Cybersecurity testing and intrusion detection for cyber-physical power systems

By

Shengyi Pan

A Dissertation
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Electrical and Computer Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

December 2014
Copyright by
Shengyi Pan
2014
Cybersecurity testing and intrusion detection for cyber-physical power systems

By

Shengyi Pan

Approved:

____________________________________
Thomas H. Morris
(Major Professor)

____________________________________
David A. Dampier
(Committee Member)

____________________________________
Bryan A. Jones
(Committee Member)

____________________________________
Yong Fu
(Committee Member)

____________________________________
James Fowler
(Graduate Coordinator)

____________________________________
Jason Keith
Interim Dean
Bagley College of Engineering
Power systems will increasingly rely on synchrophasor systems for reliable and high-performance wide area monitoring and control (WAMC). Synchrophasor systems greatly use information communication technologies (ICT) for data exchange which are vulnerable to cyber-attacks. Prior to installation of a synchrophasor system a set of cyber security requirements must be developed and new devices must undergo vulnerability testing to ensure that proper security controls are in place to protect the synchrophasor system from unauthorized access. This dissertation describes vulnerability analysis and testing performed on synchrophasor system components. Two network fuzzing frameworks are proposed; for the IEEE C37.118 protocol and for an energy management system (EMS).

While fixing the identified vulnerabilities in information infrastructures is imperative to secure a power system, it is likely that successful intrusions will still occur. The ability to detect intrusions is necessary to mitigate the negative effects from a successful attacks. The emergence of synchrophasor systems provides real-time data with millisecond precision which makes the observation of a sequence of fast events feasible.
Different power system scenarios present different patterns in the observed fast event sequences. This dissertation proposes a data mining approach called mining common paths to accurately extract patterns for power system scenarios including disturbances, control and protection actions and cyber-attacks from synchrophasor data and logs of system components. In this dissertation, such a pattern is called a common path, which is represented as a sequence of critical system states in temporal order. The process of automatically discovering common paths and building a state machine for detecting power system scenarios and attacks is introduced. The classification results show that the proposed approach can accurately detect these scenarios even with variation in fault locations and load conditions.

This dissertation also describes a hybrid intrusion detection framework that employs the mining common path algorithm to enable a systematic and automatic IDS construction process. An IDS prototype was validated on a 2-line 3-bus power transmission system protected by the distance protection scheme. The result shows the IDS prototype accurately classifies 25 power system scenarios including disturbances, normal control operations, and cyber-attacks.
DEDICATION

To my wife Yang Wang, Dad Xiaoping Pan, Mom Ruizhen Chen.
ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my advisor Dr. Thomas Morris for his continuous guidance and encouragement throughout this work. This work would not have been possible without his support and belief in me. I would like to acknowledge Dr. David Dampier, Dr. Yong Fu, and Dr. Bryan Jones for their advice and support as members of my graduate committee. I would also like to thank Mr. Vahid Madani, Ms. Sharon Xia, the Pacific Gas and Energy Company and Alstom Grid for supporting this work.

I would like to acknowledge Mr. Uttam Adhikari for his help on building the cyber-physical power system test bed, which make feasible the realistic power system data that is used for validating my algorithm. I would also like to thank him for providing me with needed insights of power system theories.

I would like to thank my colleagues and friends, Jonathan Moorhead, Jeremy Lewis and Read Sprabery, who have worked with me and helped me in vulnerability testing study.

Finally, I would like to thank my wife, my parents and my friends who never stopped in believing me and supporting me in difficult times and rejoicing my successes.
## TABLE OF CONTENTS

DEDICATION ........................................................................................................................................... ii  

ACKNOWLEDGEMENTS .................................................................................................................... iii  

LIST OF TABLES ............................................................................................................................... vii  

LIST OF FIGURES ............................................................................................................................. viii  

CHAPTER  

I. INTRODUCTION ..................................................................................................................................... 1  
   1.1 Background ................................................................................................................................. 1  
   1.2 Cyber-physical environment of power system ........................................................................ 3  
   1.3 Cybersecurity challenges in synchrophasor-based power system ........................................... 5  
   1.4 Objectives ................................................................................................................................. 7  
   1.5 Contributions ........................................................................................................................... 10  

II. RELATED WORKS .......................................................................................................................... 13  
   2.1 Current research in power system vulnerability assessment ............................................... 13  
   2.2 Current research in data mining techniques in application to event detection for power system ................................................................................................................................. 14  
   2.3 Current research in intrusion detection system for smart grid .............................................. 17  
      2.3.1 Intrusion detection system (IDS) for smart grid ............................................................... 17  
      2.3.2 Accuracy of specification-based IDS ............................................................................... 20  
      2.3.3 Data mining techniques for learning specifications ....................................................... 21  

III. CYBERSECURITY TESTING FOR SYNCHROPHASOR SYSTEM ........................................ 23  
   3.1 Introduction............................................................................................................................... 23  
   3.2 Synchrophasor system cyber security requirements development ........................................... 25  
   3.3 Synchrophasor system cyber security component testing ....................................................... 37  
      3.3.1 Testing environment configuration ................................................................................. 38  
   3.4 Cybersecurity testing ................................................................................................................ 39  
      3.4.1 Network congestion testing .............................................................................................. 39  
      3.4.2 Protocol mutations ........................................................................................................... 43
3.4.2.1 MU-4000 network analyzer .....................................................44
3.4.2.2 A fuzzing framework for IEEE C37.118 protocol ..............47
   3.4.2.2.1 Smart fuzzing ...............................................................52
   3.4.2.2.2 Dumb fuzzing ...............................................................57
   3.4.2.2.3 Fuzzing algorithms for different IEEE C37.118 frames ....59
      3.4.2.2.3.1 Configuration Frame Fuzzing ................................59
      3.4.2.2.3.2 Data Frame Fuzzing ..............................................59
      3.4.2.2.3.3 Command Frame Fuzzing ......................................59
      3.4.2.2.3.4 Header Frame Fuzzing .........................................60
3.4.3 Energy management system fuzzing framework ..................66
3.5 Developing Snort rules for detecting attacks .......................69
3.6 Conclusion ...................................................................................72

IV. DETECTION FOR FAULT AND CYBER ATTACK IN POWER SYSTEM BY MINING SYNCHROPHASOR DATA ...........................................73

4.1 Introduction .................................................................................73
4.2 Mining common path .................................................................78
   4.2.1 Sequential events for a power system scenario ....................78
   4.2.2 Mining common paths algorithm ........................................81
4.3 Power system test bed .................................................................84
   4.3.1 Power system scenarios .......................................................84
      4.3.1.1 Power System Faults ..................................................84
      4.3.1.2 Trip command injection attack ....................................85
      4.3.1.3 Aurora attack .............................................................85
      4.3.1.4 1LG fault replay attack ..............................................86
   4.3.2 Test bed architecture ............................................................86
   4.3.3 Test Data and Data Preprocessing .......................................88
4.4 Evaluation ...................................................................................91
   4.4.1 Training phase: ....................................................................92
   4.4.2 Test phase: ...........................................................................92
   4.4.3 Experiment 1 .........................................................................93
   4.4.4 Experiment 2 .........................................................................95
   4.4.5 Experiment 3 .........................................................................100
4.5 Conclusions ...............................................................................102

V. DEVELOPING A HYBRID INTRUSION DETECTION SYSTEM USING DATA MINING FOR POWER SYSTEM ........................................104

5.1 Introduction ...............................................................................104
5.2 Mining common paths ...............................................................106
5.3 Test bed architecture .................................................................109
   5.3.1 Distance protection for transmission lines .........................109
   5.3.2 Test bed architecture ...........................................................111
   5.3.3 Test bed scenarios ...............................................................112
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Recommendations and Requirements Derived from NISTIR 7628 [3]</td>
<td>27</td>
</tr>
<tr>
<td>3.2</td>
<td>Recommendations and Requirements Derived from DHS Cyber Security Procurement Language for Control Systems [46]</td>
<td>33</td>
</tr>
<tr>
<td>3.3</td>
<td>Algorithm for configuration frame fuzzing</td>
<td>62</td>
</tr>
<tr>
<td>3.4</td>
<td>Algorithm for data frame fuzzing approach</td>
<td>63</td>
</tr>
<tr>
<td>3.5</td>
<td>Algorithm for command frame fuzzing</td>
<td>64</td>
</tr>
<tr>
<td>3.6</td>
<td>Algorithm for header frame fuzzing</td>
<td>65</td>
</tr>
<tr>
<td>4.1</td>
<td>Example paths for a scenario</td>
<td>82</td>
</tr>
<tr>
<td>4.2</td>
<td>Confusion matrix for experiment 1</td>
<td>94</td>
</tr>
<tr>
<td>4.3</td>
<td>Confusion matrix for experiment 2</td>
<td>96</td>
</tr>
<tr>
<td>4.4</td>
<td>Confusion Matrix for 4 types of faults and 3 cyber-attacks</td>
<td>100</td>
</tr>
<tr>
<td>5.1</td>
<td>Example paths for a scenario</td>
<td>107</td>
</tr>
<tr>
<td>5.2</td>
<td>Confusion matrix for scenarios Q1-Q13</td>
<td>127</td>
</tr>
<tr>
<td>5.3</td>
<td>Confusion matrix for scenarios Q14-Q25</td>
<td>127</td>
</tr>
<tr>
<td>5.4</td>
<td>Confusion matrix for sub-groups in scenario Q1</td>
<td>128</td>
</tr>
<tr>
<td>5.5</td>
<td>Detection accuracy for 4 random zero-day attacks 10x validation</td>
<td>130</td>
</tr>
<tr>
<td>6.1</td>
<td>Fulfillment of seven requirements for the proposed IDS</td>
<td>135</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

1.1 Structure of electric transmission system with integration of synchrophasor technology .................................................................4

3.1 Synchrophasor system component diagram.........................................................37

3.2 Test bed configuration ..................................................................................38

3.3 Denial of service test response time chart.........................................................41

3.4 Availability chart from congestion testing .......................................................42

3.5 PMU and PDC communication pattern ............................................................48

3.6 Structure of the fuzzing framework for IEEE C37.118 protocol .................50

3.7 A general structure for EMS system ..................................................................67

3.8 Structure of fuzzing framework for fuzzing EMS system super server ....68

4.1 Ideal vs. actual SLG fault and protection system response .........................78

4.2 Distribution of timestamps for events ..............................................................80

4.3 3-bus 2-line transmission system for case study .............................................86

4.4 Hardware in the loop test bed ..........................................................................87

4.5 Relay trip time versus fault location for relays R1 and R2 ............................95

4.6 2-D coordinates documenting fault versus command injection attack common paths .................................................................................98

4.7 2-D coordinates comparing two common paths for faults ..............................99

4.8 Accuracy rates for 10-round cross validation with different PMU streaming rates ......................................................................................102

5.1 Distance protection scheme in a 3-bus 2-line transmission system ............109

5.2 Electric transmission system test bed ..............................................................110
5.3 Control flow for automation scenario implementation and data collections
5.4 Intrusion detection system training process
5.5 Dispersion of timestamps for three states that contain three events CP = R1, SNT = R1 and IR1 = 0
5.6 Relay trip time versus fault location for relays R1 and R2
5.7 2-D coordinates documenting fault versus fault replay attack common paths
5.8 2-D coordinates documenting ling maintenance versus command injection attack common paths
CHAPTER I

INTRODUCTION

1.1 Background

The need of electric market regulation and the connection of neighboring electric grids motivate the use of wide area monitoring systems from which utilities have improved visibility of the power grid. Many utilities in the United States of America (USA) received grants from the Department of Energy (DOE) under the American Recovery and Reinvestment Act (ARRA) to create wide area monitoring systems. Wide area monitoring systems are measurement systems that use information communication technology (ICT) to transmit digital and/or analogue data measured by field sensors. The wide area monitoring systems use synchrophasor technology to improve the visualization and situational awareness through high quality measurements of voltage, current, and frequency. The synchrophasor systems require installation of phasor measurement units (PMU), and substation phasor data concentrators (PDC), among other devices and software. PMUs and substation PDCs are networked appliances which use routable network protocols to communicate. They are the key components in the synchrophasor system and may become the target of cybersecurity attacks against bulk electric power systems. Threats against these devices include denial of service attacks, attacks against open ports and services intended to elevate privilege, attempts to change device settings, attempts to inject malicious device commands, attempts to hijack device access
credentials or other confidential information, and attempts to place a man-in-the-middle between devices.

Due to the critical role that the electric power systems play in our society, there is a common agreement among different organizations that the electric power grid needs to be better secured to ensure continuous power being provided to the nation [1]. The ARRA grants required recipient entities to develop a cybersecurity plan which includes a risk assessment as part of parent wide area monitoring systems projects. Also, the synchrophasor devices, i.e. PMUs and PDCs, may be declared North American Electric Reliability Corporation (NERC) Critical Infrastructure Protection (CIP) standard 002-3 [2] critical cyber assets (CCA), depending upon each individual unit’s application within the electric power system. CCA must be housed within an electronic security perimeter and undergo a cyber-vulnerability assessment. The National Institute of Standards and Technology Interagency Report (NISTIR) 7628 also documents the guidelines and requirements for industry to better secure their facilities [3].

While there is significant research in the vulnerability assessment of cybersecurity for traditional Supervisory Control and Data Acquisition (SCADA) systems, there are comparatively few publicly known vulnerabilities for synchrophasor-based monitoring and control systems. In addition, the tight integration of information communication technology and the physical process poses new challenges to the synchrophasor-based electric power system. For example, it has been shown that data delay and loss from a communication system can cause serious interruption in control application of the electric power system [15]. The United States Government Accountability Office (GAO) has realized that current security guidelines from different organizations are not sufficient to
securely implement the future electricity grid that employs synchrophasor technology and it calls for research and development to improve current security mechanisms [4].

1.2 Cyber-physical environment of power system

A typical power system is divided into four functional parts: generation, transmission, distribution, and consumers. The electric transmission system is the backbone of the power system transmitting the electric power from generators to the load centers over a long distance. The structure of a cyber-physical environment for the electric transmission system augmented with synchrophasor technology is shown in Figure 1.1. The transmission system devices are mainly composed of transmission lines, breakers, and transformers that are monitored by field sensors. In the case of a synchrophasor system these field sensors are PMUs. The PMUs attached to transmission lines provide synchronized data that is time-stamped using Coordinated Universal Time (UTC) for continuous real-time monitoring. Phasor Data Concentrators (PDCs) collect synchrophasor measurements from PMUs that are located in different locations and send the measurements to the control center through the wide-area network (WAN). PMUs in different locations and PDCs are key components in the synchrophasor based wide area monitoring system (WAMS). Compared to the traditional SCADA system where the field sensors measure the system once per several seconds, the emergence of WAMS leveraging synchrophasor technology allows much faster measuring for the transmission system at the rates ranging from 30 samples per second to 120 samples per second [8]. Nowadays, synchrophasor measurements are not the only time-synchronized data in a system shown in Figure 1.1. As more devices and power system components such as relays, breakers etc. are integrated with the capability to synchronize to UTC, the status and measurements from
these devices are also time stamped and can be sent back to control center in real time [11].

The redundant information contributed by the time-synchronized data provides benefits for reliability, efficiency, and economics in power system monitoring and control. The extreme low latency offered by time-synchronized data allows various real-time wide area control algorithms and special protection schemes to be used to increase power grid reliability and stability [9][11][12][13][14]. The information flow described above is shown as the dotted line in Figure 1.1 and is often recognized as a control loop. In the case of a distributed control, the protection components in the system sense the disturbance and react to it by themselves. The bi-directed-arrow lines in between control components and WAN indicates not only the command data sent from control center but also the time-synchronized audit information reported from intelligent electronic devices (IDEs) to the control center.

![Figure 1.1 Structure of electric transmission system with integration of synchrophasor technology](image)

Figure 1.1 Structure of electric transmission system with integration of synchrophasor technology
The system can be considered as a finite state machine. If, for example, a tripping operation is sent from control center, this will cause system state transitions because a signal-sending operation has been recorded in the control panel which is one component of the system. In general, the changes of the behaviors in different system components such as a breaker, relay, and transmission lines in a given period of time or at a definite point of time will cause the system state to transition from one state to another. These changes are reflected by the transmission line sensor readings or device logs. If the system state is represented as a set of observations (from logs of different components) and measurement data (from measurement devices) inside the system, such changes along with time can be regarded as temporal state transitions.

1.3 Cybersecurity challenges in synchrophasor-based power system

The electric power system in the past was often isolated and used proprietary devices and software. However, the synchrophasor system greatly relies on the commercial off-the-shelf (COTS) components e.g. hardware (e.g. Personal Computers (PCs), network appliances, database servers), Windows Operating System, and standardized IP-based industrial protocols such as IEC 61850 and IEEE C37.118. The commercial hardware and software are usually the popular targets of cyber-attacks. There are a large number of exploits available in an exploit frameworks such as Metasploit [64]. It is proved that power systems are vulnerable to cyber attacks [80]. Most industrial protocols use open standards without security features. In addition, there are still a large number of legacy devices in the field that do not have security control mechanisms at all. As such, the power system may be subjected to cyber-attacks. Potential catastrophic consequences have been learned from Aurora [65] and Stuxnet [7] attacks where
attackers penetrated the network, gained access to the control software and altered system states to destabilize the control systems. As a key component in a cyber-physical power system, research should be conducted to determine the adequacy of cybersecurity efforts in synchrophasor system.

Developing robust cyber infrastructure requires intrusion prevention. Intrusion prevention techniques developed from penetration testing can fix the security breaches and effectively prevent known attacks. However, failures in intrusion prevention are still likely to occur, which can result in a compromise in the cyber infrastructure. A failure in intrusion prevention may be exploited by, for example, a zero-day attack that is unknown to the intrusion prevention mechanism. While successful attacks are always possible, intrusion detection as a means for defense in depth is necessary to be deployed to detect them. Intrusion detection system is helpful in reducing the negative impact in that fast responses to stop the attack can be taken as soon as the attack is identified. Traditional signature-based intrusion detection systems e.g. Snort are helpful to detect malicious activities when a malicious network packet is found. However, due to the increasing interaction between cyber infrastructure and the physical infrastructure created by the emergence of high-speed networks in electric power system attacks against future power systems will not be limited to those of the traditional IT system. Therefore traditional intrusion detection systems may not be enough to protect the future power grid from cyber-attacks that aim to interrupt physical processes. An example of such attacks is resonance attacks in which an attacker who has compromised system sensors or controllers causes the physical system to oscillate at its resonant frequency [5]. Another
example is demonstrated in [6] where an attack can inject false data to compromise meters to bypass the existing bad data detection algorithms.

1.4 Objectives

Over 1,000 PMU have been installed across North America, and many local and regional phasor data concentrators collect real-time, high-speed, time-synchronized information about power grid conditions to enhance grid operations and protect grid reliability. This information is also shared between transmission and power plant owners and grid operators to improve wide-area visualization of power grid states across large regions and to enhance situational awareness. Such information also enables advanced monitoring and control algorithms and new types of intrusion detection system that leverage the system state information to detect malicious system behaviors.

Despite the benefits that synchrophasor systems bring to the power grid, they exist threats of cyber-attacks as synchrophasor devices are highly interconnected using information and communications technology (ICT). Synchrophasor devices are becoming more attractive to attackers. First they usually directly interact with the physical power system as they are installed in substations and measure transmission line parameters including current, voltage, and frequency. Synchrophasor device unavailability will cause the grid operators and plant owners to lose the view of power system conditions. Second, as more synchrophasor devices are integrated with protection relays, compromise of these devices may also result in attackers controlling breakers, which can lead to blackouts.

