Exploratory Data Analysis of Network Traffic in Industrial Control Systems

Carlos Ortega, Anvit Srivastav, Priya Ahuja,
Michael Christian, Alvaro Cardenas

August 21, 2014

Abstract

In this paper we have presented the different network monitoring tools and Network security monitoring tools for Intrusion Detection based on our study and evaluation. We analyzed these tools by working on sample data sets in order to get familiar with the framework and analyze the pros and cons of different tools. Later we used one of these tools to derive results from Industrial Control systems data which was 200GB in size. We derived useful results using these tools such as network architecture and various useful graphs for analyzing the behaviour of network traffic.

1 Introduction

The different network monitoring tools that we worked on are Zenoss,Snorby,Sguil and Splunk.We also analyzed and worked with Bro IDS which is a network security monitoring tool for Intrusion Detection systems.

Our goal was to study,evaluate and compare these available network monitoring tools.First we worked on all these tools with sample data sets in order to evaluate the tool based on its performance and utility. Later, we used one of these tools to study and derive results from the Industrial COntrol Systems data which have been summarized below in Section 5.

We used Bro IDS to generate Bro logs for 200GB data from Industrial Control Systems. We used logs generated from Bro to generate useful results

by writing python scripts.The results obtained have been summarized in Section 5 of the paper.

In Section 2, we have summarized our ananysis of Zenoss, Sguil, snorby and Splunk. In Section 3, Bro IDS ans the exercise which was performed to study it has been explained.In Section 4, we have summarized the Comparision of the different tools studied based on the features that each of these provide.Section 5 consist of the results obtained from the study of Industrial Control systems data.

2 Network Monitoring Tools

2.1 ZENOSS

2.1.1 Introduction

Zenoss is an open source network,application and server management platform which is based on the Zope application server. Zenoss Core provides a web interface which allows you to monitor availabilty,functionality,performance and events of networks,servers,services and applications.
Zenoss has the monitoring availabity of network devices using SNMP, SSH and WMI, network services such as HTTP, POP3, FTP and SNMP as well as host resources such as processor and disk usage on most network operating systems.Also, Zenoss has the capabilty to automatically discover network resources as well as any changes in the configuration of the network.

Zenoss can be downloaded from here [2].

It is a single product that monitors real and virtual servers and network devices.Zenoss uses a model driven architecture.Zenoss builds a model for every device it monitors to drive the monitoring.For any device we can see the availabity and uptime,CPU Utilization, the status of key components, the OS system version and the system information. It is easy to identify if all the servies are performing properly and also helps us to identify the software installed.In addition to availabity and configuration information it provides about varoius devices , it is comprehensive event manager and it collects events both from system log files and also generates its own events when its model driven policy detects an issue such as a performance threshold violation. Zenoss analyzes and collects performance information and

charts it.You can choose a longer time period to see if current performance is not normal. And look at both OS and application metrics side by side to see correlation.Zenoss supplements system provided metrics with its own sysnthetic transations that directly measure application performance and it includes predictive thresholds that automatically alert you about abnormal behaviour even if you don't know what normal is.

There is a traffice light signal which shows and alerts about any issues with the device or network. Zenoss manages both devices and networks that are ON. It models upstream and downstream connections for deives. So if a Router fails you do not get availabity notifications for every device downstream of the failed router.One of the most powerful feature of Zenoss is the ability to look at the network from the presepective of the device.Any network issues are automatically notifies of to the user associated thus helping with the complete monitoring of devices as well as issues generated.

As a part of learning exercise we analyzed a virtual machine using zenoss. All the informaion regarding the machine like the status of the machine, intefaces, devices connected, CPU utilization was studied. Also the network map of the machines showing network configurations and connections was analyzed. When we talk about using Zenoss for analyzing any existing data, it does not provide us the flexibilty to do the same. Therfore, with Zenoss you can only monitor deivces or network tht are physically associated with your machine.

2.1.2 Technologies used in Zenoss

The different technologies used in zenoss are aa follows:
a. Zope - It is an application server written in Python
b. Python - It is an extensible programming language used in Zenoss.
c. SNMP - Monitoring protocol that is responsible for collecting status information of systems
d. RRDtool - This is the tool that is responsible for generating graphs in Zenoss.
e. MYSQL - An open soure database which is used in Zenoss for maintaining data about configuation and availabity of devices and networks present.

2.1.3 Key Features

a. Discovery
b. Availability Monitoring
c. Performance Monitoring
d. Event Management

2.1.4 ZenPacks

A ZenPack is an extension to Zenoss. It is a way of adding the abiltiy to monitor new kind of devices and also to add new capabilties to Zenoss itself.Since there are a variety of hardware and software that need to be monitored , Zenpacks allow us to monitor everything.There are hundreds of ZenPacks available, some developed, supported, and maintained by Zenoss, and many others that are developed and maintained by the Zenoss user community.

Earlier we talked about Zenoss not supporting monitoring of existing network data. With the use of Zenpacks we can however achieve this functionality by using Splunk Zenpack. The Splunk ZenPack allows you to monitor the results of a Splunk search using Zenoss.

More information on Splunk can be found here [1].

2.2 SGUIL

2.2.1 Introduction

The sguil client is written in Tcl/Tk and can be run on any operating system that supports Tcl/Tk. Sguil is described as an aggregation system for network security monitoring tools.Sguil simply ties together the outputs of various security monitoring tools into a single interface, providing us with the most information in the shortest amount of time and at a same place.It gathers all the information related to IDS alerts, full content packet logs and all other useful information into the database of TCP/IP sessions.When you indentified any kind of IDS alert, Sguil client allows you to access and query the database and decide on how to deal with the situation.

2.2.2 Architecture

A Sguil system consist of a sguil client, sguil server and a number of sguil network Sensors.The sguil sensors perform all network monitioring tasks and on regular basis they feed information gathered back to the server. This information is then managed and stored in the database at the server. The Sguil clients can then query the desired information from the server database.

2.3 SNORBY

2.3.1 Introduction

Snorby is an open source ruby on rails web application for network security monitoring that interfaces with current popular intrusion detection systems (Snort, Suricata and Sagan)

2.3.2 Features

Snorby is front end web application for any application that logs events in the unified2 binary output format. It doesn't perform any network monitoring tasks by itself, instead it depends on other IDS's such as Snort to report data to it. Since snorby itself is just a dashboard, it allows them to have sensors for events of their choice or even have custom sensors in the IDS of their choice. It also supports full packet capture so that the users can see complete packets which makes it easier to determine if an attack occurred.

Snorby can generate hourly/daily/weekly or monthly graphs and reports for monitoring data. It allows classification of events based on preset classes or based on user defined custom classes. It also features hotkeys which allow users to quickly borwse, view and classify events in the web application without the use of the mouse which makes working with it exremely fast and responsive.

2.4 SPLUNK

2.4.1 Introduction

Splunk Enterprise is the leading platform for real-time operational intelligence. Its a fast and secure way to search and visualize massive streams of data. Splunk was founded to pursue a disruptive new vision: make machine data accessible, usable and valuable to everyone. Machine data is one of the fastest growing and most pervasive segments of "big data"—generated by websites, applications, servers, networks, mobile devices and all the sensors and RFID assets that produce data every second of every day. By monitoring and analyzing everything from customer clickstreams and transactions to network activity and call records—and more—Splunk turns machine data into valuable insights no matter what business you're in. It's what they call Operational Intelligence.

2.4.2 Features

There are many features within Splunk, one of the most important is Universal indexing. It's a way of aggragting all of the data from a specified data source. Also with universal indexing data can be index in real time. Another nice feature of Splunk is the Search feature. Once the data is index it can then be searched to find potential threats, attacks, viruses ...etc. this makes for a very efficent way to find your problems.

3 Bro IDS

3.1 Intro

The Bro Network Security Monitor is a powerful tool used for intrusion detection. There are two essential components that make Bro the powerful tool it is. The *Event Engine* is used to transform the packet stream into events, which abstracts much of the complexity of dealing with packets. It is the job of the *Polocy Script Interpreter* to decide what to do with the events.

3.2 Bro IDS homework

The homework was to do the exercises that were part of a workshop done in 2011. Below is an overview of these exercises. The first part covers the first four exercises. The second part covers the last three exercises.

3.2.1 Part I

In this section we will discuss the first four exercises of the Bro workshop. We will go over the set up of bro and what I think is good practice for beginners. Bro is a powerful network analysis framework that is self-proclaimed to be very different than the normal IDS and I would agree.

