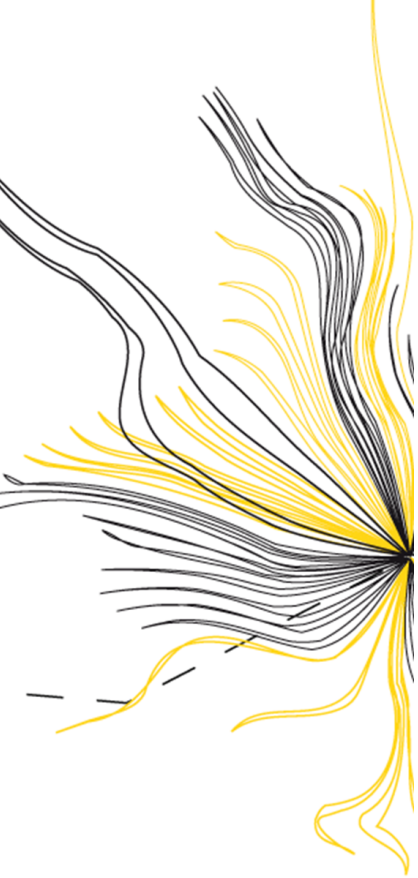


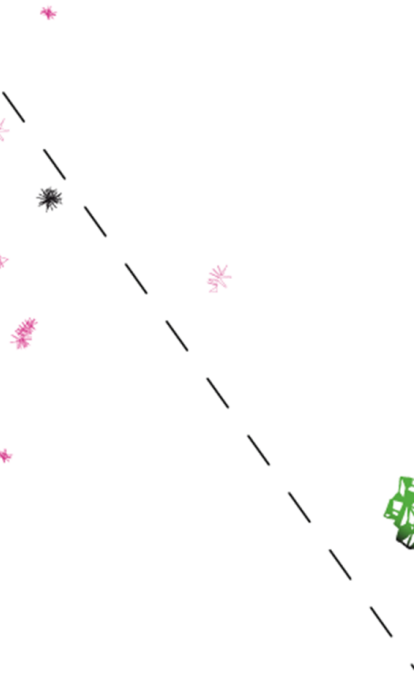
UNIVERSITY OF TWENTE.

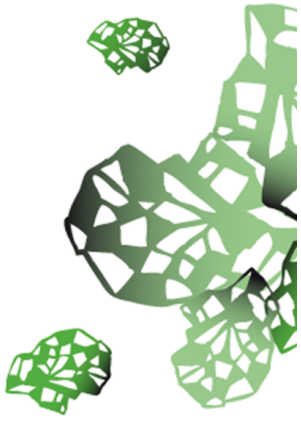
Faculty of Electrical Engineering,
Mathematics & Computer Science

Traffic Filtering Based on Subsystem Component State

Alpha O. Diallo

M.Sc. Thesis
August 2017





Supervisors:
prof. dr. ir. B.R.H.M. Haverkort
prof. dr. A. Remke
M.Sc. J.J. Chromik
Ries van Son - Compumatica Secure Networks

Design and Analysis of Communication Systems Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Abstract

Firewalls while able to filter traffic for which they have rules for are susceptible to allowing traffic that could have negative impacts on the state of an industrial control system (ICS). In order to be able to block traffic that looks legitimate but may cause the ICS system to go into unwanted states the firewall needs to consider the current state of the system and how the traffic may change this state. In this thesis the 1996 IEEE 24 bus one area reliability test system and the IEC 60870-5-104 protocol are used to represent an ICS power system and the traffic format respectively. Two methods, minimum and maximum bus connections, and branch power correlations, are used to define critical components that provide information about the ICS system. These critical components are modeled within the firewall and checked for violations whenever traffic that can change the system, identified through inspection of fields in the IEC 60870-5-104 protocol, is processed. The false positive and negative rates, and accuracy of the firewall are evaluated for different cases where critical components are identified by one of the two methods. Results from this thesis show that using branch power correlations to identify critical components and incorporating that information into the model helps to filter out traffic but can be improved with the addition of correlations between bus voltages.

Acknowledgements

First off, I would like to thank Justyna Chromik for all the encouragement. Justyna has been there from the start when I was still trying to come up with a research topic. Throughout the email exchanges, skype calls, and in person conversations she has provided helpful feedback and direction in my research and writings, and always asked questions to make me consider why I'm doing something.

In addition, I would like to thank Boudewijn Haverkort and Anne Remke for being part of my supervisory and graduation committee and providing helpful feedback during my intermediary presentations. I would also like to thank Fabio Massacci from the University of Trento for providing help in revising.

Lastly, I would like to thank Ries van Son and Compumatica Secure Networks for allowing me to work on this thesis and my minor thesis at Compumatica.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Industrial Control Systems (ICS) encompass a multitude of different systems including Supervisory Control and Data Acquisition (SCADA) and Distributed Control Systems (DCS). These systems are used in water treatment plants, buildings, factories, and electric grid systems. They are a combination of sensors for monitoring and collecting data on the real world and actuators which perform physical actions in the real world. Along with the sensors and actuators, programmable logic controllers (PLC), remote terminal units (RTUs), and intelligent electronic devices (IEDs) are used for decision making processes. ICS systems used to be isolated systems without any connections to other networks, but now, with the use of commercial off the shelf technologies and internet of things devices, they are more exposed. In order to provide some form of protection to ICS systems, firewalls, which are one of the recommended security measures for Critical Infrastructures by the National Institute of Standards and Technology (NIST) [10], are used.

Firewalls are used to segment networks and filter out traffic. Some firewalls within ICS use whitelisting and deep packet inspection to filter packets, but these techniques are not enough. Packets that contain legitimate content can be used to cause an ICS system to go into an unwanted state by sending multiple packets that each contain legitimate commands to change the systems configurations (component values). In order to be able to filter out these kinds of packets, the state of the system needs to be tracked and the impact of these packets on the systems state must be known before deciding to allow the packet through.

As an example consider the Vinyl Acetate Monomer process [11] in a chemical factory in Figure 1.1. The picture on the left shows the process without the location of the control valves, while the one on the right shows the control valves. The process produces vinyl acetate by converting ethylene, oxygen, and acetic acid through vapor reactions. In this process some of the gases are recycled through the system. The system has safety constraints, such as oxygen composition and pressure in the gas recycle loop being less than 8 mol % and 140 psia, respectively. There are

operational constraints such as the reactor and reactor inlet temperature being less than 200 Celsius and greater than 130 Celsius, respectively. The system could be attacked by sending multiple legitimate commands to open and close the control valves. By sending these commands the ratio of the chemicals can be changed, for instance making it so that the oxygen composition becomes 8 or more mol percentage, thereby violating the safety constraint. The temperature in the reactor can be made to go greater than 200 Celsius by keeping the control valve on the steam outlet closed, or by changing the setpoint level of the temperature, leading to mechanical damage.

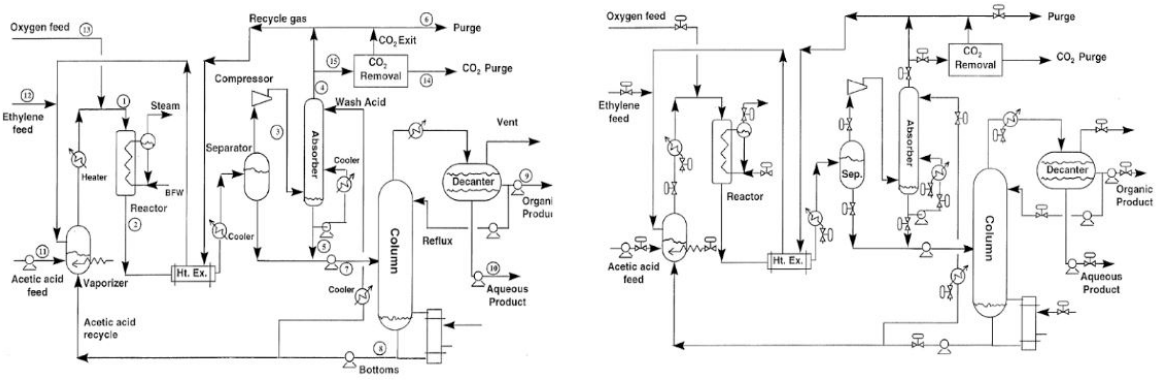


Figure 1.1: Vinyl Acetate Monomer Process

These legitimate commands can pass through firewalls that use only whitelisting or deep packet inspection leading to safety violations and mechanical damages. By taking the state of the system into consideration these commands can be stopped at the firewall, by analyzing the impact of the commands on the state of the system before letting the commands through.

Several papers have proposed different ways of filtering traffic through a firewall in an ICS system. Authors in [3] proposes the use of whitelisting along with deep packet inspection for packets that do not meet the whitelist criteria. In [7], the state of the system is monitored through the creation of a virtual system that replicated PLC component values. Packets were allowed or denied based on their impact on the state of the system which was evaluated through the use of boolean functions composed of PLC components and their values. Authors in [8] presented an analysis system that tracked the state of all observed process variables, reflecting the internal memory of PLCs. The variables would be characterized and modeled, with alerts being raised based on the differences between the real data and the predicted model data. [9] presented a semantic analysis framework for Power Grids capable of evaluating the impact of commands on the system. The system used mathematical models of the system along with the command to evaluate the commands impact using power flow analysis software. These methods will be described in more detail

in section 2.

This thesis looks at simplifying the model of the controlled physical system and using information about a subset of components which are deemed critical for the system to filter traffic. For instance, in the vinyl acetate monomer process example given, critical components could be the reactor (temperature sensor), pressure sensor in the gas recycle loop, and control valves for the steam and ethylene, acetic acid, and oxygen gases. With this concept, not all system variables need to be observed or modeled. The main research question that is addressed in this thesis is:

Whether knowing the state of critical components helps to better filter out traffic that could cause the system to go into an unwanted state?

To help answer this question several sub-questions are asked in order to learn about the system and about how traffic would be filtered. These sub-questions are:

1. What are the main critical components in the ICS system under study?
2. How would the critical components be modeled?
3. How do we know when the current traffic will make these components change the state of the ICS system?

By answering the identified questions, contributed knowledge includes whether providing knowledge on a subset of components within a system to a firewall can help it to better filter out traffic. Moreover, a proof of concept firewall is designed, that is based on the idea of using information about a subset of the system components to filter out traffic that would otherwise lead to an unwanted system state.

The rest of the document is outlined as follows: Chapter 2 provides background information on the security of ICS systems and previous work that has been done on firewalls and IDS systems for ICS systems. Chapter 3 will discuss the approaches taken to answer the research questions, as well as provide information on the steps used in the test setup to gather results. Chapter 4 presents that results from the testing of the firewall. Chapter 5 concludes the paper along with the limitations of this thesis and future work that can be done.

Chapter 2

Background

The main focus of this chapter is to provide background information on ICS systems and one of the security mechanisms, firewalls, used to protect these systems to the reader. Security issues within ICS systems are discussed in section 2.1 followed by a description of firewalls in 2.2. Section 2.3 includes information on the different types of methods that have been proposed on filtering traffic in firewalls.

2.1 Introduction to ICS Security

ICS systems were once isolated systems that did not have any connection to the internet. If someone wanted to hack into the system they would need physical access, but this has been changing. These systems now have internet protocol (IP) addresses which can allow for remote access to the system. The protocols that are used, such as MODBuS,DNP3, and IEC 60870-5-104, are encapsulated within transport control protocol (TCP) to allow for routing in the internet. The technology and integration with the web allows corporations to obtain realtime data from the ICS system to streamline their operations. Maintenance crews and workers monitoring these ICS facilities are able to take measurements of devices and perform calibrations without having to approach them [23].