One objective of this work is to develop a methodology to identify vulnerabilities associated with synchrophasor devices. This methodology will also identify gaps between available security features of synchrophasor devices and the cyber security standards or
requirements. This work demonstrates that a testing process and tools can be developed to conduct the vulnerability assessment for synchrophasor devices and protocols. One significant problem related to vulnerability testing for synchrophasor devices is the lack of protocol fuzzing tools available for mutating the IEEE C37.118 protocol. Also, current commercial protocol mutation tools are limited to mutating server to client commands and cannot mutate server to client responses. This makes commercial tools unsuitable to test the IEEE C37.118 protocol. This work demonstrates a network fuzzing framework developed to mutate IEEE C37.118 protocol packets and useful protocol mutation of other protocols used by synchrophasor devices.

While identifying and closing synchrophasor system cyber vulnerabilities is imperative, successful intrusions can still occur. Time-stamped data synchrophasor measurements and device log provide cyber security researchers a new way to enable intrusion detection. This leads to a research problem of developing unique signatures for power system scenarios (i.e. disturbances) and cyber-attacks from the large amounts of data available in a synchrophasor system. Due to power system and measurement system dynamics, events present in the data have timestamp variation which makes mining patterns difficult. A number of publications (See Chapter II) discuss using machine learning methods to learn patterns for power system disturbances or cyber attacks but none of the methods described in literature can be applied to mine patterns of both types. Traditional data mining algorithms are designed to work on a small amount of data and cannot easily be used to classify specific power system scenarios and cyber attacks [28]. Therefore, in this dissertation we describe a new data mining method called mining common paths which learns unique signatures for different power system scenarios and
cyber attacks in terms of common paths from massive heterogeneous data collected from a power system. We prove the correctness and usability of common paths by creating patterns and classifying different power system disturbances and cyber attacks on one transmission line.

Various research has been conducted to create intrusion detection systems (IDS) for smart grid, however, proposed methods suffer from different shortcomings. For example, host-based IDS is only able to monitor one location of the system. Network-based IDS only focuses on network activities. Rule and specification-based IDS suffer from limited scalability due to labor-intensive IDS creation and update procedures. One objective of this dissertation is to create a new type of IDS that overcomes able to monitor a fusion of data from heterogeneous power system sensors to classify actions or scenarios which have recently occurred. The mining common paths algorithm was applied to train an intrusion detection system for a power system of 3 buses and 2 transmission lines. The IDS was capable of classifying 25 separate disturbance, control, and cyber-attack scenarios. This work proves that an IDS trained using the mining common paths algorithm can monitor a power system implementing the distance protection scheme. This work also shows that the mining common paths algorithm can automatically learn patterns for a variety of cyber attacks, power system disturbances and valid control actions from huge amount of data with little human interaction. Finally, this work demonstrates the intrusion detection system is able to accurately classify each scenario type with relatively small training and classification time and small memory usage.
1.5 Contributions

This dissertation makes three primary contributions to industry and academia. First, this dissertation presents a vulnerability testing process for PMU, PDC, and energy management systems (EMS) which was successfully used to identify vulnerabilities in a commercial synchrophasor system. The vulnerability testing includes a new fuzzing framework for the IEEE C37.118 protocol and the EMS. The testing process has been performed on PMU, PDC, and EMS from a major commercial hardware provider and a major power system software provider. Various vulnerabilities have been identified and reported to utilities and vendors. Suggestions on how to mitigate the security risk and to improve the security features of synchrophasor devices were also provided based on discovered vulnerabilities.

Second, this dissertation documents a data mining approach called mining common path algorithm which accurately mines patterns from power system scenarios including disturbances, valid control actions, and cyber-attacks from synchrophasor data and logs of system components. These patterns, also known as common paths, represent system behaviors unique to each scenario and can be used to classify each scenario. The data mining algorithm is based on the mining sequential patterns algorithm which was found in the human health diagnosis research domain to learn patient’s physiological states. One contribution of this work is that we applied this data mining method to mine patterns for power system scenarios and cyber attacks. However, this algorithm requires that massive data to be preprocessed into a specific form called paths. To overcome this obstacle, another contribution of this work was development of a method to preprocess synchrophasor and power system state data into a set of paths usable by the algorithm.
Paths retain all events which have occurred in the system in a compressed form. As such the data can be processed by the mining common paths algorithm. Compared to other traditional data mining algorithms in [28], the classifier trained using mining common paths algorithm is able to provide more precise classification which enables automated controllers such as autonomic control frameworks in [66].

Finally, this dissertation also developed a hybrid intrusion detection system, trained using the mining common paths algorithm, to detect a variety of power system scenarios and cyber attacks in a large power system. The capability of the resulting IDS to classify the specific power scenarios and cyber attacks advances IDS state-of-art. The resulting IDS also complements current IDS techniques for Smart Grid by satisfying the following requirements.

Requirement 1, the IDS should perform stateful monitoring at the system level. This means the IDS should be able to provide monitoring at different locations of the system to be able to monitor ordered sequences of execution events. Requirement 2, the IDS must be able to monitor actions of an automated control algorithm. It should be able to distinguish actions which originate from a system operator or automated control algorithm from similar actions which originate from an attacker where the primary distinguishing feature is the state of the system when the action occurs. Namely, we expect the new IDS to be able to detect not only cyber-attacks, but also power system disturbances and normal operations. For this work, the IDS must be able to monitor a system implementing the distance protection scheme for normal and attack behaviors. Requirement 3, the IDS must be able to detect zero day attacks, i.e. attacks that are unknown to the IDS. Requirement 4, the IDS should be able to process high volumes of
data from multiple sensors. Requirement 5, the IDS must be low cost. Cost includes
development time and compute resources required to implement the IDS. This
requirement also includes that the IDS should be able to extend to new scenarios as they
are detected. Requirement 6, the IDS must be scalable to larger and more complex control
algorithms. This requirement is intended to address the dynamic nature of the power
system. The IDS must be able to continue to correctly classify behaviors as system
configuration and load change. The IDS must be able to detect events that occur in
random order. Requirement 7, the IDS must have high accuracy and minimal false
positives.
CHAPTER II
RELATED WORKS

2.1 Current research in power system vulnerability assessment

The National SCADA Test Bed Program run by the Idaho National Lab (INL) built a large scale SCADA test bed for the purpose of assessing control system cybersecurity, improving and extending the cybersecurity standards as well as training. Various common vulnerabilities associated with SCADA systems are reported in [9] from the program. Findings and learned lessons are summarized in [16] from security assessment of control systems. The program also develops recommended procurement language to enhance and improve the security in wireless systems for advanced metering infrastructure [17]. In addition, INL is engaged in cyber security standard improvement, training and cyber security assessment of software and hardware products for power system industry. However, no result is reported from INL regarding to cyber security assessment for synchrophasor system.

Researchers have performed vulnerability assessments of generation and substation devices to support development of taxonomies of vulnerabilities related to industrial control systems. In [18] Fovino et al. describe a test bed used for vulnerability assessment of components found in a Turbo-Gas Power Plant.

In [19] Skaggs et al. describe a tool, NETGLEAN, testing device for network vulnerabilities. Two well-known tools are available for network vulnerability testing of
industrial control systems. Wurldtech [20] offers the Achilles Satellite product for testing industrial control system devices. MU Dynamics [21] offers the MU Test Suite for testing networked devices, include industrial control system devices. Both products include protocol mutation and denial of service test suites.

2.2 Current research in data mining techniques in application to event detection for power system

Compared to peer works, this work is unique in that we propose a data mining algorithm that can learn patterns for both power system disturbances and cyber-attacks from heterogeneous data including synchrophasor measurements and device logs from multiple locations in the power system. Multiple traditional data mining algorithms were used to classify power system faults and cyber-attacks in [28]. The traditional data mining algorithms were able to differentiate between three broad categories with approximately 70% accuracy, power system disturbance, control action, and cyber-attack, however, the traditional data mining algorithms were not able to classify specific fault and cyber-attack types within each large category.

Current research on applying data mining to synchrophasor data for power system fault and disturbance classification can be found in [68] and [69]. These works limit algorithm input to synchrophasor measurements and do not provide cyber-attack detection. The K-nearest neighbor algorithm was used to classify three phase faults (3LG), voltage oscillation, and voltage sag scenarios in [68]. The algorithm accuracy is not provided in [68]. Stream data mining is used in [69]. This approach was able to classify 3LG and single line to ground (1LG) faults grouped for binary classification with greater than 90% accuracy. Both [68] and [69] used simulated power system data.
Many other data mining approaches have been developed to extract signatures and classify power system disturbances but they have no ability to detect cyber-attacks. Many such approaches classify power system disturbances in the time domain. Decision trees were used to classify power system disturbances in [22] and [23]. Statistical characteristics of power system frequency were used in [24] to represent the signatures of power system disturbances. Frequency domain analysis has been proposed to avoid complex transient phenomena in time-domain waveforms. The frequency domain approaches first convert time-domain waveforms to the frequency domain using wavelet or/and Fourier transforms and then extract oscillation related features. Both Support Vector Machines (SVM) [25] and artificial neural networks (ANN) [26] have been used to classify of disturbances in the frequency domain. Many works have applied neural networks to classify faults. Using current and voltage data as input, patterns for faults can be learned by radial basis function neural networks [72], Kohonen neural networks [74] and self-organizing map-based neural networks [73]. In [75] the authors used a neural network with current waveforms and data from digital fault recorders to classify faults, normal maintenance operations, and power-quality disturbances. The work presented in this work uses a sequential data mining approach to classify patterns from sequences of events. Sequential data mining is better suited for high velocity and high volume synchrophasor data streams because synchrophasor data is discrete data but continuous in time. Additionally, the mining common paths algorithm presented in this work can learn to classify traditional power system contingencies, such as faults, and cyber-attacks against power systems which masquerade as traditional contingencies.
Machine learning approaches have also been applied to detect cyber-attacks against power systems. In [27], detection rules were derived by manually specifying allowable ranges for different system measurements using domain expert knowledge. Such specification based methods have been shown to have high detection accuracy; however, the manual effort required to develop such a decision tree is too great to apply to a problem on the scale of power system protection. Traditional machine learning algorithms were applied to a dataset which captured power system disturbances, control actions, and cyber-attacks [28]. The machine learning algorithms were able to successfully differentiate between grouped power system disturbances, control actions, and cyber-attacks, but, were not able to classify specific events in each larger category. Additionally, the machine learning algorithms make a classification decision for each row of the data set, which, was sampled at a rate of 120 measurements per second. Providing a classification at each row of the dataset rate amplifies the total number of false positives. The IDS presented in this dissertation provides a classification at once per detected scenario which results in one classification per several thousand rows of the dataset minimizing the volume of false positives.

Other works have been found which provide intrusion detection for synchrophasor systems. An IDS was proposed which uses white lists to detect invalid network behaviors based on a synchrophasor network protocol specification [29]. A second proposed IDS uses timing and data volume information to identify data integrity attacks against synchrophasor systems [30]. To the best of authors’ knowledge no research has been published which detects both power system disturbances and cyber-attacks at the same time.
The data mining technique used in this dissertation uses “mining sequential patterns” technique which discovers patterns of activity sequences from time ordered data. The “mining sequential patterns” algorithm was first mentioned in [31]. It was used to discover patterns in clinical client care management process data that consists of patient records and log data over a period of treatment time [32]. This technique was extended in [33] by employing a two dimension Bayesian network to graphically represent patterns in Hemodialysis processes which consist of a sequence of medical activities over time. In order to discover patterns, a patients’ physiological “state” is defined using clinical log data and patient records (e.g. body temperature). The pattern is therefore, represented as contiguous transitions of states in a two dimension graph. The classification was made using the patterns.

For this work, the FP-growth algorithm is used to mine frequent sequential patterns. FP-Growth reduces the cost of searching for frequent sequences by adopting a divide-and-conquer strategy [34]. As demonstrated in [70], FP-graph algorithm outperforms several popular frequent pattern mining algorithms in run time and therefore it was chosen for this work.

2.3 Current research in intrusion detection system for smart grid

2.3.1 Intrusion detection system (IDS) for smart grid

In recent years, the emergence of the smart grid has motivated research into a variety of IDS techniques. People with different backgrounds have created various IDS that focus on different aspects of the smart grid. One type of IDS research focuses on intelligent electronic device (IED) security within the smart grid. For example, Chee-Wooi Ten et al. developed an anomaly-based detection technique to detect attacks against
IED [35]. The Chee-Wooi Ten IDS is host-based thus only identifies attacks against a single IED in the substation using sequential events recorded in the log from that IED. Another IDS proposed by Chen et al. provides a protection mechanism for smart household appliances [36]. Chen et al. created security rules for individual appliances by proposing homogeneous functions that models three factors of the appliance: device security, usability and electricity pricing. While these two IDS secure individual devices in the smart grid, they do not provide stateful monitoring at system level for the smart grid. More advanced IDS of this type consider behaviors of multiple devices within the system to obtain system level detection. In [27], Mitchell et al. propose a rule-based IDS for the electric grid by considering the behaviors of three types of physical devices in the electric grid: head-ends, distribution access points/data aggregation points, and subscriber energy meters. Mitchell et al. use readings from 22 sensors from the three types of devices as state components. By quantizing each of the 22 components into a limited number of ranges, they manually build three state machines with 3456, 1728, and 3456 states for the three devices respectively in terms of conjunctive normal form. To manually build such an IDS is very expensive due to the large state space. Additionally, this IDS uses a limited number of sensors therefore is able to detect a limited number of attacks. Since there are always new attacks and applications, this method is not scalable.

Network based IDS leverage communication traffic in the information infrastructure of smart grid to detect cyber-attacks. Yang et al. propose an IDS for synchrophasor systems that detects cyber-attacks by using access control white lists, protocol-based white lists and network behavior-based rules, each of which specify security rules in different layers of the synchrophasor system [29]. The Yang et al. IDS is
limited to cyber-attacks including MITM and DoS against synchrophasor devices and IEEE C37.118 protocol. Similar to Yang’s IDS, Zhang et al. propose a distributed IDS that analyzes communications traffic at different network levels of smart grid including home area networks, neighborhood area networks, and wide area networks [37]. An intelligent module is deployed at each level to classify malicious data and possible cyber-attacks using data mining algorithms. These modules then communicate to get a system level view of the status of the whole communication network to improve the detection accuracy. Hadeli et al., in [38], propose an anomaly detection technique for industrial control systems that extracts behavior patterns of devices from protocols used in industrial control systems, for example, GOOSE messages, IEEE 61850, Manufacturing Message Specification, Modbus/TCP and redundant network routing protocols. The Hadeli IDS uses a system description file to provide a full description of the overall communication pattern in the industrial control system. For the case of power system control applications, the system description file describes expected system behaviors from information carried by those protocols. Hadeli’s method, along with [29] and [37] are efficient to detect malicious activities that cause changes in network traffic, but the IDS fails to detect malicious actions that result in invalid changes to the physical system. For example, Hadeli’s method cannot detect a malicious command to trip a protection relay from a valid IP address which will take a transmission line out of service and cause a blackout. A specification-based IDS that can track sequential events of the system is reported in [39] for the advanced metering infrastructure (AMI). The authors manually built a state machine by extracting specifications from two AMI protocols and devices status. To prove the correctness of the state machine, a model checking technique was
used to verify the specifications. This IDS is also not applicable to transmission systems because transmission systems have far more control actions and disturbances than AMI. As such, manually building a state machine would be very expensive.

Other proposed IDS for smart grid leverage power system theory. For instance, in [40], Valenzuela et al. use optimal power flow programs to detect cyber-attacks which alter system measurement data as the bad data will cause the power flow to be dispatched erroneously. Talebi et al. propose a mechanism for identification of bad data attacks in a power system using weighted state estimation [41]. Zonouz et al. proposed an IDS that not only examines the measurement data using state estimation and power flow theory but also includes the results from network IDS to calculate the probability that the data is compromised [42]. Although these works are all proven capable of detecting altered data, these IDS are limited to one type of attack and cannot be extended to detect other attacks against power systems.

2.3.2 Accuracy of specification-based IDS

The detection accuracy of specification-based IDS depends on how accurately the specifications describe system behaviors. Different efforts have been made to build accurate specifications for specification-based IDS. One approach is to use a formal language, such as the declarative language MuSigs [44] to describe known attacks using temporal logic formulas [45]. The MuSigs authors formally specified attack signatures and proved the soundness and completeness of their detection rules. The use of a formal language to specify behaviors is too work intensive for a power system IDS where there are too many behaviors to specify.
A promising way to improve the accuracy of specifications is through the use of data mining. A data mining technique was applied to an IDS framework proposed by Lee et al. that combined signature-based IDS and anomaly-based IDS [43]. Data mining programs were applied to a large volume of log data to learn attack signatures and normal behavior patterns and automatically create detection rules. Lee et al. showed that the signatures for attacks and patterns for system normal behaviors created using their data mining technique are accurate by comparing their results for probing and user to root privilege escalation attacks to all other participants in the DARPA intrusion detection evaluation program prepared by MIT Lincoln Labs. The overall detection accuracy of their IDS against 4 types of attacks is 80.2% and is the highest among all participants. Lee’s IDS was originally designed for stateless IDS therefore it cannot be directly applied to specification-based IDS. A new data mining algorithm must be developed to discover sequential events for specifications.

2.3.3 Data mining techniques for learning specifications

A specification for a scenario contains a sequence of execution events or system states. The nature of specifications requires the data mining technique applied to the proposed IDS to be able to mine sequential patterns and identify the dependent relationship between events. The data mining technique used in this dissertation uses the “mining sequential patterns” technique which discovers patterns of activity from time ordered data. The “mining sequential patterns” algorithm was first mentioned in [31]. Lin et al. applied it to discover patterns in clinical client care management process data that consists of patient records and log data over a period of treatment time [32]. This technique was extended in [33] by employing a two dimension Bayesian network to
graphically represent patterns in Hemodialysis processes which consist of a sequence of medical activities over time. In order to discover patterns, a patients’ physiological “state” is defined using clinical log data and patient records (e.g. body temperature). The pattern is therefore, represented as contiguous transitions of states in a 2-dimension graph. The classification was made using the patterns.
CHAPTER III

CYBERSECURITY TESTING FOR SYNCHROPHASOR SYSTEM

3.1 Introduction

Many utilities in the United States of America received grants from the Department of Energy under the American Recovery and Reinvestment Act (ARRA) to create wide area monitoring systems. The ARRA grants require recipient entities to develop a cybersecurity plan which includes a risk assessment as part of parent wide area monitoring systems projects. Wide area monitoring systems require installation of phasor measurement units (PMU), and substation phasor data concentrators (PDC), among other devices and software. PMUs and substation PDCs are networked appliances which use routable protocols. As such, these devices may be declared North American Electric Reliability Corporation (NERC) Critical Infrastructure Protection (CIP) standard 002-3 [2] critical cyber assets (CCA), depending upon each individual unit’s application within the power system. CCA must be housed within an electronic security perimeter and undergo a cyber vulnerability assessment.

The IEEE 1402 Guide for Electric Power Substation Physical and Electronic Security [2] defines cyber intrusion or electronic intrusion as “Entry into the substation via telephone lines or other electronic-based media for the manipulation or disturbance of electronic devices.” PMU and substation PDC are networked appliances and may become the target of attacks against bulk electric power systems. Threats against these
devices include denial of service attacks, attacks against open ports and services intended to elevate privilege, attempts to change device settings, attempts to inject malicious device commands, attempts to hijack device access credentials or other confidential information, and attempts to place a man-in-the-middle between devices.

