RECREATING WIDE AREA INDUSTRIAL CONTROL SYSTEMS NETWORK WITHIN AN EMULATED ENVIRONMENT

A Thesis

by

PATRICK JASON RAYMOND WLAZLO

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTERS OF SCIENCE

Chair of Committee,	Ana Goulart
Committee Members,	Katherine Davis
	Rainer Fink
Head of Department,	Jorge Leon

December 2021

Major Subject: Engineering Technology

ABSTRACT

The integration of Information Technology (IT) and Industrial Control Systems (ICS) has made the monitoring and remote controlling of ICSs inexpensive and reliable. However, the lack of cybersecurity protection of field devices when IT was incorporated into ICS systems has allowed for malware attacks, such as Stuxnet, the Ukraine attacks, the intrusion in the European Network of Transmission System Operators (ENTSO) in 2020, and the Colonial Pipeline ransomware attack in 2021. Fortunately, today most ICS infrastructure stakeholders are re-evaluating the security posture of their cyber-physical networks. To help researchers find better solutions to a wide range of cyberattacks, this work introduces a novel approach that can recreate a communication network topology for a large-scale power system model. Several use cases were developed to show the effect a (main-in-the-middle) MiTM attack can impose on a grid when Distributed Network Protocol version 3 (DNP3) telemetry is changed. With these use cases, the Cyber Physical Resilient Energy Systems (CyPRES) research team devised a new mechanism that uses data from network monitoring tools and intrusion detection systems (IDS) to detect such attacks. In addition, a software pipeline between NP-View and the Common Open Research Emulator (CORE) is introduced that could recreate larger scales of an ICS network in an emulated environment is proposed that can be used for research.

DEDICATION

To Margie Wlazlo, Adam Wlazlo, Andrew Wlazlo, Froma Wlazlo, and extended family, I cannot thank you enough for all that you have done from me and continue to do for me. Without you by my side, it would not have been possible for me to dedicate my time toward earning my bachelor's or master's degrees. Nor would I have been able to conduct research for Texas A&M University for these past few years.

To my advisors, committee, colleagues on the CyPRES project, researchers at Sandia National Lab, Dr. Saman Zonouz from Rutgers University, and Edmund Rogers from the University of Illinois; I am very glad to have such a dedicated group of individuals mentoring me and guiding me along my education and career paths. I have been very fortunate to have met and worked alongside each and every one of you for these past few years.

To Dr. Ana Goulart and Dr. Katherine Davis. I am sad that my time here as a researcher has come to an end, as all good things do. There is still so much good work to be done and research questions that have yet to be explored. I am very excited to see the next phase of the CyPRES project has in store for all of us!

To Abhijeet Sahu, Hao Hung, Zeyu Mao, and Kolten Knesek, the long hours that you have spent working with me to set up the testbed, conduct experiments, and learn how industrial control systems operate, I cannot thank you enough. I wish you the best in your academic career and in your future careers.

To my friends, the COVID-19 pandemic, professors, teachers, and others I have not listed by name, I would like to thank you for all the disagreement, arguments, good-times and bad that we have shared. Each one of you has shaped me in ways I have yet to fully understand. But without you, I cannot be certain that I would have made it this far in my academic career and I would like to thank you for your role that you have played in my life.

ACKNOWLEDGMENTS

This research is supported by the US Department of Energy Cybersecurity for Energy Delivery Systems program under award DE-OE0000895.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of: Professor Anna Goulart from the department of Engineering Technology and Industrial Distribution, Professor Katherine Davis of the Department of Electrical and Computer Engineering, and Professor Rainer Fink from the department of Engineering Technology and Industrial Distribution of Texas A&M University. The CPTL framework was designed by current and former members of the CyPRES project; Kevin Price, Christian Veloz, Abhijeet Sahu, and Patrick Wlazlo. Parts of the data collected for the MiTM experiment was coordinated by current members of the CyPRES project; Abhijeet Sahu, Zeyu Mao, Hao Huang, Kolten Knesek, and Patrick Wlazlo. The iptable conversion script used in the NP-View to CORE pipeline was initially designed and implemented in Python by a former member of the CyPRES group member; Julian Velasquez. All other work conducted for the thesis completed by the student independently.

Funding Sources

This graduate study was supported by a fellowship from Texas A&M University and the Department of Energy Cybersecurity for Energy Delivery Systems program under award DE-OE0000895.

NOMENCLATURE

ICS	Industrial Control Systems
IT	Internet Technology
ENTSO	European Network of Transmission System Operators
MiTM	Man-in-The-Middle
RESLab	Resilient Energy System Lab
CyPRES	Cyber-Physical Resilient Energy Systems
FDI	False Data Injection
FCI	False Command Injection
DNP3	Distributed Network Protocol version 3
ML	Machine Learning
NP-View	Network Perception-View
CORE	Common Open Research Emulator
RTAC	Real-Time Automation Controllers
RTU	Remote Terminal Units
TLS	Transport Security Layer
ELK	Elasticsearch, Logstash, and Kibana
AC	Alternating Current
DMZ	Demilitarized Zone
CPS	Cyber-Physical System
SCADA	Supervisory Control and Data Acquisition
IDS	Intrusion Detection Systems

OPAL-RT	Ordinateur Parallèle et Logiciel Temps- Réel (English Translation: Parallel Simulation and Real-Time software)
PDC	Phasor Data Concentrator
CPTL	Cyber-Physical Topology Language
OOP	Object Orientated Programming
VM	Virtual Machine
RTU	Remote Terminal Units
IP	Internet Protocol
VLAN	Virtual Local Area Network
MAC	Medium Access Control
LAN	Local Area Network
UTP	Unshielded Twisted Pair
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
SSH	Secure Shell
ERCOT	Electric Reliability Council of Texas
BANC	Balancing Authority of Northern California
NYISO	New York Independent System Operator
IID	Imperial Irrigation District
LADWP	Los Angeles Department of Water & Power−PacWest
JSON	JavaScript Object Notation
XML	eXtensible Markup Language
kNN	K-Nearest Neighbour
A&M	Agricultural and Mechanical
ISP	Internet service provider
NIDS	Network Intrusion Detection System

RJ45	Registered Jack type 45
SEL	acSELerator
FreeBSD	Free Berkeley Software Distribution
PC	Personal Computer
OS	Operating System
RAM	Random Access Memory
CPU	Central Processing Unit
DNS	Dynamic Naming System
SNMP	Simple Network Management Protocol
CRC	Cyclical Redundancy Check
PCAP	Packet Capture
gRPC	Google's version of Remote Procedure Call
CRC	Cyclical Redundancy Check
MW	Megawatt
DNS	Domain Name System
URL	Uniform Resource Locator
NERC	North American Electric Reliability Corporation
CIP	Critical Infrastructure Protection
WAN	Wide Area Network
API	Application Programming Interface
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
DHCP	Dynamic Host Configuration Protocol
GUI	Graphical User Interface
PYGUI	Python's Graphical User Interface

TABLE OF CONTENTS

Page

ABSTRACT i

DEDICATION ii

ACKNOWLEDGMENTS iii

CONTRIBUTORS AND FUNDING SOURCES iv

NOMENCLATURE v

TABLE OF CONTENTS viii

LIST OF FIGURES x

LIST OF TABLES xii

1. INTRODUCTION 1

    1.1 Literature Review 2
        1.1.1 Review of CPS Testbeds 6

2. CREATING A CYBER TOPOLOGY MODEL 8

    2.1 Designing a Cyber-Physical Framework 8
    2.2 Developing Classes for Cyber-Physical Framework 10
    2.3 Validating the Cyber-physical Framework 15

3. DESIGN OF THE RESLAB TESTBED 17

    3.1 Testbed Architecture 17
    3.2 Network Emulator 18
    3.3 Network Intrusion Detection System 19
    3.4 Data Visualization and Correlation 20

4. DESIGN OF MITM ATTACK 22

    4.1 Developing a DNP3 MiTM Script 22
    4.2 Impact of Network Delays to Power System 24
    4.3 MiTM Attack Design Considerations 24
    4.4 MiTM Attack Uses Cases 27
    4.5 MiTM Detection 28

5. RECREATING ICS NETWORK IN EMULATED ENVIRONMENT 31

5.1 Overview of NP-View to CORE Pipeline 32
5.2 NP-View Data Export 33
5.3 NP-View To CORE Python Script 33
5.4 CORE-PYGUI Visualization 37
5.5 Rebuilding An Actual ICS network 39

6. CONCLUSION 42

REFERENCES 45

LIST OF FIGURES

FIGURE Page

1.1 A comprehensive diagram showing the network or cyber topology, with a substation, utility control center, and the regulatory (balancing authority) network. Source: Adapted from [1]. 5

1.2 A few of the common communication networks configurations. 6

2.1 The relations between the CyPRES projects frameworks for super-classes and subclasses. Source: Adapted from [2]. 9

2.2 Using Splunk NetViz app to display the network topology for a generic substation topology from our framework. 13

2.3 Using Splunk's NetViz app to display a utility networks star topology. 14

2.4 A visualization using Splunk's Maps to show how the kNN ML algorithm can cluster substations into utility control centers. 15

3.1 The RESLab emulation-based testbed architecture and its interconnections. 18

3.2 Data flow pipeline for ELK stack. From the lowest layer of data collection (left), data storage (middle), and data visualisation (right). Source: Adapted from [3]. 21

4.1 An example hexadecimal representation for a DNP3 packet. It shows a detailed breakdown of the DNP3 header and payload. Source: Adapted from [3] 23

4.2 A vector and node diagram showing how the DNP3 MiTM attack was programmed to be as undetectable as possible. Soure: Adapted from [3]. 25

4.3 A time series graph showing the correlation of the ARP alerts and DNP3 alerts generated by Snort during Use Case 2. Source: Adapted from: [3]. 30

5.1 The CORE communication network used to capture the DNP3 MiTM datasets. Source: Adapted from [3] 31

5.2 NP-View to Common Open Research Emulator (CORE) Pipeline 32

5.3 Algorithm used to create links between nodes from raw NP-View in the CORE. 34

5.4 Algorithm used to find the missing IP address and largest possible netmask. 35

5.5 A side-by-side comparison showing how the NP-View to CORE Pipeline can systematically recreate a network topology. The full-size version of these three figures can be found in Appendix A 37

5.6 Using CORE-PYGUI to visualization the NP-View to CORE pipeline's ability to recreate a real portion of an ICS network in an emulated environment. 40

1 Fully detailed NP-View version of CyPRES CPTL Framework networks topology. .. 43

2 Fully detailed NetworkX version of CyPRES CPTL Framework networks topology. 44

3 Fully detailed CORE PY-GUI version of CyPRES CPTL Framework networks topology. 44

LIST OF TABLES

TABLE Page

2.1 A numerical breakdown of nodes and link classes that were created to a synthetic cyber-physical topology of the McAllen region. 16

4.1 The four DNP3 MiTM use cases developed by the CyPRES team. 28

4.2 The average processing times of the MiTM FDI and FCI attack scripts based on type on type of packet being sniffed. Source: Adapted from [3]..... 29

5.1 A numerical breakdown of type of nodes inside the CORE session that was based of the CyPRES CPTL model. More information about this model can be found in [2, 1]..... 39

5.2 A numerical breakdown of type of nodes inside the CORE session that was based of Industry NP-View data. 41

1. INTRODUCTION

Automating the control of critical infrastructure such as energy, water, and chemical facilities, Industrial Control Systems (ICS) has undeniably increased the reliability of these systems. The inexpensive remote monitoring that Internet Technology (IT) networks provide is helping to optimize the utility sector [4]. However, integrating IT (cyber) and ICS (physical) systems has historically been focused on making these systems reliable, not secure [5]. This lack of consideration for cybersecurity protection in the design requirements for many ICS control protocols has permitted malware attacks, such as Stuxnet [6], the Ukraine attacks [7, 8], the intrusion in the European Network of Transmission System Operator (ENTSO) in 2020 [9], and the Colonial Pipeline ransomware attack in 2021 [10].

To improve the security and resilience of ICS systems, especially power systems and smart grids, this research investigates how to detect man-in-the-middle (MiTM) attacks in the energy utilities' communication network. Our approach uses an emulated CPS testbed named the Resilient Energy Systems Lab (RESLab) that houses a synthetic communication network topology for large-scale power system modeling. The Cyber Physical Resilient Energy Systems (CyPRES) project has developed four use cases to demonstrate the harm that a cyberattack can impose on a power grid. Two of the use cases are False Command Injection (FCI) attacks, the other two are False Data Injection (FDI). All four use cases are designed to modify Distributed Network Protocol version 3 (DNP3) packets while they travel over an ICS communication network. By using the RESLab's testbed and four use cases, the CyPRES team was able to capture custom data sets. These data sets were then used to improve the detection of these cyberattacks using data correlation technique and machine learning (ML) algorithms.

From partnering with the power utility industry, we discovered a knowledge gap between energy industry experts, network operators, and research groups. This divide can lead to vulnerabilities being unintentionally overlooked by these three groups. The lack of knowledge and resources that are shared between industry and research partners facilitates the divide. One of primary factors

is the level of abstraction between the network models that research groups have used to generate their data sets. The abstraction of an ICS network leads to inconsistent that make it harder for industry expert in implement the solutions that researchers provide. To help bridge this divide, a software pipeline from Network Perception-View (NP-View) to Common Open Research Emulator (CORE) is proposed that could help cybersecurity research groups test their proposed solutions for a cyberattack in an industry-inspired ICS network.

1.1 Literature Review

Most ICS protocols used for power generation, transmission, and distribution in North America are designed to be dependable. Most field devices currently in operation were developed in the early 1970s with design requirements stating they should operate 24 hours a day, for 7 days a week, for a 20 year duration without the need for major repairs. This short list of design requirements was reasonable at the time. However, since the 1970s the world has witnessed Stuxnet silently issue commands that caused centrifuges to wear themselves out [6], the Ukraine attack plunging over 200,000 Ukrainian citizen into darkness [7], and the Colonial pipeline disaster shutting off roughly 45% of the East Coast's of United States supply of diesel, petrol, and jet fuel. These events have been seen as wake up call for nations and utility companies around the world, fueling research on how to harden the security of field devices such as RTACs, protection relays, and RTUs while still making them robust enough to weather most natural disasters.

There have been a fair amount of publications describing how to secure ICS protocols such as DNP3 [11], mainly because DNP3 is vulnerable to various forms of cyberattack. The most troublesome attacks including eavesdropping, FDI, and FCI attacks. The aforementioned attacks are three common forms of a MiTM attack. Or to put simply, when an adversary silently positions him/herself within the communication channel of two or more devices. Both of these devices are left unaware that the MiTM is happening because either the packet data is left unchanged or the MiTM falsifies the data so that neither device knows they are being deceived. This single hacker, or group of hackers, can then change telemetry values at will. In the power grid. These cyber events can cause grid equipment to fail in some instances. Or even worse, they can cause widespread

blackouts.