With the exposure of ICS systems to untrusted networks, more and more vulnerabilities within these systems have been found over the years. Figure 2.1 shows the number of vulnerabilities, exploits, and malware targeting ICS systems for the years from 2001 to 2014. The spike in 2011 and after could be attributed to the discovery of Stuxnet, which was found to be targeting particular ICS systems that matched certain requirements. In this case it ended up being an Iranian nuclear facility where Stuxnet was able to make modifications to the system but provide false readings to observers that indicated the system was operating normally, even though it was not. After Stuxnet, attackers and more researchers alike took interest in ICS sys-

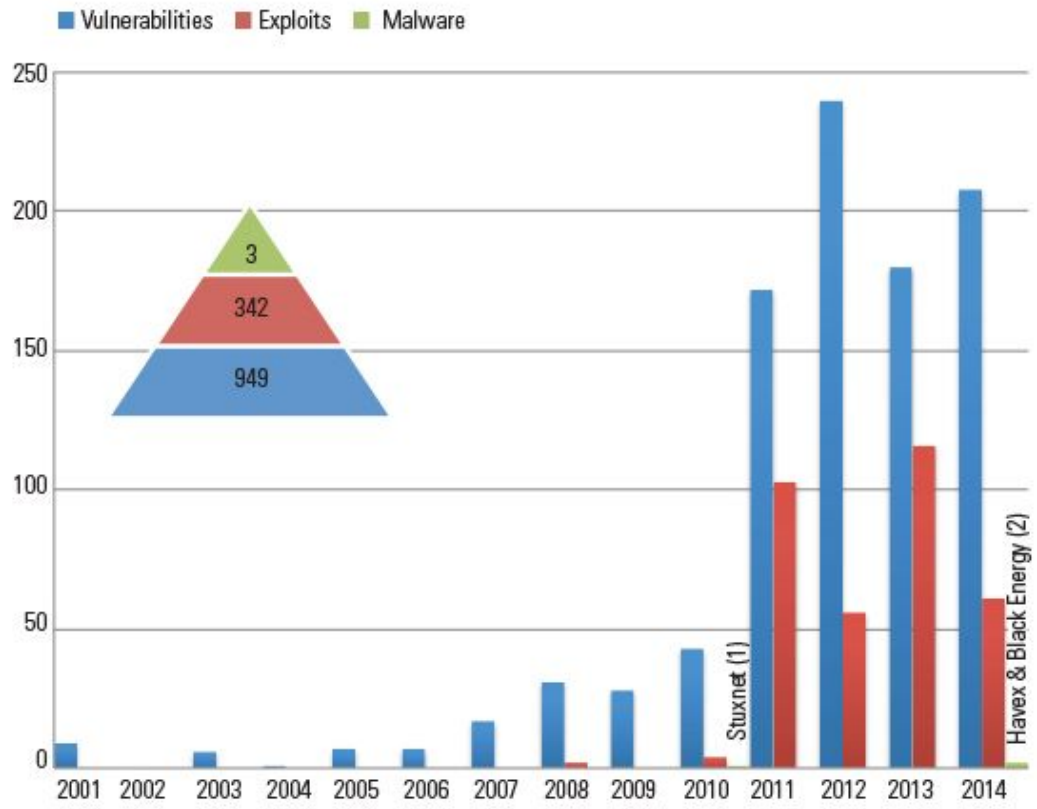


Figure 2.1: ICS vulnerability, exploit, and malware count per year [24]

tems leading to more vulnerabilities being discovered indicating that ICS systems were not very secure. A more recent example of malware targeting ICS systems is CrashOverride [25], which is the first known malware designed to attack electric grids. CrashOverride was used in a cyber attack on a transmission substation in Kiev, Ukraine in December 2016. From an analysis of the malware done in [25] different modules were found such as an IEC 60870-5-104 module, and IEC 61850 module. Both of these modules are related to electric utilities as IEC 104 and IEC 61850 are both protocols used in electric utilities. The analysis also revealed that module extensions were possible, so the malware could be used in industries other than the electric industry.

The security attributes for ICS systems in order of importance are Availability, Integrity, and Confidentiality, which is different from traditional information technology systems. Availability refers to the ability of an asset being operational to receive and handle requests within a constrained amount of time. Denial of service attacks on components of the ICS system can cause the ICS to have increased response times or to not be operational at all. Integrity is the correctness of an asset. If the asset is data then the integrity of the data holds as long as it has not been corrupted. Corruption of the data could be changing the data or exchanging real data from a

measurement with false data that may have been gathered earlier in time. The integrity of the data must hold otherwise it could cause the ICS system to go into an invalid or dangerous state. Confidentiality is the prevention of access to or use of an asset by an unauthorized entity. A violation of the confidentiality attribute within ICS systems could be an attacker gaining unauthorized access to configuration files.

These security attributes can be violated by malware that gets into the process network through another network such as the corporate network or through the use of an infected device. The protocols used by ICS systems can also be used to help in violating these security attributes. ICS systems can be vulnerable to exploit due to the architecture of the ICS system and ability to misuse the industrial protocols being used. Some of the possible ICS architectures used and the possible threats they pose will be explored next, followed by possible misuse of protocols.

2.1.1 Architecture Security

ICS systems can be connected to corporate networks as well as vendor networks. These connections can cause the ICS system to be vulnerable unless the networks are properly segregated. An architecture that provides the weakest form of pro-

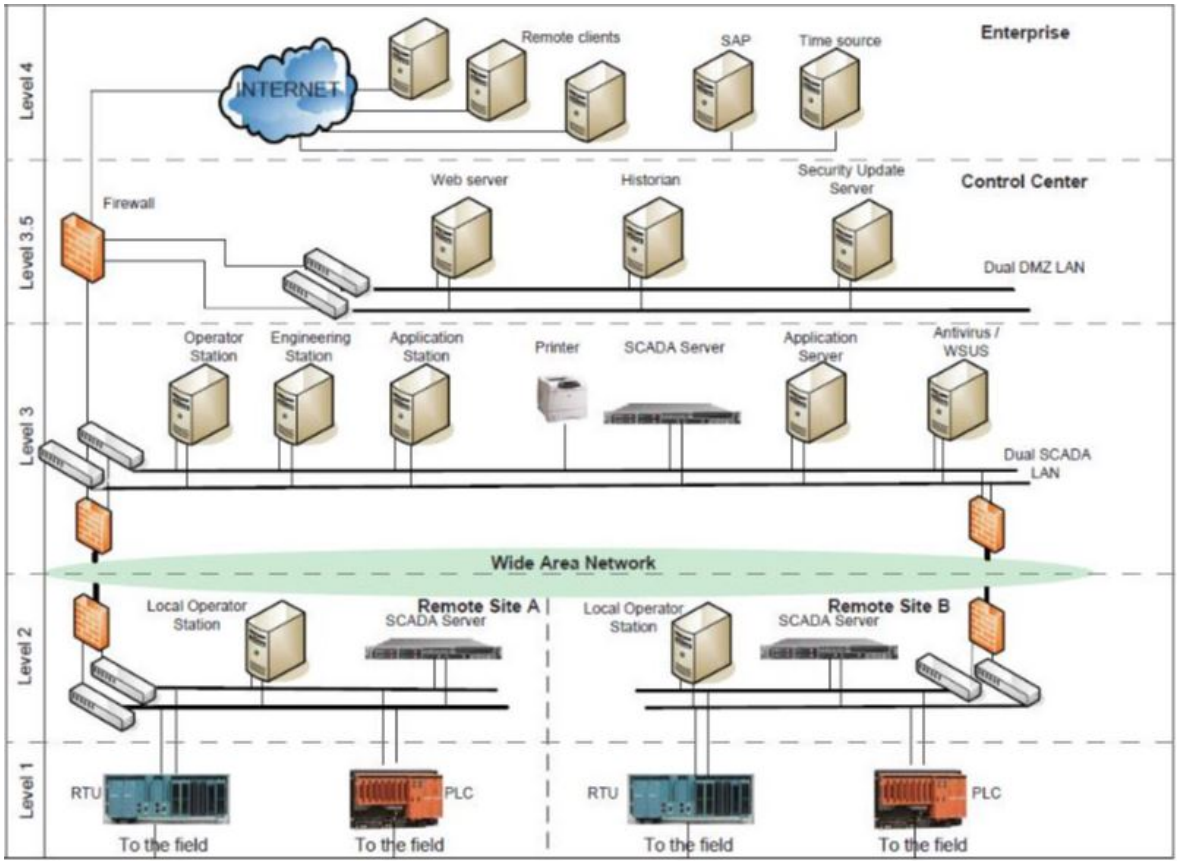


Figure 2.2: ISA-99 Purdue Model [33]

tection for an ICS system is one that uses dual network interface cards to allow for communication between two networks. In this architecture both networks have a direct connection to each other, so if there is an intruder or malware on one of the networks then they could move into the other network, without much effort. In order to provide greater security, the recommendations for ICS systems are to segregate the ICS network from all other networks and to provide a de-militarized zone [30,31,32] where untrusted networks can get data from if they need it, without having to have a direct connection to the ICS system.

The ISA-99 Purdue model in Figure 2.2, shows the type of recommended architecture for ICS systems. The model provides segregation at different levels starting from the local control at level 1 all the way up to the Enterprise network at level 4. This type of architecture provides defense in depth with multiple countermeasures in the form of firewalls. If an attacker wanted to get to the local operator stations at level 2, and did not have a way to physically access any of the devices at that level, then they would need to first compromise several systems and find ways to bypass the firewalls.

2.1.2 Protocol Security

Many of the protocols used in ICS systems lack authentication features to validate where the packets are originating from. Since there is no authentication, it is possible for an attacker to intercept a packet and modify it before forwarding the packet with potentially false information. The functions of the protocols themselves can be used to unintentionally reveal information about the ICS system, to cause denial of service attacks, or to send commands that look legitimate but end up switching the state of the ICS system into an unsafe state [26,27,28,29]. Command injection attacks are attacks that allow a malicious entity to inject configuration or control commands into the ICS system. These types of attacks can lead to overwriting the control logic used in remote terminal units [28]. Attackers utilizing command injections can modify component setpoints, setting them lower or higher, leading to an incorrect view of the component by the system. Attackers can also interrupt a components communications by causing a denial of service attack on the component, such as by sending a command with MODBUS function code 8 and sub-function code 1, in an ICS system using MODBUS [28]. Sending this command will cause the component to perform a restart and sending multiple of these commands can lead to the component continuously restarting, thereby cutting off communication with other components. In order to protect against receiving traffic from attackers and running commands which can have negative consequences on the ICS system the packets need to be inspected to determine where they came from and what

commands they are trying to run.

2.2 Industrial Firewalls

Firewalls provide network segmentation along with the enforcement of rules set by administrators, and has both advantages and disadvantages, such as those listed in Table 2.1. Modern firewalls are able to analyze and filter out data on all layers of the OSI model. With ICS systems, firewalls can be used to provide segmentation between the corporate network and the ICS system, as well as inspect the traffic that is traveling between the two networks. Traffic is filtered through the use of rules, and sometimes there are many rules that a firewall checks traffic against, which can introduce latency into the network. ICS systems have real time requirements and so this latency can prevent them from completing a task within a predefined time limit, which can cause problems.

Table 2.1: Firewall advantages and disadvantages

Advantages	Disadvantages
Powerful filtering capabilities through rules, and ability to inspect data	Introduces latency into system
Bidirectional communications allowed	Possibility of allowing unwanted traffic and data exfiltration
Lower cost to acquire	Higher running costs, in maintaining and auditing rules
Lower deployment complexity	Possibility for misconfigurations

Firewalls being able to inspect and analyze the content, at the application layer, of the traffic coming to them is very beneficial in ICS systems. ICS traffic has to be inspected in order to be able to understand what the traffic content is doing to the system, and without this inspection, the traffic would look all the same. An example of this would be with the MODBUS protocol, where the MODBUS read and MODBUS write commands would look the same without inspection. With deep packet inspection if a MODBUS write command is sent that is going to perform an unauthorized write operation, then this can be caught and the traffic can be filtered out.

The effectiveness of a firewall can be judged by its accuracy in correctly classifying the traffic that comes to it. The accuracy can be measured by collecting data on the number of false positive and false negatives, along with the true positive and true negatives. The definitions for true positive and negative and false positive and negative are given as follows:

True positive: Correctly classifying malicious traffic as malicious.
True negative: Correctly classifying safe traffic as safe.
False positive: Incorrectly classifying safe traffic as malicious.

False negative: Incorrectly classifying malicious traffic as safe.

Having too many false positives means that the firewall is restricting legitimate traffic, which could be crucial to the functioning of the ICS system, from passing through. Having too many false negatives creates a false sense of security since malicious traffic is being let through without being blocked. This malicious traffic could be to change the setpoint values of components or to turn off/on a component that should not be turned off/on in the current state of the ICS system. While a firewall many not be able to correctly classify traffic correctly 100% or the time, it should minimize the number of false positives and negatives to give an accurate security posture of the ICS system. In this thesis the false positive and false negative rates, as well as the accuracy, as calculated for the firewall that includes the created models. Equations 2.1, 2.2 , and 2.3 are used for calculating the false positive rate (FPR), false negative rate (FNR), and accuracy, respectively.

$$FPR = \frac{FP}{FP + TN} \tag{2.1}$$

$$FNR = \frac{FN}{FN + TP} \tag{2.2}$$

$$Accuracy = \frac{TN + TP}{TN + TP + FN + FP} \tag{2.3}$$

where:

FN = # of false negatives
FP = # of false positives
TN = # of true negatives
TP = # of true positives

2.3 Previous Work on Firewalls

Several types of firewalls for ICS systems have been proposed in the past, e.g., (i) firewalls that rely on the use of whitelisting [1,2,3,4]; (ii) firewalls that perform dynamic deep packet inspection to compare function code values with ones that are not allowed [5,6]; and (iii) firewalls and IDS systems that take into consideration the state of the ICS system when analyzing new traffic to determine whether to allow or deny the traffic [7,8,9]. The main points of the proposed firewall systems are described in this section along with some of their potential problems.