The first exercise in Bro is the getting started exercise. This is by far one of the simplest exercises out of the four. In this exercise we start by configuring the Bro source code. Steps one through four are only needed if you are doing a fresh instal of Bro, in my case I installed Bro along with the Security Onion application. In steps one through four we set paths and privileges for Bro. If you do not complete step number four on the Linux system then you will have to run the first bro command along with the Sudo command. Step four gives Bro the necessary privileges to run as the "root" user. Step four was the only step I found to be necessary to run if you already had Bro installed. You will also want to complete steps five through eight as well, running through these steps will get you familiar with using BroControl and running Bro directly along with teaching you what log files are and what they look like. BroControl is an interactive shell for operating Bro's installation. You may also run Bro from inside the interactive shell(BroControl). You may also run Bro directly from a temp folder. When running Bro directly it

will also store the log files in the working level directory. It would be good practice to learn how to run Bro from both BroControl and directly from a temp file. Once you understand what Log files are you can start to parse out information you need using the Bro-Cut utility. Bro-Cut is a utility that can be used to parse out specific information out of Log files.

Exercise two "Understanding and Examining Bro Logs" gives you an in depth look into Bro Log files. It will show you how Log files are generated using the –r command in Bro. Bro is filled with helpful commands and you can view them all by running the –h command. This section also shows you how to parse information from whichever Log file you would like information from. You can list the requested information in descending order as well as ascending order. This will be done using your "awk" tool for processing data you could also do this with Bro-Cut as mentioned earlier. Bro calls the Awk tool their "Swiss army knife for log processing". This tool allows users to quick retrieve any information he/she needs from the Log files. Using this tool will allow you to find all connections lasting longer than one min, find all IP addresses of web servers that send more than more than 1 KB back to a client, and show a breakdown of the number of connections by service.

The next exercise was very interesting because it's the first time we have seen Bro in action so far speak. Exercise three teaches us about notices and alarms.

Exercise three "Handling Notices" is where things started to get interesting to me, because it was able to give me real time feedback and by default Bro does some analysis on some Network traffic by itself and determines if there's any interesting information for you to look at. This interesting information is also known as Notices in Bro. In this exercise you will learn how to run Bro on a network trace and see if there's any suspicious activity currently on the network. Lastly this exercise will teach you how to write policies and how to get Bro to trigger alarms.

In exercise four "Bro Programming Primer" you will begin to learn the Bro scripting language. You will look at tables and vectors,and Records along with Functions. I believe spending plenty of time on this section will be of good value because you havet to become pretty familiar with the language to get Bro to do what you want it to do any way and this is a great place to start.

After doing these exercises I can tell you that some were a piece of cake and where just a matter of doing exactly as the instructions directed me to and others where the exact opposite with no solution online. In exercise one the only issue I ran into was due to not running step four, which allowed Bro to run as the "root" user. I solved this problem by running Bro with the Sudo command to give it the required permissions. In the first exercise I also found that beyond the bro documentation found on the Bro website there isn't much help out there and that was pretty consistent throught the rest of the excercises. Exercise two's final step gave me problems because I wasn't exactly sure how to run the script that tells Bro to change the separator for Bro. I solved this problem by downloading the file for the exercise and saving it in the working directory, then simply run the script provided.

3.2.2 Part II

The final exercises focused more on real world situations and build up from the first exercises. Overall these where very engaging, however some of the solutions were outdated. This caused some confusion that will be discussed further on.

The fifth exercises was arguably the most confusing to complete. This is not because it was difficult, but rather because the topics are covered in this exercise are not adequately discussed in the bro videos. The first part deals with certificates and how to allow Bro to trust custom certificates. Essentially this part just consists of running a few Bro command lines. The first command was to run Bro with a script that is somewhere in the installation directory. After seeing the solution the location of the scripts can be found by doing a find on the script that is used in the solution. These ssl Bro scripts in Security Onion are found here: /opt/bro/share/bro/policy/protocols/ssl/. The rest of the first part consist of the same type of problem. The last two parts of this exercise dealt with events. The actual Bro scripts that are needed to solve the problems are quite simple, but doing event scripts for the first time can be quite frustrating. It is important to note that the solution to the second part will not work if the version of Bro is 2.2+. That being said, the problem with the solution is that the parameters for the x509_certificate event have changed since Bro 2.0. Looking up the updated event in the doc-

umentation will help solve the problem.

Perhaps being one of the funner exercises, the sixth exercise was the most engaging to complete. This exercise consists of five parts, all of which revolve around a central story of a company, Huge Big Dairy, and how their security has been compromised by the hacktivist group Synonymous.

Part one deals with web authentication. The goal is to find the web server that uses weak authentication using the pcap file provided. After that it is possible to find the users password by telling bro to capture passwords and run Bro with the pcap file again. The second part is where things begin to get more interesting. This time another pcap file is given and the goal is to find the file that was leaked and determine the type of attack. Finding the file is not too complicated, but knowing the name of the attack can be tricky if someone is not familiar with the semantics of security. After figuring out that this was a directory traversal attack the next step is to see if it was successful. This can be done by looking at the status code of each HTTP request. In the end it is possible to see that there was a successful attempt at fetching the classified file.

The third part of the exercise has to do with email. The information that is provided states that HBDairy believes someone forwarded sensitive information using unencrypted email. A pcap file is provided, to solve this the SMTP traffic needs to be analyzed. At first this seems pretty straight forward. However using the solution provided does not work because of some minor differences between Bro version. For example, after finding a potential UID from the SMTP log, the provided solution is to search for that UID in the smtp_entities.log file, a file that is not created in new Bro versions. Instead the UID needs to be searched for in the files.log file. After this the file that was emailed needs to be extracted. The solution that is given does not work anymore but there is another solution that will extract all the files here:
http://www.bro.org/bro-exchange-2013/exercises/faf.html.

The last two parts of exercise six deal with examining the dns log and http log files. In the fourth part the goal is to figure out why the employees cannot access youtube.com. After that you need to see how long the downtime was. The last part deals with a CSRF attack done by Synonymous to an employee.

The last exercise is the most challenging and advance. It is definitely one to revisit. The first part is to make a sidejacking detector by basically seeing if a cookie is being reused. The second and last part is more complicated and is to make a webcat analyzer for Facebook.

3.3 Afterthoughts

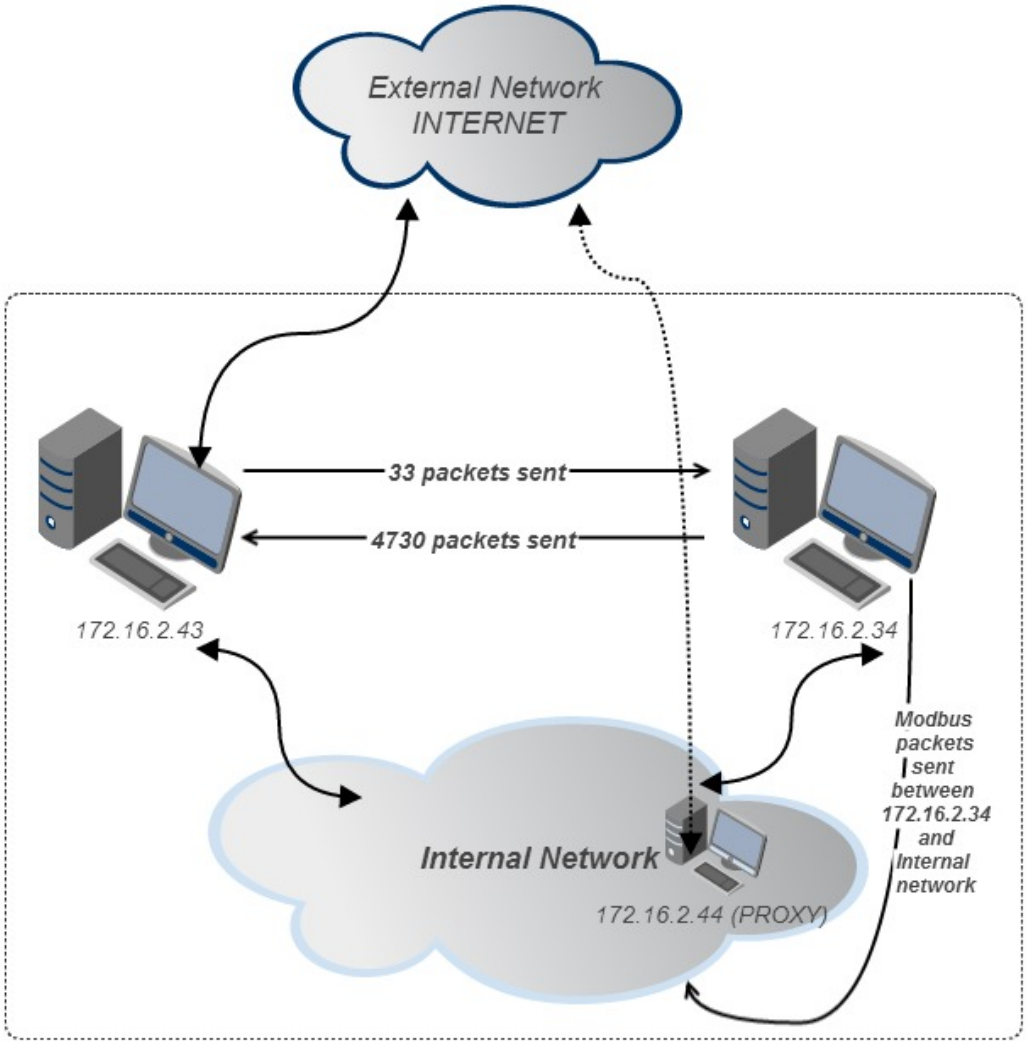
Overall after finishing the exercises there is a greater appreciation of Bro. Although there are some aspects of Bro that can be daunting to understand, overall Bro is very versatile and can do a lot more things than what was covered in the exercises. The only potential issue is that while Bro does contain a substantial amount of documentation, some of the documentation can be difficult to follow. This causes a trial and error approach to using and learning new parts of Bro.