There are some common ways to prevent MiTM attacks, such as end-to-end encryption or using an authentication mechanism. For instance, the a DNP3 packets application layer can be encrypted by using Transport Layer Security (TLS) encryption [12]; however, this has not widely adopted by energy utilities. There are a few reasons for the lack of TLS layer encryption being adopted by industry. The most prominent is the amount of downtime necessary to upgrade RTACs and RTU with a DNP3 protocol stack that includes TLS encryption and the need to reprogram field devices with an authentication mechanism. The downtime can be as much as $300,000 per hour [13], making upgrading field equipment uneconomical for most utility companies. For this reason, the CyPRES project's publications have focused on using data that field and IT devices already generate in order to detect cyberattacks.

In the MiTM attack, we assume an intruder can access the substation network where DNP3 data packets are being transmitted using cleartext, i.e., with any form of encryption or authentication mechanism. Then we use open-source networking tools like Snort and Elasticsearch Logstash Kibana (ELK) stack to detect the MiTM attacks. This requires no or minimal amount of downtime by the utility companies, making this approach more practical for utilities to implement.

There have been numerous research works that use small power system bus cases to study how natural disasters and cyber security events can affect the reliability of the power grid. In these cases, electricity busses, generators, and loads are simulated in order to study how power grid as a whole would react to a given cyber or physical event [14, 3, 15, 16]. The size of these physical network typologies can range from an 8-bus substation model [17] to a 300-bus substation model [18]. Once these synthetic power system models are created, they can be simulated using Powerworld [19] to study the effects that tripping a breaker or turning off a generator will have on other components in a power grid.

There are some examples of research groups trying to emulate large scale grid typologies. For instance, A. Birchfield *et. al* [20] created a 2000-bus synthetic power grid that was used to conduct research on a large power grid. While this synthetic model is useful in studying topological

connectivity and convergence of high voltage alternating current (AC) circuit power flows, it was not within its scope to incorporate any network communication infrastructure. Also, the physical simulator they used (Powerworld) was not meant to study the cyber aspects of the power grid. For this reason, in [2] we created a framework that would add a communication network layer on top of the 2000-bus physical topology. The synthetic cyber-physical model of a large-scale power grid included network, routers, switches, and demilitarized zones (DMZs) for Supervisory Control and Data Acquisition (SCADA) systems. The cyber-physical synthetic model we introduced included balancing authorities, utility control centers, and substations connected in a simple, hierarchical fashion. A visualization of this network is shown in Fig. 1.1.

With the MiTM attack script and large scale cyber-physical power grid emulator setup in the RESLab testbed, the CyPRES team was able to create datasets for the CyPRES team and other researchers to study. While also leveraging open source Intrusion Detection System (IDS) and ML algorithms to detect and mitigate against such attacks.

However, unlike physical transmission grids, there is no standard for how to structure a communication network. This usually leads to researchers having to simplify the topology of a network into a ring, star, tree, or a hybrid of multiple simple typologies as shown in Fig. 1.2. The realism of this network setup could not validated. It would be more ideal for researchers to set up a network that is directly inspired by an actual power grid communication network. For this reason, the last part of this thesis will show how network data obtained from industry can help us create a more realistic network model.

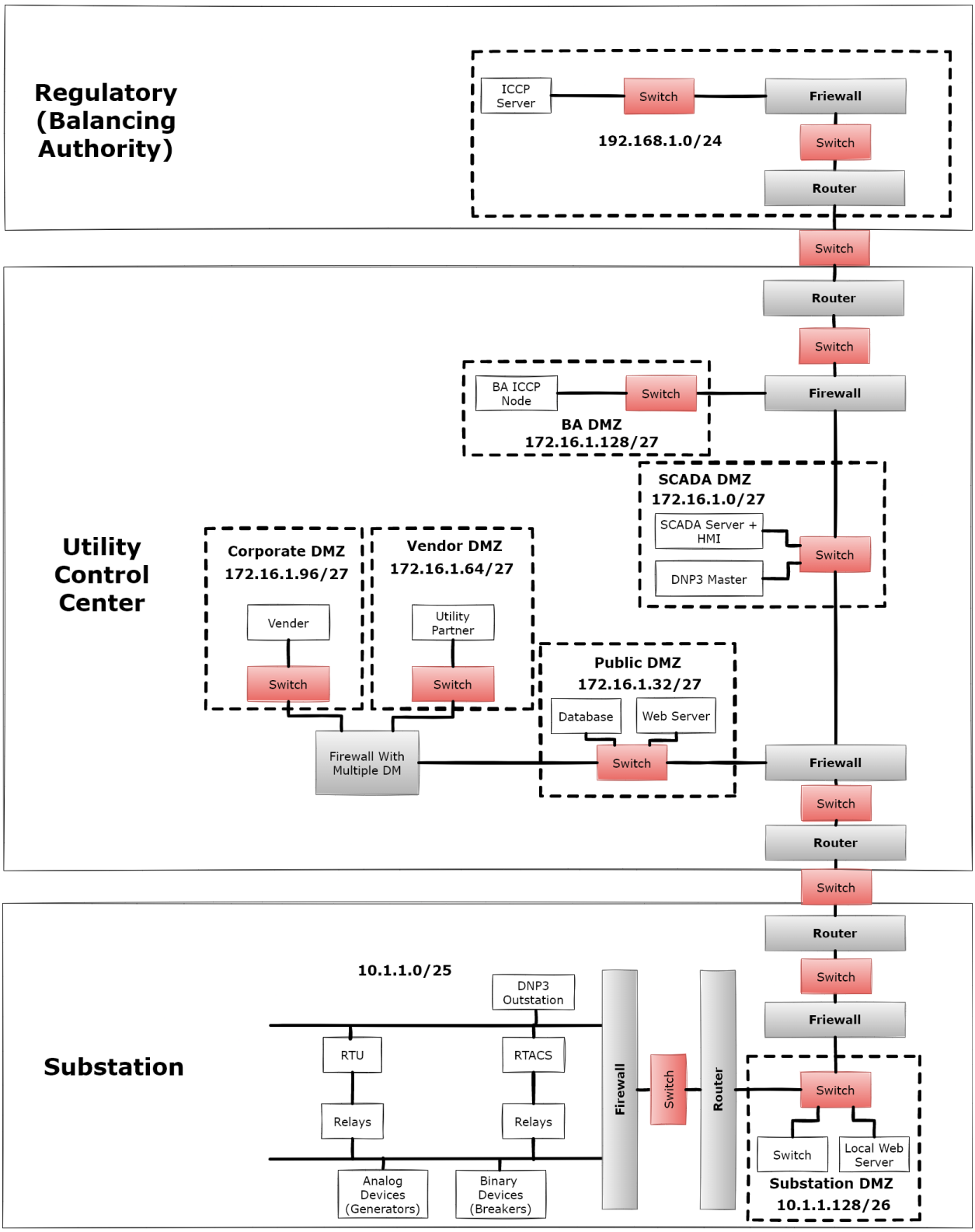

Figure 1.1: A comprehensive diagram showing the network or cyber topology, with a substation, utility control center, and the regulatory (balancing authority) network. Source: Adapted from [1].

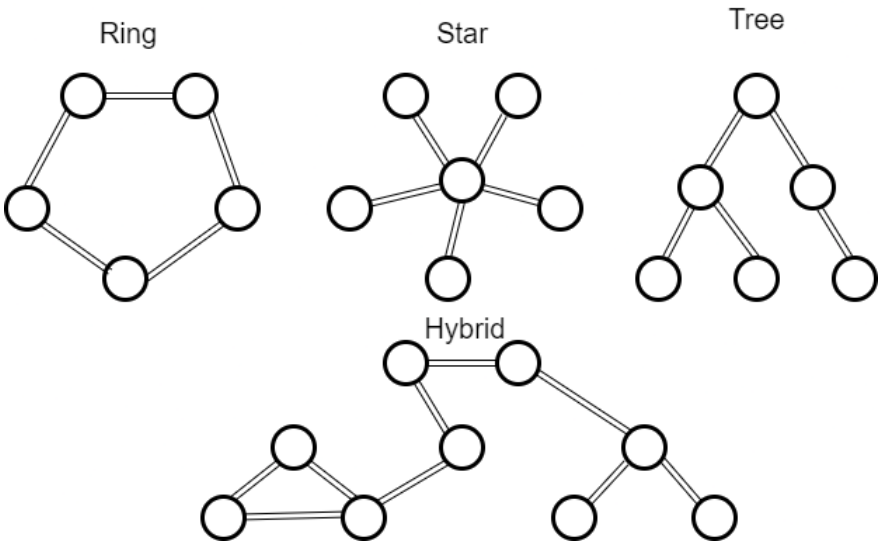

Figure 1.2: A few of the common communication networks configurations.

1.1.1 Review of CPS Testbeds

There are various other cyber-physical testbeds that have studied cyber attacks vectors against a single or many different ICS protocols. For instance, in [21] the authors created a testbed to study the wide area monitoring for a synchrophasor measurement system. The paper provides a rule-based IDS. As is implied throughout their work, a lot of knowledge from industry stakeholders and time from experts is needed to implement a proficient security program that can detect cyber intrusions in an ICS network. To over come this high barrier, the authors proposed solution is a hybrid IDS that uses data mining techniques with domain expert knowledge. The IDS system is then able to reduce the strain that is placed on operators to know their IT and ICS systems and know how to use rule-based IDS systems properly. While this work is technically sound and the authors were able to achieve their objective, it was tested on a two-line three-bus power transmission system using real-time digital simulator (RTDS) as the power simulator that had hardware-in-the-loop (HIL). The structure of the communication network was not documented and it is has not been verified that their research solution would scale well to a larger ICS network.

To give another example, the authors of [22] propose a novel approach for designing a testbed

as a foundation for a novel IDSs that can detect MiTM, Denial-of-Service (DoS) attacks, Address Resolution Protocol (ARP) spoofing, relay tripping, and disconnecting substations on wide area ICS networks. The authors use OPAL-RT, a real-time digital simulator, with Phasor Data Concentrator (PDC) and devices for HIL. While their testbed serves its purpose as an environment for testing and validating various cyber attack-defense algorithms, there is no method to validate their proposed solution. Their results could be improved by testing their IDS system in a communication network model that is directly inspired by industry.

The validation of other groups' experiments and research is of great concern within the research community. For ICS network and cybersecurity research, there has to be a lot of assumptions made on how the communication network functions in order to make advancements in the utility and networking fields. Having some way for researchers to validate their research that was developed using synthetic data or custom datasets is essential to prove the validity of their tireless research effort.

To help facilitate the creation of networking schemes for re-routing in ICS network, as a final chapter in this thesis dissertation I am proposing a software pipeline to create an ICS network within the RESLab research tested. This will be accomplished using the CORE [23] along with a software tool that is widely used by energy utilities to audit their network and firewall rules called NP-View [24, 25]. The goal is to automatically recreate a network in an emulated environment, as shown in Fig. 5.2. This emulated environment will allow researchers to test that their proposed solution will work within an emulated ICS environment. However, this is only a step toward validating their work, and only a physical-validation experiment will definitely prove that a research solution is viable for industrial use.

2. CREATING A CYBER TOPOLOGY MODEL

The initial phase of the CyPRES testbed research focused on merging a communication (cyber) network topology with a power system (physical) topology to create a cyber-physical topology. By having a cyber-physical framework a wide area network could be emulated in the RESLab environment. This gives the CyPRES team a platform to design and study more realistic cyberattacks.

2.1 Designing a Cyber-Physical Framework

The cyber-physical topology that the CyPRES team created was based on the Cyber-Physical Topology Language (CPTL) framework [26]. The CPTL framework we proposed is comprised of vertices and edges that represent links and nodes within a cyber-physical network. For instance, the vertices are the network nodes (physical devices) and the edges are used to connect the nodes to one another (communication links). The vertices and edges have defined attributes with them that can be used in modeling the network in either a simulated or emulated environment.

These vertices were implemented using Python Object-Oriented Programming (OOP). We defined Python classes to store attributes for *Switch*es, *Router*s, *Firewall*s, *Relay*s, *RTU*s, *Relay Controllers*, or *CyberNode*s. A *CyberNode* is a generic class to represent any communication device within a network. However, our framework allows any research group to create a more well-defined vertex if they choose to. The *CyberNode* is an optional class that serves as a generic communication device for a node that does not require a newly defined sub-class of the *Node* class such as a virtual machine (VM).

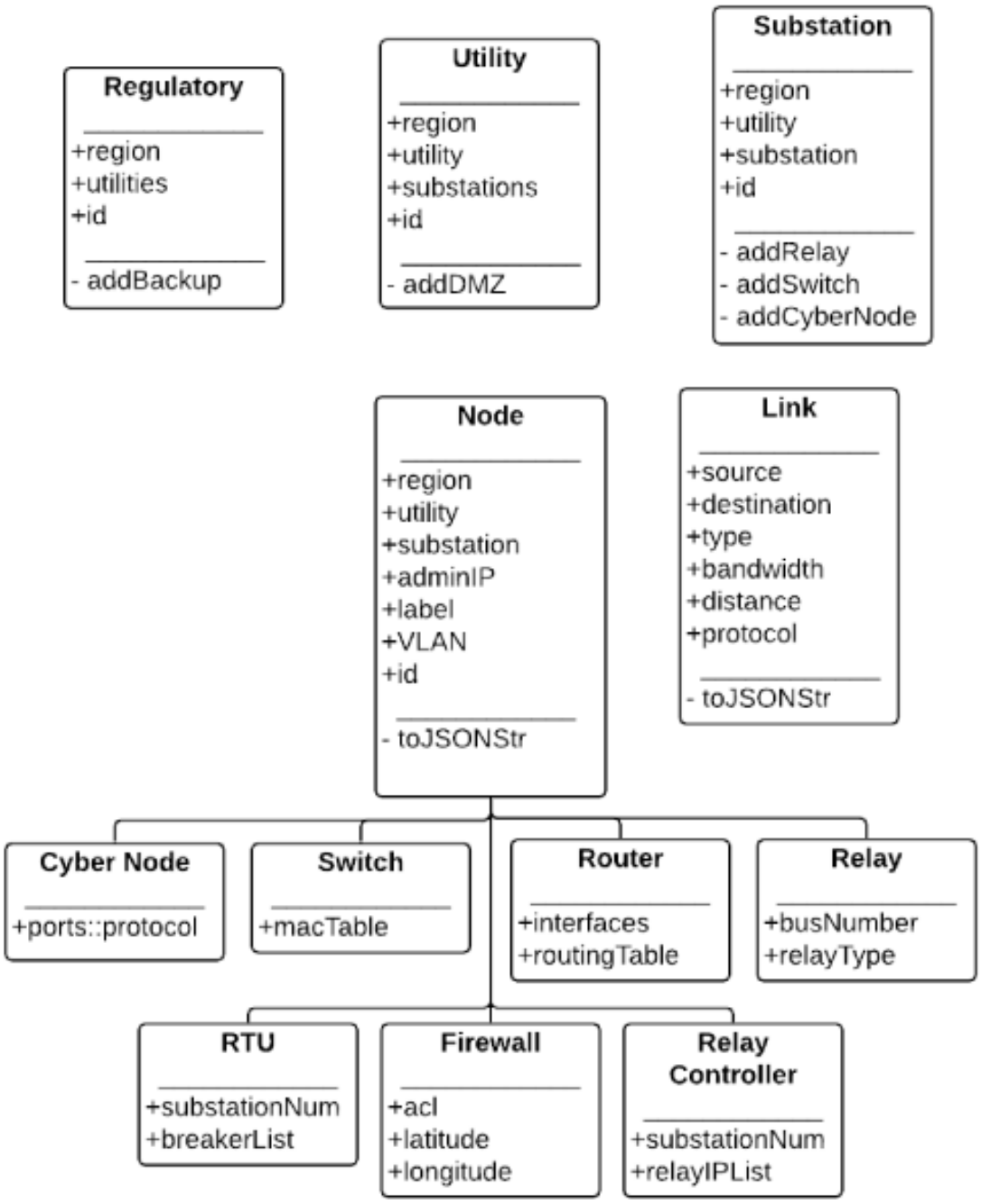

Figure 2.1: The relations between the CyPRES projects frameworks for super-classes and sub-classes. Source: Adapted from [2].