2.3.1 Whitelisting

In [1,2] a proposal and implementation of a firewall consisting of three whitelists based access control filters was presented. Traffic that came to the firewall had to go through all three filters in sequence before it could be let through the firewall. The whitelist filters consisted of (i) interface, (ii) communication, and (iii) command filters. The interface filter filtered based on MAC/IP pairings, while the communication filter filtered based on source/destination IP pairing, source/destination ports, and the protocol being used. The command filter filtered based on the control commands (function codes) that were inside the ICS protocol. If a packet failed to pass through any of the filters because it did not meet the whitelisted rules then the packet was dropped. This implementation, which was tested with the MODBUS TCP and DNP3 protocols, was successful in blocking all the packets that did not meet the rules set. However, the authors did note, but did not explain, that there were problems identified that they would need to address in the future.

In [3] the authors propose the use of whitelisting along with deep packet inspection for packets that do not meet the whitelisting policies. In this case all packets that meet the whitelisting policies are automatically allowed through the firewall without further inspection and only those that do not have to go through the deep packet inspection. For the packets that go through the deep packet inspection the function code and data values are extracted along with the source/destination IP and destination port. These extracted values are then used to determine if the packet was meant to do something malicious. The authors noted that one of the benefits of using whitelisting is its ability to reduce the load on the firewall when it came to dealing with proprietary protocols without having to detect the data content inside the packet. While whitelisting may help in reducing the load on the firewall, one concern is the potential for whitelisted devices to be compromised and then being able to send packets that contain malicious function codes and data. Since, the device is already whitelisted the firewall would let that devices traffic through without inspecting its content and this can lead to problems.

In [4] an application layer filtering system for automation networks to permit only industrial traffic and to block traffic that performs non-permitted actions is proposed. The proposed system of blocking non-industrial traffic is due to, e.g., the possibility of non-industrial traffic leading to information being improperly accessed, the use of network scanners to discover network topology, and the possiblity of denial of service attacks that would saturate the network bandwidth. Blocking non-permitted industrial traffic is to limit potential actions that could cause harm to the system such as performing unauthorized write operations on registers. The proposed system uses the Perl regular expression library, for pattern matching to identify industrial protocols, along with analysis of the protocol function field against user-defined

rules. The user-defined rules are specified in a rule file with device and device constraints, to ensure that harmful or out of context request actions are not processed. The proposed system was implemented and tested with the MODBUS TCP protocol. While the system was able to effectively filter out non-industrial and malicious traffic it introduced latency into the system that impacts the real time requirement that ICS systems have.

2.3.2 Dynamic Packet Inspection

In [5,6] the use of the u32 feature within the Linux Iptables firewall is proposed for use in performing dynamic deep packet inspection of ICS protocols. The u32 feature allows for the extraction of 32 bits from the packet being inspected at any specified location for comparison with values of interest. Using the u32 feature the authors were able to extract the header length fields from the IP and TCP headers and then use those values to move to the beginning of the ICS message. Once at the beginning of the message an offset would be used to examine the function code and data and compare those values against values that could lead to misuse. If the function code and data values matched then the packet would be dropped by the Iptables firewall. This method was applied with the MODBUS and DNP3 protocol, were the values used for comparison represented common attacks on PLCs that ran those protocols. With this method the users of the firewall would have to explicitly state the function code and data values of every type of attack that they wanted to block and if they did not state those values, then the system would be vulnerable.

2.3.3 System State Consideration

The firewalls described so far were able to correctly allow legitimate traffic or deny malicious traffic based on the policies or rules that were implemented within them. They all do so by treating each packet as an isolated packet and without considering the actions of the packet on the system. Meaning that the firewalls do not consider the content of other packets that have passed through and how those packets may have affected the ICS system. Knowing the effects of previous packets on the system can help in blocking current or future packets that could lead the system into an unwanted state, even if those packets were legitimate packets.

In [7] the actions performed by previous packets and how they may have changed the ICS system are taken into consideration. In this firewall, a virtual system that replicates part of the ICS system, the PLC component values, is created and so whenever traffic that goes through the firewall changes the value in a PLC component the firewall is able to take note of the change. These changes are monitored in

order to decide whether or not to allow new traffic through the firewall by inspecting the traffic and its values and whether or not these values will lead to a match with a rule. The rules or critical formulas, are boolean functions that identify combinations of PLC components and their values that correspond to particular states of the ICS system that would harm the functionality and integrity of the system. This state monitoring mechanism along with the application of packet filtering rules that examined the source/destination IP, source/destination ports, and specific protocol fields, allows the firewall to go more in depth than other firewall proposals. While monitoring the system state can help in blocking legitimate traffic that can lead the system to unwanted states, it puts a requirement on the virtual system within the firewall of always having to match the physical system. If the replicated components in the virtual system do not match the physical components then the firewall may perform the wrong actions which could lead to the same harm in system functionality that the firewall was trying to prevent. In order to be able to create the critical formula rules the administrators must know the combinations of the replicated components and their values that can lead to harm in integrity and functionality, so a detailed understanding of the physical ICS system is needed.

In order to detect intrusions in the system [8] created an analysis system that tracked the state of all observed process variables. This system reflected the PLCs internal memory. The process which the author took was to characterize the variables into one which was either a control, reporting, measurement, or program state variable. Models were then created for these variables and the IDS system would raise alerts based on the variances between the predicted model data and real data. Due to how variable naming can differ between programmers and vendors the automatic characterization of the variables ran into some problems since some variables could fit into more than one category.

In [9], a mathematical model of an electric power system under survey was created. In order to determine the state of the system the mathematical model along with commands sent to the system were evaluated in power flow analysis software. If through the power flow analysis the commands were found to have a negative impact on the system there were response mechanisms in place to reverse the actions. The same techniques as used in the paper may not be applicable in other industries or even in other electric power systems as they will need their own tailored models.

2.4 Discussion

This chapter discussed some of the security issues within ICS systems focusing on the architectural and protocol security issues. The architectural and protocol security issues can be mitigated through the use of firewalls, which can help provide

network segmentation and inspection of traffic at a low level. Metrics for measuring the effectiveness of a firewall were discussed which are the same metrics used in evaluating the proof of concept firewall created during this thesis. Previous work done on designing firewalls that used different methods of filtering traffic was also presented to provide information on what has been tried before.

Chapter 3

Methodology

This chapter begins with discussing the ICS system under study as well as the IEC 60870-5-104 protocol used, in Sections 3.1 and 3.2, respectively. Section 3.3 focuses on the two methods used for answering research sub-question 1. Section 3.4 provides details on the implementation of the component models from the IEEE 24 bus system. Section 3.5 focuses on how the IEC 104 protocol fields are used to distinguish packets that would have an affect on the processes in the ICS system. Section 3.6 discusses how the tools used in testing were developed, which includes how commands that were used in MATLAB were translated into IEC 104 packets. The interactions between the firewall, the model created, and the ICS system are also discussed in section 3.6. The chapter concludes with a discussion of the work done in this chapter.

3.1 1996 IEEE 24 bus Reliability Test System

The system used throughout the thesis was the 1996 IEEE one area 24 bus reliability test system (RTS) [12]. The system is shown in Figure 3.1 on page 16, where the top portion of the system is the medium voltage area and the bottom is the low voltage area. The system contains 38 branches with some duplicate branches, five transformers connecting the medium and low voltage areas, 17 bus loads, and 33 generators, including a synchronous condenser at bus 14.

3.2 IEC 60870-5-104 Protocol

The IEC 60870-5-104 (IEC 104) protocol is used in the electric utilities industry, primarily in Europe, Middle East, and Asia Pacific. The protocol is an extension of the application layer in the IEC 60870-5-101 protocol allowing for communication over TCP/IP using port 2404 on the server. IEC 104 contains different methods of

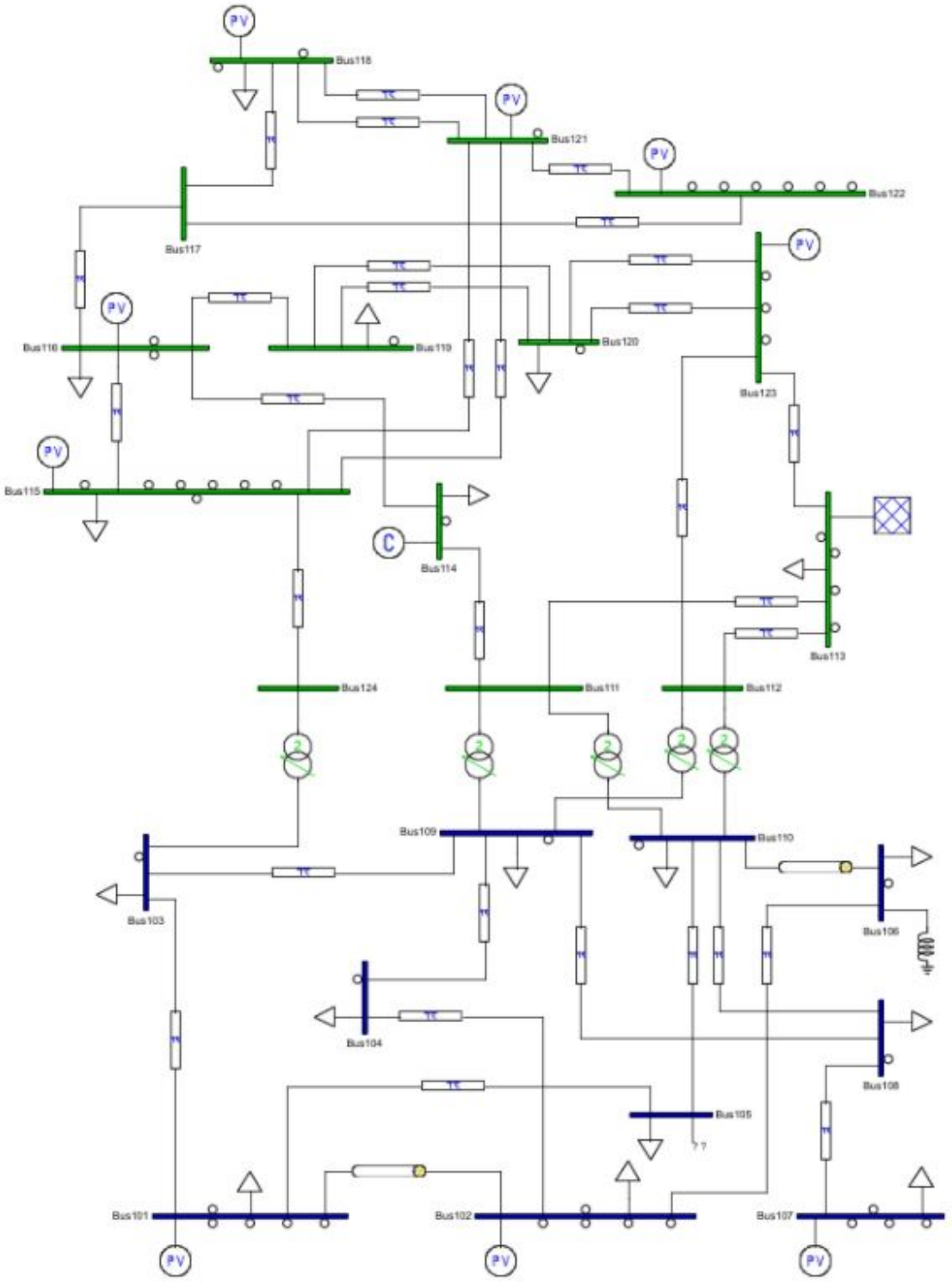


Figure 3.1: 1996 IEEE 24 bus RTS

data acquisition including request/response, cyclic data transmission, and acquisition of events. The request/response data acquisition is a client (controlling station) requesting information and the server (station being controlled) responding with the

requested information. The cyclic data transmission is when the field devices are configured to send data periodically without being requested to. The acquisition of events method allows data to be sent to the client when events occur on the server, such as failures. These data acquisition methods are identified in the cause of transmission field in the IEC 104 messages, which will be explained later on.

An IEC 104 message is made up of the Application Protocol Data Unit (APDU), which contains the Application Protocol Control Information (APCI) and Application Service Data Unit (ASDU). The APCI contains the start byte, length, and four control fields, which are explained as follows:

Start byte: One byte long and indicates the start of an IEC 104 message with the hexadecimal value 0x68.

Length: One byte long and indicates the length of the APDU without the start byte and this field.