4 Comparision of Network Monitoring Tools

Features	Zenoss	Snorby	Splunk	Sguil	Squert	OSSIM
Platform	Puthon,Java			Tcl/Tk		
AutoDiscovery	Yes					
Data Storage Method	SQL		Flatfile	SQL		
Plugins	Yes (ZenPacks)		Yes			
Triggers/Alerts	Yes	Yes	Yes	Yes	Yes	
Web-Based				No		
Generate Graphs	Yes	Yes	Yes	No	No	

5 Results : Analysis of Industrial Control Systems Data

5.1 Figure 1 : Network Architecture



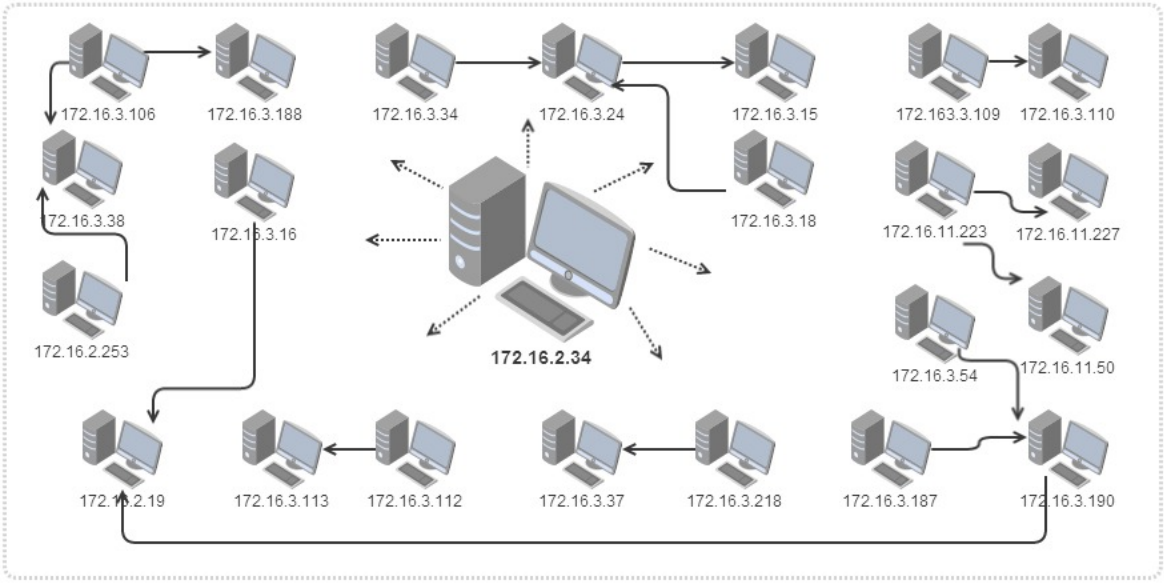
This figure summarizes the picture of the netowrk architecture that was obtained after studying and getting results from the ICS data.There were

different types of protocol used for communication in the network like TCP, UDP, ICMP, IGMP, SNMP and MODBUS.

The MODBUS traffic flow was observed to happen between the 172.16.2.34 and the internal network IPs.More detailed picture of network flow is given in Figure 2 below.

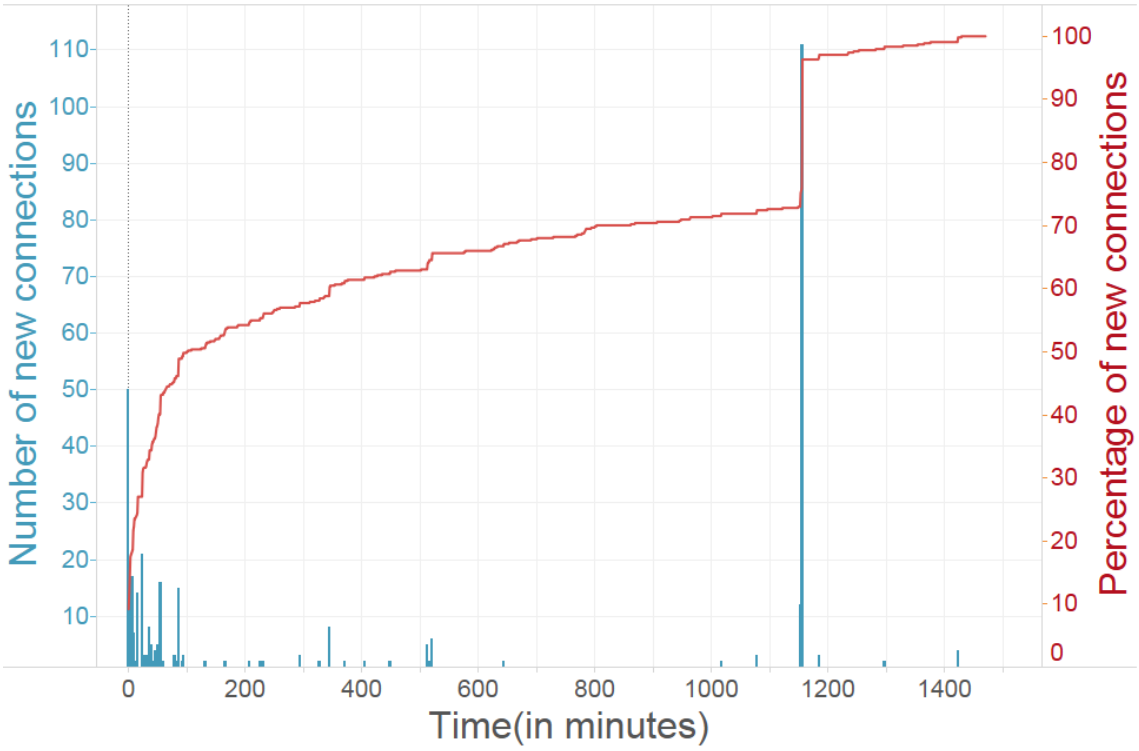
As shown in the figure, There were 2 IPs that were dominating the entire communication 172.16.2.43 and 172.16.2.34.Both, 172.16.2.43 and 172.16.2.34 talk each other as well as other Internal network IPs. All communication with the outside world was done only through the 172.16.2.43. However, 172.16.2.43 was not at all involved in the MODBUS packets communication. Based on the study, we observed that there was one IP address - 172.16.2.44 which was the only IP other than 172.16.2.43 that communicated directly with the outside world, this seemed to serve as a PROXY. Thus, this is the brief network picture that was obtained after studying the data from ICS.

5.2 Figure 2 : Modbus Traffic Flow in the Internal Network



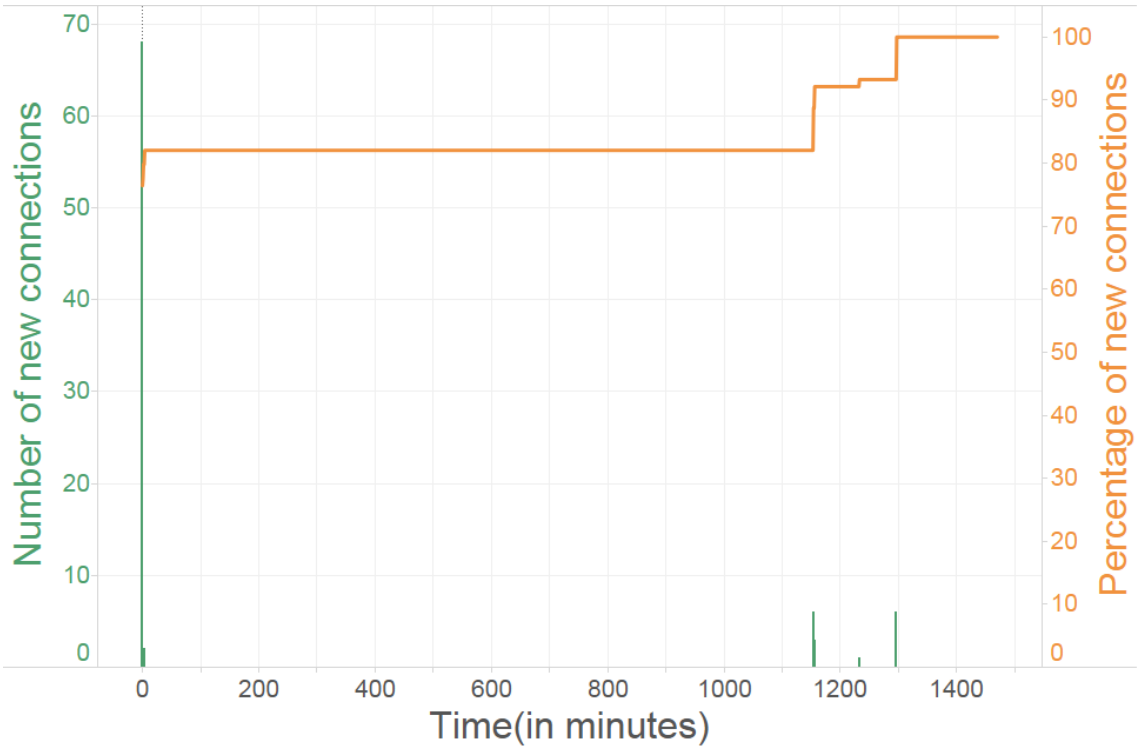
This Figure depicts the MODBUS data flow between 172.16.2.34 and the internal network.We can see from the figure that 172.16.2.34 communicated with everyone in the internal network. Whereas, there are few communication in which 172.16.2.34 is not involved. Such communications which do not involve 172.16.2.34 have been shown in the figure clearly. 172.16.2.43 is not at all participating in the MODBUS traffic flow.