2.2 Developing Classes for Cyber-Physical Framework

The vertex contains attributes to help identify and emulate nodes within a cyber-physical network. In the Python class structure, a vertex is implemented as a *(*Node) class. The attributes stored by each class in our CPTL framework is shown in Fig. 2.1. As shown, *Node* class is the super-class of the *Switch*, *Router*, *Firewall*, *Relay*, *RTU*, *Relay Controller*, and *CyberNode* classes, and has the following attributes:

- *region* - One of the three major interconnections of the United States grid (*Northern*, *Eastern*, and *Texas*). It can also be used to segment a single ISO into many regions, such as *North*, *East*, *South*, and *West*, for smaller use cases.
- *utility* - The name of the utility company that manages the substations.
- *substation* - The name of the substation that houses the device.
- *adminIP* - The IP address that is used for remote monitoring and managing of the device.
- *label* - The human readable label for that device used for easier identification of the device.
- *VLAN* - The name of the Virtual Local Area Networks (VLANs) the nodes interfaces are directly connected to.
- *id* - The unique identifier that is used to identify this node in a large-scale network.

The attributes listed above are common to each type of node underneath the *Node* class. Any attributes that are unique to a specific node are stored within a sub-class of the *Node* class. For instance, switches use Media Access Control (MAC) addresses to send information from one node to another. Therefore, the *Switch* class stores this information in our framework. This is not meant to imply that other devices do not store MAC addresses information. It is just simpler for emulation purposes to house all of this information in the switch, because the purpose of MAC addresses is to give switches the ability to forward frames within a local area network (LAN).

In this framework, the edges are implemented as the *Link* class. The *Link* class is a generic class that contains information characteristics to describe the connection media between two *Node*s that are connected together. The *Link* class is intended to be a super-class for any of the different connection types that a transmission grid could have. For our network framework, we did not have any use case that required us to distinguish between the different links. In our emulated environment the links are primarily used to add delay to packet data or stochastically drop packets as they travel over a network, it was necessary to define a sub-class for the *Link* class. However, there might be use cases that will require more attributes that are not being stored by the *Link* class. For these instances, the framework allows the user to create sub-classes for Unshielded Twisted Pair (UTP) Category 5 links, fiber optic links, or microwave links, if a use case requires it. The following attributes are the minimum *Link* attributes requires creating an emulated network:

- *region* - One of the three major interconnections of the United States grid (*Northern*, *Eastern*, and *Texas*). Can also be used to segment a single ISO into many regions such as *North*, *East*, *South*, and *West*, for smaller use cases.
- *source* - The *id* of the source node.
- *destination* - The *id* of the destination node.
- *type* - can be *Ethernet*, *serial*, *multimode fiber*, or *microwave links*.
- *bandwidth* - The maximum number of bits per second (bps) that can be sent through this link.
- *distance* - The physical distance in meters of the link from source to destination. This is a rough estimate of how far apart the nodes are so that we can calculate propagation delays.
- *protocols* - This is the type of application layer protocol the nodes are using to communicate with each other. For example: HTTP, HTTPS, SSH, and DNP3.

In the framework, there are three other classes that are used to segment the network into different domains. At the top layer is the *Regulatory* domain, which is generally referred to as a balancing authority, such as the Electric Reliability Council of Texas (ERCOT), Balancing Authority of Northern California (BANC), New York Independent System Operator (NYISO), among others. The sole purpose of the balancing authority is to regulate the generation, transmission, and distribution of electrical power within any given region of the United States' grid. In the middle of the hierarchy is the *Utility* company. The utility company is usually a private institution that is tasked with generating or transmitting high-voltage power over long distances to distribution facilities, or distributing electrical power for private and commercial use. At the lowest level is the *Substation*. A substation contains physical devices used to step-up or step-down high voltage power for the power grid. Each of the classes has its own set of attributes that can store specific information to help organize the nodes into different segments of the entire grid. A visualization of how the framework is breakdown into the three distinct domains is shown in Fig. 1.1.

Once all of or part of these attributes are defined, it can be output in a specific file storage format. For our use cases we store this data in JavaScript Object Notation (JSON) format. JSON is ideal because it is the most human-readable format of the three major formats (JSON, XML, and CSV). This allowed for the testing and debugging of the network attributes while programming the full scale network topology.

In our framework, you can see the *toJSONStr* function which first translates all the attributes to a JSON template that is used to save all the attributes in a JSON file. These JSON files can then be imported into another application for creating an emulated network. The JSON file was imported into a software tool called Splunk [27], which allowed us to visualize the information that had been generated.

The substation visualization that Splunk produced is shown by Fig. 2.2. To keep the visualization easy to understand, we showed a simplistic version of the network. This network had two two relays, a RTU, two switches, a windows machine, a printer, a router, and a firewall. Each individual switches in our model could be thought of as VLAN. This is because for emulation, it is more

important to know which devices can communicate with each other, rather than which switch they are physically connected to.

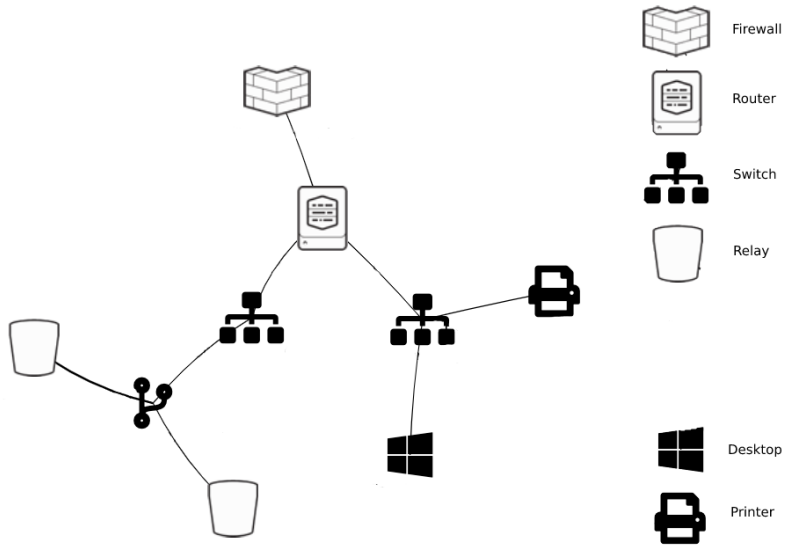

Figure 2.2: Using Splunk NetViz app to display the network topology for a generic substation topology from our framework.

Likewise, in Fig. 2.3 an example of the utility segment of the power grid cyber network topology is shown. This network is comprises the substation routers at the end of each network diagram. All the substation routers are connected to the substation facing utility control center firewall, shown by the firewall in the middle of Fig. 2.3. The DMZ at the utility level was omitted for this diagram because it is difficult to display, but it was stored in the JSON data file to display if it was necessary to debug the DMZ portion of the network. In an actual ICS network the utility and the substation firewalls might be connected via the internet or through an internet service provider (ISP). However, these firewalls are usually connected to each other using a site-to-site virtual private network (VPN) tunnel. Therefore, to emulate data traveling over an ICS network, we connected the firewalls to each other.

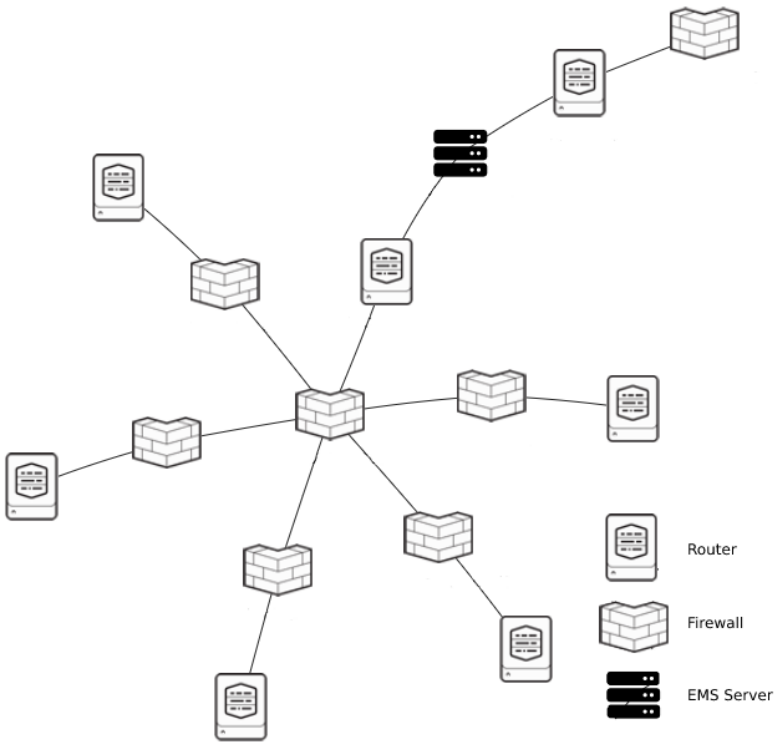

Figure 2.3: Using Splunk's NetViz app to display a utility networks star topology.

The location of each of the utility control center was not defined in the Texas 2000-bus model, but the location of each substation was included in the model[28]. As shown by Fig. 2.4, the substations were divided into seven clusters and connected to a central utility control center inside that cluster. Fig. 2.4 shows the location of seven groups that were devised using a k-Nearest Neighbor (kNN) clustering algorithm. The kNN algorithm was used to find a cluster of substations based on their geographic locations in the Texas 2000-bus model. The central point of each cluster was used as geographic coordinates for the utility control center.

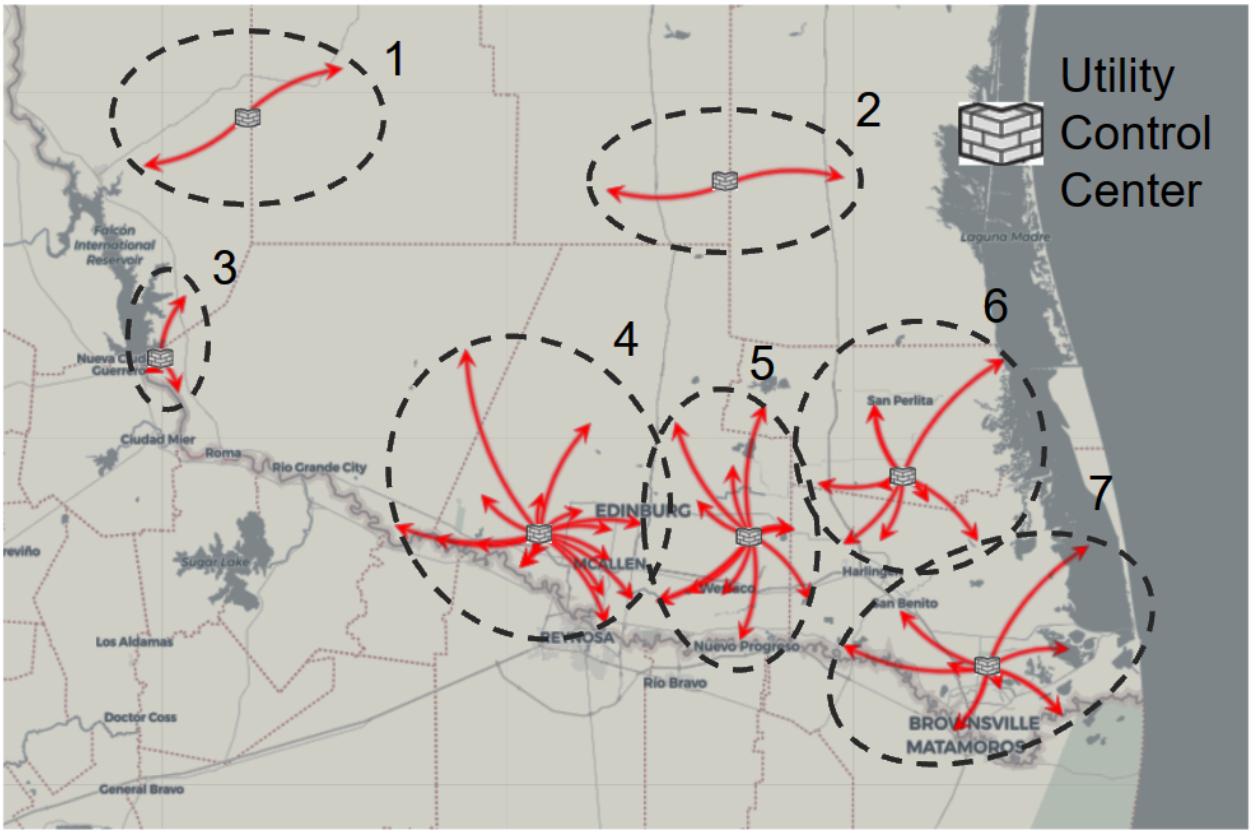

Figure 2.4: A visualization using Splunk's Maps to show how the kNN ML algorithm can cluster substations into utility control centers.