This chapter describes the process used to develop a set of cyber security requirements for PMU and PDC installation. Three primary sources were used to derive cyber security requirements. First, NISTIR 7628: Guidelines for Smart Grid Cyber Security [3] was reviewed and 28 relevant requirements were taken from this document. Second, the Department of Homeland Security: Cyber Security Procurement Language for Control Systems was reviewed. This document was used to derive project requirements and used as a basis for procurement language added to vendor contracts. Second, this dissertation describes testing performed to identify PMU and PDC vulnerabilities prior to device installation in a production control system. A MU Dynamics MU-4000 Analyzer was used to perform network congestion testing, denial of service testing, and protocol mutation testing. Testing also included device manual reviews to identify security related features, security feature testing, network traffic disclosure testing, and subjecting of devices to network isolation via introduction of extraneous VLAN, and a man-in-the-middle attack. Results from the tests were provided to the utility to enable network monitoring to mitigate identified vulnerabilities and to allow the utility to work device vendors to create corrective action plans. PMUs and PDCs from multiple vendors were tested. Vendor names and product identifiers are withheld from this article to prevent enabling attacks. Results from this testing have been shared with device vendors. Finally, this dissertation describes a methodology for
developing signature based intrusion detection system developed for us in bulk electric substations. The intrusion detection system described take information from multiple sources including, SNORT, synchrophasor data, relay data logs, and energy management system logs to provide model based classification of system occurrences as valid faults or network based attacks.

The body of this chapter includes a section describing synchrophasor system cyber security requirements development, a section describing cyber security testing of synchrophasor system components, a section describing development of Snort rules, and finally, a section on future works and conclusions.

### 3.2 Synchrophasor system cyber security requirements development

Prior to testing a set of cybersecurity requirements and recommendations were prepared from review of the National Institute of Standards and Technology Interagency Report (NISTIR) 7628 Guidelines for Smart Grid Cyber Security: Vol. 2, Security Architecture and Security Requirements [3], Department of Homeland Security (DHS): Cyber Security Procurement Language for Control Systems [46], and utility internal requirements. NISTIR 7628 Vol. 2 includes a process for deriving cyber security recommendations and requirements for smart grid systems. NISTIR 7628 requirements and recommendations are taken from NIST SP 800-53 Revision 3 [47], the Department of Homeland Security Catalog of Control Systems Security: Recommendations for Standards Developers [48], NERC CIP (1-9) [2]. Each requirement is traceable to one or more of the aforementioned source documents.

A cross functional team was formed to review and discuss cyber security requirements and recommendations. This team included representatives from the utility,
the vendor of phasor measurement unit and phasor data concentrator hardware, the vendor of the energy management system, bulk electric transmission system consultants, and a cyber security researcher from academia. Team members included cyber security engineers, power system engineers, network communications engineers, hardware and software designers, and management representatives. A subcommittee drafted an initial version of cyber security recommendations and requirements for the intended synchrophasor system. The initial draft was circulated to the larger team for review. Finally, multiple meetings were held with all team members to discuss each proposed cyber security requirement in detail. The resulting recommendations and requirements are included in Table 3.1 and Table 3.2. Table 3.1 and Table 3.2 list requirements pertinent to system hardware and software components.

Requirements related to organization and management, physical protections, services acquisition, macro information system protection, risk management and assessment, personnel security, planning, maintenance, incident response, information and document management, configuration management, training, and security program management exist but are not listed in Table 3.1 and Table 3.2 [83].
<table>
<thead>
<tr>
<th>NISTIR 7628 Req. #</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC-4</td>
<td>Access Enforcement</td>
<td>The synchrophasor system should enforce assigned authorizations for controlling access.</td>
</tr>
<tr>
<td>AC-7</td>
<td>Least Privilege</td>
<td>The synchrophasor system should assign and enforce the most restrictive set of rights and privileges or access needed by users for the performance of specified tasks.</td>
</tr>
<tr>
<td>AC-8</td>
<td>Unsuccessful Login Attempts</td>
<td>The synchrophasor system should enforce a defined number of consecutive invalid login attempts by a user during a defined time period.</td>
</tr>
<tr>
<td>AC-9</td>
<td>Smart Grid Information System Use Notification</td>
<td>The synchrophasor system should display appropriate use banners where applicable.</td>
</tr>
<tr>
<td>AC-10</td>
<td>Previous Logon Notification</td>
<td>The synchrophasor system should notify the user, upon successful logon, of the date and time of the last logon and the number of unsuccessful logon attempts since the last successful logon.</td>
</tr>
</tbody>
</table>
Table 3.1 (Continued)

<table>
<thead>
<tr>
<th>NISTIR 7628 Req. #</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC-12</td>
<td>Session Lock</td>
<td>The synchrophasor system should initiate a session lock after an organization-defined time period of inactivity or upon receiving a request from a user; and retain the session lock until the user reestablishes access.</td>
</tr>
<tr>
<td>AC-21</td>
<td>Passwords</td>
<td>The synchrophasor system should adhere to utility password complexity rules and passwords should be changed according to utility policy.</td>
</tr>
<tr>
<td>AU-2</td>
<td>Auditable Events</td>
<td>A set of auditable events should be developed for the synchrophasor system. The list should be revised based on current threat data, assessment of risk, and post-incident analysis.</td>
</tr>
<tr>
<td>AU-3</td>
<td>Content of Audit Records</td>
<td>The synchrophasor system should produce audit records for each auditable event.</td>
</tr>
<tr>
<td>AU-8</td>
<td>Time Stamps</td>
<td>The synchrophasor system should use internal system clocks to generate time stamps for audit records.</td>
</tr>
</tbody>
</table>
Table 3.1 (Continued)

<table>
<thead>
<tr>
<th>Req. #</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AU-9</td>
<td>Protection of Audit</td>
<td>The synchrophasor system should protect audit information and audit tools from unauthorized access, modification, and deletion.</td>
</tr>
<tr>
<td></td>
<td>Information</td>
<td></td>
</tr>
<tr>
<td>AU-10</td>
<td>Audit Record Retention</td>
<td>The synchrophasor system audit logs for a utility specified time period.</td>
</tr>
<tr>
<td>AU-16</td>
<td>Non-Repudiation</td>
<td>The synchrophasor system should protect against an individual falsely denying having performed a particular action.</td>
</tr>
<tr>
<td>CP-10</td>
<td>Smart Grid Information</td>
<td>The utility must have the capability to recover and reconstitute the synchrophasor system to a known secure state after a disruption, compromise, or failure.</td>
</tr>
<tr>
<td></td>
<td>System Recovery and Reconstitution</td>
<td></td>
</tr>
<tr>
<td>IA-5</td>
<td>Device Identification and Authentication</td>
<td>The synchrophasor system should uniquely identify and authenticate devices before establishing a connection where technically feasible.</td>
</tr>
<tr>
<td>SC-3</td>
<td>Security Function Isolation</td>
<td>The synchrophasor system should isolate security functions from non-security functions.</td>
</tr>
<tr>
<td>NISTIR 7628 Req. #</td>
<td>Title</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SC-5</td>
<td>Denial-of-Service Protection</td>
<td>The synchrophasor system should mitigate or limit the effects of denial-of-service attacks based on an organization-defined list of denial-of-service attacks.</td>
</tr>
<tr>
<td>SC-7</td>
<td>Boundary Protection</td>
<td>The synchrophasor system should be appropriately placed within electronic security perimeters.</td>
</tr>
<tr>
<td>SC-8</td>
<td>Communication Integrity</td>
<td>The Smart Grid information system protects the integrity of electronically communicated information.</td>
</tr>
<tr>
<td>SC-9</td>
<td>Communication Confidentiality</td>
<td>The synchrophasor system should protect the confidentiality of sensitive communicated information.</td>
</tr>
<tr>
<td>SC-10</td>
<td>Trusted Path</td>
<td>The synchrophasor system should establish a trusted communications path between the user and the synchrophasor system.</td>
</tr>
<tr>
<td>NISTIR 7628 Req. #</td>
<td>Title</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>SC-12</td>
<td>Use of Validated Cryptography</td>
<td>All of the cryptography and other security functions that are required shall be NIST Federal Information Processing Standard (FIPS) approved.</td>
</tr>
<tr>
<td>SC-19</td>
<td>Security Roles</td>
<td>Specific security roles and responsibilities for users of the synchrophasor system should be defined.</td>
</tr>
<tr>
<td>SC-20</td>
<td>Message Authenticity</td>
<td>The synchrophasor system should provide mechanisms to protect the authenticity of device-to-device communications.</td>
</tr>
<tr>
<td>SC-22</td>
<td>Fail in Known State</td>
<td>Devices and software used in synchrophasor system should fail in a known state to prevent loss of confidentiality, integrity, or availability.</td>
</tr>
<tr>
<td>SC-26</td>
<td>Confidentiality of Information at Rest</td>
<td>Synchrophasor system hardware and software should employ cryptographic mechanisms for all critical security parameters to prevent unauthorized disclosure of information at rest.</td>
</tr>
</tbody>
</table>
NISTIR requirements address access control (AC), audit requirements (AU), continuity of operations (CP), identification and authentication (IA), and smart grid information system and communication protection (SC). The requirements were derived using the NISTIR 7628 Logical Interface Category 3: Interface between control systems and equipment with high availability, without compute or bandwidth constraints. This interface category specifically includes communication interfaces between phasor measurement units and a wide area measurement system. It was assumed that the synchrophasor system would eventually be used to source measurements to wide area protection system applications and therefore high availability was a requirement. It was also assumed that new computer systems and new communication bandwidth would be added to support the synchrophasor system and therefore not compute or bandwidth constraints were assumed.
Table 3.2  Recommendations and Requirements Derived from DHS Cyber Security Procurement Language for Control Systems [46]

<table>
<thead>
<tr>
<th>Req. #</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC.1</td>
<td>System Hardening</td>
<td>Vendor(s) shall list required ports and services for normal and emergency operation.</td>
</tr>
<tr>
<td>PROC.2</td>
<td>Least Privilege</td>
<td>Vendor(s) shall configure systems with least privilege file and account access and provide documentation of the configuration.</td>
</tr>
<tr>
<td>PROC.3</td>
<td>Hardware Configuration</td>
<td>Vendor(s) shall disable all unneeded communication ports and removable media drives.</td>
</tr>
<tr>
<td>PROC.4</td>
<td>Upgrade Access Control</td>
<td>Vendor(s) shall password protect the BIOS from unauthorized changes.</td>
</tr>
<tr>
<td>PROC.5</td>
<td>Patch Management</td>
<td>Vendor(s) shall have a patch management and update process.</td>
</tr>
<tr>
<td>PROC.6</td>
<td>Perimeter Protection</td>
<td>Vendor(s) shall provide detailed information on all communications (including protocols) required through a firewall.</td>
</tr>
<tr>
<td>PROC.7</td>
<td>Session Management</td>
<td>Vendor(s) shall not permit user credentials to be transmitted in clear text.</td>
</tr>
<tr>
<td>PROC.8</td>
<td>Concurrent Logins</td>
<td>Vendor(s) shall not allow multiple concurrent logins, applications to retain login information between sessions, provide any auto-fill functionality during login, or allow anonymous logins.</td>
</tr>
<tr>
<td>Req. #</td>
<td>Title</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PROC.9</td>
<td>Account Logout and Timeout</td>
<td>Vendor(s) shall provide user account-based logout and timeout settings.</td>
</tr>
<tr>
<td>PROC.10</td>
<td>Warning Banner</td>
<td>A standard warning banner developed by the utility and must be displayed when users logon to a utility computer system and/or network.</td>
</tr>
<tr>
<td>PROC.11</td>
<td>Least Privilege</td>
<td>System owners must restrict privileges for all users, interconnected systems, and software based on the principle of least privilege. Where possible, system role accounts and programs must not run with elevated privileges.</td>
</tr>
<tr>
<td>PROC.12</td>
<td>Configurable Password Complexity</td>
<td>Vendor(s) shall provide a configurable account password management system that allows for selection of password length, frequency of change, setting of required password complexity, number of login attempts, inactive session logout, screen lock by application, and denial of repeated or recycled use of the same password.</td>
</tr>
<tr>
<td>Req. #</td>
<td>Title</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PROC.13</td>
<td>Password storage</td>
<td>Vendor(s) shall not store passwords electronically or in vendor-supplied hardcopy documentation in clear text unless the media is physically protected.</td>
</tr>
<tr>
<td>PROC.14</td>
<td>Emergency Security Rollback</td>
<td>Vendor(s) shall provide a mechanism for rollback of security authentication policies during emergency system recovery.</td>
</tr>
<tr>
<td>PROC.15</td>
<td>Password Encryption Algorithm</td>
<td>Passwords must be encrypted using a utility approved cryptographic algorithm.</td>
</tr>
<tr>
<td>PROC.16</td>
<td>Password Complexity</td>
<td>User account passwords to utility defined complexity requirements.</td>
</tr>
<tr>
<td>PROC.17</td>
<td>Activity Logging</td>
<td>Vendor(s) shall provide a system whereby account activity is logged and is auditable both from a management (policy) and operational (account use activity) perspective.</td>
</tr>
<tr>
<td>PROC.18</td>
<td>Audit Log Time Stamping and Encryption</td>
<td>Vendor(s) shall time stamp, encrypt, and control access to audit trails and log files where feasible.</td>
</tr>
<tr>
<td>Req. #</td>
<td>Title</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PROC.19</td>
<td>Audit Log Impact on System Performance</td>
<td>Vendor(s) shall ensure audit logging does not adversely impact system performance requirements.</td>
</tr>
<tr>
<td>PROC.20</td>
<td>Audit Log Entry Contents</td>
<td>Log data shall include the date and time of the event, the unique ID used to initiate the event, the type of event, success or failure, and the name of the object involved.</td>
</tr>
<tr>
<td>PROC.21</td>
<td>User Accounts with Defined Role</td>
<td>Vendor(s) shall provide for user accounts with configurable access and permissions associated with the defined user role.</td>
</tr>
<tr>
<td>PROC.22</td>
<td>TCP/IP Cybersecurity Features</td>
<td>Vendor(s) shall provide physical and cyber security features, including but not limited to authentication, encryption, access control, event and communication logging, monitoring, and alarming to protect the device and configuration computer from unauthorized modification or use.</td>
</tr>
<tr>
<td>PROC.23</td>
<td>Approved Cryptographic Algorithms</td>
<td>The use of cryptographic algorithms must be limited utility approved algorithms.</td>
</tr>
</tbody>
</table>
3.3 Synchrophasor system cyber security component testing

The cyber security requirements from the above section were applied to hardware, software, and communication systems throughout the synchrophasor system. A diagram was developed which included all system components and communication interfaces to each component. A sanitized version of the synchrophasor system component diagram is shown in Figure 3.1. The energy management system components depicted in this figure may not be complete but are necessary for adopting synchrophasor technology. This work provides a third party testing methodologies for phasor measurement unit (PMU), phasor data concentrator (PDC) and the energy management (EMS) system. The following sections give the cybersecurity testing procedures and methodologies for PMU and PDC. The testing results with vulnerabilities are ranked using a risk scale proprietary to the utility who grant this work.

![Synchrophasor system component diagram](image)

Figure 3.1 Synchrophasor system component diagram
### 3.3.1 Testing environment configuration

Three PMUs and one PDC were tested. A MU Dynamics MU-4000 Analyzer was used to perform denial of service, network congestion, and protocol mutation tests for well-known protocols such as TCP/IP etc. A personal computer (PC) was used with Wireshark to capture network traffic data logs and to host software used to configure and remotely control the PMU and PDC. The PMUs were connected to a Real Time Digital Simulator (RTDS) in a hardware-in-the-loop configuration. The RTDS provided simulated high voltage AC busses for the PMU’s to measure. PMUs were connected through a substation router to PDC. PDC concentrated synchrophasor measurement streams from the PMU and forwarded this data to an OpenPDC installation which served as a historian for the system. Noted that, there is another PC depicted as attacker’s, which is used to perform penetration testing such as Man-in-the-middle attacks and IEEE C37.118 protocol mutation tests. Figure 3.2 shows the test bed configuration.

![Test bed configuration](image)

Figure 3.2 Test bed configuration
PMUs periodically measure voltage, current, and transmit voltage and current phasors (based upon a reference cosine waveform) at 120 samples per second. PMUs are time synchronized devices with clocks synchronized to Universal Time Coordinated (UTC) with 1 microsecond accuracy. Synchrophasor network packets are transmitted from the PMUs to a PDC. PMUs adhere to the IEEE C37.118 standard which specifies measurement requirements and the synchrophasor measurement format. PMUs may communicate over Ethernet or Serial port. Three PMU’s were tested for this work. PMU A and PMU B shared the same vendor, while PMU C was manufactured by a second vendor. All PMUs communicate over Ethernet using the IEEE C37.118 protocol. PDC collect synchrophasor streams from multiple PMU and create a single stream for retransmission to historian. PDC perform stream data rate conversion and can be configured to interpolate when data is missing from a stream. PDC adhere to the IEEE C37.118 standard and communicate over Ethernet.

3.4 Cybersecurity testing

3.4.1 Network congestion testing

The MU-4000 Network Analyzer was used to perform network congestion testing. The MU-4000 denial of service test suite includes tests for multiple network protocols across all network OSI layers. The denial of service tests validate a device’s ability to withstand large volumes of traffic directed at the device. The relevant network protocols for testing PMU and PDC include various types of protocols in different ISO/OSI layers.

Each network congestion test attempts to stress a separate portion of the device’s network stack. The tests target a device’s ability to process large volumes of a single type
of network traffic. The PMUs and PDC are usually based on embedded systems or simplified PC structure therefore contain limited memory which can be exhausted and lead to operating system exceptions, cause services to stall, and or cause the device to reset itself. A set of network layer tests send floods of ARP requests, PPPOE packets, and IPv4 packets to the target device. Network layer variations send random packets of all three types, IP packets with random sizes and random payload, and IP packets with large numbers of IP fragments. A set of ICMP tests were also used. ICMP tests send floods of ICMP echo requests (aka. Ping flood or Smurf attack), ICMP echo packets with large payloads, address mask requests, and source quench messages.

Transport layer tests send floods of TCP and UDP packets to the device under test. TCP tests include variations which stress a device’s ability to create and teardown TCP sessions with floods of TCP SYN and TCP FIN packets targeting individual TCP ports and to random TCP ports. UDP tests include random headers and payloads to the UDP ports which are open in the target devices.

Two tests validate device behavior for illegal packet types. A LAND test sends floods of IP packets with both the source and destination IP address set to the target’s IP address. A teardrop test sends fragmented IP packets which have overlapping IP fragments.

All devices tested eventually became unresponsive when the traffic volume increases beyond that devices ability to process packets. Figure 3.3 shows typical device behavior to denial of service tests. The brown triangle shows the rate packets are being transmitted to the target device. As the tester ramps the packet rate it periodically sends the target an instrumentation packet (a query which the tested device is known to support)
to test if the device is able to respond. The instrumentation packet may be a TCP session request on a supported port or an ICMP echo request or any other type of packet the target is known to be capable of responding to. The blue vertical lines show the target device responding to instrumentation requests. A taller blue line indicates a slower response time. The red dots indicate failed instrumentation request. As the packet rate increases devices become unresponsive. Some devices may hang or reset themselves when subjected to high packet rates. Many devices are unresponsive during the test, but, become responsive again when the packet rate returns to acceptable levels.

![Denial of service test response time chart](image)

Figure 3.3 Denial of service test response time chart

Understanding the packet rate which causes a device to become unresponsive is important for system planning and for creating an effective denial of service mitigation approach. Figure 3.4 shows a typical availability chart for a single denial of service test against a device. The availability shows the percent availability (Y-axis), percentage of time that a device is able to respond to instrumentation requests, versus packet transmission rate (X-axis).
The availability chart can be used by utility engineers and network administrators to define a maximum threshold for traffic congestion at the switch or router within the substation for the different traffic types. Based upon testing results it is recommended that utilities monitor network traffic volume in control system networks to detect transmission of high volumes of traffic. Monitoring systems should alert a human administrator to enable mitigation. Routers in the control system network may be configured to limit traffic sent to the PMU or PDC or may be configured to close ports sourcing offensive amounts of network traffic. Automatically closing router ports is potentially dangerous since critical traffic may use the port. A thorough system review should be performed before enabling automatic port closure. Maximum traffic rate thresholds should be defined for all relevant traffic types.