5.3 Figure 3 : New non modbus connections per minute



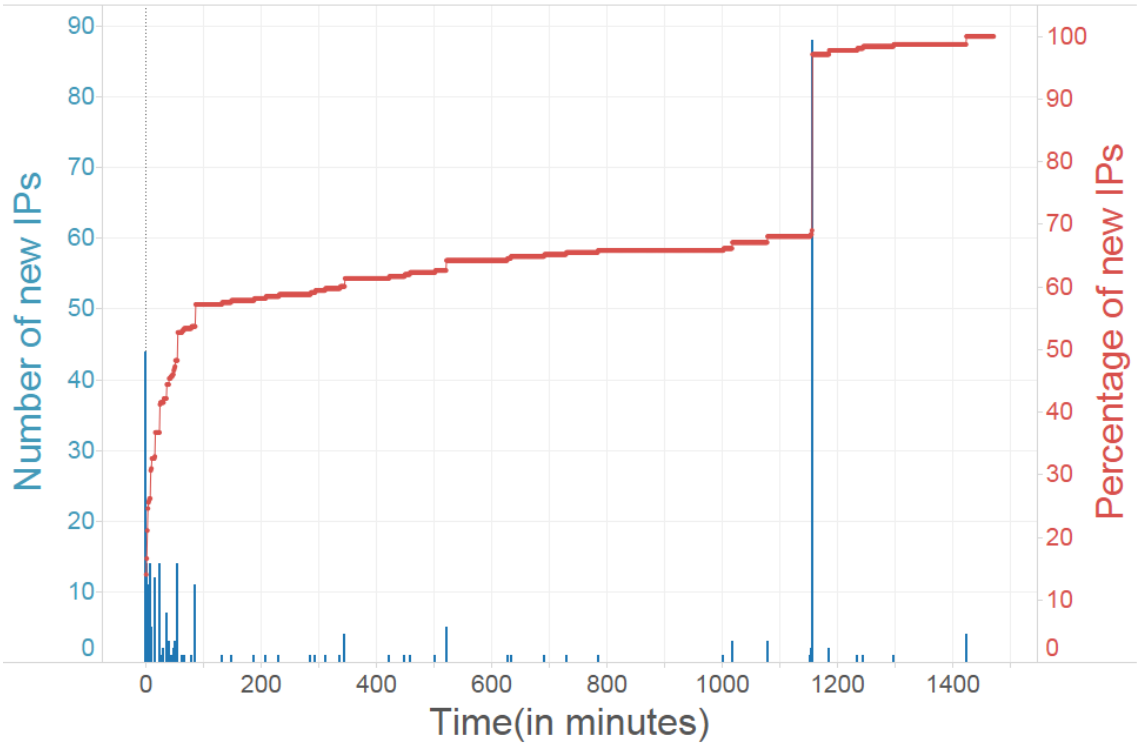
It can be inferred from the figure that the majority of connections are formed within the first 1.5 hours. After that there are a few new connections that take place intermittently. However, there is a sudden surge in connections at 1156 minutes when 111 new connections were formed.

5.4 Figure 4 : New modbus connections per minute



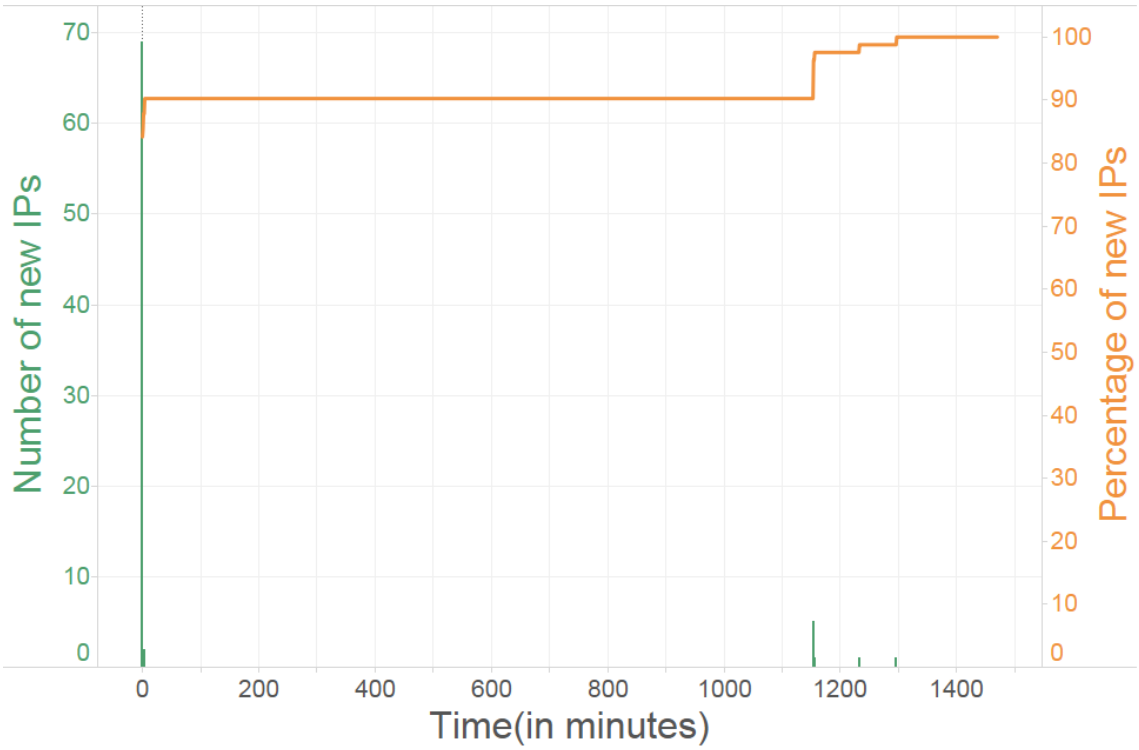
As can be seen from the graph, almost all the connections are made within the first 3 minutes. In fact, the data showed that most of the connections were made in the first 6 seconds. After the first 3 mins, there are absolutely no new connections except in the interval 1154-1297 minutes 4 small peaks were observed. It is interesting to note that the first and the second peak occur at 1154 and 1156 minutes, which is almost the same time period where the peak from the non modbus connection graph was observed.