Control fields: Four bytes long and is used to determine the format of the APCI, and contains sequence numbers. This firewall in this thesis inspects the control field when an IEC 104 packet is received to determine whether more fields need to be inspected.

The third byte in the APCI, which is the first control byte, determines whether or not an ASDU is transmitted along with the APCI. Inside this byte the first and second bit determine the the format of the APCI. If the first bit is 0 then the APCI is in information format (I-format), as shown in Figure 3.2.

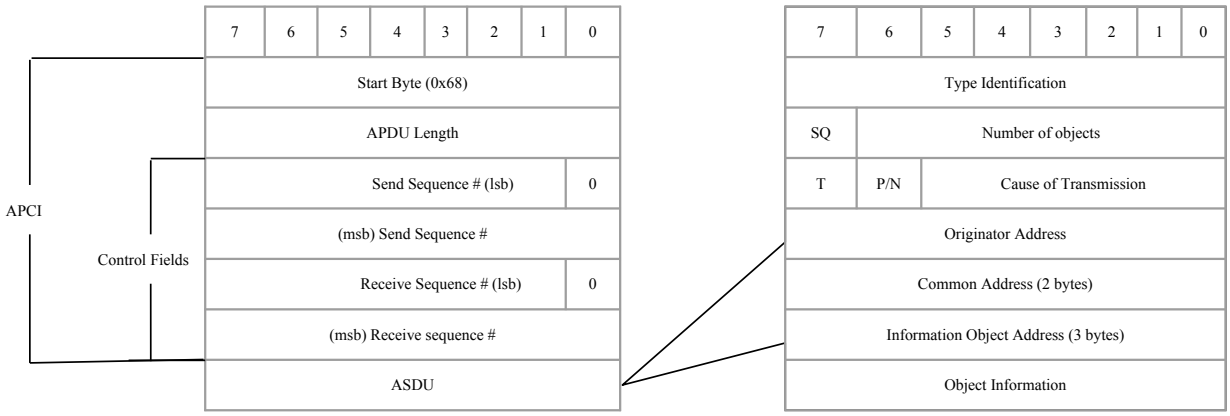


Figure 3.2: IEC 104 APCI I-Format with ASDU

If the first bit is “1” and the second bit is “0” then the APCI is in Supervisory format (S-format) as shown in Figure 3.3.If the first bit is “1” and the second bit is also “1” then the APCI is in Unnumbered format (U-format) as shown in Figure 3.4. The S and U format APCI makes it so that the APDU only contains the APCI, without the ASDU, but the I-format APCI makes it so that the ASDU is included in the APDU. When the APDU only contains the APCI then the length field in the APCI will be

fixed at four bytes, but when the APDU contains the APCI and the ASDU then the length field will be greater than four and a maximum of 253 bytes. The I-format APCI contains sequence numbers in the control fields for both sent and received data which are used for transmission confirmations.

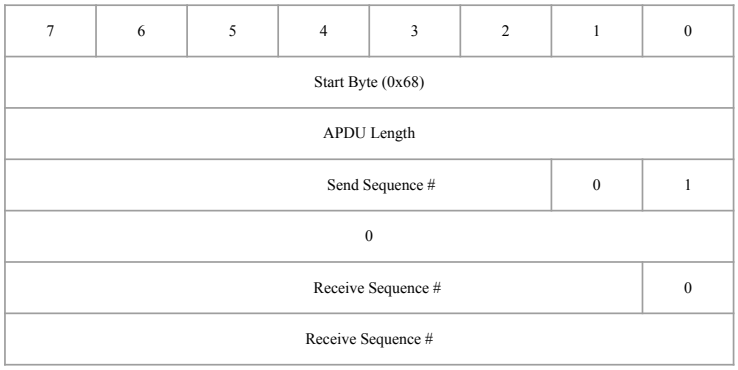


Figure 3.3: APCI S-Format

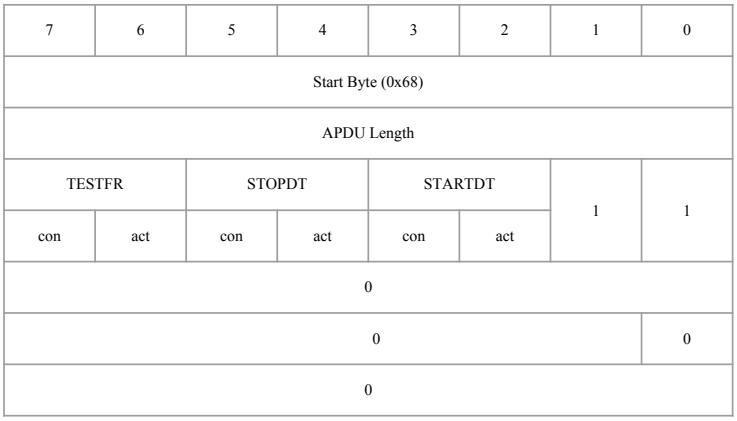


Figure 3.4: APCI U-Format

The S-format contains only sequence numbers, indicated in the third and fourth control bytes, for the received data and is used for acknowledging the receipt of I-format frames. The sender increments the send request number, and when the receiver responds the receive sequence number will be equal to the number of sent APDUs, if all the APDUs were received properly. When there is a long data transmission in one direction, an S format APDU is sent by the receiver to acknowledge the APDUs before a timeout or a buffer overflow occurs. The U-format APCI does not contain any sequence numbers and is instead used for connection management functions TESTFR, STOPDT, and STARTDT. TESTFR is used to check if the communicating hosts are responding. STARTDT and STOPDT are used to start and stop data transmissions respectively. The ACT and CON bits are used to represent a request and a positive response respectively. STARTDT must always be sent by

the controlling station and confirmed by the controlled station before any user data transfer from the controlled station is initiated.

The ASDU, format shown in Figure 3.2, is used to send application data and consists of the following fields: type identification, variable structure qualifier, cause of transmission, originator address, common address (ASDU address fields), information object address, and object information fields. A description of these fields is as follows.

Type Identification: One byte and indicates the type of information object sent which can include a time tag (timestamp) that is three bytes. 256 possible values, but 0 is invalid, 128-135 are reserved for routing messages, 136-255 are for special use. Only 1-127 are defined in the standard. Type Identification defines the structure, type and format of the information object.

Variable Structure Qualifier: One byte and is comprised of the SQ bit and the number of objects.

- **SQ:** One bit and indicates whether information objects are single (0) or a sequence (1) of information elements. With single, the information element is addressed by the Information Object Address, and each information object has an address associated with it. With sequence, the Information Object Address specifies the address of the first information element, and the following information elements are identified by an increment of 1 starting from the first address for each element.

- **Number of objects:** Seven bits and indicates number of information objects

Cause of Transmission: One byte and indicates the data acquisition method. Contains T, P/N bits.

- **T:** One bit and indicates if ASDU was generated during testing (1) conditions or not (0)

- **P/N:** One bit and indicates positive (0) or negative (1) confirmation of a request

- **Cause:** Six bits and indicates specific reason for transmissions, such as periodic, requested, etc. Zero is not defined and only 1-13 and 20-41 are defined.

Originator Address: One byte and used when a host is relaying a message to other hosts or when directing mirrored ASDUs in monitor direction (Server $\rightarrow$ Client). The relaying host would use this field to identify the original sender. Field is zero when not relaying messages.

Common Address: Two bytes and indicates the station address of the sender. Common address can also be set to broadcast address (65535) or to zero indicating that address is not used.

Information Object Address: Three bytes and indicates the address of the requested object. If address is not relevant (not used) then it is set to zero. Each requested information object has an address. Used as destination address in control direction (Client $\rightarrow$ Server), and as source address in the monitor direction (Server $\rightarrow$ Client).

Object Information: Variable size depending on the Type Identification. Contains the actual data.

In answering the research questions and developing the tools used for testing, the IEC 104 protocol fields were utilized.

3.3 Critical components of an ICS system

Two different methods for determining critical components within the IEEE 24 bus system were used. These methods were (i) minimum and maximum connections to other buses, and (ii) finding correlations between branch powers and using one of the buses attached to those branches. Using buses with minimum and maximum number of connections was a way to see how the firewall would perform if it knew the state of components with a certain amount of connections in the system. For instance, how would the results compare if the firewall knew the states of buses with three connections versus buses with 5 connections. The initial hypothesis was that observing buses with greater number of connections would result in the firewall being able to better filter out malicious traffic, but in some ways this was not the case as will be shown in the Results section.Table 3.1 shows the number of connections and the buses that had that number of connections. For example bus 7 only had one connection to another bus, while buses 9, 10, and 21 had connections to 5 other buses each.

Table 3.1: Bus connections

Connections	Buses
1	7
2	4,5,6,14,22,24
3	1,2,3,8,13,17,18,19
4	11,12,15,16,20,23
5	9,10,21

In this thesis correlations between components were only found for branch real powers, but this could be extended to finding correlations between voltages, and generator real and reactive powers. This method is similar to the invariant induction method used in [13]. The real power readings for all the branches within the IEEE

24 bus system were collected as different actions were taken in the system. These actions were activating, deactivating switches and increasing, decreasing bus load power. The data was collected using the IEEE 24 bus case MatLab file in [14], which is the format needed to be used with Matpower. A script, located at [15], was created using MATLAB which took commands corresponding to the actions mentioned (activate switch, deactivate switch, increase load, and decrease load). Once an action was taken an AC power flow analysis was done using using Matpower to determine the effect of the action on the system. The branch powers from the result of the power flow is used to determine if there are any correlations between the branch powers.

After collecting the branch real power data the MATLAB function 'corrcoef' was used to obtain the correlation coefficient matrix between all the branches. The resulting correlation coefficient matrix, located in [16] under the "Orig branch power coefficient" tab, was used to help determine which branches were correlated. One of the two buses which makes up the branch was chosen based on trying to minimize the number of components that needed to be observed. After choosing branches considered as critical components, we made graphs, where the branch with the chosen bus was on the x-axis and the branch it correlated with was on the y-axis, along with a linear regression line. The code to generate the linear regression equations is in [17]. The graphs and the linear regression equations are shown and talked about in more detail in Chapter 4. The linear regression equation was in the form of y = Ax + B, where x is the branch with the critical component and y is the branch that has a strong correlation with the branch containing the critical component. The linear regression equations were incorporated into the model developed to determine the branch power of the branch that correlated to the branch with critical component.

3.4 Modeling the Critical Components in the Firewall

A model of the components to be created was made using Java, where the different components in the IEEE 24 bus system were divided into different classes. Each component such as a bus, load, branch, and generator had a class associated with it. Figures 3.5 to 3.8 show the classes for the branches, loads, generators, and buses named BranchItem, LoadItem, GeneratorItem, and BusItem respectively.

Each class contains values for the components, such as limit constraints, as well as values that change over time such as the power in the component. Each class is explored in a bit more detail below, but the functions will be not explained in detail as they are available at [18].

LoadItem Class: The LoadItem class models a load and has two variables loadPower, and loadPowerBase which contains the actual power of the component and

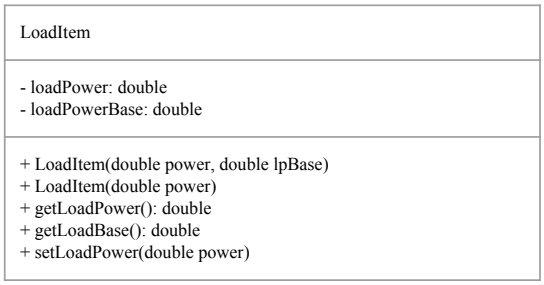


Figure 3.5: LoadItem classes with variables and methods

the power base respectively. The power base in the IEEE 24 bus system is 100 MVA, but for other systems this could change so this value is kept track of. There are two constructors for the class both taking in the power, and one allowing the inclusion of a power base. If the power base is not passed in it defaults to 100 MVA. The remaining classes are get and set methods for getting the power base and power value and for setting the power of the load respectively.