2.3 Validating the Cyber-physical Framework

This framework verified by defining a sample of the Texas 2000 bus network. The McAllen region was chosen as the use case to demonstrate the features of this model. In the McAllen region there are seventy-one substations, seven utilities, and one balancing authority.

Class Type	**Number of Instances**
Firewall	86
Router	157
Switches	142
Relays	166
Total Nodes	**772**
Links	456
Total Links	**456**

Table 2.1: A numerical breakdown of nodes and link classes that were created to a synthetic cyber-physical topology of the McAllen region.

To give a better sense of the scale of the network, between the three class domains, there were 79 JSON files generated. These files contain 86 firewall nodes, 157 routers, 142 switches, 166 load and line relays. There were a total of 772 nodes and 456 links, as shown in Table 2.1. Further details about this framework and the McAllen use case can be found in [2].

3. DESIGN OF THE RESLAB TESTBED

The CyPRES project has been granted the ability to create a cyber-physical power grid testbed here at Texas A&M University, which is known as RESLab. There have been numerous design choices that have gone into creating the testbed in order for it to have the ability to emulate the cyber-physical network topology described in Chapter 2. This allows the CyPRES group to develop real world inspired cyberattacks and custom datasets. RESLab is also about to interchange hardware and software components to help validate experiments from other research groups. The design choices that have been made to devise this thesis are described in this chapter.

3.1 Testbed Architecture

The testbed has many different components which include: the network emulator, the Network Intrusion Detection System (NIDS), and the data visualisation and correlation software [29]. These three major components can be swapped out and replaced for different use cases, if necessary. This allows the CyPRES team to recreate and validate the results of experiments from fellow research universities or national labs. An example of the CyPRES team using RESLab to validate results from an experiment originally conducted by SANDIA can be found in [30]. A simplified model of RESLab testbed architecture is shown in Fig. 3.1, and the three major components are described in the following paragraphs.

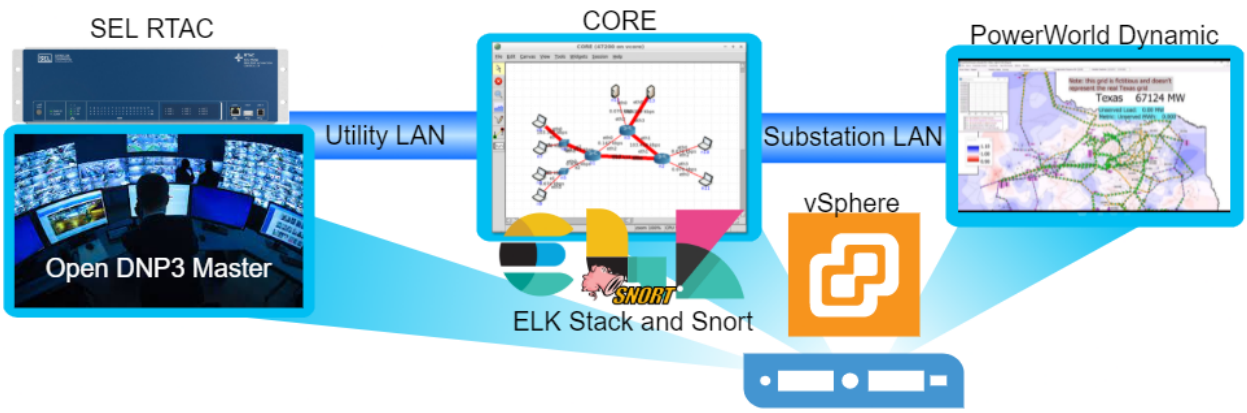

Figure 3.1: The RESLab emulation-based testbed architecture and its interconnections.

3.2 Network Emulator

CORE is used to emulate the communication network because of its ability to integrate real world field equipment by using virtual Ethernet ports. It is also programmed to integrate Linux containers and bridges to form emulated communication networks. The Linux bridges can be configured to have realistic attributes, such as transmission delays to packets and stochastic packet drops and connecting two or more devices together. For this reason, CORE has been programmed to use Linux containers as links and switches.

Like some other real-time network simulators and emulators, CORE allows traffic to flow into and out of physical networking equipment (RTAC, RTU, Physical switches, ect) via vEthernet cards. These vEthernet cards are a virtual version of a real networking card. They allow for packets to be sent to and from the host's physical Ethernet card as if they were a real computer directly connected to one another. Within a CORE session, these vEthernet cards at referred to as RJ45 ports because they allow the user to connect devices that are plugged into the physical Ethernet adapter. The CyPRES project has used these RJ45 ports to integrate HIL, like the *SEL RTAC* as shown in Fig. 3.1. Likewise, the RJ45 ports are used to connect the Open DNP3 Master VM to *PowerWorld Dynamic studio VM* through the CORE's emulated network environment.

CORE uses FreeBSD Jail by default, which is an early form of a Linux container. These

containers have some similarity to a VM but are not the same technology. Both VM and Linux containers can be thought of as a virtual personal computer (PC) that are hosted within a host's operating system (OS). However, traditionally a VM is better suited for use cases that require dedicated computing resources, such as random access memory (RAM), hard disk storage, networking capabilities, graphics rendering, and processing from the central processing unit (CPU) of the host machine. Alternatively, Linux containers are better for use cases that can share resource dynamically with other Linux containers. Due to CORE already being hosted in a VM it was decided that Linux containers would be better for creating an emulated network environment. Additionally, Linux containers allow for any of the applications that are loaded onto the host OS, such as Nmap [31] or ELK stack, to be accessible by all the Linux containers within CORE. Therefore, more Linux containers could be loaded into CORE's emulated environment than VMs.

To help with the scaling of the network, CORE uses gRPC [32]. This protocol is used to coordinate multiple servers together via Python scripts and CORE's frontend application, called CORE-PYGUI. This allows the CORE-daemon to be run on multiple different servers or VMs and distribute the computing resources utilization across a collective group of servers while still having access the same network emulation session. This feature gives CORE the unique ability to scale the number of emulated nodes.

3.3 Network Intrusion Detection System

Snort [33] has been chosen as the NIDS because we can configure custom alerts for ICS protocols, including DNP3 protocol. Snort uses rule- based preprocessors that is able to scan through network traffic for any anomalies in ICS and IT packet. These preprocessors will sniff through traffic and generate alerts on any packets that match the rules that the user programs. It also includes common preprocessors that can be used to sniff ordinary IT traffic, i.e., DNS, HTTPS, SNMP, ARP, among others. In addition, Snort has a community of researchers and developers that can create custom preprocessors for ICS protocols, such as Modbus, Goose, and DNP3.

For the four use cases listed in Section 4, Snort was configured to monitor ARP and DNP3 traffic. The Snort service was only run on the substation router, as shown in Fig. 1.1. Because the

substation router is the default gateway, all traffic leaving the Substation has to travel through it. This makes the substation router an ideal location to perform data correlation. However, it can only see packet information coming from the field devices. If the MiTM has changed the value of the packet and made sure the response from the field device coincides with the original value sent by the outstation, there is no anomalous DNP3 packet from the router's perspective. For this reason, Snort was configured to log and alert any ARP traffic that is destined for the substation router.

In Snort, MAC addresses can be whitelisted. The whitelisting process will ensure that only one MAC address and one IP address are associated. This is accomplished by statically assigning one IP address with a MAC address [33] in Snort. In addition, the average number of ARP packets increases during an ARP MiTM attack and Snort can log each ARP update as an alert. Therefore, we can assume that when more ARP packets are being logged by Snort, that either a device is being added to the network, or a MiTM has started.

We can improve the correlation of the MAC address even further by checking that the cyclical redundancy check (CRC) values are correct using Snort's DNP3 processors. This was useful information that was noticed after we created the DNP3 MiTM scripts. During the experiment it was noted that DNP3 has multiple CRCs inside the same DNP3 packet and one of them can be easily miscalculated by the MiTM attack while trying to process the packets as fast as possible. Therefore, a Snort Rule ensured that the integrity of the packet was not changed by validating all the CRC were correct.

Snort is configured to generate alerts for DNP3 and ARP traffic and save theses alerts to a log file. The contents of the log file are then pipelined into another application for active network monitoring. For ICS networks, it makes sense for operators to have real-time update of these packets streamed to a data visualisation tool. In the RESLab testbed, we use the Elastic Search-Kibana (ELK) stack [34] for data visualization and correlation.

3.4 Data Visualization and Correlation

For data visualization, ELK stack [34] was chosen due to its ability to capture large amounts of packet data (PCAP) data, along with NIDS alerts. In Fig. 3.2 we can see that Snort alerts are be-

ing forwarded to the Elasticsearch database using Logstash, an open source log forwarder used by cybersecurity experts. Along with this data, packet information is being forwarded to the Elasticsearch database using Packetbeats [35]. Packetbeat is another application in the ELK stack that will forward packet capture data (PCAP) to an Elasticsearch database. By housing all of this important networking information in one centralized location, the Elasticsearch database, we can perform data correlation. The data correlation and visualization is accomplished using Kibana, a front-end application for ELK that used Lucid queries to pull information for the Elasticsearch database. In practice, this allows an operator to create custom dashboards and perform data correlation for MiTM attacks and other cyber events occurring within a communication network.

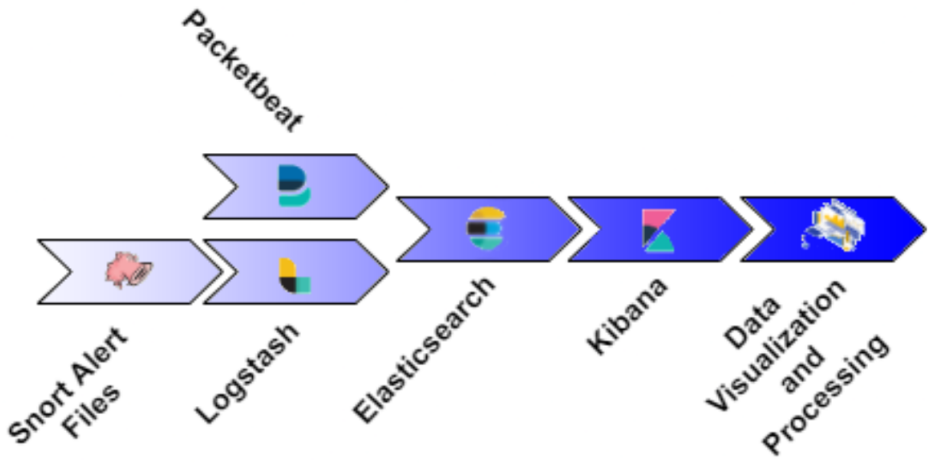

Figure 3.2: Data flow pipeline for ELK stack. From the lowest layer of data collection (left), data storage (middle), and data visualisation (right). Source: Adapted from [3].

4. DESIGN OF MITM ATTACK

4.1 Developing a DNP3 MiTM Script

DNP3 is an ICS protocol used in SCADA systems for remotely monitoring and controlling field devices [36]. The protocol was released in 1993 and originally intended to communicate over RS-485 serial links; it has since been upgraded to an application layer protocol for the TCP/IP stack [5], allowing DNP3 to travel over the internet to remote substations and generation plants.

While the clear-text nature of this protocol was not a major issue when it was communicating over RS-485 links, it has been proven to be a major security concern for TCP/IP-based networks [3]. The primary concern is that a well-funded, nation-state actors could theoretically launch a cyber campaign that would implant a kind of MiTM attack in a power grid. Since DNP3 is the most common ICS protocols used in by power and water utilities in the United States [37], it has been chosen as the primary protocol to study at the RESLab [38] testbed.

An example of a DNP3 packet is shown in Fig. 4.1. Like all TCP/IP based protocols, DNP3 packet is broken down into a header and a payload. The DNP3 header is ten bytes in length, and includes the following fields:

- *synchronization* - Sometimes referred to as a Start Sequence; it is a two-byte field used to show that the rest of the information being received is DNP3 data.
- *frame length* - The total number of bytes in the DNP3 header and payload. The maximum length of the application layer can be 249 bytes [37].
- *Destination Address* - Used to determine what DNP3 device should process the DNP3 data contained in the DNP3 payload.
- *Source Address* - Used to identify which DNP3 device sent the DNP3 application layer data.
- *Control* - An optional field used for link layer confirmation. It can be used for the DNP3 master to confirm the data is being received by the field device.

- *CRC* - Used to ensure the integrity of the DNP3 header fields. There are multiple CRCs in the application layer of the DNP3 packet. They are used to ensure there are no communication errors [37].

The DNP3 data payload is a collection of datapoints from all field devices. Theses datapoints are classified into analog inputs, binary inputs, analog outputs, and binary output datapoints. The binary data is used for field devices that only have two states. In our testbed, these devices are circuit breakers that can only be in an OPEN or CLOSED state. Likewise, the analog datapoints are used to store information for devices that require more than two states. In our testbed these are the generator setpoints. The output datapoints are used to indicate that the message is coming from the DNP3 master and are intended to update the state of field devices. The inputs are status updates from the field devices to the DNP3 master [39].

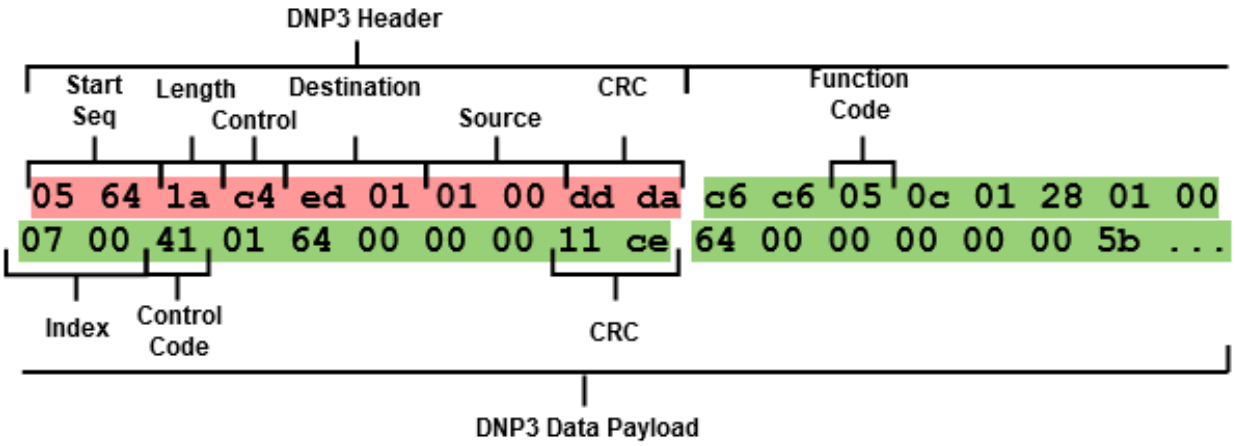

Figure 4.1: An example hexadecimal representation for a DNP3 packet. It shows a detailed breakdown of the DNP3 header and payload. Source: Adapted from [3]

Since DNP3 is a peer-to-peer protocol, it can have multiple network setups. One common network setup is called a multi-drop network. In a multi-drop network one DNP3 master communicates directly with more than one field device. Another common DNP3 network configuration is a one-to-one network. This configuration is characterised by a single DNP3 master communicating

with only one field device. In the RESLab testbed we have chosen a multi-drop network due to the large number of field devices that PowerWorld allows us to simulate.