5.5 Figure 5 : New non modbus IPs per minute



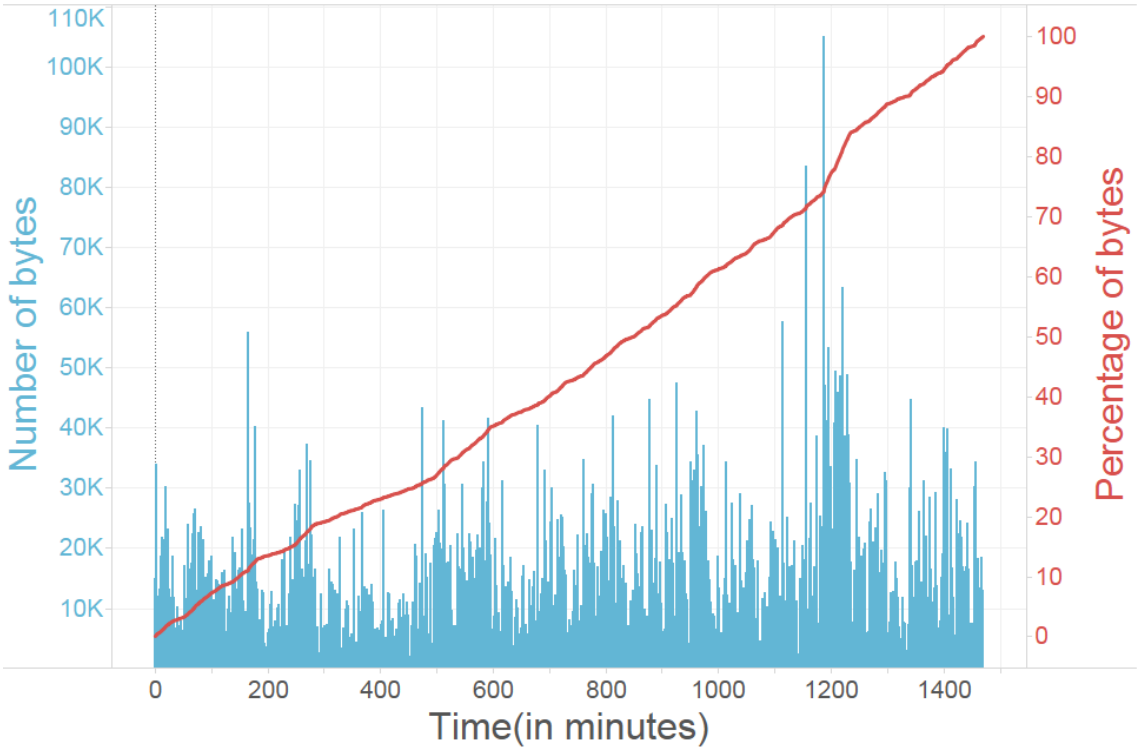
This graph is similar to the non modbus connection graph for the most part. It shows that most of the new IPs are discovered within the first 1.5 hours after which a few new IPs are discovered intermittently. It also features a sudden surge at 1156 minutes when 88 new IPs are discovered. It can be inferred that most of the times that a new connection is made, the IPs being connected to are new IPs.

5.6 Figure 6 : New modbus IPs per minute



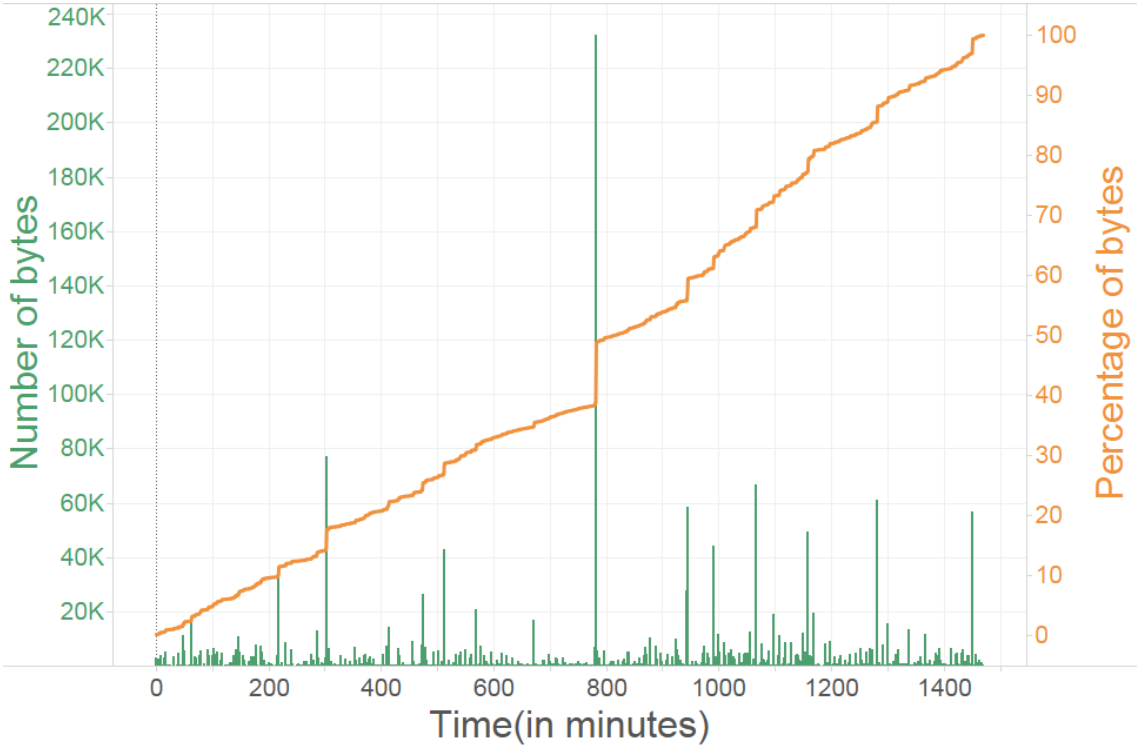
It is interesting to see that this graph also runs parallel to the new modbus connection graph. The majority of connections take place within the first 3 minutes and then there are no new IPs except the 4 small peaks observed around 1154-1297 minutes. Hence, even for the modbus connections, most of the connections are being made to new IPs.

5.7 Figure 7 : Number of non modbus packets per minute



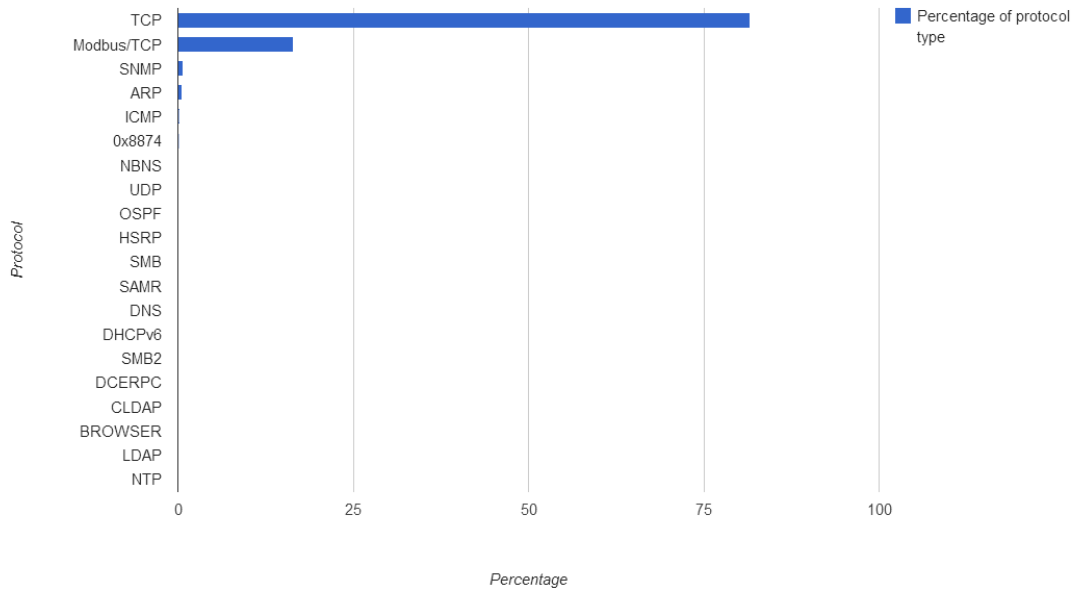
The number of bytes being transferred for non modbus packets fluctuates but for the most part, the number of bytes being sent per minute stays under 20,000 and it shoots up close to 30,000-40,000 intermittently. There are few peaks which shot above even 50,000 bytes a minute in the range 1115-1220 minutes.

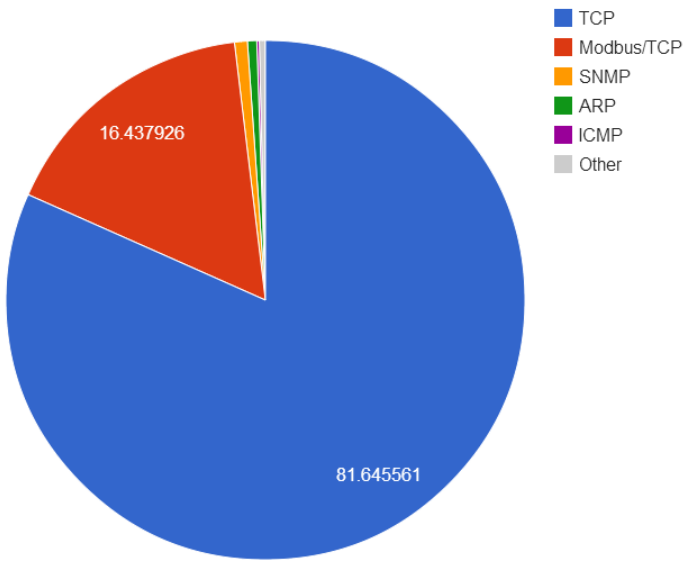
5.8 Figure 8 : Number of non modbus packets per minute



Unlike the non modbus graph, the number of bytes transferred over modbus stayed within 5,000 and it crossed 20,000 only 13 times. However, There is a huge peak at 782 minutes when 232,314 bytes were transfered.