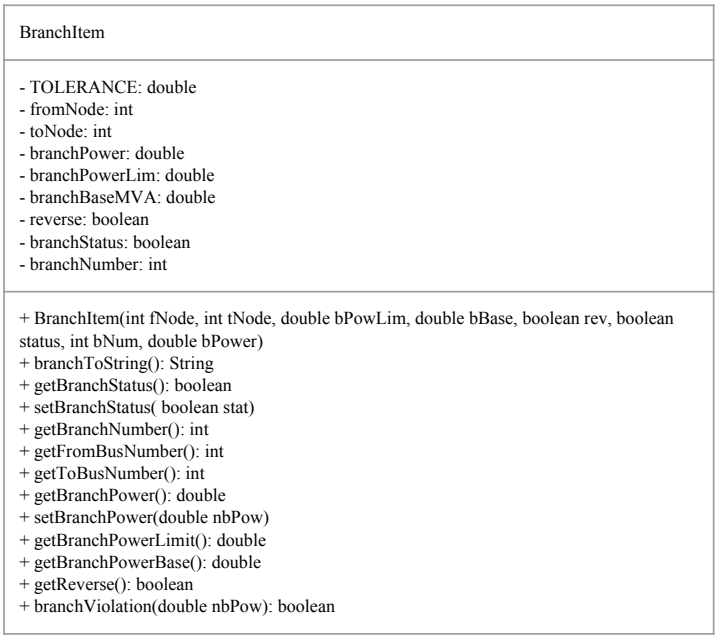


Figure 3.6: BranchItem classes with variables and methods

BranchItem Class: The BranchItem class contains variables that store the two buses that make up the branch, the power, power limit, power base, connection status, and the branch number. The branch number indicates what branch it is on the bus that it connects to as multiple branches can connect to the same bus. There is also a reverse variable. The reverse variable is based on whether the bus is classified as “fbus” or “tbus” in the IEEE 24 bus case file. For example, for some buses appear only in the “tbus” column of the Matpower case file and if the observed component is in that column then reverse will be true, which means that it was changed to be the “from bus” and the original “from bus” was changed to the “to bus”. There is a TOLERANCE variable which is the same throughout all the classes

where it is included and this is to increase the range of limits by a small value. The TOLERANCE was chosen to be 0.01 to correct error differences between decimal precisions in JAVA and MATLAB. Other TOLERANCE values could be used such as having it be 10% of the component value to allow for larger error corrections, but differing TOLERANCE values were not investigated as part of this thesis. With the TOLERANCE value added the limit constraints are as follows:

$$Limit_{min} - TOLERANCE \leq Value_{actual} \leq Limit_{max} + TOLERANCE \tag{3.1}$$

A version of equation 3.1 is also used to check for violations within each component. The constructor of the class takes in all the values needed to set the variables mentioned. The remaining functions are get and set functions along with a branchViolation function. The branchViolation function does a simple check for power limit violations on the branch as expressed in equation 3.2.

$$|Power_{actual}| \leq Limit_{BranchPower} + TOLERANCE \tag{3.2}$$

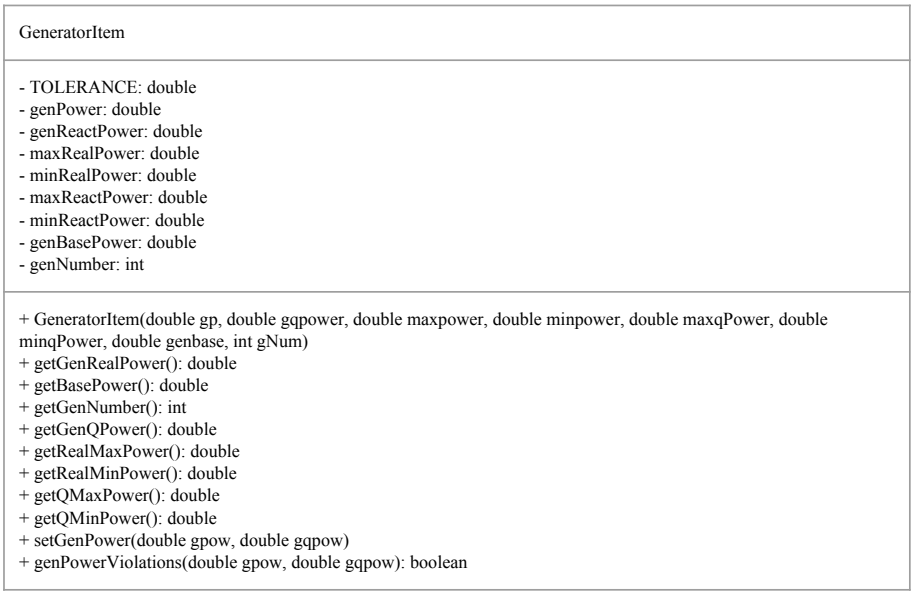


Figure 3.7: GeneratorItem classes with variables and methods

GeneratorItem Class: The GeneratorItem class also has the TOLERANCE variable along with variable to store the generator real and reactive power, the minimum and maximum real and reactive power allowed, the power base, and the generator number. In the matpower case files a single bus can have multiple generators and so the generator number is used to store which generator is represented. The constructor for the classes takes in parameters to initialize all the variables in the class mentioned. The remaining functions are get and set functions. The set function in the class takes two parameters which represent the real and reactive power and is used to update those values as every time the model is updated. There is a generator power violation function which checks to make sure that the both the real and reactive powers are within the allowed limits. The checks used is as follows:

$$RealPower_{min} - TOLERANCE \leq |RealPower_{actual}| \leq RealPower_{max} \tag{3.3}$$

$$ReactPower_{min} - TOLERANCE \leq ReactPower_{actual} \leq ReactPower_{max} + TOLERANCE \quad (3.4)$$

If any of the checks fail then the function returns a boolean true value to alert the model driver program of the violation.

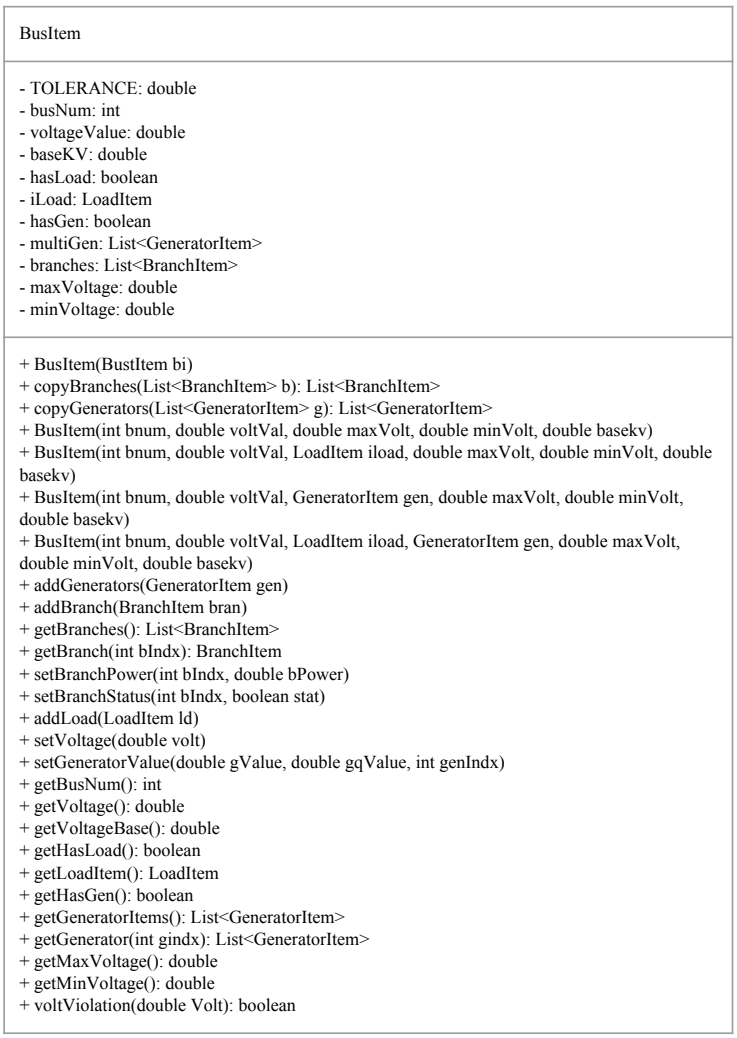


Figure 3.8: BusItem class with variables and methods

BusItem Class: The BusItem class makes use of the LoadItem, BranchItem, and GeneratorItem classes since the buses are are chosen as the critical components to observe and the components that connect to those buses need to be kept track of. The BusItem class has variables to store the bus number, bus voltage, voltage base which is in kV, minimum and maximum voltage allowed, and whether the bus has a load and generators. If the bus has a load connected to it a LoadItem object is created and if there is generators are present, GeneratorItem objects are created and stored in a list. BranchItem objects are also created and stored in a list variable.

The BusItem class contains several constructors which all atleast take in parameters which are for the bus number, the voltage value, the maximum and minimum voltage allowed, and the voltage base. The constructors differ based on whether the bus has a load or generators or both. There are functions to add generators to the bus in case more generators are added later on and also to add branches and loads. The remaining functions are get and set functions to retrieve values and to set bus voltage, branch, generator, and load power. There is also a voltage violation function for verifying that the bus voltage is within the range of allowed values. The check used to check the bus voltage value was as follows:

$$Voltage_{min} - TOLERANCE \leq Voltage_{actual} \leq Voltage_{max} + TOLERANCE \tag{3.5}$$

If this check is violated a boolean true value is sent to the driver program to indicate that a bus violation has occurred.
The component models were created in this way to allow for a modular design for future work where new components could be added, such as transformers. Each component contains only the necessary data to know its states and the ability to view and update that data quickly.

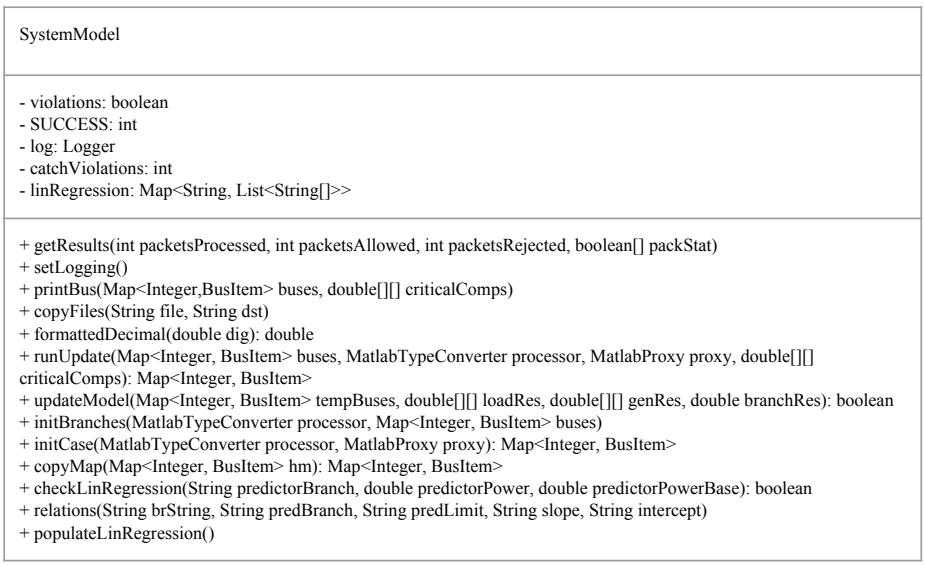


Figure 3.9: Main model driver program

Figure 3.9 shows the main model driver program class, SystemModel, which is what interacts with the firewall, by receiving the commands and values and sending whether there are any violations. The driver program is communications with MATLAB for running the power flow analysis. The SystemModel class contains global variables that indicate violations that occurred in the critical components, a success variable which helps to indicate whether the powerflow analysis done in MATLAB converged or not, a logger object to log details of the model such as when violations occur. There is a variable which is set based on what type of violations a user

would like the model to take note of, and a variable which stores branches and the branches that they correlate with along with the linear regression equations.

The Main program of the model driver gets messages, from the simulated firewall through a socket, which contains the command that is in the packet along with the value. The commands are restricted to process commands, certain Type Identifications, in the IEC 104 protocol only as those are the commands that will be directly affecting the components. In this thesis only three process commands were considered, which are “Single command”, “Double command”, and “Setpoint command scaled value”. These commands can be with or without the CP56Time2a time tag. The process commands and their Type Identification are shown in Table 3.2 along with the information object element types that are associated with them.

Table 3.2: Process commands with type identification and information object elements.

Command	Type Identification	Information Object Element Information
Single Command (w/ time tag)	45 (58)	SCO - 1 byte, (CP56Time2a - 7 bytes)
Double Command (w/ time tag)	46 (59)	DCO - 1 byte, (CP56Time2a - 7 bytes)
Setpoint Command Scaled Value (w/ time tag)	49 (62)	SVA - 2 bytes, QOS - 1 byte, (CP56Time2a - 7 bytes)

The firewall only sends the command, value, and information object address, which indicates which component to target, to the model driver if the IEC 104 packet has the type identifications in Table 3.2; as well as having a Cause of Transmission of 6 or 8 along with the S/E bit set to 0 meaning to execute. A Cause of Transmission of 6 or 8 indicates that the command is going towards the process component rather than being a response from a previous command, as specified in the IEC 60870-5-104 standard. Once the command and value, which is stored in the SCO, DCO, or SVA information element (c.f Table 3.2, is received by the driver program, the value is sent to a MATLAB program along with the critical components list where a power flow analysis is ran. For instance, if the command received is a single command with a value of 0, indicating to turn off, and the information object address is for a branch then the MATLAB program will perform the power flow analysis with the indicated branch set to inactive. Once the power flow analysis is completed the new values for the critical components are sent back to the SystemModel program where the critical components models are then updated and checked for violations.

