan. Email: justinm@bu.edu

Justin is a Computer Engineering major that has taken classes such as Smart and Connected Systems and Operating Systems and has prior networking experience. Justin's role is to work on the packet logic after the parsing of the protocols. He will deal with generation of the metadata. Justin will also assist in writing from FPGA to the external SSD.

The entire group plans to take Computer Networking in Spring 2020 to help with the project.

Figure 8. Web Application



The screenshot shows a web browser window with a single tab titled 'Network Monitoring'. The page has a light blue header with the title 'Network Monitoring' in bold. Below the header is a table with 7 columns: Packet No., Time, Source, Destination, Protocol, Length, and Info. The table contains 8 rows of data, each representing a network packet. The background of the table rows is light pink.

Packet No.	Time	Source	Destination	Protocol	Length	Info
1	3:36:07 PM	192.168.1.30	192.168.1.10	TCP	-	ack
2	3:36:12 PM	192.168.1.30	192.168.1.10	TCP	-	903
3	3:36:16 PM	192.168.1.30	192.168.1.10	TCP	-	http
4	3:36:20 PM	192.168.1.30	192.168.1.10	TCP	-	opt
5	3:36:24 PM	192.168.1.30	192.168.1.10	TCP	-	903
6	3:36:33 PM	192.168.1.30	192.168.1.10	TCP	-	hsh
7	3:36:39 PM	192.168.1.30	192.168.1.10	TCP	-	in2
8	3:36:47 PM	192.168.1.30	192.168.1.10	TCP	-	ctr

The web client of the system which allows users to view up-to-date and older network activity and search through the packets.