5.9 Figure 9 : Different Protocol Types with their Percentage




TCP
Modbus/TCP
SNMP
ARP
ICMP
Other
16.437926
81.645561

Protocol	Percentage
TCP	81.645561
Modbus/TCP	16.437926
SNMP	0.798312
ARP	0.578608
ICMP	0.151973
0x8874	0.122386
NBNS	0.052091
UDP	0.031368
HSRP	0.030064
SMB	0.029413
SAMR	0.024677
DN	0.013685
DHCPv6	0.010818
SMB2	0.007777
DCERPC	0.00517
CLDAP	0.00391
BROWSER	0.003041
LDAP	0.002998
NTP	0.001651

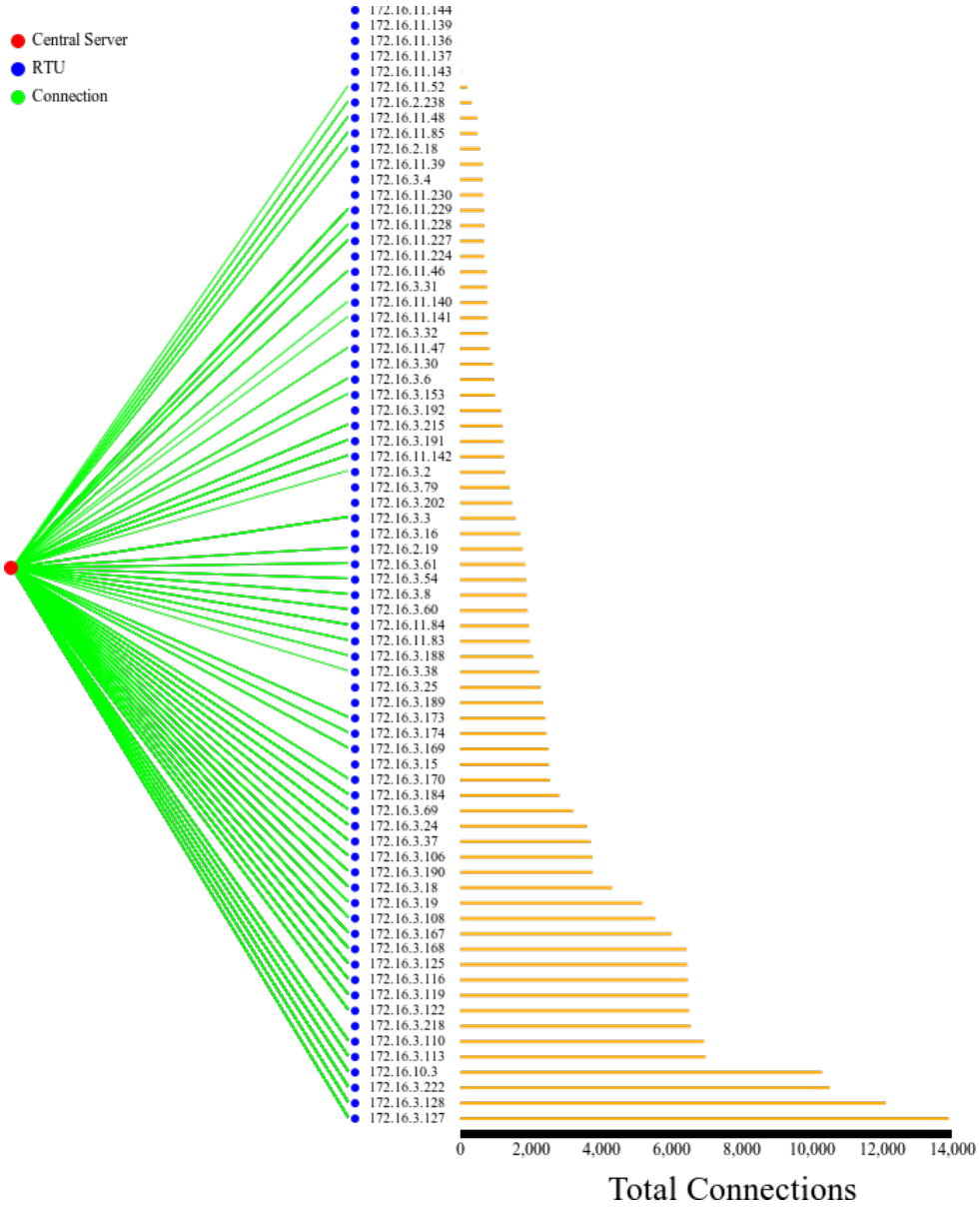
6 Modbus Visualizations With D3 and Gephi

In order to analyze and visualize the network data, Bro was used to create logs. The Bro logs made it possible to extract the relevant information much easier and faster than if we used the pcap files directly. Since there were 166 pcap files, there were 166 sets of Bro logs. Each Bro log contained logs for the modbus, dns, and connection, as well as others. Since the logs were easy to parse, using python it was possible to extract relevant information and create json or csv files to be used with D3 or Gephi.

6.1 The First PCAP File

In order to have an understanding of the modbus network, the initial exploratory analysis was of the first pcap file. This served as a way to try different theories before committing to the entirety of the data.

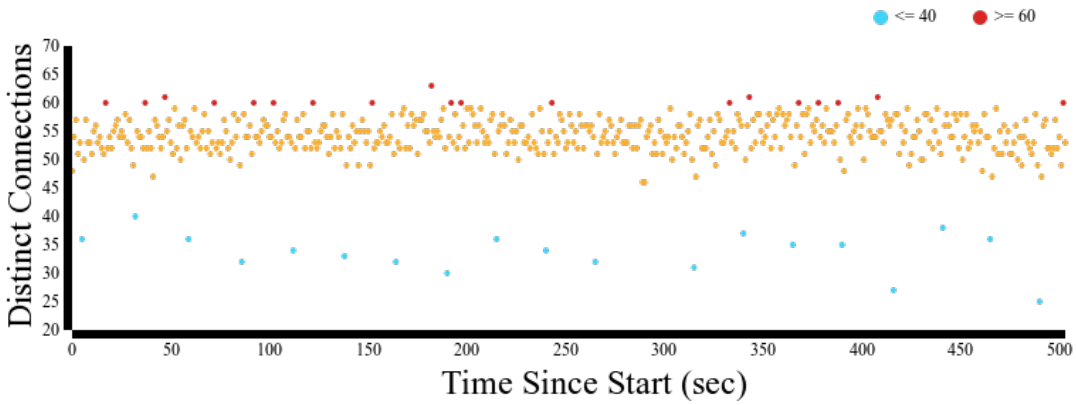
6.1.1 Figure 9 : Distribution of Modbus Connections



This image was taken from an animation done in D3 showing the connections from the central server to different RTUs. In this image you can see

how in the last second of the first pcap file there is a large disparity in the number of connections being made to the different RTUs.

6.1.2 Figure 10 : Distinct Modbus Connections at Each Second



In this graph we observe the distinct modbus connections for the first pcap file at each second. This means that each second we count the number of distinct connections for that second. When graphed it is possible to see that there is a uniformity in the way that the network behaves. What is interesting to see is the blue dots, which represents when the number of distinct connections drops significantly. This happens about every 26 secs starting at second 5.

6.2 Using The Entire Data

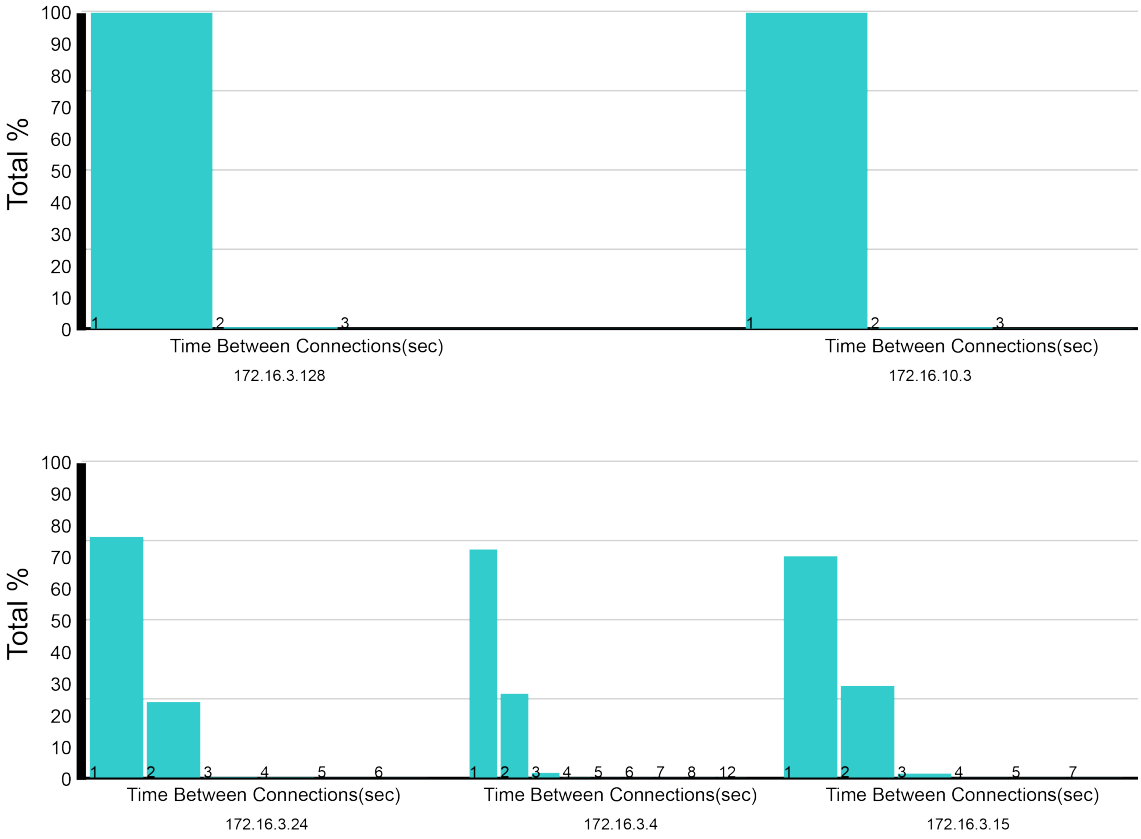
After examining the first pcap file, attention was shifted into analyzing all of the data. Using what was learned from the previous examinations from the first pcap file we further examined the data to see what was the pattern of connections.