The Matpower IEEE 24 bus case file is used to represent the ICS system and the power flow analysis is ran using this case file. The case file gets updated whenever a command is executed without any violations occurring as indicated by the data received by the SystemModel program after the power flow analysis. If violations do occur then both the case file and the critical component models are not updated from the state before the command was executed. This process is used to mimic the act of the firewall allowing or rejecting the packet. The code for the model program

as well as the MATLAB scripts used can be viewed at [19].

3.5 Identifying Changes in ICS System Due to Traffic

In order to determine if a packet will make changes to components in the ICS system, the Type Identification of the IEC 104 ASDU had to be examined. The Type Identifications that directly change the process components, in addition to those mentioned in the 3.2, include: regulating step, set point normalized, setpoint floating point number, and bitstring of 32 bits commands. As mentioned, this thesis considers only the three commands, single, double, and setpoint scaled value command, to signify that the packet will be changing a component. Any other commands that are not process commands can be handled differently by the firewall using whitelisting but this is implemented in this thesis. If during inspection of a packet the firewall determines that the packet has a destination port of 2404 and contains an ASDU with a type identification that matches that of a process command then the cause of transmission along with the information object information is inspected. Figure 3.10 shows the fields within the ASDU that are inspected if the type identification is for a single command. After verifying that both the type identification and the cause of transmission, the “S/E” bit is the final check in order to determine whether to send the packet to the model for testing. The S/E bit is for select (1) and execute (0), and it has to be set to execute in order to send it to the model. If the S/E bit is set to select than it means that the component is being prepared and the packet will not actually do anything to change the component or the system. The SCS bit for single commands is set to 0 for turning a component off and 1 for turning the component on. This value would be sent along with the IOA and a string indicating the command to the model.

Once the packet has been sent to the model program then the determination of whether to allow or reject the packet is as described in the previous section. The variable structure identifier in the ASDU for the process commands always have a value of “1” since only a single information will be sent.

3.6 Test Setup

As data from a real ICS system was not available which could show what commands may cause violations, a series of commands were generated and tested for violations in MATLAB. The commands, command Sequence A, shown in Table 3.3 perform actions that include increasing or decreasing the load power at a bus, e.g., command #1, and activating or deactivating a branch switch, commands

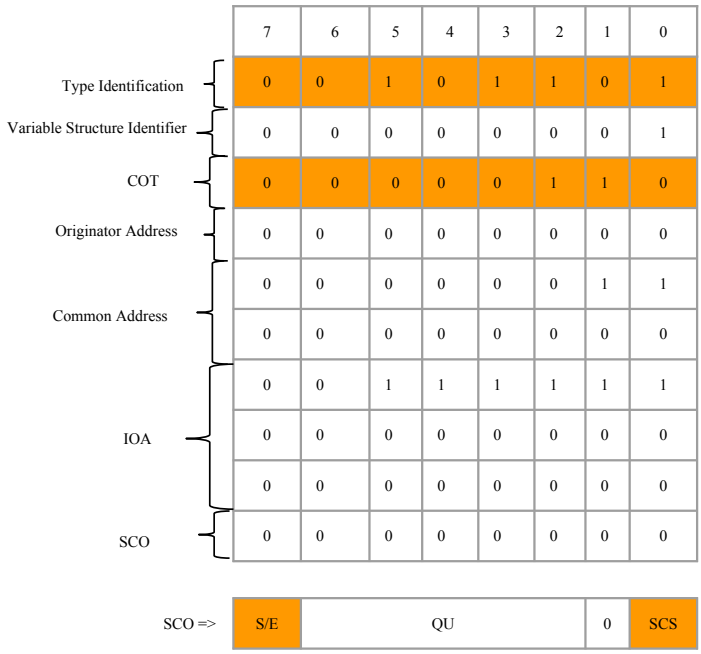


Figure 3.10: ASDU with fields that are inspected to determine if packet information is sent to model

#12 and #14 respectively. The order that the commands were given is shown in the "#" columns and the commands with red backgrounds indicates commands that resulted in either voltage violations or branch power violations or both. The data associated with each command can be seen in the excel sheet in [16], under the "Origtest volt" and "Origtest branch power" sheets.

Table 3.3: Command Sequence A. Commands with red background indicate violations occurred.

#	Command	#	Command	#	Command	#	Command	#	Command	#	Command
1	load 1 .5	11	load 4 1.392	21	cutoff 9 4 0	31	load 14 1.803	41	cutoff 19 20 0	51	cutoff 22 17 1
2	cutoff 2 4 0	12	cutoff 9 8 1	22	cutoff 9 8 0	32	cutoff 14 11 0	42	cutoff 13 23 0	52	load 19 1.2006
3	cutoff 9 8 0	13	load 8 1.497	23	load 1 1.296	33	load 18 1.4008	43	cutoff 22 17 0	53	load 4 0.7961
4	load 9 1.097	14	cutoff 10 8 0	24	load 3 1.4	34	load 9 .802	44	cutoff 19 20 1	54	load 8 1.1992
5	cutoff 10 5 0	15	cutoff 21 22 1	25	cutoff 3 24 0	35	load 1 1.5	45	cutoff 21 15 0	55	cutoff 21 15 1
6	cutoff 16 15 0	16	load 20 1.5	26	load 14 1.199	36	cutoff 16 15 1	46	load 3 1.4006	56	cutoff 7 8 0
7	load 19 1.099	17	cutoff 23 20 0	27	cutoff 16 15 1	37	cutoff 10 5 1	47	load 15 1.2	57	load 1 1.4
8	load 18 1.498	18	load 10 1.4	28	cutoff 1 3 1	38	cutoff 10 8 1	48	load 19 1.2014	58	cutoff 13 11 0
9	cutoff 21 22 0	19	load 14 1.397	29	load 3 1.2976	39	load 15 1.4984	49	load 9 1.2987	59	
10	cutoff 1 3 0	20	cutoff 17 16 0	30	cutoff 3 24 0	40	load 19 1.39698	50	load 2 1.2989	60	

The load command format is "load [bus number] [increase/decrease amount]" where the load at the specified bus would be multiplied by the increase/decrease amount. If the command is "load 1 .5" as in command #1, then the load at bus 1 would be decreased by half. If the command is "load 1 1.1", the load would be increased by 10%. Activating/deactivating the branch switch command is formatted

as “cutoff [from bus] [to bus] [status]”. Status is either 0 for deactivation or 1 for activation. The MATLAB script used to enter and run these commands is in [15].

After the commands and their data is verified in MATLAB, they are translated to fit into the IEC 104 protocol format using an open source IEC 104 packet generator [21]. The packet generator contains client and server programs where the commands are sent from the client to the server and the server echoes the clients commands back. After a few modifications to the Python files the commands were written in the “ieee_24_traffic.py” file [22]. The commands are in the following format:

```
(‘START’, ‘auto’, ‘if’, [TI, VSQ, CoT, OA, CA, (IOA,  IOA information)])
```

An example for the first command “load 1 .5” is as follows:

```
(‘START’, ‘auto’, ‘if’, [49, 1, Cause_Act, 0, 3, (25,  54, QOS_Ex)])
```

The Originator Address (OA) is set to 0 to indicate that there are no relays being used, and the Common Address (CA) was arbitrarily chosen to be 3. The Originator and Common addresses were the same for all the packets generated. The Type Identification (TI) used for the load command is 49 which is the *setpoint scaled value* command and the variable structure qualifier (VSQ) is set to 1 since there will only be a single information object. The “Cause_Act” value is a variable which is set to 0x06 to indicate an activation Cause of Transmission (CoT). The Information Object Address (IOA) is set to 25, to indicate bus 1, which is a value that was chosen based on a configuration file created, c.f. “ieee24_iec.properties”[19], where each component is given an address. The SVA value for the setpoint scale value command is 54 since the previous load power at bus 1, 108 MW, is now multiplied by .5, so the load power will be decreased by half. QOS_Ex, set to 0x00, is a variable to indicate that the packet will be executed since the S/E bit will be set to 0. After the commands were created both the client and server programs were started along with the network monitor program CommView 6.5. The CommView program allowed for capturing the traffic between the client and the server on the loopback interface as both were running on the same computer. Once the client finished sending all the commands to the server the network capture was stopped and a pcapng file, “ieee_24_traffic.pcapng” [22] was created for later use in testing the model.

Figure 3.11 shows the interactions between the firewall, model for observed components, and the ICS system. The packet going into the firewall is a replay of the captured traffic from the creation of the commands as described previously. The command translation portion of the firewall is where the packet is inspected to determine the destination port. The IEC 104 portion of the packet is inspected further to verify the Cause of Transmission and the Type Identification, IOA, and Object Information are extracted. The extracted information is sent to the system model

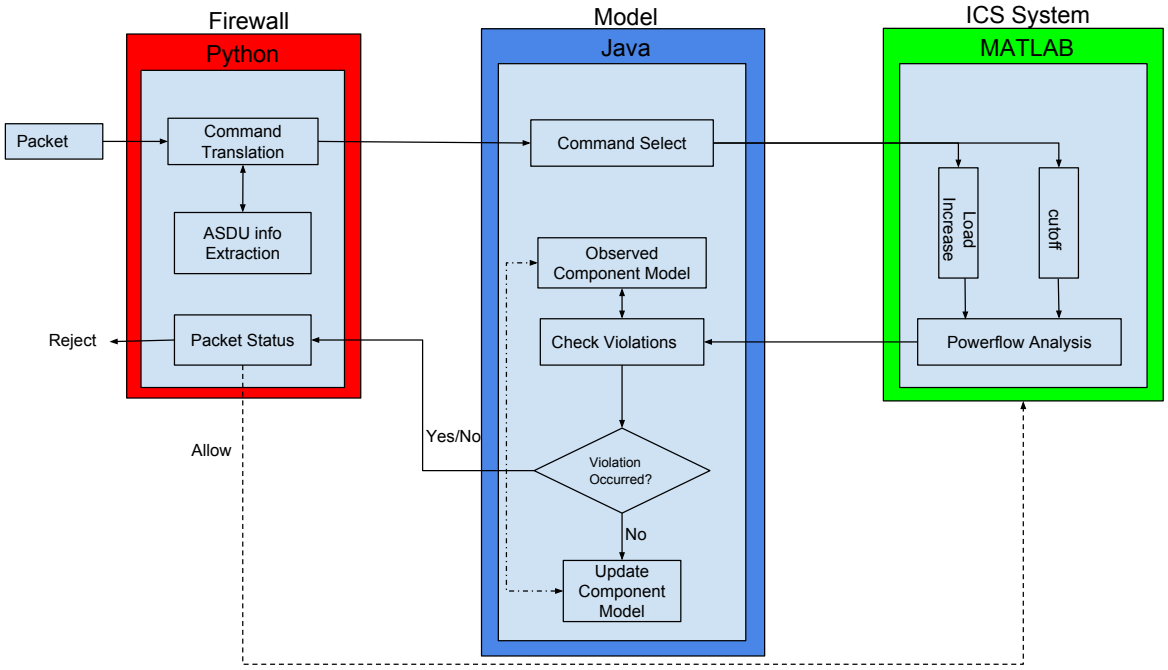


Figure 3.11: Interactions between firewall, model, and ICS system

where the correct scripts to be executed in MATLAB are chosen based on the command. The MATLAB scripts get executed with the extracted information from the packets and a power flow analysis is ran. The data for the critical components being observed from the power flow analysis is related back to the system model where violations on both the observed components and the components which they may have a correlation to are checked. If no violations occur the model is updated. The system model then sends back the results to the firewall indicating whether the packet should be rejected or allowed through.

Since results of the command executions are known beforehand, from Table 3.3, the false positive and negative rates can be calculated, as well as the accuracy of the entire model and firewall in helping to filter out the packets that are considered malicious. The false positive and negative rates, along with the accuracy of the entire system are explored in chapter 4.

3.7 Discussion

Initially, a naive method, minimum and maximum connections, was used to determine the critical buses. However, this method is limited in that only the bus and the components directly connected to it such as branches, loads, and generators can be observed for violations. Violations can occur on components that are not connected to any of the observed components. In those cases the minimum maximum connec-

tions method can lead to the firewall allowing the packet through unless there are violations on the observed components. Finding valid correlations between components would help to determine the state of components that are not being observed by using the values of those components that are observed and are correlated with the unobserved component. As mentioned, in this thesis the correlations were only focused on the branch real powers but correlations on bus voltages and generator real and reactive powers could also be tested for to create a better model. This method of finding correlations between components to determine critical components could potentially be applied to other ICS systems as well such as water and waste management, and chemical processing plants.