![Availability Chart](image)

**Figure 3.4** Availability chart from congestion testing

It is important to understand PMU and PDC behavior after DOS event completes. Testers should confirm that tested devices and network appliances in the route do not
queue large volume of IEEE C37.118 data packets which then leads to a synchrophasor stream which is perpetually delayed. PDC hold data from on time PMU to wait for data packets from late arriving PMU streams. A denial of service attack can have a persistent effect if the attacked PMU’s date stream becomes consistently late after the attack. PDC eventually drop old data packets and begin to interpolate. PMU and PDC which recover from a denial of service attack should clear their transmit queues to avoid the aforementioned effects.

3.4.2 Protocol mutations

A second method to test for denial of service vulnerabilities is through protocol mutation, also known as protocol fuzzing. Fuzzing is a general term for a type of software testing technology that uses unexpected input and monitors for exceptions for discovering faults in software. The results of fuzzing can help to ensure that exceptions can be handled appropriately, filter out unwanted values while allowing the full range of acceptable inputs [49]. In the rest of this chapter, it is abbreviated as fuzzing. The program that performs fuzzing work is called fuzzer. Two types of protocol mutation methods are used in this work. One is brute force fuzzing that creates network packets with random changes [49]. The fuzzer that implements brute force fuzzing starts from a valid sample of a protocol or data format and keeps mangling every individual bit or byte or word within the data packet. This type of fuzzing is pretty straightforward because all it needs to do is to modify the data and pass it along to the target. It requires little research to the format of the data massage and therefore it is also called “dumb fuzzing”.

The other type of fuzzing is through manually modifying protocol packets according to how the protocol specification works and therefore it is relatively “smart”. 

43
This type of fuzzing is called “smart fuzzing” in this work. In smart fuzzing each field in a mutated packet’s header, payload, and trailer is assigned a set of variant values that are designed specifically for the target. The selection of protocols for mutation testing was based on port scanning and device manual review results. Variant values for a field may include legal values and illegal values. The protocol mutation tester creates a set of packets which include all combinations of all fields with all variant values. The number of combinations grows quickly and protocol mutation can be a slow process. However, the benefit of protocol mutation, no matter dumb or smart is that combinations of packet fields which may not be thought of by a human can be tested to confirm that the device network stack does not hang or reset when the test packet is processed. Protocol mutation is intended to discover vulnerabilities before they are discovered by an adversary and become exploited zero day vulnerabilities.

All communication protocols supported by a device are tested. Mutated protocols for the PMU and PDCs included ARP, TCP, UDP, IP, ICMP, DNP3, MODBUS, IEEE C37.118, and HTTP. The protocol mutation testing of these protocols are done using the MU 4000 Network analyzer except IEEE C37.118 where a fuzzing framework is developed.

3.4.2.1 MU-4000 network analyzer

The MU-4000 Network Analyzer was used to perform protocol mutation testing. As with the denial of service testing the tester sends groups of mutated packets to the target device. The tester periodically sends instrumentation packets (queries which the tested device is known to support) to confirm that the device under test can still respond. Protocol mutation requires two types of instrumentation packets. The first
instrumentation is a communication packet and response pair which is known to work on the target device. This instrumentation is typically unrelated to the mutated protocol. This instrumentation confirms the device network stack is still functioning and responsive. It is possible the portion of the network stack associated with the mutated protocol will hang without affecting other parts of the network stack. For example, a UDP mutation may hang the UDP stack, but leave the TCP stack functioning correctly.

The second instrumentation request type is a known good packet of the type being mutated. This instrumentation confirms the portion of the network stack related to the mutated protocol is still functioning and responsive.

Some services were capable of assignment to a variable TCP or UDP port number. In this case, protocol mutation was repeated for multiple ports. A good strategy for testing services with variable ports is to repeat testing with port assigned to multiple port numbers in the well known space (0-1023), multiple port numbers in the registered port range (1023-49151), and multiple port numbers in the private range (49152-65535). Some services are capable of assignment to a fixed set of port numbers. In this case, it is good practice to test at all legal port assignments.

The MU-4000 includes built-in protocol mutation capabilities for many well known protocols. Some protocols are not supported. For example, IEEE C37.118 is not natively supported. Also, newly developed protocols may not initially be supported. The MU-4000 is capable of learning protocols from Wireshark packet captures. After learning a protocol the MU-4000 scenario builder can generate protocol mutations to test a device. The scenario builder feature was used for IEEE C37.118 protocol mutation. Only frames received as input by the target device should be mutated and sent to the
target device. Mutated IEEE C37.118 commands frames were mutated and sent to PMUs. Mutated IEEE C37.118 configuration and data frames were sent to the PDCs.

Protocol Mutation testing may indentify individual packets which cause device failures including hanging network stacks or causing the device under test to reset itself. Protocol Mutation testing may also indentify combinations of packets which cause similar device failures. In both cases careful study is required to determine the root cause of the failure. Mitigation of detected vulnerabilities can be achieved with a firewall or signature based intrusion prevention system (IPS) rules to block problem traffic. Vulnerabilities identified using protocol mutation should also be reported to the device vendor. Protocol mutation identified multiple issues on devices tested for this work. Issues included crashing of individual network services, crashing of applications running on devices, and unintended soft resetting of affected devices.

The MU-4000 works best as a client which sends mutated packets to a server. The MU-4000 uses randomization algorithms and constrained randomization algorithms to fuzz servers. The MU-4000 is less capable of fuzzing server to client responses, especially responses which are dependent upon the previous packet sent from the client. To overcome this issue an in-line fuzzer was developed to mutate server to client packets. The current version of the in-line fuzzer simply varies random bits of the server to client response to attempt to break random protocol rules. This method has proven effective at identifying vulnerabilities. A fuzzer is needed which properly mutates server to client responses based upon previous client to server packets and system state. The next section is analysis of the needs of such a fuzzer for IEEE C37.118 packets.
3.4.2.2 A fuzzing framework for IEEE C37.118 protocol

The fuzzer for IEEE C37.118 protocol must be able to fuzz server to client responses, especially responses which are dependent upon the previous packet sent from the client. To achieve this a fuzzing framework was developed to mutate server to client packets. The current version of the fuzzing framework is able to perform both dumb fuzzing and smart fuzzing for IEEE C37.118 packets. Dumb fuzzing flips bits of the server-to-client response to attempt to break random protocol rules, while smart fuzzing mangles the specific fields of data packets according to the protocol specifications.

The fuzzing framework is designed based on the communication patterns of the IEEE C37.118 protocol. The fuzzing framework mutates server to client responses based upon previous client to server packets and system state. There are four types of packets defined in IEEE C37.118 protocol [50]: header, command, configuration, and data. The communication pattern between PMU and PDC is depicted in Figure 3.5 where PMU is the server and PDC is the client. The IEEE C37.118 protocol is an application layer protocol that can be carried in the TCP payload or UDP payload. Since UDP protocol is much faster and easy to implement, in this testing the communication between the PMU and PDC is configured to use IEEE C37.118 based on UDP. Communication is initiated by the PDC (i.e. client) through sending a command frame that tells the PMU (i.e. server) to start to stream measurements to the PDC. The command frame can also deliver other commands such as a stop command, request configuration frame command, and others. The PMU which receives the start command then knows where to send its C37.118 packets. However, before sending the data frames the PMU sends a configuration frame to the PDC in which the PMU communicates the organization and size of its data frames.
There are two types of configuration frames: configuration frame 1 and configuration frame 2. The configuration frame 1 contains the constant part of the PMU configuration and therefore it usually needs to be sent only once the first time PMU is connected to PDC. The configuration frame 2 contains a variable part of the PMU configuration e.g. number of phasors. Hence the configuration frame 2 has to be sent every time the PMU is connected to the PDC such that PDC will know whether there are changes in the PMU and update them. The PMU then starts to send the data frames at a rate ranging from 30 to 120 frames per second. The data frame contains real time phasor data e.g. magnitude, phase angle, frequency, analog, digital data. The Header frame contains up to 80 characters of human readable/ASCII with comments on the PMU, the data sources, scaling algorithms or any other information. The Header frame is rarely used.

![PMU and PDC communication pattern](image)

Figure 3.5 PMU and PDC communication pattern

Only frames received as input by the target device should be mutated and sent to the target device for the purpose of testing whether the target can handle the mutated packets properly. According to different synchrophasor devices since each of them
accepts different frames shown by the arrows in Figure 3.5, different fuzzing strategies are used. For testing the PDC, three types of frames should be considered: configuration frame 1 or 2, data frame, and the header frame. As for the PMU, the command frame and the header frame are mutated for testing.

A typical fuzzing framework usually consists of three parts: test case generation; target monitoring, and logging pertinent data on failure [49]. A fuzzing framework for network protocols requires a tool to capture valid protocol traffic [51]. One such tool can be a sniffer that dynamically captures the network packets at runtime. The structure of the fuzzing framework designed for IEEE C37.118 protocol is shown in Figure 3.6. The fuzzing framework is executed on a Linux computer that connects to the substation switch. The fuzzing framework consists of five major parts: man-in-the-middle (MITM) server, protocol parser, fuzzing engine, validation engine and log, in which the MITM server captures the IEEE C37.118 protocol packets; the fuzzing engine generates the test cases; validation engine monitors the devices under test and log stores not only the test cases that crash the devices but also the activities of the fuzzing framework.
The MITM server is implemented using Ettercap. Ettercap is a sniffer written in C language that features passive and active sniffing in the communication channel, content filtering on the fly and many other MITM attacks. The Ettercap in this fuzzer redirects all the packets between PMU and PDC to the fuzzing framework PC by spoofing the Media Access Control (MAC) addresses of the PMU and PDC such that it pretends to be the PMU for the PDC and pretends to be the PDC for the PMU. This is achieved by sending the Address Resolution Protocol (ARP) packets to the victim devices with the MAC address replaced with the MITM server’s. Ettercap then captures
the IEEE C37.118 packets off the wire, layers the new IEEE C37.118 packets onto UDP packets and finally transmits the new packets onto the network. The new IEEE C37.118 packet has its fields altered according to the type of fuzzing chosen. The content of IEEE C37.118 packet is modified using Scapy [54]. Scapy is an open source Python program that provides a Domain Specific Language (DSL) that enables the users to describe any kind of packet using the Python syntax and interpreter. In the fuzzing framework, Scapy is used to accomplish IEEE C37.118 packet assembly, packet editing, packet re-play and packet decoding.

When an IEEE C37.118 packet is captured by the MITM server, it hands the packet to the “protocol parser” where the packet is decoded into one of the four types of the IEEE C37.118 frames. The protocol parser is a Python module that has been developed to describe the IEEE C37.118 protocol based on DSL provided by Scapy. The frame type field is examined to determine which type of IEEE C37.118 frames the packet belongs to. Then the packet will be forwarded to the corresponding fuzzer for mutations by the “fuzzing engine”.

The fuzzing engine is composed of four frame fuzzers that mutate different types of IEEE C37.118 frames. Each frame fuzzer accomplishes packet editing by applying either dumb fuzzing or smart fuzzing to the packet. The smart fuzzing has predefined mutations to the fields of the packet according to the protocol standard. For example, the field FRAMESIZE of the packet is changed such that it does not match the actual size of that packet [56]. Besides the common fields that all types of frames have, each type of frame also has its unique fields. For example, the IEEE C37.118 command frame has two unique fields. One of them is the CMD field that is a 2 bytes field specifying the
command. There are 6 defined values for this field. However, a smart fuzzer for command frame is designed to replace the CMD with a value that is not one of the 6 legal values.

3.4.2.2.1 Smart fuzzing

Smart fuzzing mutates the packets based on the protocol specifications. IEEE C37.118 includes 4 types of packets; header, command, configuration, and data. Header and command packets are transmitted from the PDC to the PMU. Configuration and command packets are transmitted from the PMU to the PDC.

All 4 frame types include a 2 byte synchronization word (SYNC). The first byte of the SYNC is defined as always 0xAA. It is important to check other values for this field. The second byte of the SYNC field includes a reserved bit, 3 bits to designate the frame type, and 4 bits for version number. There are 5 legal frame types. Illegal frame types should be sent; 0b101, 0b110, 0b111. All 16 possible version number possibilities should be sent; though only some have been defined to date. All 4 IEEE C37.118 frame types include a 2 byte frame size field. Frames should be sent with frame sizes which do not match the actual FRAMESIZE. Also, very large frame sizes should be sent to test for buffer overflow possibilities. A 4 frame types include a 2 byte IDCODE field. This value is the PMU or PDC ID number. The values 0 and 65535 are reserved and therefore should be tested. PMU and PDC typically have preprogrammed ID values. Frames with IDCODE values not assigned to the target device should be tested. All 4 IEEE C37.118 frame types include a 4 byte SOC field. The SOC field is a time stamp that counts the number of seconds since Jan-01-1970. The field is limited to 136 years which means the max value is 0xB34C00. Above 0xB34C00 the count is supposed to roll over. It is
important to test values greater than 0xB34C00. All 4 IEEE C37.118 frame types include a 4 byte FRACSEC field. This field is broken into two parts. The most significant 4 bits of FRACSEC (bits 31-28) are used to document the presence of a leap second. Bit 31 is reserved and therefore transmitting a 1 in this bit should be tested. Bits 30 (LEAP) indicates a leap second is occurring. Bit 29 (LEAPED) indicates a leap second occurred in the last 24 hours. Bit 28 (TOLEAP) indicates a leap second will occur in the next second. Various fuzzing scenarios can be derived for these fields. First, the leap second bits should be asserted at times and dates when they are not expected. Seconds, LEAP should be set without first setting TOLEAP in the previous second. LEAP should be set without setting LEAPED in the following second and 24 hours. TOLEAP should be set with no following LEAP assertion. LEAPED should be asserted when not preceded by TOLEAP or LEAP combinations. Finally, all three bits (LEAP, LEAPED, TOLEAP) should be asserted at random times. The next 4 bits of FRACSEC (bits 27-24) are defined by a table to indicate clock faults and clock synchronization values. There are multiple reserved values (0b1100, 0b1101, 0b1110) which should be tested. The remainder of the FRACSEC field is a number fraction of a second. This value is depended upon the TIMEBASE value from the PMU configuration frame. This value can be changed when configuring the PMU. FRACSEC values which do not match with the programmed TIMEBASE should be tested. Finally, All 4 IEEE C37.118 frame types include a 2 byte CHK field which is a 16-bit CRC. Frames with invalid CRC values should be tested. Some fuzzers make changes to valid packets by randomly flipping bit values. In this case the fuzzer should ensure that the CHK field is correct to ensure that more that the CRC logic is being tested.
The IEEE C37.118 data frame has multiple unique fields. Since data frames are transmitted from the PMU to PDC fuzzing data frames is limited to the PDC. The STAT field is a 2 byte field which provides PMU status. This field includes multiple reserved and user defined bits. All combinations of these bits should be tested. The PHASORS, FREQ, DFREQ, ANALOG, and DIGITAL fields all vary in size according to values in the configuration frame. The configuration frame is sent from the PMU to PDC during initial session start-up. Tests should include varying the number of values in these fields to not match the configuration frame definitions. Variation should include 0 bytes, larger, and smaller number of bytes for each field. PDC concentrate multiple synchrophasor streams from PMU into a single stream of IEEE C37.118 data frames. As such the size of the data frames output from PDC varies according to the number of PMU which is defined in a configuration from sent from the PDC to its upstream client, an EMS, state estimator, or openPDC. It is important to test varying data frame sizes. Very large sizes should be tested to check for buffer overflow vulnerabilities. Also, it is important to test data frame sizes which do not match the configuration frame.

The IEEE C37.118 configuration frame has multiple unique fields. Since configuration frames are transmitted from the PMU to PDC fuzzing data frames is limited to the PDC. Fuzzing PDC configuration frames is a challenge because the PDC typically requests the configuration frame only once when the session is initiated. The PDC can be forced to request a configuration frame update by asserting bit 10 in the STAT word of a data frame send from the PMU to PDC. Bit 10 of the STAT word indicates the configuration has changed and the PDC should request to read the configuration files. The TIME_BASE field is 4 bytes. The most significant byte of
TIME_BASE is reserved. Tests should be conducted with these bits set to non legal values (0-255). The NUM_PMU field 2 byte field which specifies the number of PMU in a data frame. This field can legally be up to 65535. However, the actual limit is less than 65535 since the maximum FRAMESIZE is 65535. The actual limit depends upon the values of PHNMR, ANNMNR, DGNMR, and FORMAT which set the number of phasors, analog values, digital values, and format of said values for each PMU in the frame. Testing combinations of NUM_PMU and the PHNMR, ANNMNR, DGNMR, and FORMAT which result in greater than 65535 bytes in the data frame is important. Also, testing combinations of NUM_PMU and PHNMR, ANNMNR, DGNMR, and FORMAT which result in do not match the data in the data frames is important. The CHNAM field is specified as 16*(PHNMR+ ANNMNR +16 *DGNMR). Testing combinations of CHNAM, PHNMR, ANNMNR, and DGNMR which do not adhere to the previous definition is important. The FORMAT field specifies the data type of FREQ, DFREQ, PHASORS, and ANALOG fields from the data frame. Testing combinations of FORMAT which do not match the values in the FREQ, DFREQ, PHASORS, and ANALOG fields in the data frame is important. Bits 15-4 of the FORMAT field are reserved. Testing non-zero fields in this field is important. The PHUNIT field of the configuration frame is 4 bytes. The most significant byte has legal values of 0 or 1. Testing should be completed to send values 2-255 in this byte. The ANUNIT field is a 4 byte field. The most significant byte of this field has several constraints. Values 3-4 are undefined by the specification. Values 5-64 are reserved. Values 65-255 are user definable. All values from 3-255 should be tested. THE DIGUNIT is 4 byte mask of the DIGITAL field from the data frame. Bits 63-48 and 32-16 are a mask which indicates the
normal status of the digital bit corresponding to that bit lane. Test should be conducted to change normal status bit values for bits not in use in the DIGITAL field of the data frame. Test should also be conducted to invert the normal value for bits which are in use in the DIGITAL field in the data frame. Bits 47-33 and 15-0 are masks which indicate which bytes are in use. Tests should be conducted to deselect DIGITAL field bits which are actually in use and select DIGITAL field bits which are not actually in use. The FNOM field is a 2 byte field which sets the nominal frequency. Only two values are allowed 0 and 1. Tests should be conducted for values from 2-65535. The DATA_RATE field is a 2 byte signed integer representing the number of frames per second. Typically this value will be 30, 60 or 120 frames per second. However, the legal values are [-32767, 32767]. Testing should be conducted for multiple values throughout this range. Additionally, the value 0x8000 should also be tested since it fits in the field but is not specified as legal since it is effectively -0. CFGCNT is a 2 byte field which indicates the number of configuration changes since installation. This value should be varied out of order and changed to large values to test PDC response.

The IEEE C37.118 command frame has two unique fields. Command frames are sent to PMU. Command frames may also be sent to the upstream facing interface of the PDC. The CMD field is a 2 byte field specifying the command. There are 6 defined values for this field. Undefined values should be sent to the device to test behavior. EXITFRAME is a variable length field from 0-65518 bytes. This size is limited by the FRAMESIZE field in the command frame. The value of EXITFRAME is user defined. Tests should be conducted to send non-zero size EXITFRAMEs. Also, test should be
conducted in which the FRAMZSIZE is too large or too small based upon the size of the EXITFRAME field.

The IEEE C37.118 header frame has one type of unique field. Header frames are read from the PMU and therefore fuzzing of header frames is directed at the PMU. The header frame may have up to K ASCII bytes of data. The number of bytes of data is the FRAMESIZE – 16. The maximum number of data bytes is therefore 65519. Header frames should be tested with non-ASCII characters in the data bytes of a header frame. Header frames with non-printable characters should also be tested in the data byte fields. Finally, testing should be conducted when the FRAMESIZE specified incorrect for the number of data bytes transmitted.