4.2 Impact of Network Delays to Power System

The authors in [40] discovered that the maximum tolerable delay for a binary TRIP or CLOSE command to be received by a utility control center is between 3 ms to 16 ms. Therefore, the maximum delay for a HMI workstation to receive updates is between 16 ms to 100 ms. This is a considerably short time frame for a MiTM intrusion to modify packet data. For this reason, the MiTM attack script that I implemented had to send and modify packets in the least amount of processing time as possible. If the CyPRES group had used a simulation rather than an emulation, this important detail might have been overlooked.

To further highlight the importance of timely ICS telemetry, the authors in [41] varied the transmission delay of AGC telemetry. By doing so they discovered that when the transmission delay was increased, the overshoot amplitude and delay time of the generators both increased. This in turn caused a temporary increase in the frequency of the transmission grid. Their experiment demonstrated the importance of timely delivery of SCADA packets. This supports the idea that the delay time of ICS telemetry in field devices can also influence the behavior of the power grid.

4.3 MiTM Attack Design Considerations

The MiTM script worked in a similar fashion to Snort, but for the opposite purpose. The script was designed to eavesdrop traffic and modify the values stored in the packet while in transit. It was programmed to do this as fast as possible to keep within the delay threshold previously described.

The sequence of events required to perform the ARP based MiTM attack are shown in Fig. 4.2.

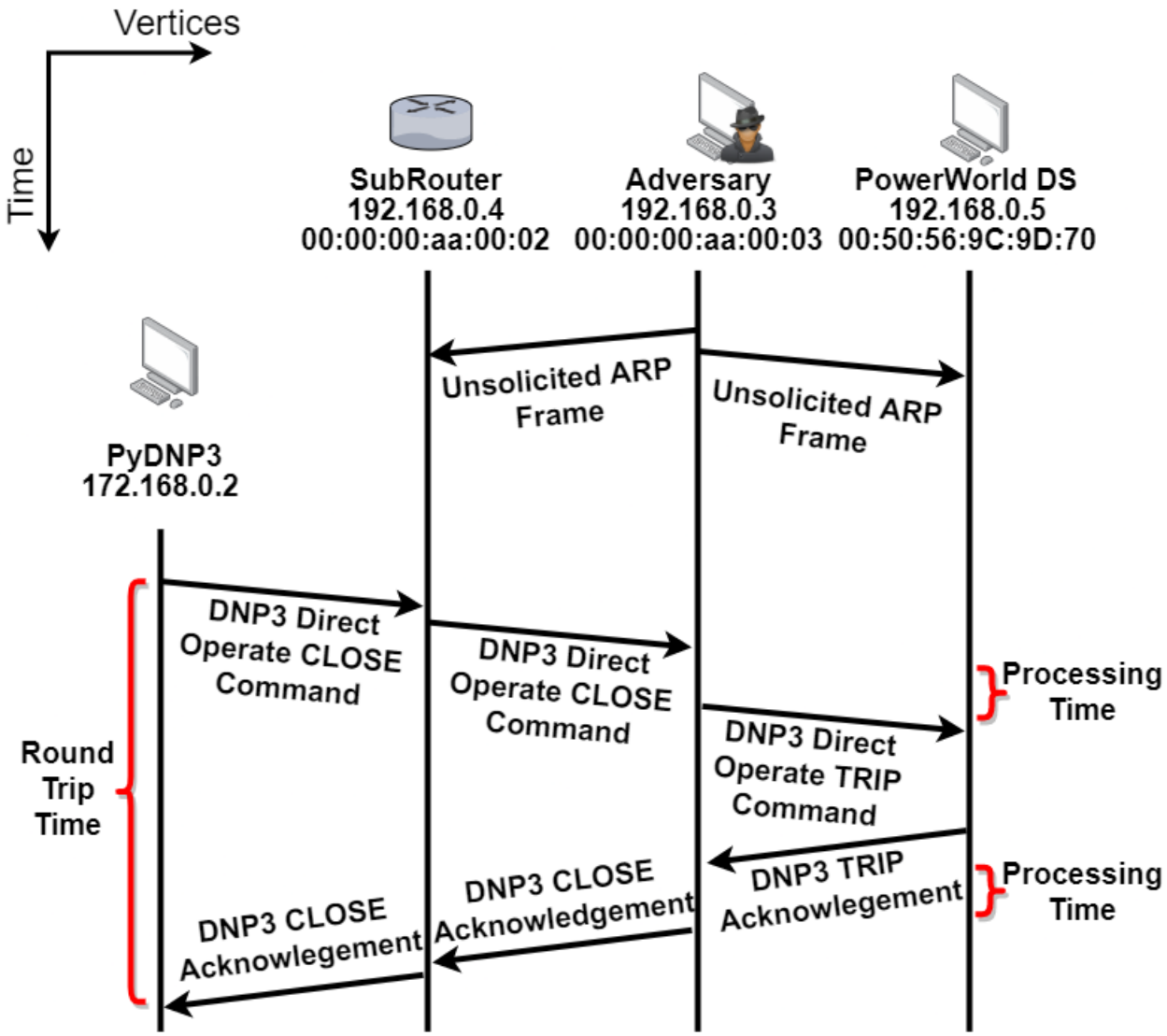

Figure 4.2: A vector and node diagram showing how the DNP3 MiTM attack was programmed to be as undetectable as possible. Soure: Adapted from [3].

First, the adversary must position itself between the gateway and the field devices. Then, the MiTM attack must send an ARP packet directly to the gateway, claiming to be the field device. This will associate the MAC address of the machine running the MiTM attack with the IP address of the field device in the gateway's ARP table. Similarly, an ARP packet is sent to the field device that will force the field devices in Powerworld to associate the MiTM attack machine's MAC address with the gateway's IP address [3].

To keep this route ARP-cache poisoned, the adversary must send ARP packets to the gateway and the field device at a higher rate than either the field device or substation gateway would send ARP packets. The ARP packet is typically sent every 30 seconds from the gateways, but could be as little as every 1 second, and the gateway generally has the shortest ARP update interval. In our emulation, the gateway broadcasts ARP packets every 60 seconds. Since packet information is a discrete signal, we found it appropriate to use the Nyquist frequency, as shown by Equation 4.1, to determine the rate at which to send ARP packets. Moreover, the Nyquist period shown by Equation. 4.2 is the derivation of the Nyquist frequency modified to have variables that represent time periods. Therefore, if the gateway is sending a packet every 30 seconds (T_s), the MiTM attack must send a ARP every at most 15 seconds (T_n) to ensure that the route is continuously ARP-cache poisoned.

The full equation used to determine the time period between ARP packets sent by the MiTM attack is described by Equation 4.3, where the variables are defined as follows:

$$f_n = \frac{1}{2} f_s \tag{4.1}$$

$$T_s = \frac{1}{2} T_n \tag{4.2}$$

$$T_{min} = \min(T_G, T_F) \therefore T_n = \frac{1}{2} * T_{min} \tag{4.3}$$

- T_G - The time period between two ARP packets originating from the gateway.
- T_F - The time period between two packets originating from the field device.
- T_{min} - The smaller of either the T_G or T_F time periods.
- T_n - The time between ARP packets sent from the MiTM attack.

Once the ARP cache poisoning has been put in place, the MiTM can now start to change the header information, and binary input, binary output, analog input, and analog output data to any

value the adversary desires. This is true as long as the attacker does not have a delay time that will cause the TCP timeout clock to expire and cause the packet to be retransmitted by the sender.

For instance, in Fig. 4.2 a binary direct operate packet is being changed while it is making its way from the *PyDNP3 master* VM to the *PowerWorld DS* VM. The packet will start off as the CLOSE packet, which is intended to close a breaker simulated within the PowerWorld DS simulator. As shown in the figure, the MiTM does add some transmission delay because the packet has to be rerouted to a device in the substation network where the MiTM attack is running. In addition, when the MiTM attack changes the value of the packet from a CLOSE to a TRIP (OPEN) command, the CRC has to updated according, which adds a few more milliseconds of delay. If the CRC is not updated, then the packet will be disregarded by the field device. Likewise, the TCP layer has changed because the DNP3 application data has been falsified by the MiTM attack script. Therefore, the TCP layer checksum has to be updated as well in order for the forged data in the packet to be accepted by the field device. The delay time that is added by the DNP3 MiTM is called processing delay.

4.4 MiTM Attack Uses Cases

There are four distinct use cases that were created using the MiTM attack developed in Section 4.3. From each of these four uses cases, a dataset with the PCAP data from the substation router in CORE, the Open DNP3 Master VM, and the PowerWorld Dynamic Studio VM were captured, along with the Snort logs that were generated by during the attack.

A brief description of the four use cases is given in Table 4.1. Use case 1 was focused on modifying the binary command packet. It was designed to invert a DNP3 direct operate command packets value between a TRIP and a CLOSE command. The binary command in this use case was used to modify a breaker's state to either TRIP or CLOSE. This FCI attack is also designed to change the DNP3 response packet to match the original command that was sent by the Open DNP3 master. Therefore, the utility control center operators and substation router are left unaware of the current state of the relay.

Use cases 2 through 4 focus on modifying the analog command data, commands, and response

	Name of Use Case	Description of Use Case
Use Case 1	Branch Control Modification	Change binary direct modification from TRIP to CLOSE. Falsify the DNP3 response packet to the operator.
Use Case 2	Generator Set-Point Modifications	Force a generator's setpoint to 20MW. Falsify the DNP3 response packet to the operators.
Use Case 3	Measurements and Status Modifications	Change multi generator setpoint to 20MW at the same time.
Use Case 4	Status and Response Modifications	Falsify generators setpoint. Falsify the response packet to the operators.

Table 4.1: The four DNP3 MiTM use cases developed by the CyPRES team.

packets. The DNP3 binary control commands are used to remotely set a generator's setpoint for our use cases. For instance, use case 2 is designed to modify a generator's setpoint to a relatively low value that was not 0 MW. This ensured that it was not easily detectable by the operators but still would decrease most generator's supply to the grid substantially, or force generators with a lower setpoint to ramp up their supply of electrical power. Use case 3 modified the status updates from the generator. Forcing the operator or any automated control system to send an incorrect setpoint to the generator. Therefore, the FCI was not made by the adversary but rather the operator that was trying to correct the lower than normal generation supply for that particular generator. Use case 4 was designed to force the value of multiple generators setpoints to 20 MW. The intention of each one of these uses was to cause cascading grid failure. More details about these usecases can be found in [3, 42].

4.5 MiTM Detection

It is challenging to detect MiTM attacks within substation networks, mainly because the attacker impersonates both devices that it communicates with. For this research, a custom DNP3 MiTM attack was based on ARP because the switches determine where to forward traffic to and from field devices in our network setup based on MAC address. Other forms of MiTM, such as a DNS centric MiTM's were considered, but were not used because field devices use statically assigned IP address and not a URL to communicate with control centers. Therefore, a DNS-based MiTM would not work well for targeting field devices.

Packet Type	Processing Time
Bypass	22.775 ms
Analog FCI	27.693 ms
Binary FCI	30.217 ms
Status FDI	35.415 ms

Table 4.2: The average processing times of the MiTM FDI and FCI attack scripts based on type on type of packet being sniffed. Source: Adapted from [3].

The processing times for each of the packets are shown in Table 4.2. The bypass traffic was any packet forwarded between the control center and field device that did contain any DNP3 application layer data. This traffic was mainly TCP traffic that was used to establish the TCP session between the DNP3 master and field device. Therefore, the data inside the bypass traffic was not updated and the processing time was shown through our experiments to have the lowest processing time. Conversely, the DNP3 analog packet that had its command setpoint modified took on average 27.693 ms of processing delay. This is because the analog values have to be adjusted. Also, the DNP3 layer CRCs and the TCP layer checksum have to be recalculated by the MiTM script. Likewise, the binary input took on average 30.217 ms to be modified by the FCI MiTM attack. Finally, the status binary and analog outputs that were sent to the control station were contained in one large DNP3 packet (referred to as the DNP3 status update packet), with multiple CRC checksums that had to be recalculated, taking on average 35.415 ms for the FDI portion of the script to modify all of the targeted datapoints, recalculate each of the DNP3 layers CRCs, and update the TCP header checksum.

This processing delay is one metric that an IDS system could use to detect if there is an MiTM. However, the average time added to the round trip time was between 22.775 ms and 35.415 ms, which means that any network change could would also have a similar effect on the round trip time. For instance, if a routing table is updated or router goes offline, the packets will travel along a different path that will increase the latency of the packet. This suggests that the time delay added by the MiTM attack's processing time should not be the only metric used to detect a DNP3 MiTM attack.

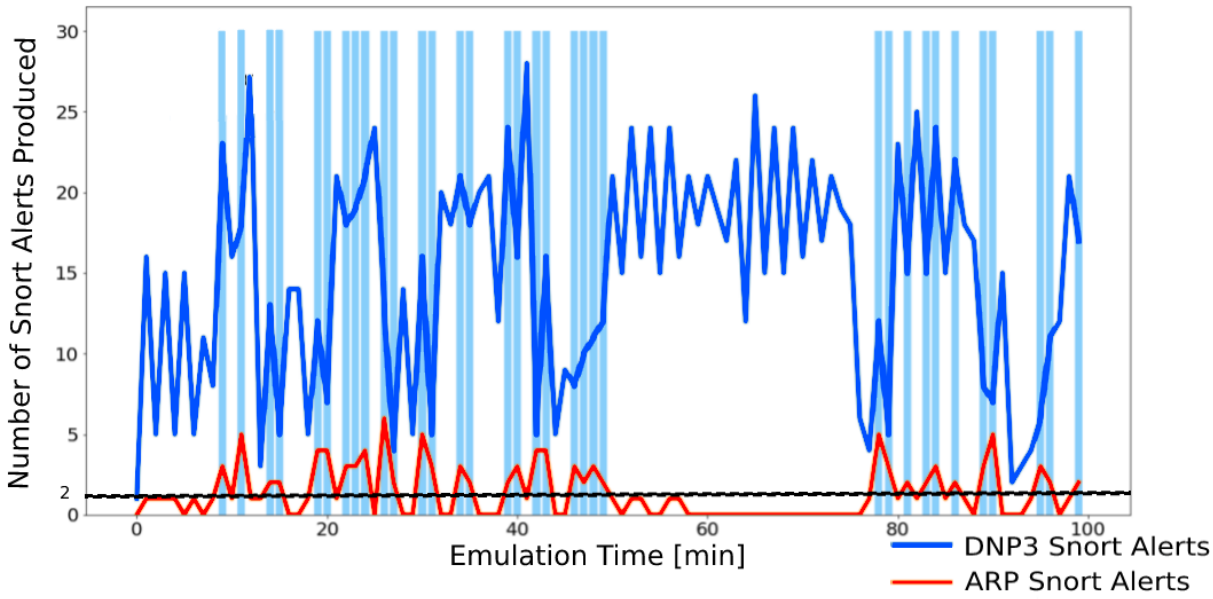

Figure 4.3: A time series graph showing the correlation of the ARP alerts and DNP3 alerts generated by Snort during Use Case 2. Source: Adapted from: [3].