6.2.1 Frequency of Modbus Connections

The next set of graphs show how often the main server was communicating with the RTUs. What this means is that the graphs show how often sets of

communications occur. It is important to note that 0 secs is not included. This is because it was observed that "bursts" of connections to a single RTU occur within a second. For example the server might communicate with a single RTU 300 times in one second, this "burst" might not be seen again for 5 secs. This turned out to be a fair assumption to make, however it did not always provide clear results.

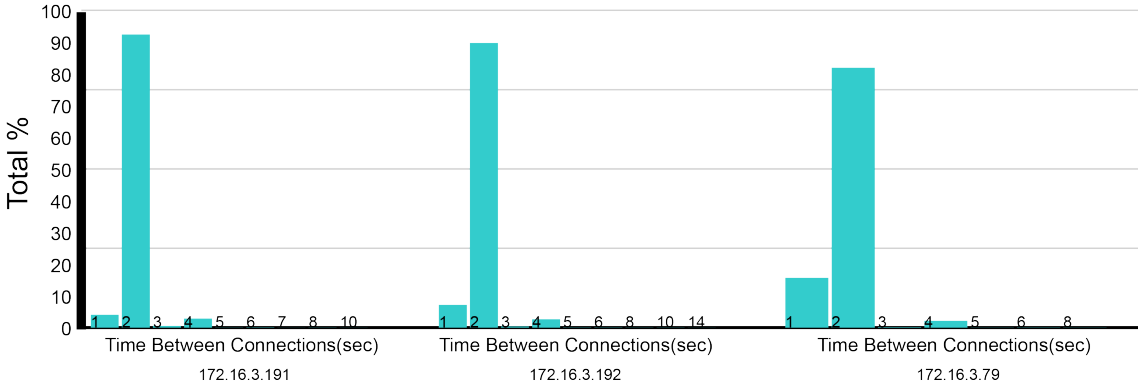
Figures 11,12 : Selected RTUs With Majority of Intervals of 1sec



The top image shows two of the addresses of the RTUs that have the top total connections as seen on Figure 9. As you can see nearly 100"%" of the time there are communications every second. This might also indicate that these RTUs are critical to the system and therefore need constant monitoring.

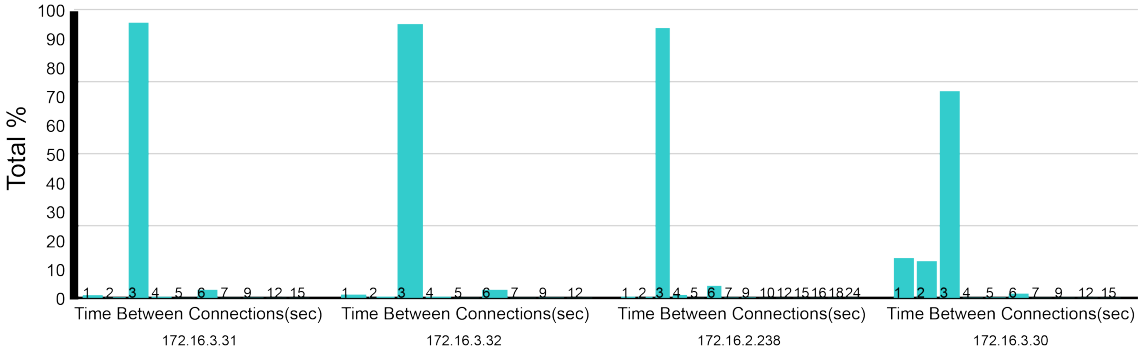
The bottom image shows different addresses of the RTUs that have about 75"%" of their intervals at 1 second. The remainder is mostly of 2 seconds. At first this seemed odd, but since we are using a second as our time frame, what seemed to happened was that a series of communications spanned more than one second. For example let A be an RTU. Now lets say that the server started a "burst" of communications with A at time x and finished at time x+3. It then continued to make the same "burst" every 2 seconds. Based on the current way that the graphs were made, there would be three 1 second intervals and just one 2 second interval. This suggests that using a second as the time frame might be too large.

Figure 13 : Selected RTUs With Majority of Intervals of 2sec



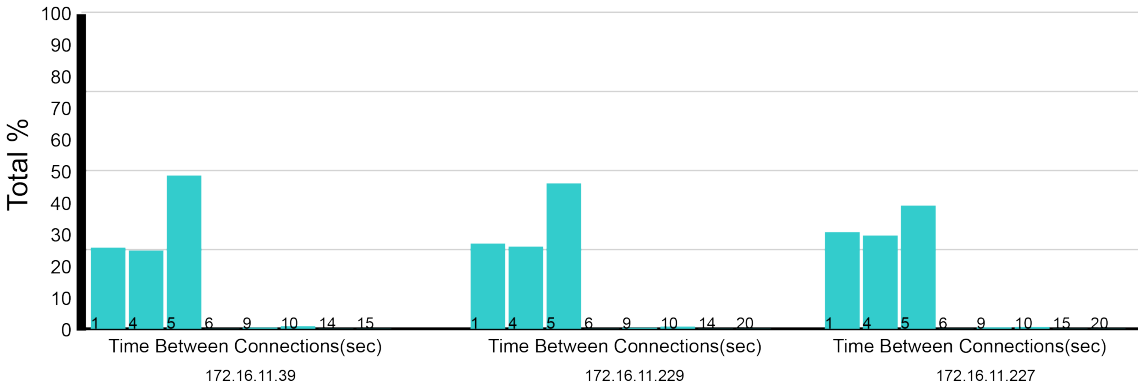
Here we have some RTUs which the server communicates to for the most part every 2 seconds. This further shows how the modbus network is rigid, controlled, and to an extent predictable.

Figure 14 : RTUs With Majority of Intervals of 3sec



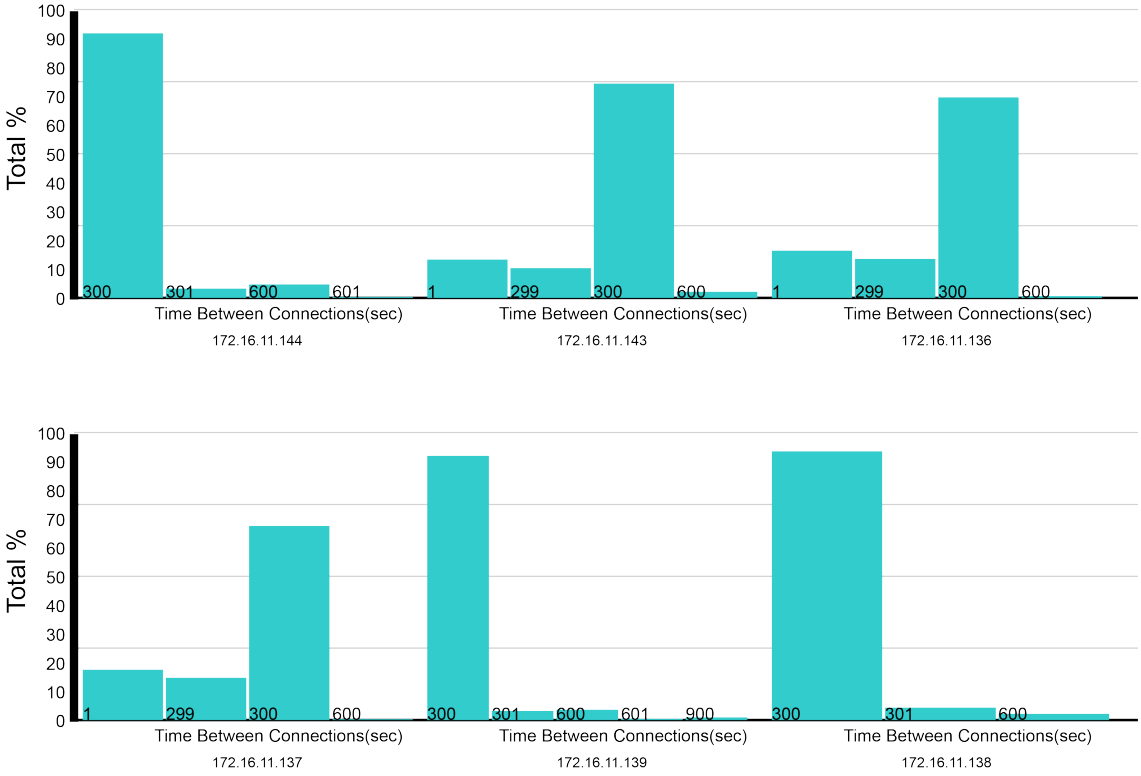
This image is similar to Figure 13 except that the communications happen every 3 seconds.