3.4.2.2 Dumb fuzzing

Dumb fuzzing perform packet mutation without knowing the protocol specifications. Therefore, a dumb fuzzer create test cases by flipping bits in a capture packet. In this work, the dumb fuzzing is implemented through a file fuzzer – ZZUF [53]. ZZUF was originally designed as an application input fuzzer that intercepts file operations and changes random bits in the program’s inputs. Therefore ZZUF has no ability to mutate network packets. To resolve this, the about-to-be-mutated IEEE C37.118 packet is written into a temporary file as binary strings. Instead of fuzzing the packet, the binary file is mutated by ZZUF. The output from ZZUF is layered upon a UDP packet and forwarded by the MITM server to the target.

How ZZUF mutates the binary file is exclusively determined by two of its parameters: fuzzing ratio and seeds. This feature of ZZUF is convenient to the tester for easily reproducing the mutated packets and later replaying the bugs. The fuzzing ratio
indicates the proportion of bits that ZZUF changes. A fuzzing ratio of 1/command frame length (bits) was used to cause ZZUF to invert 1 bit per fuzzed packet. The fuzzing ratio is approximate. The actual test results show each packet had 1-10 randomly changed bits. The fuzzing results can be reproducible by specifying the “seed” parameter if the ratio is fixed. For example, if the ratio is fixed to 0.01 which means 1% of bits in the packet are inverted, the seed number 1 will always choose the same bits for inversion by ZUFF. Therefore, if a mutated packet crashes the target device, we are able to reproduce this packet if we recorded the original packet and the ratio and seed. This task is done by the logging system of the fuzzing framework. The logging system records all activities of the fuzzing framework as well as the malicious packets that crash a device under test.

The validation system starts sending validation packets through the MITM server after each mutated packet being sent to the target. There are two methods in this framework to validate whether the target survives the malicious input: Internet Control Message Protocol (ICMP) echo and HTTP request. The first method is suitable for all types of devices that are networked to the switch while the second method is only available to the devices that are running an HTTP server. The ICMP validation sends ICMP echo request packets and expects a response from the target device. The HTTP validation sends a “GET” request and expects the webserver to return a webpage. The tester can choose either or both as the validation method. If the target device fails to respond to the ICMP request or HTTP GET request it will be regarded as a crash and then the original packet (packet before being mutated) is recorded in the file system and the ratio and seed information about ZZUF is logged for the future analysis.
3.4.2.2.3  Fuzzing algorithms for different IEEE C37.118 frames

This section describes four algorithms to mutate IEEE C37.118 frames using both smart fuzzing and dumb fuzzing via ZUFF. The fuzzing ratio for ZUFF is fixed such that only 1 bit of the frame is flipped.

3.4.2.2.3.1  Configuration Frame Fuzzing

The configuration frame fuzzer alters data frames and forwards these altered configuration frames to the device under test, i.e. PDC. Before the fuzzing starts the connection between PMU and PDC needs to be established. This is done by the MITM server by examining whether data frames streaming from the PMU to PDC have been captured. To provide a large set of fuzzable configuration frames the MITM server makes a change data frames request. A data frame has its STATUS field changed to value of 0x0004. This informs the PDC that the configuration of PMU has been changed and thus the PDC should send a command frame requesting the new configuration frame. The approach to fuzz a configuration frame is shown in Table 3.3.

3.4.2.2.3.2  Data Frame Fuzzing

The data frame fuzzer alters data frames and forwards these altered data frames to the device under test, i.e. PDC. Data frame fuzzing only requires a connection between PMU and PDC to be established before running the MITM server. The algorithm for fuzzing the data frame is listed in Table 3.4.

3.4.2.2.3.3  Command Frame Fuzzing

The command frame fuzzer alters command frames and forwards these altered command frames to the device under test, i.e. PMU. A simple way to provide a large set
of fuzzable command frames is to use SCAPY to generate a normal command frame from which the mutated command frames are created. This is done by the protocol parser. In this case the MITM server does not necessarily need to perform ARP spoofing to intercept the connection between PMU and PDC. Instead, the MITM server initiates the connection to PDC by sending a “Transmission On” command. The PMU responds to this command by starting to stream data frames. When the fuzzer detects the connection between the MITM server and PDC has been established the fuzzing process starts. Besides the two validation methods, ICMP echo request and HTTP GET request, there is one more way to inspect whether the target device survived the command frame fuzzing. This is through examining whether the target device continues streaming data frames. This is necessary because it is possible that the device’s network stack is working properly (by responding ICMP echo request) but the Synchrophasor processing unit has crashed due to the fuzzing. However, this validation is not suitable for fuzzing the command “Transmission Off” as this command is originally used to stop the streaming.

Algorithm 3 lists the approach to fuzz a command frame. There are 6 types of command frames: “Transmission On”; “Transmission Off”; “Send Header”; “Send Configure frame 1”; “Send Configure frame 2” and “Send extra frames”. To fuzz all 6 types of frames a loop is created between step 2 and step 16 such that the protocol parser can create the 6 types of command frames. The algorithm for fuzzing command frames are summarized in Table 3.5.

3.4.2.2.3.4 Header Frame Fuzzing

The header frame fuzzer alters header frames and forwards these altered header frames to the device under test, i.e. PDC. This fuzzing can performed after the connection
between PMU and PDC is established. To provide a large set of fuzzable header frames, a command frame that requests a header frame from PMU needs to be created by the fuzzer and sent to the PMU. This informs the PMU to send a header frame from which the mutations can be made. The approach to fuzz a header frame is shown in Table 3.6.
Algorithm 1 configuration frame fuzzing approach

1: Initiate fuzzing ratio and set seed number to 0.

2: MITM server captures a packet and hands to Protocol Parser

3: if the packet is data frame:
   4:   MITM server alters data frame to cause configuration frame transmission.
   5:   go back to step 2

6: elseif the packet is configuration frame:
   7:   ZZUF alters bits in the frame OR smart fuzzer for deliberate changes in specific fields

8: else    go back to step 2

9: MITM server forwards altered frame to device under test.

10: Validation Engine sends ICMP echo request or/and HTTP GET request to the device.

11: if responses are captured:

12:   (device survived) pass

13: else    recorder the configuration frame, fuzzing ratio and the seed number.

14: Increase seed number by 1

15: if seed number == maximum seed number:

16:   stop fuzzing

17: else    go to step 2
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Initiate fuzzing ratio and set seed number to 0.</td>
</tr>
<tr>
<td>2</td>
<td>MITM server captures a packet and hands to Protocol Parser</td>
</tr>
<tr>
<td>3</td>
<td><strong>if</strong> the packet is data frame:</td>
</tr>
<tr>
<td>4</td>
<td>ZZUF alters bits in the frame OR smart fuzzer makes deliberate changes in specific fields.</td>
</tr>
<tr>
<td>5</td>
<td><strong>else</strong> go back to step 2</td>
</tr>
<tr>
<td>6</td>
<td>MITM server forwards altered frame to device under test.</td>
</tr>
<tr>
<td>7</td>
<td>Validation Engine sends ICMP echo request or/and HTTP GET request to the device.</td>
</tr>
<tr>
<td>8</td>
<td><strong>if</strong> responses are captured:</td>
</tr>
<tr>
<td>9</td>
<td>(device survived) pass</td>
</tr>
<tr>
<td>10</td>
<td><strong>else</strong> recorder the data frame, fuzzing ratio and the seed number.</td>
</tr>
<tr>
<td>11</td>
<td>Increase seed number by 1</td>
</tr>
<tr>
<td>12</td>
<td><strong>if</strong> seed number == maximum seed number:</td>
</tr>
<tr>
<td>13</td>
<td>stop fuzzing</td>
</tr>
<tr>
<td>14</td>
<td><strong>else</strong> go to step 2</td>
</tr>
</tbody>
</table>
Table 3.5  Algorithm for command frame fuzzing

<table>
<thead>
<tr>
<th>Algorithm 3 command frame fuzzing approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Initiate fuzzing ratio and set seed number to 0.</td>
</tr>
<tr>
<td>2: MITM server sends a “Transmission on” command</td>
</tr>
<tr>
<td>3: if the received packet is NOT data frame:</td>
</tr>
<tr>
<td>4: go to step 2</td>
</tr>
<tr>
<td>5: else:</td>
</tr>
<tr>
<td>6: Protocol Parser creates a normal command frame</td>
</tr>
<tr>
<td>7: ZZUF alters random bits in the frame or smart fuzzer makes deliberate changes in specific fields</td>
</tr>
<tr>
<td>8: MITM server forwards altered frame to device under test.</td>
</tr>
<tr>
<td>9: Validation Engine sends ICMP echo request or/and HTTP GET request to the device.</td>
</tr>
<tr>
<td>10: if responses and data frames (not suitable for “Transmission off” command) are captured:</td>
</tr>
<tr>
<td>11: (device survived) pass</td>
</tr>
<tr>
<td>12: else recorder the command frame, fuzzing ratio and the seed number.</td>
</tr>
<tr>
<td>13: Increase seed number = seed number + 1</td>
</tr>
<tr>
<td>14: if seed number == maximum seed number:</td>
</tr>
<tr>
<td>15: stop fuzzing</td>
</tr>
<tr>
<td>16: else go to step 2</td>
</tr>
</tbody>
</table>
Algorithm 4 header frame fuzzing approach

1: Initiate fuzzing ratio and set seed number to 0.
2: MITM server captures a packet and hands to Protocol Parser
3: if the packet is data frame:
4:   MITM server sends a command frame to cause header frame transmission. Then go to step 2
5: elseif the packet is header frame:
6:   ZZUF alters bits in the frame OR smart fuzzer for deliberate changes in specific fields
7: else  go to step 2
8: MITM server forwards altered frame to device under test.
9: Validation Engine sends ICMP echo request or/and HTTP GET request to the device.
10: if responses are captured:
11:   (device survived) pass
12: elseif recorder the configuration frame, fuzzing ratio and the seed number.
13: Increase seed number = seed number + 1
14: if seed number == maximum seed number:
15:   stop fuzzing
16: else  go to step 2
The structure of the in-line fuzzer is also suitable for fuzzing other network protocols. As long as the protocol to be fuzzed is an open standard a corresponding protocol parser can be developed. This is referred to as “white box” testing. For proprietary protocols in which protocol specification is unavailable only dumb fuzzing is possible. The next section provides a methodology to fuzz a typical energy management system in which an intelligent dumb fuzzing framework is proposed.

3.4.3 Energy management system fuzzing framework

The energy management system (EMS) consists of a set of computer applications and databases that help operators in electricity utilities perform monitoring, control, and optimization of the performance of the electricity power grid. In general, the EMS system structure is shown in Figure 3.7 where the client is an application programming interface (API) that provides a human machine interface (HMI) to the operators. The client API communicates with different applications that may be located in a super server. Each application performs corresponding analysis and calculation according to the requests from the client. Multiple applications may run simultaneously. The databases provide necessary data and operations descriptions for the applications and they can be running in the same super server or distributed in different super servers. The communication between the client API and applications usually uses proprietary protocols that are not open to the public. The EMS system fuzzing framework in this work is meant to test applications within the EMS system that listen on a network interface of the super server. The fuzzer sends anomalous input to attempt to crash EMS applications in order to reveal the security vulnerabilities of the EMS system. The structure of fuzzing framework is not
only limited to EMS super server but also could be used as a network fuzzer to any other
distributed database systems.

The test bed for testing the proposed fuzzing framework has one client machine
and one super server. This test bed assumes that client can be run either remotely or
locally along with an application. The test bed also assumes that the tester is able to
access the application machine and source code for the purpose of debugging. In
addition, the assumption that the protocol does not have checksum validation or the
checksum is disabled in the communication is necessary.

Figure 3.7   A general structure for EMS system
The fuzzing framework for the EMS system is based on the Basic Fuzzing Framework (BFF) [55]. The BFF was originally designed to test applications running locally on Linux and Mac OS X platforms. The fuzzing framework designed for this work aims to test applications running over a network with the proprietary protocol. Nevertheless, the design is consistent with the four functions of a typical fuzzing framework as stated in the previous section. The structure of the fuzzing framework is depicted in Figure 3.8. The fuzzing method for this framework is limited to dumb fuzzing because the protocol specifications are not available. And the fuzzing framework fuzzes application layer protocols carried by the TCP/IP protocol.

Figure 3.8 Structure of fuzzing framework for fuzzing EMS system super server

The fuzzing framework runs on the super server along with the EMS applications. The fuzzing framework is composed by five components: MITM server; fuzzing engine;
The applications running on the super server are the targets of all fuzzing tests. The applications are configured to listen on ports of the local network interface (127.0.0.1) of the super server. The client (10.0.0.1) in Figure 3.8 (outside the boundary of super server) is configured to communicate with the super server over the local area network at 10.0.0.2. However, rather than using a sniffer as with the fuzzing framework for the IEEE C37.118 protocol, the MITM server is a multi-threaded process that runs on the super server and listens to TCP ports associated with IP address of 10.0.0.2. When the client sends requests to 10.0.0.2, one of the threads of MITM server captures these packets and forwards them to the application that is listening to the destination port. The MITM server then will act like a proxy for relaying traffic from the client to appropriate application on the super server. The fuzzer ZZUF mutates packets captured by the MITM server and ZUFF is controlled by the fuzzing engine according to some algorithm. In the engine configuration file, parameters for ZZUF are specified such as start seed and maximum seed. A debug agent with local access to the application source code is attached to the application process to detect when an exception is raised. The debug agent uses GDB. When GDB detects a program exit number other than 9 (which is normal exit). It records the debugging information to the log and also communicates with the fuzzing engine to record the packet along with seed number and fuzzing range.

3.5 Developing Snort rules for detecting attacks

Snort rules are capable of tracking the number of packets from a given source in a specified time period. Such Snort rules can alert if a flooding attack is detected. A SYN flood rule for an IEEE C37.118 interface should take into account the normal and
extraordinary, yet still valid, volumes of traffic expected on the network interface. In normal operation, using IEEE C37.118, the PDC sends commands to the PMU to request a configuration file. The PMU responds with a configuration file and then begins to stream synchrophasor measurements, data packets, at 30, 60, or 120 packets per second. This process should generate one TCP session and therefore only one TCP SYN packet should be sent per synchrophasor session. A PDC may connect to multiple PMU and therefore may have multiple active TCP sessions on port 4712, the port assigned for IEEE C37.118. PMU and PDC also commonly have other TCP services. Each open port of the tested PMU and PDC was tested with TCP SYN flood attacks. In all case devices were 100% responsive to TCP SYN floods of less than or equal to 1000 packets per second. The two rules below detect TCP SYN flood attacks against any port on PMU or PDC. The rules alert for more than 1000 TCP packets in one second. This threshold value can likely be significantly decreased without causing spurious alerts.

```
alert tcp any any -> $PDCIP any (msg:"Syn Flood to PDC"; flags:S,CE; flow:to_server; threshold: type threshold, track by_src, count 1000, seconds 1; priority:3; sid:1000001;)

alert tcp any any -> $PMUIP any (msg:"Syn Flood to PMU"; flags:S,CE; flow:to_server; threshold: type threshold, track by_src, count 1000, seconds 1; priority:3; sid:1000002;)
```
Flooding attacks performed in device testing included ARP floods, IP floods, TCP SYN floods, TCP SYN FIN floods, UDP floods, ICMP floods. In each case SNORT rules can be derived to detect the floods.

Protocol mutation testing was performed with the MU-4000. Protocol mutation, also known as fuzzing, checks device response to broken protocol rules. Protocol mutation can be performed at any network layer. In this section we provide MODBUS/TCP and IEEE C37.118 protocol mutation examples.

One MODBUS/TCP device tested reset itself when the LENGTH field of was less than the actual length remainder of the MODBUS/TCP packet. The rule below confirms that the specified bytes remaining are actually in the packet. This rule was taken from a rule set developed by Digital Bond [57].

```
alert tcp $MODBUS_SERVER 502 <> $MODBUS_CLIENT any
(flow:established;\
byte_jump:2,4; isdataat:0,relative; msg:"SCADA_IDS: Modbus TCP - \ 
Incorrect Packet Length, Possible DOS Attack"; \reference:url,digitalbond.com/tools/quickdraw/Modbus-tcp-rules; \classtype:non-standard-protocol; sid: 1000003; rev:1; priority:2;)
```

Because much of this work was done under confidentiality agreement, other Snort rules written were not included in this chapter as they would indirectly divulge the
vulnerabilities identified in testing. In addition, a set of Snort rules are developed for malicious C37.118 packets as in [58], which were validated by aforementioned fuzzer.

### 3.6 Conclusion

In this chapter, cyber security testing methodologies for synchrophasor system are provided. Results of the testing were reported to the sponsoring electric utilities and to the hardware and software vendors to aid in understanding the impacts of cyber-attacks to the synchrophasor system and assist vendors and the utility to deploy defense mechanisms. Two fuzzing frameworks for network protocol and distributed computing systems were developed to identify vulnerabilities of synchrophasor devices and the energy management system. Different fuzzing methodologies were used depending on different characteristics of the target under test. The fuzzing frame work for IEEE C37.118 protocol can be used as an attack tool that modifies measurements carried by IEEE C37.118 data frames. Additionally, the fuzzer for IEEE C37.118 data frames can be easily changed to modify any value in C37.118 data frames but make the data frames still compliant with the protocol specification. Such frames therefore may not be detected by a traditional IDS such as Snort.
CHAPTER IV
DETECTION FOR FAULT AND CYBER ATTACK IN POWER SYSTEM BY MINING SYNCHROPHASOR DATA

4.1 Introduction

Situational awareness technologies have been studied and continuously improved for decades. The need to continue situational awareness improvements is motivated by recent power disturbances which have led to large scale blackouts [59]. A power system disturbance, such as a transmission line fault, can initiate a chain of reactions which lead to a cascading blackout if timely actions from operators are not taken. Poor visibility across the power system may also cause the significance of an event to be misunderstood and lead to incorrect control actions by operators in control centers. Additionally, as power systems increasingly depend on communication infrastructures to provide the wide-area monitoring and control, power systems are exposed to the threat of cyber-attacks. Cyber-attacks are another form of power system contingency. Attacks that target power systems can exploit vulnerabilities in control devices and communication links to corrupt the control and measurement signals [7][60], and interrupt monitoring algorithms [67]. Cyber-attacks which corrupt control and measurement signals can be disguised as power system disturbances or control actions. Situational awareness technologies are needed which distinguish between actual power system disturbances related to natural events, and cyber-attacks. The emphasis of this work is not on classifying disturbance
types as quite a number of methods have been proposed to do so in the power system, but on distinguishing between disturbances and cyber-attacks. There are three reasons it is important to distinguish disturbances from cyber-attacks. First, in the case that a cyber-attack impersonates a disturbance or control action, proper classification will lead to proper response. Classifying a cyber-attack as a disturbance or control action can lead to improper response and cause an outage or other negative impact on the power system. Conversely, incorrectly classifying a disturbance or control action as a cyber-attack can lead to improper response within the information and communications technology (ICT) system. Second, a single classifier which identifies all types of power system contingencies is needed as an input to automated event response algorithms such as autonomic management frameworks, system integrity protection schemes (SIPS) [13], and wide area protection systems (WAPS) [12]. This work presents a methodology to mine the patterns for disturbances and cyber-attacks using a two-dimensional graph from logged heterogenous system data, use common paths in the graph as signatures of each type of modeled scenario, and finally, to classify specific disturbances and cyber-attacks. For proof of concept, in this work we consider disturbances as different types of line-to-ground and line-to-line faults.