To help detect an ARP-based MiTM attack, we have used ELK stack, a data aggregation application, that can query any information from network devices. With Snort generating alerts for DNP3 packets as the travel in-to and out-of the substation network and PacketBeat capturing ARP packets as they are sent to substation router we can now correlate these two pieces of information to detect when DNP3 packets are being changed by a MiTM attack [3].

Fig. 4.3 shows the correlation between the ARP Snort alerts being generated by the ARP packets sent to the substation gateway and the number of DNP3 packets being exchanged between the DNP3 master and field devices. The black horizontal line shows that average baseline of two ARP packets being sent to the substation router during normal operation. When the number of ARP packets is above this threshold, it indicates that there is an ARP attack happening because higher than normal level of ARP packets are necessary to keep the route ARP cache poisoned. Correlating this information with which DNP3 packets were sent during this time will give an indication of which DNP3 packets were likely modified by an FDI or FCI attack.

Fig. 5.1 shows the communication network that was manually created in order to capture the datasets the four use cases described in Section 4.4. This communication network is a simplified version of the communications network for a power grid. It in has four different LANs directly connected to one another via four routers. At the top of Fig. 5.1 is the *Balancing Authority* (10.3.20.0/24) network which has two devices in it that can be used to send ICCP commands to the control center and substation networks. Likewise, the *Control Center* (172.16.0.0/24) network is connected to the DNP3 master network that has two computer nodes. The *Substation* (192.168.0.0/24) network has two computer nodes. *SubDev1* computer node was compromised in each of the four use cases and used as a platform to launch the DNP3 MiTM attack. In addition, there is an *RTAC* network (172.168.2.0/24). It is an alternate utility control center RTAC network that can be used to control the field devices with real hardware-in-the-loop [29].

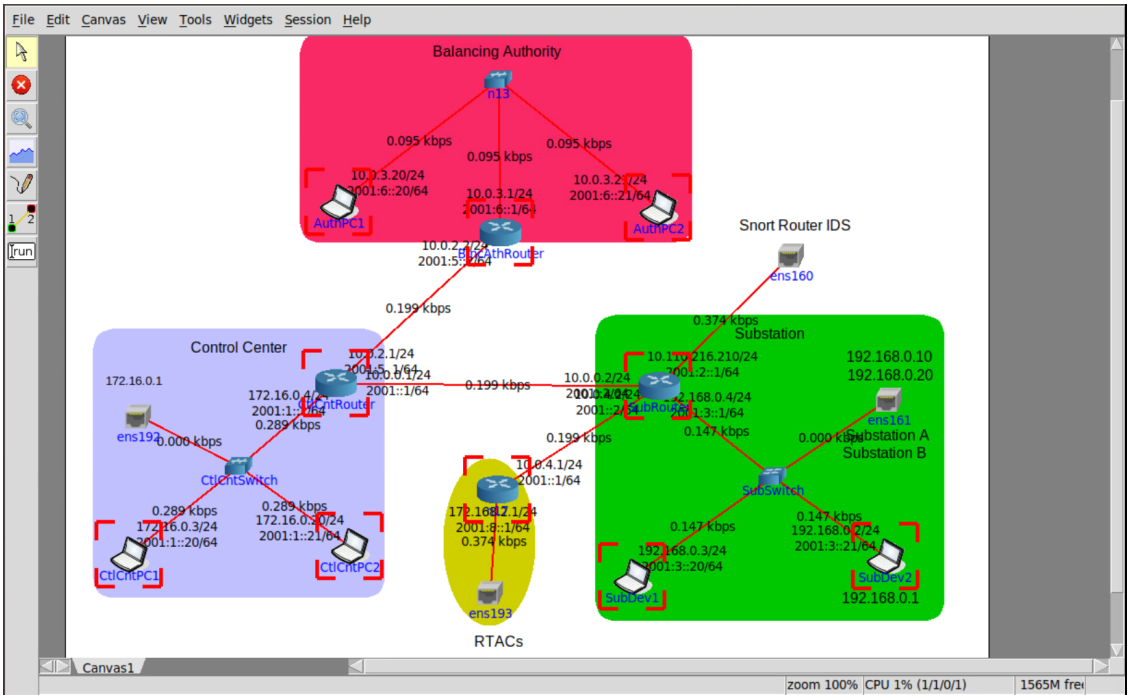

Figure 5.1: The CORE communication network used to capture the DNP3 MiTM datasets. Source: Adapted from [3]

However, Fig. 1.1 is a more realistic model of an actual ICS cyber-physical model. In [1] authors Gaudet *et. al* have recreated the CyPRES projects cyber-physical model in NP-View and established firewall rules for the CPTL model described in Chapter 2. NP-View is a commonly used application in the ICS industry that will conduct network Firewall path analysis, allowing operators to find vulnerabilities in their network. More details for how the cyber-physical model was implemented in NP-View and the firewall rules that were added to the CPTL framework are described in [1].

5.1 Overview of NP-View to CORE Pipeline

With the device configurations already put into NP-View to ensure NERC-CIP compliance, a software pipeline can be created that would automatically generate an emulated version of the communication network using CORE. The process for how this software pipeline is structured is shown in Fig. 5.2. The only update that the NP-View project from [1] required was to put each device into a labeled group. These groups are used to keep track of which networks the nodes are connected to. The labeled groups in NP-View are referred to as regions when exported into a CSV format.

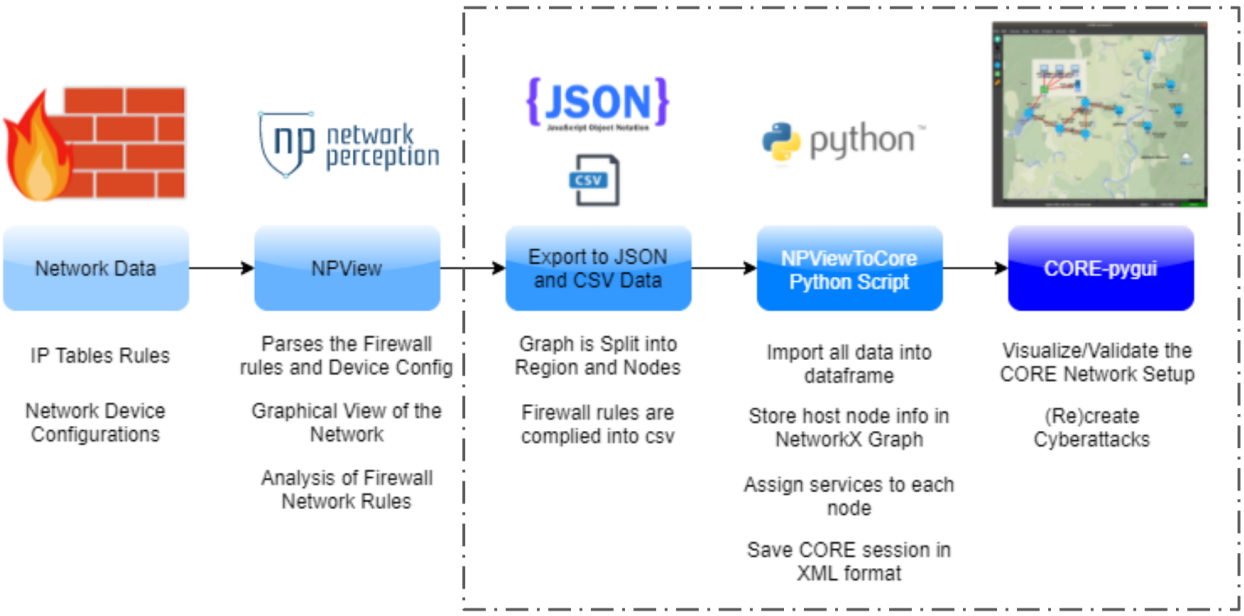

Figure 5.2: NP-View to Common Open Research Emulator (CORE) Pipeline

After the network was segmented into these different regions using NP-view, the NetworkX Region and Nodes plus firewall information could be exported in a CSV format and imported into the NP-View to CORE Python script.

5.2 NP-View Data Export

There are three key files in NP-View that contain most of information necessary to recreate the communication network within CORE network emulation environment. The files are called the Node/Region CSV data, the Host IP Lookup CSV file, and the Firewall CSV File.

The Node/Region CSV data has information about the latitude and longitude for each node within NP-View, along with the region (a.k.a labeled group) information that can be used to determine which nodes are within a LAN or VLAN. The IP Lookup CSV file contains information about what is the IP address for each of the end-nodes in NP-View. However, for routers and firewalls, which can have are multiple interfaces, each interface requires an IP address. The process of how an IP address is found is discussed in Section. 5.3. Fortunately, NP-View contains firewall rules that can be systematically translated into Linux *iptable* rules for emulating the network in CORE. Each one of these exported files are considered raw data for the CORE emulator and needs to be refined in order to create the emulated network.

5.3 NP-View To CORE Python Script

The raw data from NP-View is imported into the NP-View to CORE Python script. It allows for easier data formatting in order to recreate the network in the CORE environment. First, the nodes are loaded into a data frame where the host name given by NP-View can be reformatted so that it does not exceed the 50 characters maximum host name that CORE requires. Then, the data frame is loaded into a NetworkX graph to create a relational mapping of the communication network with nodes and links.

The next step is to add the connections between each node by using the region's information from the Node/Region CSV data. However, the raw data from the NP-View CSV has to be formatted using the algorithm shown in Fig. 5.3. The region information is given in the following format:

set A containers B, C, D and set E contains A, B, C, D, F, G. Giving this information directly to CORE would incorrectly establish a link between node E and B, C, D. In order to avoid this linking issue, the algorithm will check if set A is a subset of set E. If A is a subset of E then simply remove the duplicated elements from set A. This results in set A containing B, C, D and set E containing A, F, G. Therefore, the proper links between a router, a switch, and a PC can be added in the network emulator. The short-hand for the algorithm described above is stated in Equation 5.1

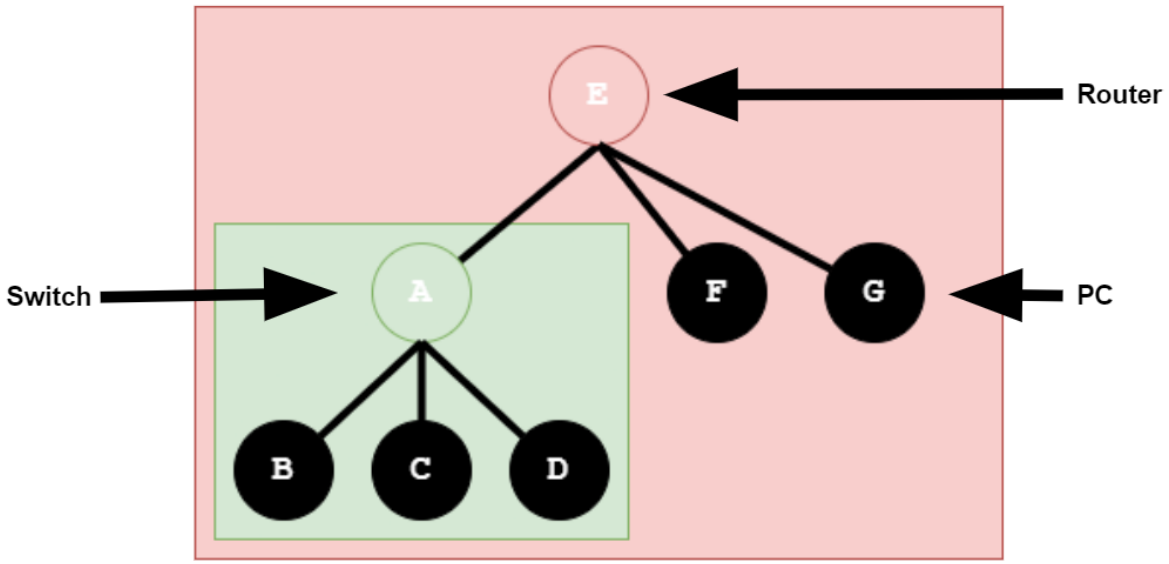

Figure 5.3: Algorithm used to create links between nodes from raw NP-View in the CORE.

$$E_n \supseteq A_n \Rightarrow E_{n+1} = E_n \cup \bar{A}_n \cup \{A\}, A_{n+1} = \bar{E}_n \cup A_n \tag{5.1}$$

With the links applied, then the type of emulated device can be added to CORE. By default, all the devices are labeled as a PC node in the NetworkX graph. When a node is detected to be connected to multiple nodes, it is labeled as a switch. If the name of the node that NP-View has provided contains a keyword, then it is considered a router. The keyword for the work in [1] was "asa". In CORE, the only difference between the router and firewall is whether or not that device

has *iptable* rules to filter traffic. Therefore, if there are firewall rules from the exported file by NP-View, then NetworkX is programmed to label that node as a firewall. Otherwise, if no *iptable* rules exists, the node is tagged as a router.

Then, PC, routers, and firewalls interfaces are checked to make sure their network interfaces have IP address. If an IP address is not found, then an address within the network range is given to that device. This is accomplished by first counting the number nodes that are connected to a switch within NetworkX. Then an creating an inventory of all the IP address that are already assigned to nodes within the LAN. Afterwards, the algorithm shown in Fig. 5.4 is used to determine the smallest possible number of network bits that can be used for all the hosts inside the network.