The model created allows for simple definitions of components as well as for future additions of components should they be needed, such as the addition of transformers. The checks for violations within the model are currently only on the single component values, power or voltage value, compared against the constraints placed on the component. Other checks could be incorporated such as those in [20], which include using Kirchoff's current law to check that the current going into a bus is approximately the same as the current leaving the bus.

This section described the process taken for answering each of the sub-research questions and the develop of materials to help answer the overall question of whether knowing the state of critical components helps to better filter out traffic that could cause the system to go into an unwanted state? The answer to the overall research question is answered and discussed in the results section. In addition the steps taken for testing and the interactions between the firewall, the model, and the MATLAB programs, were laid out as well.

Chapter 4

Results

This chapter discusses results obtained from two test runs, explained in the following. Section 4.1 focuses on the correlations found between branch powers and the linear regression equations formed. Section 4.2 focuses on the firewall results of the Sequence A commands in Table 3.3 of section 3.6. Section 4.3 will focus on the firewall results based on a different set of commands, Sequence B, but using the same linear regression equations show in Figure 4.1 of Section 4.1.

4.1 Correlation Results From Command Sequence A

Using the data from the power flow analysis gathered after running command Sequence A commands, the correlation coefficients were calculated using MATLAB, as explained in section 3.3, the results of which are in [16]. From those coefficients several buses were chosen to be the critical components due to them being connected to a branch having strong relationship with another branch.

Other buses were also chosen if they were connected to a branch that had very weak relationships with other branches. For example, the branch connecting bus 7 and 8, had a correlation coefficient of 0 with other branches. The buses chosen included buses 1,2,3,7,9,10,16,21, and 23. These buses were chosen after reviewing the correlation coefficients in [16] under the “Orig branch power coefficient” tab. With these buses all of the branches in the IEEE 24 bus system could be observed either due to the bus being connected directly to a branch or being connected to a branch that it correlates to. Figure 4.1 shows the graphs of branches that contain some of the buses and the branches that they correlated with. The X-axis represents the first branch and the Y-axis represents the second branch. For example with Figure 4.1a, the X-axis represents branch 16-17, and the Y-axis represents branch 18-17. The linear regression equation is also represented in the graph. The code used to generate these graphs can be found in [17].

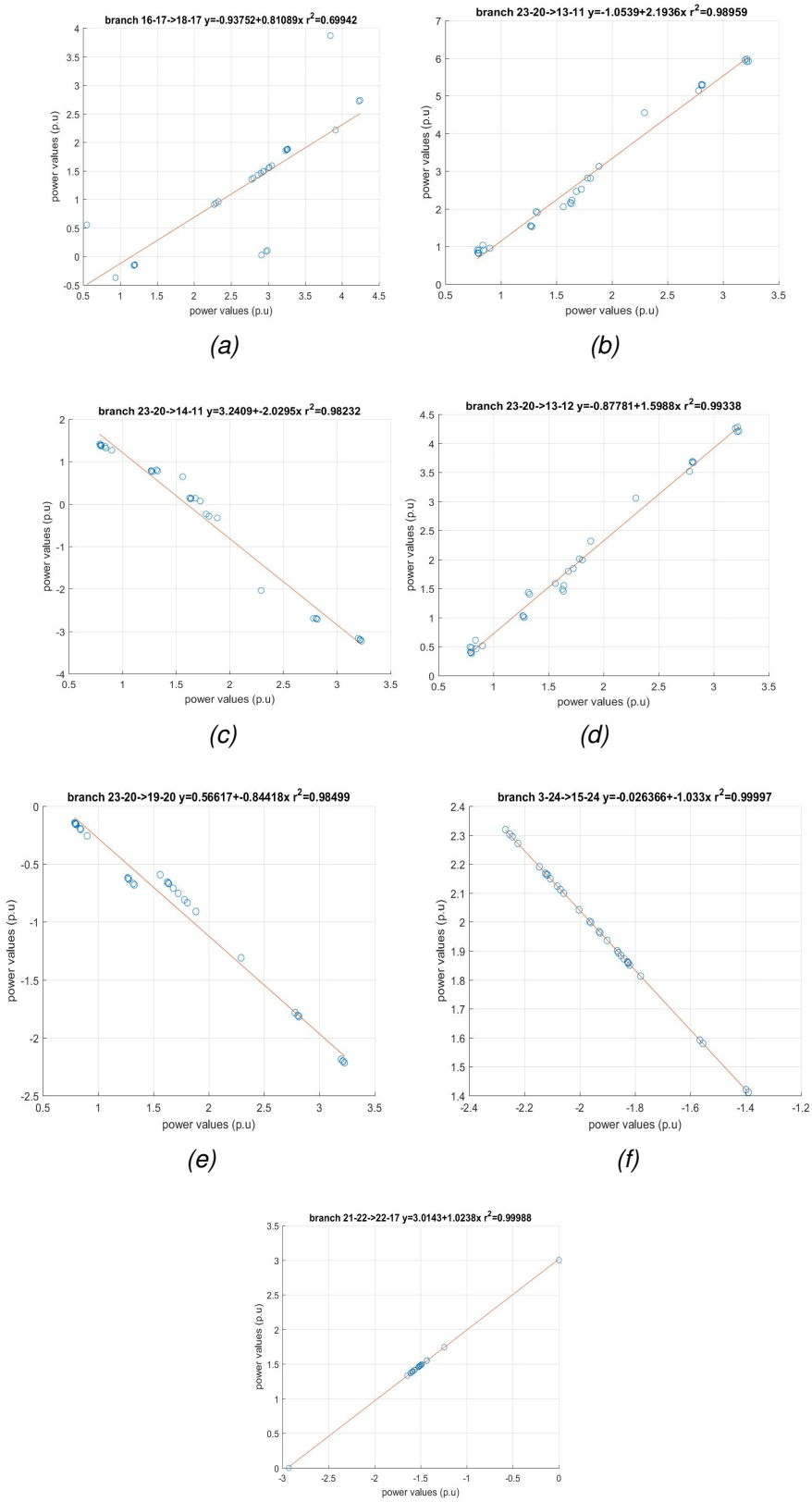


Figure 4.1: Graphs showing branch correlations with linear regression equation

The graphs show that the chosen branches do correlate well with each other based on the r-squared value, with the exception for branches 16-17 and 18-17.

4.2 Firewall Results For Command Sequence A

Several tests were ran with different critical components chosen each time. The traffic used was based on the captured traffic described in section 3.6. Table 4.1 shows the contingency table describing the false positive and negative and true positives and negatives. The number of packets along with the number of positive samples (packets that cause violations) and number of negative samples (packets that would not cause violations) are also included.

Table 4.1: Contingency table for command Sequence A packets indicating false positive and negative and true positive and negative conditions.

Packet Population = 58 # PS = 22, # NS = 36		Predicted Condition	
		Predict packet is harmful (true)	Predict packet is safe (false)
True Condition	Packet is harmful (true)	True Positive	False Negative
	Packet is safe (false)	False Positive	True Negative

Table 4.2 shows the results of the experiment where each case used different critical components as described.

Case 1: Observing bus 7, having 1 branch connection.

Case 2: Observing buses 4,5,6,14,22,24 which all have 2 branch connections to other buses.

Table 4.2: Firewall results for command Sequence A.

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7
# of false positives	2	0	7	4	5	5	1
# of false negatives	7	9	6	5	5	10	2
# of true positives	15	13	16	17	17	12	20
# of true negatives	34	36	29	32	31	31	35
FPR	2/36= 5.6%	0/36= 0	7/36= 19.4%	4/36= 11.1%	5/36= 13.9%	5/36= 13.9%	1/36= 2.8%
FNR	7/22= 31.8%	9/22= 40.9%	6/22= 27.3%	5/22= 22.7%	5/22= 22.7%	10/22= 45.5%	2/22= 9.1%
Accuracy	49/58= 84.5%	49/58= 84.5%	45/58= 77.6%	49/58= 84.5%	48/58= 82.8%	43/58= 74.1%	55/58= 94.8%

Case 3: Observing buses 1,2,3,8,13,17,18, and 19, which all have 3 branch connections to other buses.

Case 4: Observing buses 11,12,15,16,20, and 23, which all have 4 branch connections to other buses.

Case 5: Observing buses 9,10, and 21, which all have 5 branch connections to other buses.

Case 6: Observing buses 1,2,3,7,9,10,16,21, and 23, chosen based on the correlation coefficients. This case uses linear equations for checking other violations

on branches that may not be connected to one of the buses chosen.

Case 7: Like case 6 but with bus 4 included as a critical component since some Sequence A commands cause only voltage violations on bus 4.

Case 1-5 is based on the concept of the number of connections to other buses as explained in section 3.3 and does not include the linear regression equations in the checks for violations. Case 6 and 7 both incorporate the linear regression equations to check for violations on other branches that may not be directly connected to the buses chosen as critical components.

The violations that occurred when the commands were ran were voltage violations on buses 3, 4, 7, 9, 11, 12, and 24 and branch violations on branches 7-8, 9-3, 10-8, 13-11, 13-12, 14-11, and 16-14. Commands 10, 11, 14, 17, and 18 from Table 3.3, resulted in only voltage violations occurring on bus 4. Since these commands were the first commands with violations, how they were classified would change the outcome of the results in Table 4.2. When bus 4 was added to the list of critical components, case 7, commands 42 and 49 from Table 3.3 resulted in false negatives and command 43 resulted in a false positive occurring. The first 5 commands which resulted in only voltage violations at bus 4 were correctly classified after adding bus 4 to the list of critical components, and so the number of false positive and negatives decreased, increasing the accuracy of the firewall, as indicated in case 7 of Table 4.2. Commands 42 and 49 were suppose to cause branch power violations on branch 13-11, but these were not detected and this could be due to errors in the regression equation that shows the relationship between branch 23-20 and branch 13-11. However, commands 40, 47, 48, 50, 52, and 57 which only caused branch power violations occurring on branch 13-11 were observed using the regression equation for branches 23-20 and 13-11. The regression equations, while not perfect, when incorporated into the model along with the buses where the voltage violations occur does help in filtering out traffic as can be seen by the low false positive and negative rates, and the high accuracy in case 7. With case 6, since the first 5 commands that cause violations only caused voltage violations on a bus which was not observed, the errors in the ICS system would propagate with other commands. The propagation of these errors led to false classification of commands, as show with the high false negative rates and low accuracy in case 6.

4.3 Firewall Results For Command Sequence B

The second test was ran with a different set of commands but using the same linear regression equations as in the first test. The commands used for the second test are shown in Table 4.3 and the number of positive and negative samples are shown in Table 4.4. The commands in 4.3 caused voltage violations on buses 3, 4, 6, 8, 9,

and 24, and branch power violations on branches 1-3, 9-3, and 10-6.

Table 4.3: Command Sequence B

#	Command	#	Command	#	Command	#	Command
1	cutoff 9 3 0	11	cutoff 13 11 0	21	load 8 2.5	31	cutoff 16 14 1
2	cutoff 15 24 0	12	cutoff 10 5 0	22	cutoff 2 4 0	32	cutoff 15 24 0
3	cutoff 9 3 1	13	cutoff 10 6 0	23	cutoff 2 4 1	33	cutoff 15 24 1
4	cutoff 23 12 0	14	cutoff 10 5 1	24	cutoff 3 24 0	34	cutoff 17 16 0
5	cutoff 10 8 0	15	cutoff 10 8 1	25	cutoff 3 24 1	35	cutoff 17 16 1
6	cutoff 7 8 0	16	cutoff 13 11 1	26	cutoff 10 6 0	36	cutoff 19 16 0
7	cutoff 23 12 1	17	load 3 2.0	27	cutoff 10 6 1	37	cutoff 19 16 1
8	cutoff 1 3 0	18	load 6 2.0	28	cutoff 7 8 0	38	cutoff 18 17 0
9	cutoff 7 8 0	19	load 3 2.5	29	cutoff 7 8 1	39	cutoff 18 17 1
10	cutoff 1 3 1	20	load 6 2.5	30	cutoff 16 14 0	40	

Table 4.4: Contingency table for command Sequence B.

Packet Population = 39 # PS = 14, # NS = 25		Predicted Condition	
		Predict packet is harmful (true)	Predict packet is safe (false)
True Condition	Packet is harmful (true)	True Positive	False Negative
	Packet is safe (false)	False Positive	True Negative

This test was ran to see what the firewall would filter traffic for the same system but with a different set of commands and also to see if the linear regression relations between the branches would hold. The results of the second test are shown in Table 4.5. For this second test, case 7 includes bus 6 instead of bus 4 since some commands cause only voltage violations to occur on bus 6.