A new trend in power system situation awareness is the use of high-speed and time-synchronized data. Compared to traditional supervisory control and data acquisition (SCADA) systems that poll field sensors once per several seconds, synchrophasor systems allow measurement of up to 120 samples per second. Synchrophasor systems provide measurements such as voltage, current, and frequency. Synchrophasor data was used in this work for two reasons. First, the mining common path algorithm uses a set of system
states in temporal order as a signature for each observed event type. High frequency synchrophasor measurements enable identification of fast moving power system events. Some power system events involve very fast changing behaviors and may last only a few milliseconds [61]. For example, zone 1 faults are typically set to be cleared instantly. The presence of a fault and system response of opening the breaker to clear the fault take just a few cycles. These events can be missed by slower speed measurement systems. Second, synchrophasor systems provide more accurate system state visibility due to the use of time synchronized measurements. The mining common paths algorithm can leverage this improved visibility to track events related to a single event from multiple synchronized sensors. The high measurement frequency and time-synchronized characteristic offered by synchrophasor systems create very large volumes of data and enable various applications including Wide Area Monitoring Systems (WAMS), Wide Area Protection Schemes (WAPS), and System Integrity Protection Schemes (SIPS) [12][13][11]. However, using synchrophasor data alone is not enough to detect cyber-attacks. Such example can be a cyber-attack that mimics a real fault by first injecting false measurements then tripping the relay. The status of other power system components such as relays and breakers is also available as time-synchronized data via synchrophasor systems [11]. Combining synchrophasor data with other system logs such as relay status log and network event monitor logs can extend the situational awareness capabilities provided by a synchrophasor system to detect cyber-attacks. But, this creates the challenge of how heterogeneous data sources can be merged to train and use such a classifier. This work provides a solution to this problem by proposing a data mining approach that leverages the time-stamped data to extract temporal patterns which can be
used to describe system behavior related to disturbances, control actions, and cyber-attacks. Henceforth, disturbances, control actions, and cyber-attacks are collectively referred to as scenarios.

In this work, a pattern for a scenario is presented as a common path that consists of a sequence of system states in temporal order. A system state in a common path is made up of multiple instantaneous readings from available sensors from the system. One advantage of the common path is that it overcomes the difficulty in analyzing time domain waveforms by discovering the critical system states across very short time intervals (in milliseconds). These common paths are mapped into a state machine with two-dimensional coordinates for the scenario classification. The automatic process of discovering common paths is introduced by using a case study in a simulated 3-bus 2-line transmission system. For this work, a case study is provided which considers disturbances including symmetric and unsymmetric faults and different cyber-attacks that mimic the 1LG fault to confuse operators in the control center. The cyber-attacks studied in this work belong to masquerading and/or man-in-the-middle (MITM) attacks that target physical devices such as PMU and relays. These attacks may originate from a compromised node in control center, sending control commands or measurement packets covered by legitimate source IP addresses and legal packet formats. As such, it is assumed the masquerading packets cannot be detected by traditional network intrusion detection systems. Validation of the mining common paths algorithm is based on simulated data because actual synchrophasor data is not available for researchers due to the proprietary nature of data, confidentiality issues, and lack of proper sharing mechanism among researchers and institutes. Additionally, data sets captured from
utilities contain a limited number of scenarios. This limits diversity in the data set. Some power system scenarios are rare, especially cyber-attacks, hardware-in-the-loop (HIL) simulation allows targeted dataset creation realistic scenarios captured from the same commercial devices found in utilities. The same data sets used in this work has also been used in [28] for synchrophasor data mining research.

This work has three primary contributions. First, this work demonstrates a new classifier capable of distinguishing power system disturbances and cyber security attacks that interrupt power system control actions and mimic real disturbances. Second, we use the sequential pattern mining algorithm to mine fused heterogeneous data and create common paths for each known scenario. Third, power systems are dynamic in nature which leads to minor variations in system state for known scenarios. The classifier presented in this work learns by parsing datasets marked with scenario type. The training process results in an ordered sequence of system states, i.e. a path, representing each unique instance of a scenario found in the dataset. To avoid overfitting the mining common path algorithm was developed to discover critical states shared by similar paths representing the same scenario. The result of the common path algorithm is a merged set of paths representing all scenarios in the dataset. The classifier matches monitored state transition patterns to common paths of known scenarios to provide a specific classification of the observed behavior.

The remainder of this work is organized as follows. Section 4.2 discusses the methodology, the process of mining common paths, and the classifier training and validation phases. Section 4.3 introduces the case study test bed, test data, and test data
preprocessing procedure. Section 4.4 presents the classification results of three experiments. Section 4.5 concludes this work and proposes future work.

4.2 Mining common path

4.2.1 Sequential events for a power system scenario

Figure 4.1 Ideal vs. actual SLG fault and protection system response

Power system scenarios can be described as an ordered sequence of measureable events. For example, Figure 4.1 depicts phase \(a\) current magnitude during a single-line-to-ground (1LG) fault on a transmission line. The current magnitude can be quantized into 3 ranges; high, normal, and low which are represented by dark grey, white, and light grey shading on Figure 4.1. When the system is in a stable state, the current stays in the normal range, marked as node A in Figure 4.1. When the 1LG fault occurs, current increases to the high range via node B. The protection scheme will operate two relays, R1 and R2, at the both ends of the transmission line to open breakers and isolate the fault. Current magnitude then drops through node C to zero. If following six notations are used
to denote six events: “$I_{R_1}=H$” as node “B”, meaning “Current measured by $R_1$ increases to High”; “$I_{R_2}=H$” for “Current measured by $R_2$ increases to High”; “$R_1=$Trip” for “$R_1$ trips”; “$R_2=$Trip” for “$R_2$ trips”; “$I_{R_1}=0$” as node “C” for “Current measured by $R_1$ drops to Zero”; “$I_{R_2}=0$” for “Current measured by $R_2$ drops to Zero”.

The timestamps of 1LG fault and resulting protection scheme operation can be represented by expression (1) where $t(\cdot)$ stands for the timestamp of corresponding events.

$$t(I_{R_1}=H) = t(I_{R_2}=H) < t(R_1=$Trip$) = t(R_2=$Trip$) < t(I_{R_1}=0) = t(R_2=0)$$

Expression (1) assumes a fault which appears at both relays at the same time and assumes both relays operate at the same time. In fact, the fault may occur at different locations along the line leading to variations in the time each relay observes the fault and variations in relay operation time. Power systems are dynamic. In Figure 4.1, the dashed line shows an ideal waveform of current magnitude during a fault and the solid line graphs a waveform captured from Real Time Digital Simulator (RTDS) simulation of a 1LG fault. The actual waveform includes multiple variations from the ideal waveform. A power system’s response to load variation, fault location variation, and transient behaviors results in irregular waveforms. Such variations are reflected as dispersions in the timestamps of node B and node C for different instances of the same scenario. The dispersion in timestamps can be seen not only in the events related to the current magnitude but also events related to other features. Figure 4.2 shows box plots of timestamps of six events for three fault scenarios and one scenario where relays $R_1$ and $R_2$ are tripped by attackers. The Figure 4.2 x-axis is the set of observed events. The box
plots represent 40 instances of each scenario. To provide an ordered sequence the time stamp of the first event in a sequence was subtracted from timestamps of all later events in the sequence. The box plots and the interconnecting edges of a scenario are depicted using the same color. As shown in Figure 4.2, events take place in temporal order. Event timestamps vary due to system dynamics. For each scenario, a track can be drawn by connecting box plot medians. The tracks shown in Figure 4.2 generally agree with expression 1. Expert knowledge can be used to create similar expressions for all known system behaviors. However, time variation prevents these from serving as signatures for classification. This leads to the need for a graph to describe an ordered set of events describing a scenario while comprehending the variation in times stamps.

Figure 4.2 Distribution of timestamps for events

Tracks are an ordered list of events with measurements where each vertex is an event measured at a single sensor. The classifier presented in this work uses paths which
are an ordered list of system states where a *state* is snapshot of measurements from all available sensors at a given time instant. The steps taken to convert heterogeneous data collected during a scenario into a path will be introduced in next section. Path vertices are states and path edges are transitions between states. Paths are a means for providing stateful monitoring of the system. The training process performed to create paths is subject to over-fitting due to the time variations seen in Figure 4.2. In the over-fitting case, different instances of the same scenario may have different paths. A technique for mining common paths is provided below to identify shared critical states between a set of paths for a scenario leaving a *common path* which comprehends the variation in timestamps.

4.2.2 Mining common paths algorithm

Describing the Mining Common Paths algorithm requires definitions of the concepts of state, feature, sequence, and path.

A state is used to represent a system’s instantaneous status. A state consists of a set of observed system measurements or features $f$ as well as a normalized time stamp $TS$, i.e. $S = \{TS, f_1, \ldots, f_n\}$. The value of a feature is read from a sensor. The possible values for a feature are in a range called its domain. A feature that has continuous values in its domain should be discretized to finite ranges to avoid an infinite state space.

A path $P$ is a list of observed system states arranged in temporal order according to their timestamps, namely $P = \{S_1, S_2, \ldots, S_n\}$, ordered by increasing time. A sequence $s$ is a subset of a path, i.e. $s \subseteq P$. We denote a sequence $s$ by $\{S_{i+1}, S_{i+2}, \ldots, S_{i+m}\}$. A path $P$ contains sequence $s$ if all of the elements in $s$ appear in $P$ in the same order. In a set of sequences, a sequence is maximal if the sequence is not contained in any other sequences.
Let $G$ be the set of all observed paths for a scenario $Q$ so $G = \{P_1, P_2, \ldots, P_n\}$ where $n$ is the number of observed paths for $Q$. A path supports sequence $s$ if the sequence is contained in the path. Support can be defined as a metric in which the support of sequence $s$ is the percentage of paths in $G$ that contain sequence $s$.

A common path for scenario $Q$ is any sequence whose support is greater than a minimum support threshold and is maximal. There may be multiple common paths for a single scenario. Common paths reflect the states that occur most frequently for a scenario. The process of mining common path is similar to mining frequent sequence patterns as defined in [31].

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>S1</td>
<td>S2</td>
<td>S3</td>
<td>S4</td>
<td>S5</td>
<td>Ideal Case</td>
</tr>
<tr>
<td>P2</td>
<td>S1</td>
<td></td>
<td>S2</td>
<td>S3</td>
<td>S4</td>
<td>S5</td>
</tr>
<tr>
<td>P3</td>
<td>S1</td>
<td>S10</td>
<td>S2</td>
<td>S3</td>
<td>S4</td>
<td>S5</td>
</tr>
<tr>
<td>P4</td>
<td>S1</td>
<td>S11</td>
<td>S3</td>
<td>S4</td>
<td>S5</td>
<td>Modified States</td>
</tr>
<tr>
<td>P5</td>
<td>S21</td>
<td>S22</td>
<td>S23</td>
<td>S24</td>
<td>S25</td>
<td>Error Path</td>
</tr>
</tbody>
</table>

**Example**  
Consider the set of paths shown in Table 1. For the example $G = \{P1, P2, P3, P4, P5\}$. If the minimum support threshold is set to 60%, the set of sequences in $G$ which meet the minimum support threshold includes $\{S_1, S_2, S_3, S_4, S_5\}$ and $\{S_1, S_3, S_4, S_5\}$. For this example, $\{S_1, S_2, S_3, S_4, S_5\}$ is maximal and is therefore the common path. The sequence $\{S_1, S_3, S_4, S_5\}$ is not maximal because it is contained in $\{S_1, S_2, S_3, S_4, S_5\}$. If the minimum support threshold is changed to 70%, the set of
sequences in \( G \) which meet the minimum support threshold includes only \( \{ S_1, S_3, S_4, S_5 \} \). Since only one sequence meets the threshold it is maximal and is a common path.

Table 4.1 also provides examples of types of paths. P1 represents the ideal case for a path representing a scenario. P2 matches P1 except a subset of states are delayed. This may occur due to timestamp variation of events or due to system dynamics. P3 contains an extra state. Dynamics may occur when a feature oscillates during a state transition. P4 represents the case when a path is similar but a state is different from the ideal case. This could happen when an event in a state (i.e. \( S_2 \)) does not occur due to the variation in the timestamp, which results in a different state (i.e. \( S_{11} \)). P5 represents an error path. In the error path no sequences match the ultimate common path.

The common path is used as a signature during classification. Changing the minimum support threshold changes the number of states in a common path and can affect classification accuracy. It is not necessary to find a common path which matches the ideal path, rather the goal is to find a common path which is unique for a scenario and which leads to maximum classification accuracy. For a noisy system a shorter common path may yield better classification results.

The common path must contain critical states about a group of paths, \( G \). For example, a common path for a 1LG fault should have a sequence of critical states representing “current going high”, “relay trip” and “current falling to zero.” The ability to find a common path is greatly dependent on the “quality” of paths in \( G \). For example; if there are a many error paths in \( G \) it will be difficult to find sequences which meet the minimum support threshold.
The rest of this work presents a case study which applies the mining common path algorithm to a 3-bus 2-line transmission system for classifying four types of power system symmetric and unsymmetrical faults and three cyber-attacks scenarios.

4.3 **Power system test bed**

A real world power system is dynamic and consists of thousands of buses, loads, transmission lines, and other components. The power system operation goes through various states and is a continuous process. The 3-bus 2-line transmission system used in this work is modified from the IEEE 9-bus 3-generator system [62] according to our simulation requirements. Although this system is relatively small it captures the essence of the larger power system and is small enough to be comprehensible in every detail. This system uses commercial PMU and relays from different major vendors. The test bed and data sets exhibit behaviors of a real power system, yet fit into the resources available in the lab in terms of hardware and software limitations. The transmission system used for HIL simulation for this work is shown in Figure 4.3.

4.3.1 **Power system scenarios**

The power system disturbances and three types of attacks simulated for this work are described as follows.

4.3.1.1 **Power System Faults**

In this work we consider symmetric and unsymmetrical faults in a power system as the examples of disturbances. A power system fault is a condition where the system voltage, current and frequency are abnormal. Typically, single line to ground (1LG) faults, double lines to ground (2LG) faults, three lines to ground (3LG) faults and line to
line (LL) faults represent greater than 95% of faults in a power system [63]. In this work, for proof of concept we simulated phase-a-to-ground fault for 1LG faults, phase-\(a-b\)-to-ground faults for 2LG faults, phase \(a-b-c\)-to-ground fault for (3LG) faults, and phase-\(a\)-to-\(b\) line to line fault for LL faults.

### 4.3.1.2 Trip command injection attack

Trip command injection attacks create contingencies by remotely sending unexpected relay trip commands from an attacker’s computer to relays at the ends of a transmission line. The trip command injection attack used for this work closely mimics the 1LG fault. The attack was implemented against relay R1 and R2 by replaying relay trip commands captured from MODBUS/TCP network traffic. However, we assume these commands are sent from a compromised legitimate computer such that these commands cannot be detected by network event monitor (e.g. Snort) as attacks since they are from a valid source and have valid formats. The two relay trip commands open the breakers at the ends of transmission line L1. This attack stresses the system by forcing L2 to carry more power flow which may cause cascading failures in a power system. However, for this work, cascading failures were not simulated. The trip command injection attack instances were created under random load conditions in the same range used for faults.

### 4.3.1.3 Aurora attack

The Aurora vulnerability refers to potential harm caused to a generator by intentionally opening and closing a breaker near the generator in rapid succession [71]. In this work, an aurora cyber-attack was simulated which periodically sends opening-
closing commands to relays that cause the breaker on the transmission line to open and close at a very fast pace.

### 4.3.1.4 1LG fault replay attack

The 1LG fault replay attack attempts to emulate a valid fault by altering system measurements to mimic a 1LG fault followed by sending an illicit trip command from a compromised computer to relays at the ends of the transmission line. This attack may lead to confusion and potentially cause an operator to take invalid control actions. A Python script is used to initiate a Man-in-the-middle attack between the hardware PDC and the historian that replays synchrophasor measurements from a valid 1LG fault then replays commands to trip the relays on the affected line.

### 4.3.2 Test bed architecture

![Diagram of 3-bus 2-line transmission system for case study](image)

Figure 4.3  3-bus 2-line transmission system for case study
The HIL test bed shown in Figure 4.4 was used to simulate the distance protection scheme on the 3-bus 2-line transmission system and implement the faults and cyber-attacks scenarios. The RTDS was used to simulate transmission lines, breakers, generators, and load. Four physical relays were wired to the RTDS in a HIL configuration. The relays implemented a two-zone distance protection scheme. The relays trip and open the breakers once a fault occurs on a transmission line. Fault logic for different types of faults were created in RSCAD then the faults were implemented in the RTDS. Prior to each implementation of a fault, the system load was randomized in the range of 200-399MW. Each fault instance was implemented at a random location in 1% increments from 10% to 90% of line L1.

The relays used in this work are the GE-D60 and SEL-421. Both are digital relays with integrated PMU functionality. However, PMUs and relays were drawn separately in Figure 4.4. The PMUs stream real-time synchrophasor measurement data, using the IEEE C37.118 protocol at a rate of 120 samples per second, to the PDC. Then aggregated
synchrophasor data is forwarded to the OpenPDC software. A python script processes the synchrophasor measurement data received by OpenPDC into a comma separated values format (CSV) file for each instance of a scenario. A row in the CSV file includes readings of frequency, current phasors, voltage phasors, and sequence components from the four PMUs, and a timestamp. Each CSV file is labeled with the instance number, scenario name, as well as load ranges and/or fault location at the moment the instance of the scenario occurs. The label is useful for grouping instances as will be discussed in Section 4.4. The label is also used for training and classifier testing. The four relays were sources of time stamped relay state changes. There is also a network event monitor that logs any trip command packets sending to relays. All logs and synchrophasor measurement CSV files were stored in a historian. The details of this test bed can be found in [81] [82].

For this work, simulation of all scenarios starts from a stable state and ends at a stable state. Faults last for one second and the relay closes the breaker 2 seconds after opening. Also, the distance protection scheme was simplified by disabling reverse time delay backup and limiting the number of protection zones for each relay to 2. Each relay provides primary protection up to 80% of the line (Zone 1 protection) and backup protection (Zone 2 protection) up to 150% of the line. The trip time for Zone 1 protection is set to instantaneous while the trip time for the Zone 2 protection is set to 20 cycles.

4.3.3 Test Data and Data Preprocessing

In total 1,023 instances of 1LG faults, 274 instances of 2LG faults, 584 instances of 3LG faults, 272 instances of LL faults, 274 instances of command injection attacks, 225 instances of aurora attack, and 703 instances of 1LG fault replay attack were
simulated. Test data consists of the synchrophasor measurement CSV files, the four relay logs and network event monitor logs collected during all of these scenarios. One relay log is extracted from one of the relays, containing timestamp and corresponding events (trip or not trip). Network event monitor log contains timestamp and corresponding network events (trip command seen or not seen). Each CSV file contains tuples with 52 synchrophasor measurements as each PMU provides 13 measurements including voltage and current phasor magnitude ($V_a$, $V_b$, $V_c$ and $I_a$, $I_b$, $I_c$), zero, positive and negative sequence voltage and current phasor magnitude ($V_0$, $V_1$, $V_2$ and $I_0$, $I_1$, $I_2$) and apparent line impedance ($Z$). A single CSV file has approximately 2,000 tuples for an instance of a single scenario. Since the PMU stream at 120 samples per second, 2,000 tuples corresponds to 17 seconds of simulated system time per scenario. The test data was separated into training and testing data sets, each of which was the input to training and testing phases of the classifier described in the previous section. The data preprocessing step in the training and testing phases converts a data set into paths. This preprocessing process constitutes following steps.

Step 1: Feature selection. Rather than using all recorded input features from the dataset, only a portion of measurements was retained as selected features. In this work, the selected features contain relay status and the three phases current magnitudes ($I_a$, $I_b$, $I_c$). Relay status was used as features because all cyber-attacks studied in this work maliciously trip relays via the network. The network event monitor log was selected as one of the features for the same reason. The three phase current magnitudes were selected because the current magnitudes of the three phases were the most significant
measurements during symmetric and unsymmetrical faults. Other unselected measurements were discarded from the input data.