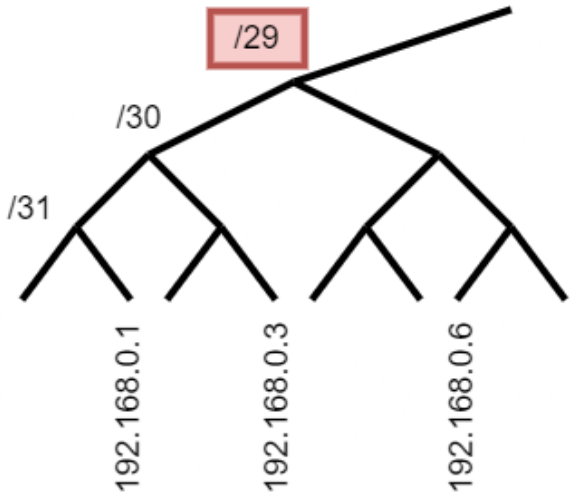

Figure 5.4: Algorithm used to find the missing IP address and largest possible netmask.

First, the algorithm will start with the 192.168.0.1/32 network address and check if all other IP addresses can fall under this network address. If not, then the algorithm will decrease the number of network bits in order to find the smallest host bits that can incorporate all the IP addresses within the list of IPs it is given. When it finds the correct number of host bits, it will conclude that this is the largest possible netmask that can be used for the given LAN. For the example shown in Fig. 5.3, the given inventory of IPs are 192.168.0.1, 192.168.0.3, and 192.38.0.6. From the list of IPs, the

algorithm can find that the largest possible netmask that can incorporate all the IP addresses is /29. Therefore, the 192.168.0.0/29 is the subnet number used for this LAN. From there, any of the missing IP addresses can be assigned to the interfaces that do not have an IP address. In this instance, the list of IP address that can be given to the interfaces that do not have IP addresses would be: 192.168.0.2, 192.168.0.4, and 192.168.5.

The algorithm shown in Fig. 5.4 implies that at least half of the usable IP addresses are assigned to interfaces of nodes in the NetworkX graph. This was not a problem for the NP-View data that was given, and should not be a problem for most LANs. However, if fewer than half of the IP address are assigned to nodes in the region, then the inventory of used IP addresses is removed from all the nodes and a new network address and netmask is given. In this way, it can incorporate all the interfaces within the LAN.

With this information stored in NetworkX, the CORE gRPC API can be used to create routers, switches, and PCs. Depending on how the nodes are labeled, the default services will be allowed in CORE. If the node is labeled as a PC, then it will be assigned a default route. If the node is labeled as a router, then it will be given static routes to forward the packets. If the node is a firewall, then the *iptables* rules and routes will be added to that node. Switches are simply Linux bridges and therefore do not require any services to be added to them because they will route information using MAC addresses, similar to a physical switch.

In addition, CORE allows for custom scripts to be added alongside predefined default services. It was noted that the default route service for the PCs will always default to the network address incremented by one. For example, if the network address is 192.168.0.0/24 then the default route for any PC in that LAN is set to 192.168.0.1. In recreating the network automatically, this was proven to be an issue because the default gateway is not always the network address incremented by one. To solve this issue, the NetworkX graph made the routers broadcast the IP address of the interface to all devices in the LAN that interface is connected to. If the device was labeled as a PC, it would set this IP address as its default route. This is a process similar to how a DHCP server would do for a new host in a LAN. However, since DHCP is not traditionally used in an

ICS network, it made little sense to enable the DHCP service on the routers during the run-time of the CORE session. Regardless, once the NP-View to Python script is started, each of the PCs can communicate with one another because their default route is added as a custom service to each PC in the wide area network (WAN).

5.4 CORE-PYGUI Visualization

The CORE GUI-daemon framework makes it relatively easy to visualize the network topology that is being created by the NP-View to CORE pipeline. Through CORE gRPC application program interface (API), nodes and links can be remotely added to what is called a session. The session will contain all the nodes and links connected as specified by the script, allowing the GUI to be on one server and remotely connected to the server or cluster of servers that is hosting the CORE-daemon server.

Fig. 5.5 shows the process of converting the raw data exported from NP-View into the CORE session. As is shown in Fig. 5.5, the NP-View data is first refined in order to create the NetworkX topology graph.

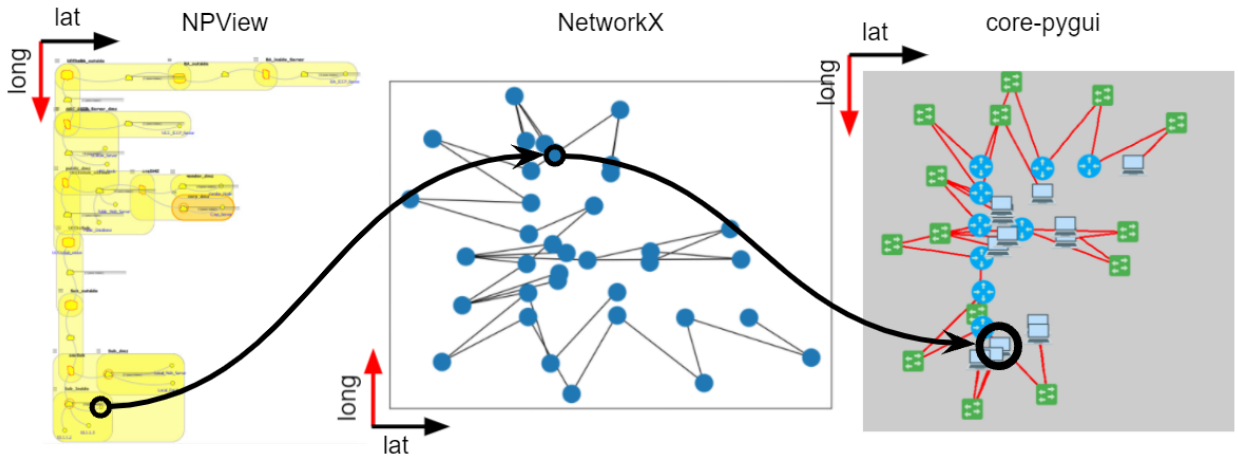

Figure 5.5: A side-by-side comparison showing how the NP-View to CORE Pipeline can systematically recreate a network topology. The full-size version of these three figures can be found in Appendix A

The following attributes are stored for each node in the NetworkX graph:

- *core_type* - The label used to determine what kind of node this should be in CORE. It can be a PC, Firewall, Router, or Switch.
- *core_node_id* - The unique identifier that can be used when referencing the node within the CORE-daemon session. It is used to load the custom scripts onto that node after the node has been initialized in the daemon's session.
- *node_ip* - The list of IP addresses of the nodes that are used to route information through CORE.
- *lat* - The NP-View latitude (or x-coordinate) used for visualizing the network topology.
- *long* - The NP-View longitudinal (or y-coordinate) used for visualizing the network topology.
- *core_node* - A direct reference to the CORE node. This will show all the information that CORE stores about the node, similar to the metadata of packets. This should not be confused with a memory reference address linking directly back to the node in the CORE daemon's session.
- *gateway_ip* - An optional attribute that is only used for PC nodes in order to store the IP address for the default gateway.

As shown by Fig. 5.5, the origin, latitude, and longitude are different between NP-View and NetworkX. Since the raw data is coming from NP-View and is being stored in NetworkX, it was not seen as an issue that NetworkX has a different coordinate system than NP-View. Fortunately, the coordinate system of for the NP-View and CORE-PYGUI graphs are similar. The red and black arrow are being used to show a relative location for where the origin is, along with the direction the latitude and longitude attributes are when the network topology is being graphed.

As shown in Table 5.1, there were a total of 35 nodes emulated in the CORE session. The servers and computing nodes in Fig. 1.1 were made into PC nodes in the emulator. For an emulation

it was unnecessary to distinguish between servers and computing nodes because they are both containers with applications loaded onto them within the CORE framework. Therefore, they are both represented using a PC node. There were four routers and five firewalls with *iptables* loaded onto them. The firewalls, routers, servers, and computing nodes were connected to each other via a switch which held the network address information in the emulation.

CORE Node Type	**Number of Nodes**
PC	13
Switch	14
Router	4
Firewall	5
Total	**36**

Table 5.1: A numerical breakdown of type of nodes inside the CORE session that was based of the CyPRES CPTL model. More information about this model can be found in [2, 1].

5.5 Rebuilding An Actual ICS network

The CyPRES project was able to work with its industry partners and receive data from a real ICS network in order to test the NP-View to CORE pipeline. The data configuration data for 166 nodes inside the ICS network was loaded into NP-View. With NP-View's ability to parse through this configuration data, the NP-View to CORE pipeline was able to automatically recreate this in WAN in the CORE emulated environment.

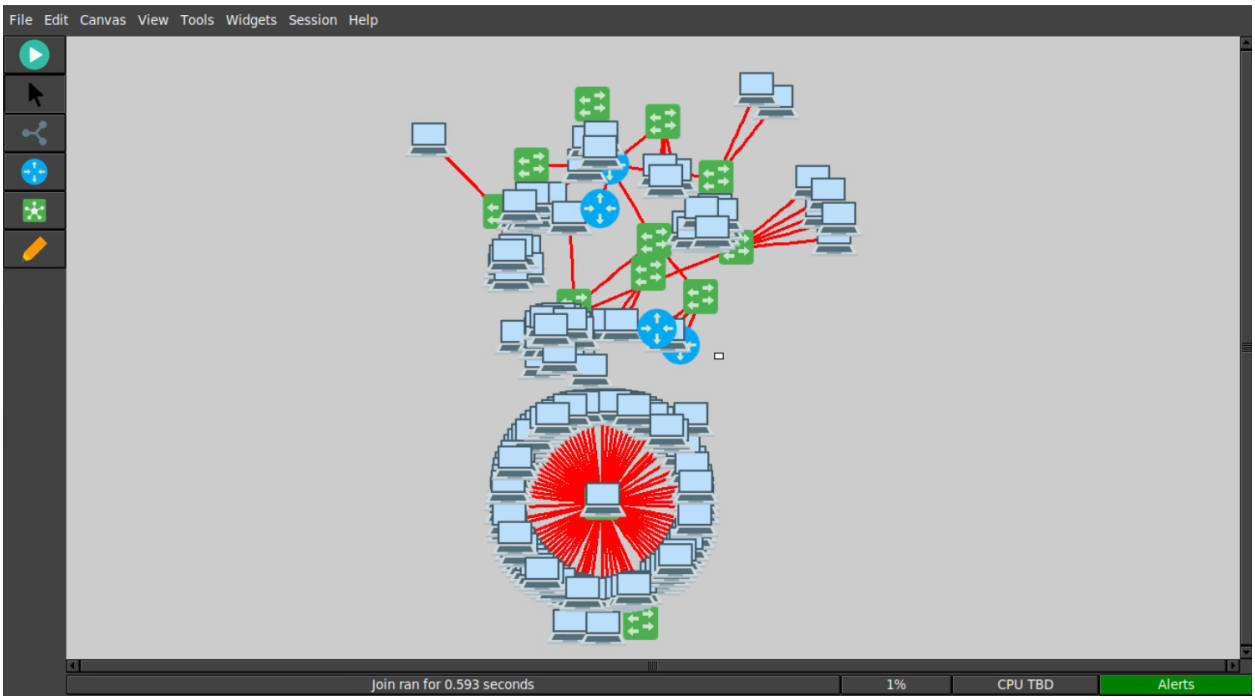

Figure 5.6: Using CORE-PYGUI to visualization the NP-View to CORE pipeline's ability to recreate a real portion of an ICS network in an emulated environment.

The pipeline was able to automatically recreate the 150 PC nodes, 12 switches, four routers, and a firewall, as shown in Table 5.2. It is not known which devices the PCs represent in CORE. This information could be extracted from NP-View through its configuration files. However, this level of realism was intentionally omitted to keep the recreated industry network sanitized but still usable for research.

The custom script for adding default routes to PC was automatically added to each of the PC nodes in the CORE session. This allowed the PC nodes to communicate with one another during the CORE session's runtime. Therefore, the default route did not have to be manually updated on each PC before or during the running session.

Another custom script was added to link PCAP files to the appropriate node, if there were any given traffic data by the industry partners. This could allow for more realistic datasets to be generated by the researchers. However, the packets stored within the PCAP files are limited to the duration of the PCAP capture and the data stored in these packets might be confidential. For

this reason, the type of traffic (TCP or UDP), the average duration between packets being sent, the length of the packet (in bytes), and the source and destination address of the packet was extracted by the NP-View to CORE script and stored in a JSON format. With this information, the custom script on the node was created to generate dummy traffic that would not expose any information about the data that was being sent and could emulate network traffic for longer duration. Thus, larger datasets could be generated with more realistic networking constraints.

CORE Node Type	**Number of Nodes**
PC	150
Switch	12
Router	4
Firewall	1
Total	**167**

Table 5.2: A numerical breakdown of type of nodes inside the CORE session that was based of Industry NP-View data.

6. CONCLUSION

The implications of this work are critical to the power industry since we live in a society that is plagued with an ever-growing list of cyber attacks and vulnerabilities. According to the National Institute of Standards and Technology the electrical generation industry is the target for 57% of the annual cyber attacks for the United States [43].

The benefit of this work over other ICS publications is that it attempts to validate their work in an ICS environment without having to put their solution into a production ICS environment. It can also be used to create more realistic datasets that can train machine learning algorithms that are intended for those to be placed in an actual ICS environment.

At a bare minimum, the validation method that is presented by the NP-View to CORE pipeline can be used by other research groups to validate their own method for detection of cyberatacks, such as FDI and FCI attacks. The NP-View to CORE pipeline will enable researchers to determine what assumptions they made about ICS cyber-physical network that are not accurate, and to correct those assumptions before their proposed solution is put in a production ICS environment. This will help bridge the gap between researchers and utility network operators to work together and patch the holes in today's ICS networks.