Figure 15 : RTUs With Majority of Intervals of 5sec



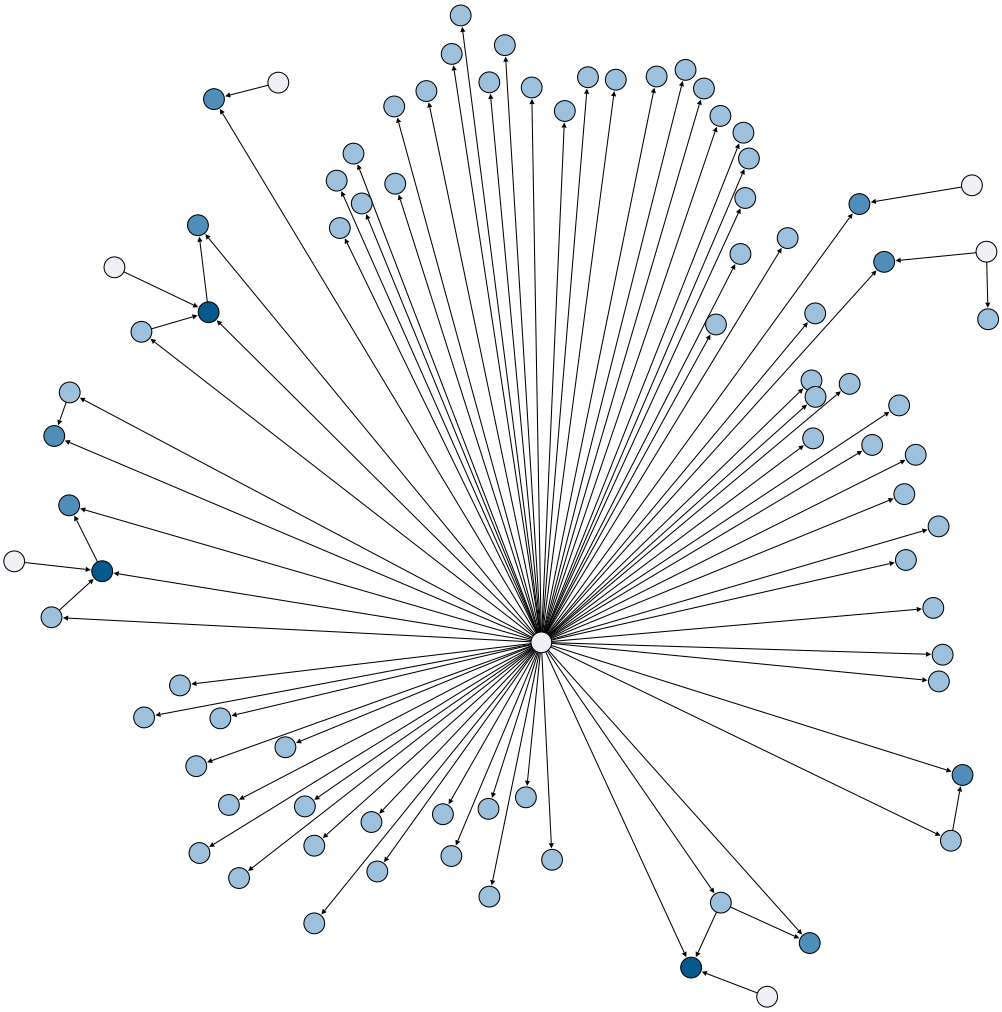
These RTUs also seemed to behave in an odd way. However just like Figure 12, the time frame of 1 second might not be small enough to have conclusive results.

Figure 16,17 : RTUs With Majority of Intervals of 300sec



The last set of RTUs are very unique since the server communicate with them almost every 300 seconds. A possible explanation for this is that these RTUs are not critical to the system and therefore do not require constant monitoring.

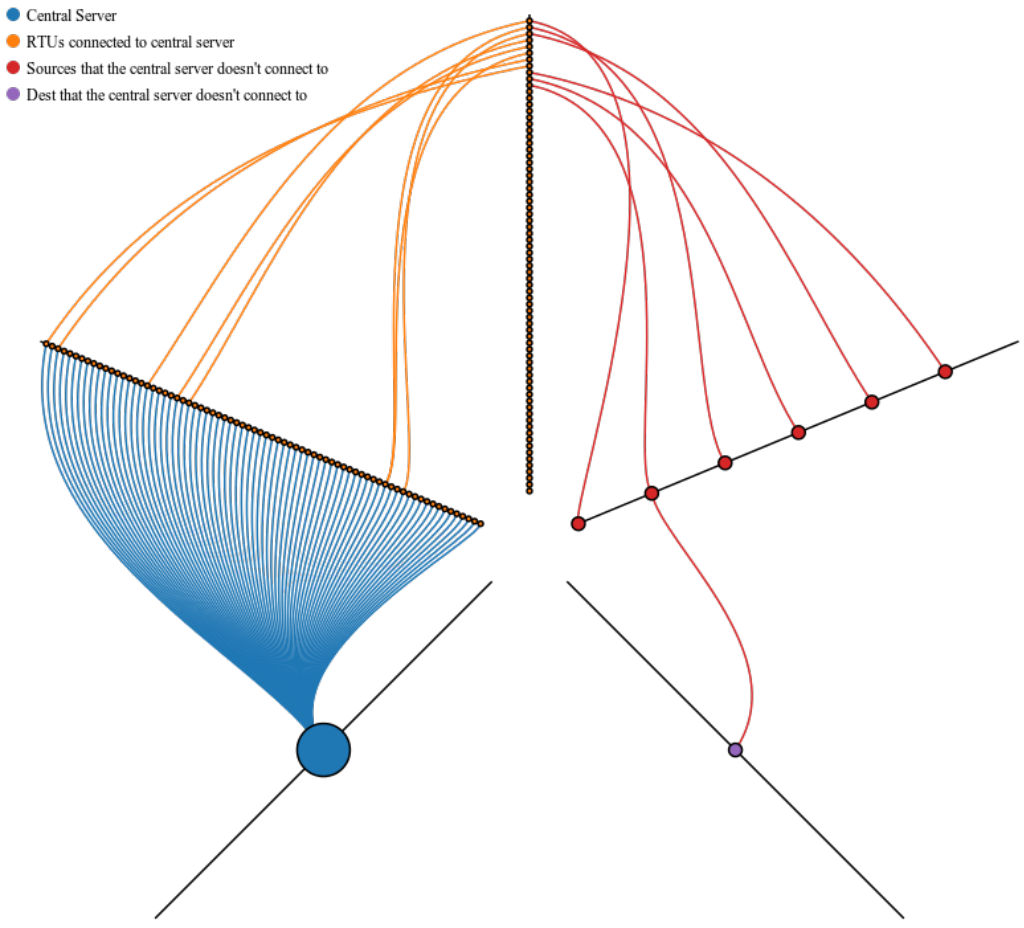
6.2.2 Figure 18 : Complete modbus network



This graph was made using Gephi. The Graph shows the complete modbus network. The nodes are color coded based on indegree where white is 0, light blue is 1, medium blue is 2, and dark blue is 3. These represent the number of sources that connect to that particular node.The center white circle represents the central server which can be seen in Figure 2 with address 172.16.2.34. As you can see most of the connections happen from the central

server to different RTUs. However there are 7 other nodes that the central server does not communicate with. It is also possible to see that some RTUs communicate with each other.

6.2.3 Figure 19 : Hive plot of the complete modbus network



This graph shows the same information as Figure 18 but in a way that shows the connections between RTUs more clearly. There are 8 different connections between the RTUs. These are represented by the yellow lines. The graph also shows the addresses that the central server does not communicate with. These are the nodes in red and the one in purple. It is clear

that there are some communications from the red nodes to the RTUs(yellow nodes), these are represented in red lines. There is also a single connection from the red nodes to a purple node. More analysis would need to be done to conclude the purpose of the red and purple nodes(address). One possibility is that these addresses may also be RTUs.

References

[1] Splunk. http://www.splunk.com/.

[2] Zenoss user community. http://www.zenoss.org/.