Step 2: Quantizing features. Each feature was first quantized into finite ranges. The quantization of features requires an expert’s domain knowledge. Continuous features such as phase current for this case study were quantized into nominal ranges to create a finite state space. The phase currents were quantized into low, normal, and high ranges. The low range was 0-99 Amperes (A). The normal range was 100-1199 A. The high range was greater than 1200 A. The relay status was quantized into two values; tripped and not tripped. The network event monitor log was also quantized into two values; trip command seen and trip command not seen.

Step 3: Merge quantized features into a measured events database. The measurement data from the PMU and relay log were merged into a single measured events database for one instance of a scenario. The PMU current magnitude measurements were measured at 120 samples per second while relay status was updated only on a relay state change. To merge the features phase current was chosen as a reference and the relay status was up sampled prior to merging into the measured events database.

The aggregated features with their quantized values in a single row of the measured events database describe the system state at a given timestamp. A system state thus is a vector of timestamps and features with quantized measurements. An example of such state that describes relay R1 and R2 tripping due to high current magnitude can be represented as a vector \{Timestamp, IR1 = High, IR2 = High, R1 = Trip, R2 = Trip, \ldots\}, where “IR1 = High” and “IR2 = High” in the vector represent high current magnitudes.
measured by PMUs in R1 and R2. “R1 = Trip” and “R2 = Trip” in the vector represent relay trip status of the two relays. Note that there will be other features with quantized values in the vector but they are not displayed in this example. The time difference between two states is same as that between two rows in the measured events database, which is the reciprocal of the synchrophasor measurement rate; 1/120 samples per second = 8.33 milliseconds (ms). The timestamps of rows in the measured events database are normalized by subtracting the time of the first row from all other rows. This causes all measured events databases to start from time 0.

Step 4: Mining paths from measured events databases. A path is mined from a measured events database by first merging contiguous rows with unchanged state in the measured events database. The remaining rows contain unique states. A state space database is used to track the unique states. Each unique state is given a state identifier (S_id). Rows are updated to include the state identifier of the system state with the time stamp of the state.

Paths are an ordered list of states. Each instance of a scenario will have a path. Many instances of a scenario will have unique paths due to system and measurement dynamics. The paths mined from training data sets over fit the actual system behavior they are intended to model. If raw paths are used for classification the classifier accuracy will be low. Common paths are needed which represent the scenario across all variations found in the training data set.

4.4 Evaluation

Three experiments were performed to validate the Mining Common Paths algorithm. The first experiment classifies two classes, 1LG fault and command injection
attack. To further stress the algorithm, we design the second experiment which performs classification on different 1LG fault locations classes and command injection attack class. The second experiment uses the same training data set and testing data set as experiment 1, but it requires an extra step in training in which system expertise is used to divide the 1LG fault class into multiple subclasses representing different fault locations. In experiment 3, we tested the algorithm for 4 types of short-circuit faults and 3 types of attacks. Experiment 3 uses 10-round cross validation to validate the correctness of the classifier. Experiment 3 also includes a comparison of classification accuracy using different PMU streaming rates. The training phase and test phase used for each experiment are summarized as follows.

4.4.1 Training phase:

*Input*: Training data set for $N_{\text{training}}$ instances of $M$ classes (each class associates with one scenario)

*Output*: $M$ sets of common paths (cp) for $M$ classes

**Step 1**: Training data set is preprocessed into $N_{\text{training}}$ paths as stated in Section IV.C for $N_{\text{training}}$ instances.

**Step 2**: $N_{\text{training}}$ paths are grouped into $M$ groups for $M$ classes.

**Step 3**: Common paths are computed for each group of paths.

4.4.2 Test phase:

*Input*: Test data set for $N_{\text{test}}$ instances of $M$ classes

*Output*: Classify instance by scenario type
**Step 1:** For each instance in test data set, preprocess instance into one path under test (PUT$_j$).

**Step 2:** Compare the path to all common paths in cp by repeating Step 3 for each cp$_i$ in cp.

**Step 3:** If cp$_i$ $\subseteq$ PUT$_j$ then cp$_i$ is a candidate common path.

**Step 4:** The PUT$_i$ is classified as class of the maximal length candidate common path. If more than one maximal candidate common path are maximal then PUT$_i$ is classified as unknown.

**Step 5:** If PUT$_i$ is classified as none of the known classes, then it is marked as unknown.

**Step 6:** Repeat Step 1 to Step 5 for all N$_{testing}$ instances.

### 4.4.3 Experiment 1

For the first experiment, approximately half of the test data for 1LG fault and command injection attack is randomly chosen as training data set while the rest is used as a testing data set. This resulted in 519 instances of 1LG fault and 127 instances of the command injection attack which were used for training. Table 4.2 is a confusion matrix for experiment 1.

For this work, accuracy, misclassification, and unknown rates were defined as follows. The accuracy rate is the percentage of instances correctly classified. Misclassification rate is the percentage of the instances of a class which were misclassified as another scenario. The unknown rate is percentage of the instances of a scenario which were not classified as any scenario. Unknown instances either match no common paths or match more than one common path from more than one class.
Table 4.2  Confusion matrix for experiment 1

<table>
<thead>
<tr>
<th></th>
<th>Fault</th>
<th>C. Inj.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault</td>
<td>491</td>
<td>0</td>
</tr>
<tr>
<td>C. Inj.</td>
<td>0</td>
<td>123</td>
</tr>
<tr>
<td>Unknown</td>
<td>28</td>
<td>4</td>
</tr>
</tbody>
</table>

For the first experiment, the overall classification accuracy was 95%. No instances were misclassified. A total of 5% of tested scenario instances were unknown. All unknown instances matched at least one fault and at least one command injection common path.

There were a total of 221 common paths found for the two scenarios; 203 for 1LG fault scenario and 18 for the command injection scenario. This high number of paths results from the dynamic nature of the power system. Figure 4.5 is a plot of the fault location, from the perspective of relay R1, versus relay trip times for relays R1 and R2. Figure 4.5 clearly shows zone 1 and zone 2 trip boundaries for both relays. Additionally, Figure 4.5 shows that the relay trip times vary with fault location especially in the fault location region from 24-79% of the transmission line. The large number of common paths for the 1LG fault injection scenario is primarily due to this variation. System behavior also varies as the system load changes. Behavior changes due to system lead to multiple common paths being found for both the 1LG fault and command injection scenarios.
4.4.4 Experiment 2

Ideally, faults between 0-20% of the transmission line should have instant trip time for relay R1 and trip after 20 cycles for relay R2. Faults between 80-100% of the transmission line should trip after 20 cycles for relay R1 and instantly for relay R2. In the 21-79% range both relays should ideally trip instantly. Observed trip times match the ideal case for the 0-20% and 80-100% ranges. Note, the apparent impedance setting for zone 2 for relay R2 causes the zone 1 to zone 2 transition to occur at approximately 23% of the line (77% of the line from relay R2’s perspective) instead of at the expected 20% of the line (80% of the line from relay R2’s perspective).

The trip times from 24-80% of the line are always instantaneous. Observed trip times tended to increase as the fault approached the zone 1 to zone 2 boundary points. To compensate for this observed behavior the 1LG fault paths were grouped by fault location per the following groups; (10-23%, 24-29%, 30-35%, 36-40%, 41-60%, 61-65%, 66-70%, 71-80%, 81-90%). Additionally, it was observed that trip times partially correlated
to the system load. As a result, the 1LG fault class used in experiment 1 is divided into multiple classes by fault location and load. Four load ranges were used; (200-249, 250-399, 300-349, 350-399 MW). This subdivided the 1LG fault class into 9*4 = 36 sub classes.

The command injection attack class in experiment 1 was also divided using 4 load ranges, which results in 4 command injection attack classes.

The extra step of subdividing the 1LG fault class and command injection attack results in a total of 40 classes, i.e. M = 40 in the training phase of the classifier. The training data set and testing data set in this experiment is the same as that used in experiment 1.

Table 4.3  Confusion matrix for experiment 2

<table>
<thead>
<tr>
<th></th>
<th>10-23%</th>
<th>23-29%</th>
<th>30-35%</th>
<th>36-40%</th>
<th>41-60%</th>
<th>61-65%</th>
<th>65-70%</th>
<th>71-80%</th>
<th>81-90%</th>
<th>C. Inj.</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-23%</td>
<td>191</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>23-29%</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30-35%</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>36-40%</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>2</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>41-60%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>41</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>61-65%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>65-70%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>3</td>
<td>14</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>71-80%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>38</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>81-90%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>135</td>
<td>0</td>
</tr>
<tr>
<td>C. Inj.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>127</td>
</tr>
<tr>
<td>Unk. Fault</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Unknown</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.3 is a confusion matrix for all scenarios for experiment 2. As previously mentioned, the 1LG fault classes were divided by fault location and system load. To save space the groups in the confusion matrix were combined to just show the fault location
classes and one command injection class. An extra row (marked Unk. for unknown) was added to the confusion matrix to show instances of scenarios which were not classified.

The experiment 2 classification accuracy, misclassification, and unknown rates can be viewed from multiple perspectives. The overall accuracy rate for the groups shown in the confusion matrix was 87.6%. Misclassification and unknown rates for the same groups were 9.1% and 3.3% respectively. From the confusion matrix the majority of misclassification occurred when 1LG fault groups were classified as members of a neighboring or nearby fault group. The unknown cases are separated into unknown instances which resulted from an instance matching multiple fault common paths (“Unk. Fault” in Table 4.3) and unknown instances which matched no common path. The 16 cases of faults which matched common paths from more than one group all occurred because both the (30-35%) and (36-40%) shared a common path.

The intent of subdividing the 1LG fault class was not to classify 1LG faults by a specific fault location. Correctly classifying a fault as a fault is sufficient as many algorithms are available to provide fault location information. The accuracy rate when the fault location classes were combined into a single class is 96.7%. The misclassification rate was 0% and the unknown rate was 3.3%.

Common paths can be mapped into two-dimensional coordinates with the Y-axis indicating the state identification code (state ID) and the X-axis indicating normalized timestamps. An edge between two vertices represents the temporal transition between two states. Each vertex is marked with state information. shows common paths for two scenarios, a fault in the 36-40% fault location group and a command injection attack. The fault and command injection common paths both start at the system normal state.
These paths differ immediately because for faults the PMU will measure high current when a fault is present. This makes the second state of the fault common path high current detected at relay R1. The command injection attack occurs when there is no fault present. As such, the second state for the command injection attack has normal current at both relays while both relay’s status indicates a trip.

![Diagram of fault versus command injection attack common paths](image)

**Figure 4.6** 2-D coordinates documenting fault versus command injection attack common paths
Figure 4.7 shows common paths for two different 1LG fault locations. Note that not all features are displayed in the graph. The 10-23% fault is in relay R2 zone 2 and the 24-29% fault is in relay R2 zone 1. This difference is the primary reason for different paths for the two fault sub groups.

Figure 4.7 demonstrate that common paths contain the critical states for different scenarios. The primary contribution of the mining common paths algorithm is the ability to create unique paths for each scenario type.

Training and testing processing time and memory usage were measured using an Ubuntu Linux Virtual Machine with 3.5GHZ CPU and 2GB memory. For experiment 1, training required 202 seconds and 25.3 MB memory. Experiment 1 testing required 550 seconds to complete and 25.3 MB of memory. For experiment 2, training required 205
seconds and 25.2 megabytes (MB) memory. Experiment 2 testing required 540 seconds to complete and 25.2 MB of memory.

### 4.4.5 Experiment 3

A third experiment was conducted for classifying 4 types of symmetric and unsymmetrical faults and 3 types of cyber-attacks. The training phase used the same methodology as experiments 1 and 2. Validation in this experiment used 10-round cross validation. In each round, half of the test data was randomly chosen as a training dataset and the remaining data was used as the testing data set. Table 4.4 is a combined confusion matrix for 10 rounds of validation for the 1LG, 2LG, 3LG, LL faults, command injection, Aurora, and fault replay attacks. Each entry in the table sums up numbers for 10 rounds in the corresponding location.

**Table 4.4 Confusion Matrix for 4 types of faults and 3 cyber-attacks**

<table>
<thead>
<tr>
<th></th>
<th>1LG Flt.</th>
<th>2LG Flt.</th>
<th>3LG Flt.</th>
<th>LL Flt.</th>
<th>Cmd. Inj.</th>
<th>Aurora</th>
<th>Flt. Replay</th>
</tr>
</thead>
<tbody>
<tr>
<td>1LG Flt.</td>
<td>5009</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>109</td>
</tr>
<tr>
<td>2LG Flt.</td>
<td>6</td>
<td>1248</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3LG Flt.</td>
<td>86</td>
<td>1248</td>
<td>2905</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>LL Flt.</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>1089</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cmd. Inj.</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>1380</td>
<td>0</td>
<td>0</td>
<td>124</td>
</tr>
<tr>
<td>Aurora</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>971</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Flt. Replay</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3138</td>
<td>0</td>
</tr>
<tr>
<td>Unknown</td>
<td>177</td>
<td>58</td>
<td>15</td>
<td>238</td>
<td>0</td>
<td>159</td>
<td>97</td>
</tr>
</tbody>
</table>

The total number of classifications made in Table 4 is 16,851, of which 15,740 instances are correctly classified. The average accuracy for the seven classes shown in
Table 4 is 93.41%. Only 488 instances of faults (177 of 1LG fault, 58 of 2LG fault, and 15 of 3LG fault and 238 for LL) were classified as unknown. And only 6 instances of faults are misclassified as cyber-attacks. The lowest accuracy for an individual class or scenario type was for fault replay attacks. Fault replay attack classification accuracy was 90%. Fault replay attacks were misclassified as a fault for 3.6% of the tested instances and misclassified as a command injection attack for 3.5% of tested instances. The fault replay attack is intended to mimic a 1LG fault and as such is sometimes able to confuse the classifier. The fault replay includes elements from the command injection attack. This leads to similarities which cause occasional misclassification as a command injection attack. Table 4 demonstrates that the classifier is able to distinguish faults and cyber-attacks.

The accuracy rate for 10-round validation when the PMU is sample rate at 20, 30, 60, and 120 Hertz (Hz) is plotted in Figure 4.8. Classification accuracy is higher when the PMU is streaming at 120 Hz and lowest at 20 Hz. This is reasonable as higher PMU samples rates gives better visibility of the system states when fast moving events, such as faults, are considered.
4.5 Conclusions

The mining common paths algorithm creates common paths from heterogeneous data in the power system. A common path represents a set of critical states that a system will step through in temporal order for a scenario such as a disturbance or a cyber-attack. Common paths can be used as signatures to classify power system behaviors with high specificity. Such a classifier is a useful tool for use with automated system integrity protection systems and wide area control systems which include responses for both natural, equipment failure, and cyber-attack related contingencies.

Simple paths can be derived from monitored instances of scenarios applied to a test bed. However, the transients present in time-domain measurement data lead to different paths for different instances of the same scenario. The mining common paths algorithm uses a sequential pattern mining approach to overcome this challenge and common paths for the scenario.
To validate the correctness of the algorithm, a case study was performed which applied the mining common paths algorithm and classifier to detect disturbances and cyber-attacks. The classifier provides a capability to accurately distinguish between different types of power system faults and cyber-attacks including command injection, aurora attacks and fault replay attacks. Three separate experiments were performed. The first experiment applied the mining common paths algorithm to data with 2 classes; 1LG fault and command injection. The second experiment adds an extra step prior to the training phase where the 1LG fault class is divided into a number of subclasses by taking advantage of power system domain expertise. The extra step of sub-dividing classes in training produces slightly better accuracy, misclassification, and unknown classification. Both experiments required similar training time, testing time, and memory usage. A third experiment was conducted using the same training as experiment 2. Ten round cross validation was performed with varying PMU sample rates. The ten round validation shows the classifier has not overfit the data. Comparison of varying PMU sample rates shows the highest accuracy is achieved with PMU sampled in 120 Hz. This is expected since faults are fast moving events and 120 Hz sample rate provides the most visibility of system state changes.

This work demonstrates a methodology to leverage synchrophasor measurements for power system disturbance and cyber-attack detection and highlights the promise of the mining common paths algorithm.
5.1 Introduction

The next generation power system, also known as smart grid, will rely on advanced technologies such as synchrophasor systems for wide area monitoring and control in order to meet the increasing demand of reliable energy. While in the past, power system components were isolated, they are now interconnected via information infrastructure e.g. Ethernet, and therefore are under the threat of cyber-attacks. Due to the critical role that the power system plays in our society, there is a common agreement that the electric power grid needs to be better secured to ensure continually available power for the nation [1]. There have been multiple documents from different organizations which provide recommendations and guidelines for industry to better secure their facilities[2][3]. However, the United States Government Accountability Office (GAO) has realized that current guidelines are not sufficient to securely implement the smart grid and calls for research and development to improve upon current security mechanisms [4].

Intrusion detection is a process which identifies activities that violate the security policy in a computer system or network. Intrusion detection is a necessary complement to preventive mechanisms such as firewalls because intrusion detection has the ability to detect attacks that exploit system design flaws or bugs and to help people understand the
cause of attacks and thus take proper reactions [7]. The increasing coupling of cyber
infrastructure and physical devices of the smart grid makes a traditional host-based
intrusion detection system (IDS) inadequate because host-based IDS only monitor one
location or one host in the system while power system control algorithms such as the
distance protection scheme usually involve multiple devices at multiple locations.
Therefore, new IDS should have the ability to take multiple data sources into account and
perform stateful monitoring at the system level. Manually building a stateful system level
IDS is a knowledge-intensive task which requires vulnerability analysis and manual
creation of rules and patterns which describe attacks, system specification, or system
normal behaviors. The manual development process results in limited scalability and
updates are slow and expensive.

This chapter documents a systematic and automated approach to building a hybrid
IDS that leverages features of signature-based and specification-based IDS. The IDS
classifies system behaviors over time as specific disturbances, normal control operations,
and cyber-attacks. Sequence of critical states, called common path, provide a
specification or signature for each scenario. A fundamental ingredient of the IDS
presented in this chapter is a data mining technique that aggregates synchrophasor
measurement data and audit logs from multiple system devices to learn the common
paths. The automatic approach eliminates the need to manually analyze and hand-code
patterns and is able to handle very large amounts of data. A case study is included to
demonstrate that the proposed IDS provides high detection accuracy for both known and
unknown scenarios and thus is suitable for a mission-critical environments such as power
systems.
Common paths are signatures of events present in a training database. Common paths are also specifications since they describe expected system behaviors related to known scenarios; normal expected system behaviors and cyber-attacks behaviors. The IDS matches a temporal set of monitored system states to common paths, a signature based technique, to make a classification. Behaviors which do not match a common path are considered unspecified events and are either zero day attacks or unknown system behaviors.

The rest of the chapter is organized as follows. The mining common path algorithm is reviewed in Section 5.2. The overview of the test bed and simulated power system scenarios are presented in Section 5.3. Section 5.4 introduces the procedures to construct the proposed IDS. Experiments and results are discussed in Section 5.5. Conclusions and future work are provided in Section 5.6.

5.2 Mining common paths

A state is used to represent a system’s instantaneous status. A state consists of a set of observed system measurements or features \( f \) as well as a normalized time stamp \( TS \), i.e. \( S = \{TS, f_1, \ldots, f_n\} \). The value of a feature is read from a sensor. The possible values for a feature are in a range called its domain. A feature that has continuous values in its domain should be discretized to finite ranges to avoid an infinite state space.