APPENDIX A

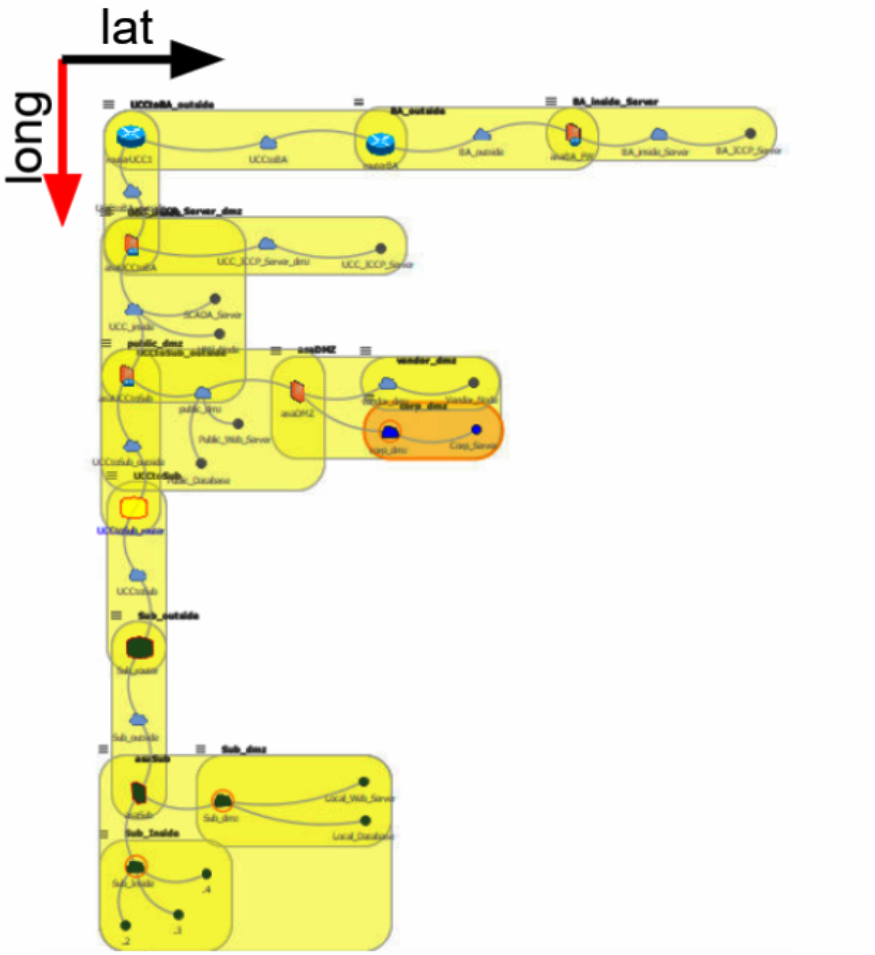

Figure 1: Fully detailed NP-View version of CyPRES CPTL Framework networks topology.

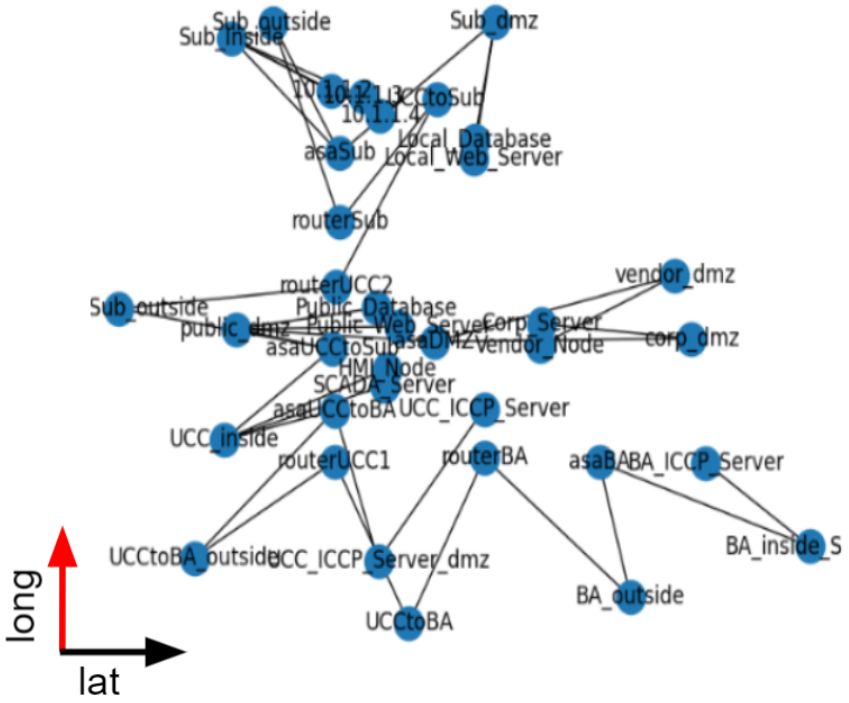

Figure 2: Fully detailed NetworkX version of CyPRES CPTL Framework networks topology.

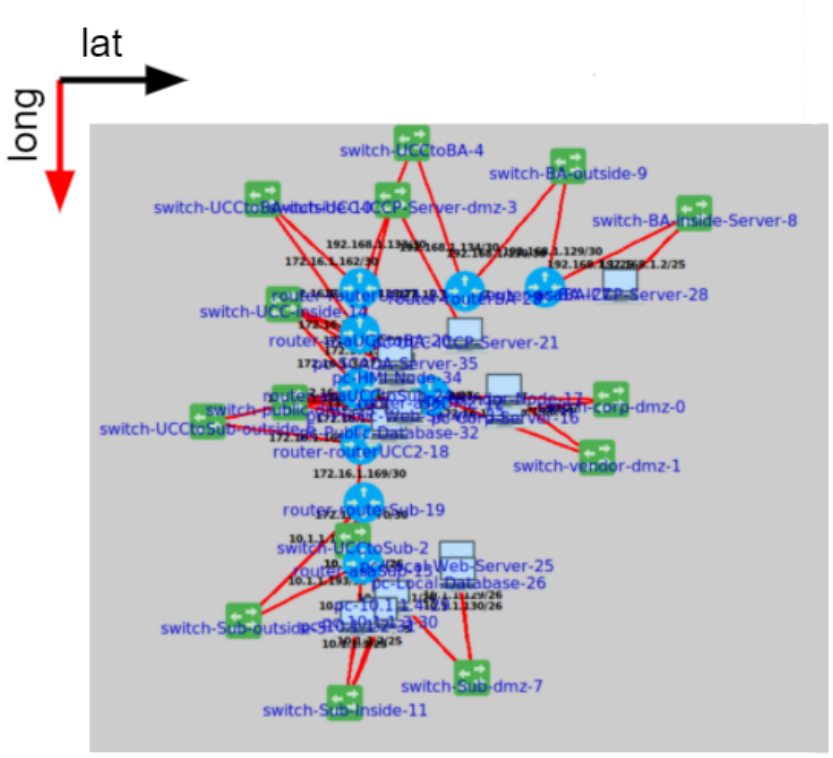

Figure 3: Fully detailed CORE PY-GUI version of CyPRES CPTL Framework networks topology.

REFERENCES

[1] N. Gaudet, A. Sahu, A. E. Goulart, E. Rogers, and K. Davis, "Firewall configuration and path analysis for smartgrid networks," in *2020 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*, pp. 1–6, 2020.

[2] P. Wlazlo, K. Price, C. Veloz, A. Sahu, H. Huang, A. Goulart, K. Davis, and S. Zounouz, "A cyber topology model for the texas 2000 synthetic electric power grid," in *2019 Principles, Systems and Applications of IP Telecommunications (IPTComm)*, pp. 1–8, 2019.

[3] P. Wlazlo, A. Sahu, , Z. Mao, H. Huang, A. Goulart, K. Davis, and S. Zonouz, "Man-in-the-middle attacks and defence in a power system cyber-physical testbed," *IET Cyber-Physical Systems Theory & Applications*, https://doi.org/10.1049/cps2.12014, June 2021.

[4] J. J. Fritz, J. Sagisi, J. James, A. S. Leger, K. King, and K. J. Duncan, "Simulation of man in the middle attack on smart grid testbed," in *2019 SoutheastCon*, pp. 1–6, 2019.

[5] Y. Xu, Y. Yang, T. Li, J. Ju, and Q. Wang, "Review on cyber vulnerabilities of communication protocols in industrial control systems," in *2017 IEEE Conference on Energy Internet and Energy System Integration (EI2)*, pp. 1–6, 2017.

[6] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

[7] R. M. Lee, M. J. Assante, and T. Conway, "Analysis of the Cyber Attack on the Ukrainian Power Grid: Defense Use Case by SANS ICS ." `https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf`.

[8] G. Liang, S. Weller, J. Zhao, F. Luo, and Z. Dong, "The 2015 ukraine blackout: Implications for false data injection attacks," *IEEE Transactions on Power Systems*, vol. PP, pp. 1–1, 11 2016.

[9] E. Targett, "High Voltage Attack: EU's Power Grid Organisation Hit by Hackers." https://dispel.io/blog/how-digital-transformation-reduces-unplanned-downtime-in-the-energy-sec#:~:text=1%20hour%20of%20downtime%20costs,downtime%20extend%20beyond%20monetary%20costs., March 2020.

[10] J. R. Reeder and C. T. Hall, "Cybersecurity's pearl harbor moment: Lessons learned from the colonial pipeline ransomware attack," 2021.

[11] G. Gilchrist, "Secure authentication for dnp3," in *2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century*, pp. 1–3, 2008.

[12] C. Rosborough, c. Gordon, and B. Waldron, "All About Eve: Comparing DNP3 Secure Authentication With Standard Security Technologies for SCADA Communications," in *13th Australasian Information Security Conference*, vol. 161, 2019.

[13] B. Burke, "How Digital Transformation Reduces Uplanned Downtime in the Energy Sector." https://www.cbronline.com/news/eu-power-grid-organisation-hacked, March 2021.

[14] A. Dabrowski, J. Ullrich, and E. R. Weippl, "Grid shock: Coordinated load-changing attacks on power grids: The non-smart power grid is vulnerable to cyber attacks as well," (New York, NY, USA), Association for Computing Machinery, 2017.

[15] Z. Zhang, Y. Mishra, D. Yue, C. Dou, B. Zhang, and Y. C. Tian, "Delay-tolerant predictive power compensation control for photovoltaic voltage regulation," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4545–4554, 2021.

[16] E. Naderi, S. Pazouki, and A. Asrari, "A remedial action scheme against false data injection cyberattacks in smart transmission systems: Application of thyristor controlled series capacitor (tcsc)," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2021.

[17] G. A. Weaver, K. Davis, C. M. Davis, E. J. Rogers, R. B. Bobba, S. Zonouz, R. Berthier, P. W. Sauer, and D. M. Nicol, "Cyber-physical models for power grid security analysis: 8-substation case," in *2016 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pp. 140–146, IEEE, 2016.

[18] M. Adibi, "Power systems test case archive." `http://labs.ece.uw.edu/pstca/pf300/pg_tca300bus.htm`, 1993.

[19] T. J. Overbye, "Powerworld." `www.powerworld.com`, 2021.

[20] A. B. Birchfield, T. Xu, K. M. Gegner, K. S. Shetye, and T. J. Overbye, "Grid structural characteristics as validation criteria for synthetic networks," *IEEE Transactions on Power Systems*, vol. 32, no. 4, pp. 3258–3265, 2017.

[21] S. Pan, T. Morris, and U. Adhikari, "Developing a hybrid intrusion detection system using data mining for power systems," *IEEE Transactions on Smart Grid*, vol. 6, no. 6, pp. 3104–3113, 2015.

[22] G. Ravikumar, B. Hyder, and M. Govindarasu, "Next-generation cps testbed-based grid exercise - synthetic grid, attack, and defense modeling," in *2020 Resilience Week (RWS)*, pp. 92–98, 2020.

[23] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim, "Core: A real-time network emulator," in *MILCOM 2008 - 2008 IEEE Military Communications Conference*, pp. 1–7, 2008.

[24] S. Singh, *Automatic Verification of Security Policy Implementations*. PhD thesis, Univ. of Illinois at Urbana-Champaign, Urbana, 2012.

[25] "Network perception." `https://www.network-perception.com/np-view//`, 2020.

[26] G. A. Weaver, C. Cheh, E. J. Rogers, W. H. Sanders, and D. Gammel, "Toward a cyber-physical topology language: Applications to nerc cip audit," in *Proceedings of the first ACM workshop on Smart energy grid security*, pp. 93–104, ACM, 2013.

[27] F. Abd Elmajid, "Integrating splunk into some of cybersecurity courses," 2021.

[28] T. A. E. E. Station, "Activsg2000: 2000-bus synthetic grid on footprint of texas." https://electricgrids.engr.tamu.edu/electric-grid-test-cases/activsg2000/, ACTIVSg2000: 2000-bus synthetic grid on footprint of Texas.

[29] A. Sahu, P. Wlazlo, Z. Mao, H. Huang, A. Goulart, K. Davis, and S. Zonouz, "Design and evaluation of a cyber-physical testbed for improving attack resilience of power systems," *IET Cyber-Physical Systems Theory & Applications*, https://doi.org/10.1049/cps2.12018, June 2021.

[30] T. Tarman, T. Rollins, L. Swiler, J. Cruz, E. Vugrin, H. Huang, A. Sahu, P. Wlazlo, A. Goulart, and K. Davis, "Comparing reproduced cyber experimentation studies across different emulation testbeds," in *Cyber Security Experimentation and Test Workshop*, pp. 63–71, 2021.

[31] G. F. Lyon, *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure. Com LLC (US), 2008.

[32] X. Wang, H. Zhao, and J. Zhu, "Grpc: A communication cooperation mechanism in distributed systems," *ACM SIGOPS Operating Systems Review*, vol. 27, no. 3, pp. 75–86, 1993.

[33] X. Hou, Z. Jiang, and X. Tian, "The detection and prevention for arp spoofing based on snort," in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, vol. 5, pp. V5–137–V5–139, 2010.

[34] "Elasticsearch, Logstash, Kibana (ELK)." https://www.elastic.co/what-is/elk-stack.

[35] "Packetbeat in elk stack." https://www.elastic.co/beats/packetbeat.

[36] G. Clarke, D. Reynders, and E. Wright, *Practical modern SCADA protocols: DNP3, 60870.5 and related systems*. Newnes, 2004.

[37] K. Curtis, "A dnp3 protocol primer,"

[38] J. Nivethan and M. Papa, "A linux-based firewall for the dnp3 protocol," in *2016 IEEE Symposium on Technologies for Homeland Security (HST)*, pp. 1–5, 2016.

[39] T. Day, "Dnp, distributed network protocol v3 an introduction." https://na.eventscloud.com/file_uploads/b68188f3ce5b22895a67b1afe5e51b6a_DNP3IntroductionHORS.PDF, 2018.

[40] X. Lu, Z. Lu, W. Wang, and J. Ma, "On network performance evaluation toward the smart grid: A case study of dnp3 over tcp/ip," in *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, pp. 1–6, 2011.

[41] Z. Zhang, C. Dou, D. Yue, B. Zhang, S. Xu, T. Hayat, and A. Alsaedi, "An event-triggered secondary control strategy with network delay in islanded microgrids," *IEEE Systems Journal*, vol. 13, no. 2, pp. 1851–1860, 2019.

[42] A. Sahu, Z. Mao, P. Wlazlo, H. Huang, K. Davis, A. Goulart, and S. Zonouz, "Multi-source multi-domain data fusion for cyberattack detection in power systems," *IEEE Access*, vol. 9, pp. 119118–119138, 2021.

[43] K. Zeng and Z. Li, "Best practices in cybersecurity for utilities: Secure remote access," 2020.