Looking at the results in Table 4.5 it could be said that the critical components in case 2 does a better job than almost all the other components in the other cases except for case 7. Upon, further inspection of which buses and branches the violations occurred at, the critical components in case 2 are directly connected to the buses and branches with the violations. Case 2 has buses 4, 5, 6, 14, 22, and 24 and so for the majority of commands that cause violations these buses are involved. In this test the use of the linear regression equations was not necessary since the violations occurred directly on the components being observed, which explains the 100% accuracy in case 7 in Table 4.5.

Table 4.5: Firewall results for command Sequence B

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7
# of false positives	1	1	2	1	1	2	0
# of false negatives	6	1	4	6	5	4	0
# of true positives	8	13	10	8	9	10	14
# of true negatives	24	24	23	24	24	23	25
FPR	1/25= 4%	1/25= 4%	2/25= 8%	1/25= 4%	1/25= 4%	2/25= 8%	0/25= 0%
FNR	6/14= 42.9%	1/14=7.1%	4/14=25.6%	6/14= 42.9%	5/14=35.7%	4/14=28.6%	0/14=0%
Accuracy	32/39= 82.1%	37/39= 94.9%	33/39= 84.6%	32/39= 82.1%	33/39= 84.6%	33/39= 84.6%	39/39= 100%

4.4 Discussion

Using only critical components that have a minimum or maximum number of connections within the IEEE 24 bus system only allows for the observation of violations on those components. As shown in case 2 of Tables 4.2 and 4.5, this can sometimes help the firewall in efficiently filtering traffic. If an attacker is able to cause violations on components that are not being observed the firewall will not be able to block the traffic that causes those violations. Finding relationships within the system and using them to provide comprehensive monitoring of the components within the system can allow a firewall to better filter out malicious traffic. The linear regression equations used in these tests, while not perfect, were able to help detect violations of components not being observed in the system by using the values of those components that were being observed and had good relations with the unobserved component. Only voltage and branch power violations were considered but linear regression equations for branches were the only ones formed, and voltage regression equations were not. By finding and including relations between bus voltages the firewalls performance could be improved, as evidenced by case 7 in Table 4.2. This could be something to look at in the future, to see if there are relationships between the bus voltages in the system.

Chapter 5

Conclusion

This chapter concludes the thesis by reviewing the overall research question and the sub-questions posed. The limitations faced during this thesis are discussed in section 5.1. Future work that extends the research conducted in this thesis is discussed in section 5.2.

In order to be able to correctly filter legitimate commands that could change the state of an ICS system to an undesired state, firewalls need to be able to take the commands' impact on the system into consideration. This thesis focused on determining whether information about a subset of the ICS system components could be used in order to depict the state of the system and to test the impact of the commands on the system. To this end several questions were posed, whose answers led to being able to develop a firewall system that is able to filter out legitimate commands with negative impacts on the ICS system. Below we repeat the posed subquestions and provide a concise answer to them.

1. **What are the main critical components in the ICS system?**
   To identify critical components within the ICS system two methods were used as discussed in section 3.3. The first method consisted of using components with a minimum or maximum number of connections to other components. The use of this method was based on the idea that the more connections a component has the greater impact it can have in the system. The second method consisted of choosing components which had high correlations with other components. In this thesis the correlations were based on the branches in the systems and linear regression equations were created to show the relationships. Both methods were tested by providing information about components classified as critical by the methods to the firewall and model developed. The method using minimum and maximum number of connections showed differing results, and using components with a higher number of connections as critical components did not improve the firewalls accuracy.

Using components with high correlations to other components resulted in the firewall having a high false negative rate and lower accuracy. This could be explained by some commands causing only voltage violations at buses that were not being observed, and no relationships between the bus voltages were looked at in this thesis. When these buses were added the false negative rate reduced and the accuracy increased, so choosing components with high branch power and bus voltage correlations as critical components may increase the accuracy further.

2. **How are these components modeled in the firewall?**
   A system model was created using JAVA were each component to be modeled was created separately. The system model allows for the creation of new components should they be needed in the future. The component models only contained the information needed to view their states and verify whether or not they adhered to the constraints placed on them. The methods used for identifying critical components were incorporated into the model and when a packet was processed different checks would be done based on the method chosen for identifying critical components. When using the minimum or maximum connections method only the components directly connected to the identified critical component would be checked for possible violations. When using linear regression, other components which may not be directly connected to the identified critical component, but were related to the critical component, would also be checked for violations using the linear regression equations. The violations checked for to assess whether the current traffic is malicious or not only included checking the component constraints, such as voltage limits and branch powers. Other violation checks could be incorporated such as checking Kirchoff's voltage law on the component.

3. **How do we know when the current traffic will make these components change the state of the ICS system?**
   To verify that a packet would have an effect on the processes in the ICS system several fields within the packet and the IEC 104 APDU were checked. These fields included the destination port, IEC 104 APCI format, Type Identification, and Cause of Transmission, and the Select/Execute bit. ICS components using IEC 104 for communications must use port 2404, as specified in the IEC 104 standard. The destination port was checked for the value '2404', and if the destination port specified was not this port then the packet was ignored. If the destination port matched, the APCI format is checked to verify that it is an I-format type, indicating that it contains an ASDU. All other format types, S and U format, are ignored. The Type Identification in the ASDU is checked to

verify that the command is a process command. In this thesis only three process commands were used, *single command*, *double command*, and *setpoint scaled value command*. Once the Type Identification is verified the Cause of Transmission is checked to verify that the it is an activation (value of 6) or deactivation (value of 8). When the Cause of Transmission is verified, the Select/Execute (S/E) bit is checked for a value of '0' (Execute). If the S/E bit is set to '1', then the packet will be getting the component ready, but will not have change any values. Using these checks, the packets are processed and only those passing the checks are processed further.

5.1 Limitations

There were several limitations in performing this research. Firstly, there was no access to a real world system and data which would help in validating this approach of filtering malicious traffic. Having real world data from components being used in actual systems would help in determining whether relationships between components could be found within the ICS systems in place today. While testing on a real life system is not feasible, incorporating the model used in this thesis into a real firewall that is used would help in better understanding the impact it would have on an ICS system. Impacts such as the added time in processing commands at the firewall, and how that could impact the real-time requirement in ICS systems.

5.2 Future Work

Future work following this thesis will be the integration of linear regression equations for bus voltages and generator real and reactive powers, to see if the performance of the firewall would improve when provided this information. This thesis only focused on the IEEE 24 bus system but the methods used in identifying critical components and in filtering traffic need to be tested on other ICS systems to verify that the methods can be applied to those systems. Another avenue for future research could be to investigate the use of the relationships between components in the system to detect tampering of devices, and manipulation of data by malicious actors.

Chapter 6

References

[1] B. Kim, D. Kang, J. Na and T. Chung, “Abnormal traffic filtering mechanism for protecting ICS networks”, 2016 18th International Conference on Advanced Communication Technology (ICACT), 2016.
[2] D. Kang, B. Kim, J. Na and K. Jhang, “Whitelist Generation Technique for Industrial Firewall in SCADA Networks”, Lecture Notes in Electrical Engineering, pp. 525-534, 2014.
[3] W. Shang, Q. Qiao, M. Wan and P. Zeng, “Design and Implementation of Industrial Firewall for Modbus/TCP”, Journal of Computers, vol. 11, no. 5, pp. 432-438, 2016.
[4] Batista, A. B., Kobayashi, T. H. and Medeiros, J. P. S. (2010). Application filters for TCP/IP industrial automation protocols. Critical Information Infrastructures Security, 111-123.
[5] J. Nivethan and M. Papa (2016) On the use of open-source firewalls in ICS/SCADA systems, Information Security Journal: A Global Perspective, 25:1-3, 83-93, DOI: 10.1080/19393555.2016.1172283
[6] J. Nivethan and M. Papa, “A Linux-based firewall for the DNP3 protocol”, 2016 IEEE Symposium on Technologies for Homeland Security (HST), 2016.
[7] A. Carcano, I. Nai Fovino and M. Masera, “Modbus/DNP3 state-based filtering system”, 2010 IEEE International Symposium on Industrial Electronics, 2010.
[8] D. Hadziosmanovic, The process matters. Enschede: University of Twente, 2014.
[9] H. Lin, A. Slagell, Z. Kalbarczyk, P. Sauer and R. Iyer, “Runtime Semantic Security Analysis to Detect and Mitigate Control-related Attacks in Power Grids”, IEEE Transactions on Smart Grid, vol., no. 99, pp. 1-1, 2016.
[10] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams and A. Hahn, “Guide to Industrial Control Systems (ICS) Security”, 2015.
[11] M. Luyben and B. Tyrus, “An industrial design/control study for the vinyl acetate monomer process”, Computers and Chemical Engineering, vol. 22, no. 7-8, pp. 867-877, 1998.

[12] C. Grigg, P. Wong, P. Albrecht, R. Allan, M. Bhavaraju, R. Billinton, Q. Chen, C. Fong, S. Haddad, S. Kuruganty, W. Li, R. Mukerji, D. Patton, N. Rau, D. Reppen, A. Schneider, M. Shahidehpour and C. Singh, "The IEEE Reliability Test System-1996. A report prepared by the Reliability Test System Task Force of the Application of Probability Methods Subcommittee", IEEE Transactions on Power Systems, vol. 14, no. 3, pp. 1010-1020, 1999.

[13] J. Bigham, D. Gamez and N. Lu, "Safeguarding SCADA Systems with Anomaly Detection", Lecture Notes in Computer Science, pp. 171-182, 2003.

[14] https://github.com/alphad05/Master_Thesis/blob/master/case24_ieee_rts_alpha.m

[15] https://github.com/alphad05/Master_Thesis/blob/master/SystemModel/simpleRuns.m

[16] https://github.com/alphad05/Master_Thesis/blob/master/test%20data.xlsx

[17] https://github.com/alphad05/Master_Thesis/blob/master/graph_br_vs_br.m

[18] https://github.com/alphad05/Master_Thesis/tree/master/SystemModel/src/model

[19]https://github.com/alphad05/Master_Thesis/tree/master/SystemModel

[20] J.J Chromik, A. Remke, and B.R Haverkort, "Improving SCADA security of a local process with a power grid model" In: 4th International Symposium for ICS & SCADA Cyber Security Research, ICS-CSR 2016, 23-25 August 2016, Belfast, UK (pp. 114-123).

[21] https://github.com/RocyLuo/IEC104TCP

[22] https://github.com/alphad05/Master_Thesis/tree/master/IEC104TCP-master

[23] M. Krotofil and D. Gollmann, "Industrial Control Systems Security: What is happening", [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6622964&tag=1

[24] "Has Your ICS Been Breached? Are You Sure? How Do You Know?", POWER Magazine, 2017. [Online]. Available: http://www.powermag.com/has-your-ics-been-breached-are-you-sure-how-do-you-know/?printmode=1

[25] "CRASHOVERRIDE Analysis of the Threat to Electric Grid Operations", DRAGOS INC, 2017. Available: https://dragos.com/blog/crashoverride/CrashOverride-01.pdf

[26] P. Huitsing, R. Chandia, M. Papa and S. Shenoi, "Attack Taxonomies for the Modbus protocols", International Journal of Critical Infrastructure Protection, vol. 1, pp. 37-44, 2008.

[27] East, S.; Butts, J.; Papa, M.; Shenoi, S. A Taxonomy of Attacks on the DNP3 Protocol; Critical Infrastructure Protection III; Springer Berlin Heidelberg: Berlin, Germany, 2009; pp. 6781.

[28] T. H. Morris and G. Wei, “Industrial Control System cyber attacks”, in BCS Learning and Development Ltd. Proceedings of the 1st international symposium for ICS & SCADA cyber security research, 2013.
[29] Z. Drias, A. Serhrouchni and V. Olivier, “Taxonomy of attacks on Industrial Control protocols”, in International Conference on Protocol Engineering and International Conference on New Technologies of Distributed Systems, 2015.
[30] “FIREWALL DEPLOYMENT FOR SCADA AND PROCESS CONTROL NETWORKS”, 2005. [Online]. Available: https://www.ncsc.gov.uk/content/files/protected_files/guidance_files/2005022-gpg_scada_firewall.pdf.
[31] “Guide to Industrial Control Systems (ICS) Security”, 2015. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r2.pdf.
[32] “Secure Architecture for Industrial Control Systems”, Sans.org, 2015. [Online]. Available: https://www.sans.org/reading-room/whitepapers/ICS/secure-architecture-industrial-control-systems-36327.
[33] “Industrial Cybersecurity Research Lab Opens — Automation World”, Automationworld.com, 2015. [Online]. Available: https://www.automationworld.com/industrial-cybersecurity-research-lab-opens.