A path \( P \) is a list of observed system states arranged in temporal order according to their timestamps, namely \( P = \{S_1, S_2, \ldots, S_n\} \), ordered by increasing time. A sequence \( s \) is a subset of a path, i.e. \( s \subseteq P \). We denote a sequence \( s \) by \( \{S_{i+1}, S_{i+2}, \ldots, S_{i+m}\} \). A path \( P \) contains sequence \( s \) if all of the elements in \( s \) appear in \( P \) in the same order. In a set of sequences, a sequence is maximal if the sequence is not contained in any other sequences.
Let \( G \) be the set of all observed paths for a scenario \( Q \) so \( G = \{P_1, P_2, \ldots, P_n\} \) where \( n \) is the number of observed paths for \( Q \). A path supports sequence \( s \) if the sequence is contained in the path. Support can be defined as a metric in which the support of sequence \( s \) is the percentage of paths in \( G \) that contain sequence \( s \).

A common path for scenario \( Q \) is any sequence whose support is greater than a minimum support threshold and is maximal. There may be multiple common paths for a single scenario. Common paths reflect the states that occur most frequently for a scenario. The process of mining common path is similar to mining frequent sequence patterns as defined in [31].

**Table 5.1  Example paths for a scenario**

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>S1</td>
<td>S2</td>
<td>S3</td>
<td>S4</td>
<td>S5</td>
<td>Ideal Case</td>
</tr>
<tr>
<td>P2</td>
<td>S1</td>
<td></td>
<td>S2</td>
<td>S3</td>
<td>S4</td>
<td>S5</td>
</tr>
<tr>
<td>P3</td>
<td>S1</td>
<td>S10</td>
<td>S2</td>
<td>S3</td>
<td>S4</td>
<td>S5</td>
</tr>
<tr>
<td>P4</td>
<td>S1</td>
<td>S11</td>
<td>S12</td>
<td>S4</td>
<td>S5</td>
<td>Modified States</td>
</tr>
<tr>
<td>P5</td>
<td>S21</td>
<td>S22</td>
<td>S23</td>
<td>S24</td>
<td>S25</td>
<td>Error Path</td>
</tr>
</tbody>
</table>

**Example**  Consider the set of paths shown in Table 5.1. For the example \( G = \{P1, P2, P3, P4, P5\} \). If the minimum support threshold is set to 60\%, the set of sequences in \( G \) which meet the minimum support threshold includes \( \{S_1, S_2, S_3, S_4, S_5\} \), \( \{S_1, S_3, S_4, S_5\} \) and \( \{S_1, S_4, S_5\} \). For this example, \( \{S_1, S_2, S_3, S_4, S_5\} \) is maximal and is therefore the common path. The sequences \( \{S_1, S_3, S_4, S_5\} \) and \( \{S_1, S_4, S_5\} \) are not maximal because they are contained in \( \{S_1, S_2, S_3, S_4, S_5\} \). Alternatively, if the minimum  

107
support threshold is changed to 70%, the set of sequences in $G$ which meet the minimum support threshold includes only \{S_1, S_4, S_5\}. Since \{S_1, S_4, S_5\} meets the threshold in this case, it is maximal and is a common path.

Table 5.1 also provides examples of possible types of paths that could be found in the dataset. P1 represents the ideal case for a path representing a scenario. P2 matches P1 except a subset of states are delayed. This may occur due to a measurement error or due to power system dynamics. P3 contains an extra state. Dynamics may occur when a feature oscillates during a state transition. P4 represents the case when a path is similar but a state is different from the ideal case. This could happen when an event that should have occurred at T2 occurs at T3 instead, which mangles states S_2 and S_3 (they change to S_{11}, S_{12}). P5 represents an error path. In the error path no sequences match the ultimate common path.

The common path is used as a specification during classification. Changing the minimum support threshold changes the number of states in a common path and can affect classification accuracy. It is not necessary to find a common path which matches the ideal path, rather the goal is to find a common path which is unique for a scenario and which leads to maximum classification accuracy. For a noisy system a shorter common path may yield better classification results.

A common path for a single line to ground (SLG) fault should have a sequence of critical states representing “current going high”, “relay trip” and “current falling to zero.” The ability to find a common path is greatly dependent on the quality of paths in $G$. For example; if there are a many error paths in $G$ it will be difficult to find sequences which meet the minimum support threshold.
Classification is performed by comparing observed system states to the states of known common paths. The path under test (PUT) is compared to all common paths. If $cpi \subseteq PUT$ then $cpi$ is a candidate common path. The PUT is classified as matching the scenario of the maximal candidate common path from the set of candidate common paths. If more than one candidate common path are maximal the PUT is classified as unknown.

The rest of this chapter presents a case study which applies the mining common path algorithm to a 3-bus 2-line transmission system for classifying 25 power system scenarios.

5.3 Test bed architecture

5.3.1 Distance protection for transmission lines

![Distance protection scheme in a 3-bus 2-line transmission system](image)

Figure 5.1 Distance protection scheme in a 3-bus 2-line transmission system
The distance protection scheme is the most popular scheme for protecting transmission lines. The principle of operation recognizes that the impedance of a high-voltage transmission line is approximately proportional to its length. This means the impedance “seen” by the relay during a fault is proportional to the distance between the point of fault and the relay. Distance relays are encoded with multiple protection zones. Each zone is assigned an apparent impedance threshold and a trip time. Relays have overlapping protection zones to provide system protection redundancy. One relay’s Zone 1 is part of another relay’s Zone 2 and so forth. For this case study, the distance protection scheme was simplified by disabling reverse time delay backup and limiting the number of protection zones for each relay to 2. Figure 5.1 shows a 3-bus 2-line transmission system that is modified from IEEE 4-bus 3-generator system. Relay R1’s zones 1 and 2 are shown as dashed line boxes. Each relay provides primary protection up to 80% of the line (Zone 1 protection) and backup protection (Zone 2 protection) up to 150% of the line in
case that the primary protection fails. The trip time for Zone 1 protection is configured to be instantaneous while the trip time for the Zone 2 protection is time-delayed to avoid false tripping unless the primary relay fails.

5.3.2 Test bed architecture

The hardware-in-the-loop test bed shown in Figure 5.2 was used to simulate the distance protection scheme on the 3-bus 2-line transmission system and to implement 25 power system disturbance, control action, and cyber attack scenarios. The RTDS was used simulate transmission lines, breakers, generators, and load. Four physical relays were wired to the RTDS in a hardware-in-the-loop configuration. The relays implemented the two zone distance protection scheme. The relays trip and open the breakers when a fault occurs on a transmission line. All relays included integrated PMU functionality to measure power system transmission line state; however, the PMU(s) were drawn separately in the graph because relays are controlled by Modbus/TCP and PMUs stream synchrophasor measurements using the IEEE C37.118 protocol. The PMU(s) streamed real-time synchrophasor measurement data at a rate of 120 samples per second, to the PDC which aggregate network frames from multiple PMU and forward the aggregated synchrophasor frames to the OpenPDC software. A python script processes the synchrophasor measurement data received by OpenPDC into a comma separated file. The synchrophasor measurement data includes readings of frequency, current phasors, voltage phasors, and sequence components. The four relays were sources of time stamped relay state changes. The signature-based intrusion detection system Snort runs on a PC to detect network activities. Snort provides alerts when it detects remote tripping command activities in the network. Snort, by itself, cannot distinguish between legitimate and
illegitimate remote trip commands since they appear the same on the network. A control panel computer simulates energy management system (EMS) functionality. The EMS simulation was used to disconnect a transmission line for maintenance by remotely tripping relays via a MODBUS/TCP network packet. An EMS log provides the timestamp of such a line maintenance event. For this work, it is assumed that an attacker computer has successfully penetrated the utility’s operational network and can launch cyber-attacks from a node on the operational network. Scenarios of power system disturbances, normal operations and power system cyber-attacks are applied against the simulated power system and its components. Data logs were captured from the synchrophasor system, relays, Snort, and the simulated EMS. All data logs were time stamped and logged events were labeled with the name of the scenario being simulated.

5.3.3 Test bed scenarios

The power system scenarios used to train and validate the IDS presented in this chapter have been grouped into three categories; power system single-line-to-ground faults, normal operations, and cyber-attacks. Each category is described in this section with details. There are a total 25 scenarios each named with capital “Q” along with a number. The system load was randomized at the beginning of each scenario. Power system SLG faults belong to the shunt fault family and account for up to 70% of faults in a power system [23]. For this work, only phase-a-to-ground faults were simulated as each phase to ground fault has similar characteristics. The phase-a-to-ground fault is abbreviated as “fault” in the rest of this chapter. There are 2 SLG fault scenarios as named Q1 and Q2 that simulate faults on one of the two transmission lines. Each fault instance was implemented at a random location in 1% increments from 10% to 90% of
the corresponding transmission line. Faults were simulated for 1 second after which an automatic reclosing algorithm restored the transmission line to service.

The transmission line maintenance scenario simulates the situation when an operator remotely trips relays to open breakers at both ends of a transmission line to take the line out of service for line maintenance. The operator initiated remote trip commands are recorded and time stamped in the control panel log. Two scenarios of this type, Q5 and Q6, were implemented, one for each of the two transmission lines.

Power system cyber-attacks may originate from insiders, amateur hackers, political activists, criminal organizations, governments, and terrorists. Cyber-attacks may appear as a nuisance or may bring the system to collapse [24]. Attacks can be carried out from within power system substations, control center, transmission and distribution infrastructures by exploiting weaknesses in physical security policies. Alternatively, attacks may take advantage of security flaws and vulnerabilities in software, devices, communication infrastructures, and protocols to electronically infiltrate power system operational networks. Three types of attacks are simulated; relay trip command injection; disabling relay function; SLG fault replay.

Relay trip command injection attacks create contingencies by sending unexpected relay trip commands remotely from an attacker’s computer to the relays at the ends of the two transmission lines. The trip command injection attack used for this work closely mimics the line maintenance scenario. The malicious trip command originates from another node on the communications network with a spoofed legitimate IP address. Since the attack is not from the control panel computer there will not be no record in the control panel log, however, the Snort network traffic monitor will detect this remote trip
command. There are 6 scenarios of this type of attack targeting either one relay (Q7, Q8, Q9, Q10), or two relays at the same time (Q11, Q12).

The disabled relay attack mimics the effects of insiders taking illicit control actions or malware taking control of software systems to manipulate control devices. A python script accesses a relay’s internal registers via MODBUS/TCP commands sent from the attacker’s computer which modify the relevant relay settings. A total of 12 scenarios of this attack are simulated in the test bed. There are 6 scenarios that disable one or two relays which cause the relay not to operate when a valid fault occurs (Q13, Q14, Q15, Q16, Q21, Q22). Another 6 scenarios disable one or two relays to interrupt line maintenance operation by disabling relay tripping function and causing the breakers not to open (Q17, Q18, Q19, Q20, Q23, Q24).

The SLG fault replay attack attempts to emulate a valid fault by altering system measurements followed by sending an illicit trip command to relays at the ends of the transmission line. This attack may lead to confusion and potentially cause an operator to take invalid control actions. A Python script is used to initiate a Man-in-the-middle (MITM) attack between the hardware PDC and the historian computer. The attack replays synchrophasor measurements from a valid single line to ground fault then replays commands to trip the relays on the affected line. There are 2 scenarios of this type of attack simulating relay faults at one of the two transmission lines (Q3 and Q4).

The final scenario, Q25, represents a stable system state. For this scenario the load may change, but, no other attacks, disturbances, or control actions are simulated.
All scenarios start and end with the system in a stable state. As such, all faults are cleared, transmission lines taken out of service for maintenance are returned to service, and all attacks end.

5.3.4 Scenario implementation

Figure 5.3 shows control flow between test bed components. The intention of this design is to simulate large numbers of scenarios in random order with random test bed parameters such as load level and fault locations. The AutoIT script controls different scripts for the implementation of different scenarios as shown in the dash line box in the top right corner.

Figure 5.3 Control flow for automation scenario implementation and data collections
There are a couple of attack scripts that are developed in Python, which implements different cyber-attacks in the test bed. For example, the command injection attack script to remotely trip a relay sends a Modbus trip command to relays. The SLG fault replay attack script sets up a man-in-the-middle attack using Ettercap, then alters PMU data in flight, and finally trips the relays to imitate a legal relay trip.

Line maintenance scenarios are also implemented using Python scripts, however, while sending the trip command to relays, they will also put an entry in the control panel log indicating the legality of this action. Single line to ground faults are simulated using a Matlab routine which triggers a SLG fault on target line locations in RTDS. Either line from Figure 5.1 will be taken out of service.

The disabled relay attack requires coordination between different scripts. First, an attack script sends a “relay function OFF” command via Modbus to the target relay. Second, the Matlab instructs the RTDS to execute the proscribed SLG fault. This coordination is done by the master AutoIT script invoking the two scripts in the specified order.

The master AutoIT script will call each script with randomized parameters in random order at random times. Randomized parameters will include the relay targets, system load level, and fault location where appropriate. The master script will also include relatively long periods of normal operation.

With this test bed, the implementations of a large number of scenarios can be easily scheduled through the AutoIT script. Each implementation is also configured to run with random test bed parameters to simulate real world power system. The parameters are fed to different scripts as arguments. For example, load level and fault
location are two arguments to the Matlab routine, the values of which result in SLG fault at a specified load level and fault location. After each implementation, relay logs, control panel logs and synchrophasor measurement data can be collected from the output of log retrieving script, synchrophasor data retrieving script and control panel script. The data then will be marked with corresponding scenario number.

The test bed also facilitates the implementation of new scenarios. New scenarios can be developed in scripts with interfaces that are callable by the master AutoIT scripts.

5.3.5 Test data

Test data used for this work includes data logs associated with 10,000 simulated instances of the 25 aforementioned scenarios. The data log is a comma separated file with labeled tuples that include 56 sensor measurements and a timestamp. The 56 data sources consist of 52 synchrophasor measurements; 13 from each relay location on Figure 5.1. The synchrophasor data from a single relay consists of phase $a$ voltage and current phasor magnitude ($V_a$, $V_b$, $V_c$ and $I_a$, $I_b$, $I_c$), zero, positive and negative sequence voltage and current phasor magnitude ($V_0$, $V_+$, $V$ and $I_0$, $I_+$, $I$) and apparent line impedance ($Z$). The synchrophasor data was sampled at 120 times per second. Relay status information, breaker events, Snort alerts, and control panel alerts were also logged. All logged data was merged into a single dataset.

An instance of a single scenario is represented by approximately 2,000 tuples in the test data set. This corresponds to approximately 17 seconds of simulated system time per scenario. In total the test data has more than 2 million tuples. Each tuple in the test data is labeled. Approximately half of the test data was used to train the classifier and half was used to test classification accuracy.
For this work, 15 features were used; phase current magnitude measured at each relay, relay status for each relay, snort alert status for each relay, and control panel remote trip status.

5.4 Training the IDS

This section documents the IDS construction process. An overview of the IDS construction process is shown in Figure 5.4. The data formatting step converts input data logs to a measured events database (MED). Next, the specification learning steps process the MED to learn common paths, a unique set of system states in temporal order, for each labeled scenario. Finally, a graph is constructed which includes common paths for all scenarios.

Figure 5.4 Intrusion detection system training process

5.4.1 Data Formatting

The first step of the data formatting process is feature quantization. Feature quantization requires domain expertise. Features with values which can take continuous values are mapped into finite ranges to limit state space size. Features which take discrete values are generally left unchanged unless the number of discrete values is large.
The phase current measurement is a real number and therefore should be grouped into discrete ranges. Phase current magnitude was separated into normal and high ranges. The normal range was 0-1199 Amperes (A). The high range was all values greater than or equal to 1200 A. The relay status, snort alert, and control panel remote trip status features are all binary. Possible relay status values are tripped and not tripped. Possible Snort alert status values are alert and no alert. Possible control panel remote trip status values are tripped and not tripped.

The MED is a merged compressed data set with quantized features. Data from sensors with lower sample rates is up sampled to match the sampling rate of the sensor with the highest sampling rate. The up sampling process depends upon the sensor type. Continuously sampled sensors update their value at each sample period based upon the current measured state. The current magnitude and relay status are continuously sampled. Event based sensors provide a single message when a state change occurs. The snort alert and control panel remote trip status features are event based. For each, when the sensor detects the presence of an event the sensor provides a message indicating the event occurred. In a data log a continuously sampled sensor measurement takes a value and holds that value across multiple samples until the state changes. Conversely, in the data log event based features are asserted for a single sample for each measured event.

When up sampling, continuously sampled sensor measurements are mapped to the nearest sample period after the measurement. All samples without a value take the value nearest proceeding sample. Event based sensor measurements are also mapped to the nearest sample period after the measurement. All samples without a value take the non asserted value. For this work, the current magnitude measurements were measured at 120
samples per second which is the highest sampling rate of all features. Relay status, snort alerts, and control panel log features were up sampled according to the aforementioned procedure.

An MED represents one instance of a scenario. As such, data formatting requires copying each scenario into a separate file. The timestamps of rows in the MED are normalized by subtracting the time of the first row from all other rows. This causes all measured events databases to start from time 0.

### 5.4.2 Creating and grouping paths

#### 5.4.2.1 Creating paths

A path is a list of observed system states arranged in temporal order. Mining paths is performed by down sampling the MED while preserving all state transitions. A state change is a change on any sensor value between two MED samples. The MED is parsed to identify all periods of consistent state. Consistent state periods are down sampled using a user defined sample period. For this work, the sample period was 0.5 seconds. Each unique state is assigned a state identifier (S_id) and all known states are stored in a state data base.

A path is mined for each MED. A single scenario will have many unique paths due to the dynamic nature of power systems, variations in the order of states within a path, and due to variations in event timing. Using the paths derived from the mining paths process for classification results in poor classification accuracy. The mining common paths algorithm is used to shrink the larger group of paths into a representative set of common paths which represent normal variation and serve as a set of signatures for each scenario.
5.4.2.2 Error features

There will be situations where a feature (or more features) used to construct MED has big deviation in its timestamps due to transmission delay or tight computational resources of a computer. This causes the paths created from the MED merging step to be inaccurate. We call such feature “error feature”. Figure 5.5 shows timestamps of three states from left to right that contain event “control panel sending trip command to relay R1” (CP = R1), event “snort detecting trip command to R1” (SNT = R1) and event “current measured by R1 dropping to zero” (IR1 = 0) respectively. The most left box plot shows that feature “control panel log” is an error feature because the timestamps of “CP = R1” present wide dispersion over a number of instances. This will result in the failure in mining a common path. To deal with “error features”, we design an error feature sensor which correct the timestamps of the error feature. The error feature sensor first
excludes the error feature from the MED before merging rows in MED. After merging rows, it adds the feature back to the first state of each path. For example, if a scheduled trip command to R1 is seen in the control panel log for the scenario that the path represents then CP = R1 is added to the first state of the path, otherwise, CP = 0 will be added.

5.4.2.3 Grouping paths

Grouping is an optional step which preprocesses input data to separate large classes into smaller sub-classes. Grouping can lead to more accurate classification when the sub-classes are sufficiently different from one another.

Figure 5.6 clearly shows zone 1 and zone 2 trip boundaries for both relays. Additionally, Figure 5.6 shows that the relay trip times vary with fault location especially in the fault location region from 24-79% of the transmission line. System behavior also varies as the system load changes.

Figure 5.6 Relay trip time versus fault location for relays R1 and R2
Ideally, instances of SLG fault at transmission line L1 scenarios (i.e. Q1) from a two zone distance protection scheme can be separated into 3 groups according to the area of the line in which the fault occurs. Group 1 includes faults from the length of the line which is protected by relay R1’s zone 1 and relay R2’s zone 2. From Figure 5.6, group 1 includes faults which occur between 10-23% of the line. For group 1 faults, relay R1 should trip instantly and R2 should trip after 0.4 seconds. Group 2 includes faults protected by relay R1’s and R2’s zone 1. Both relays should trip instantly for group 2 faults. From Figure 5.6, group 2 faults occur between 24-79% of the line. Group 3 includes faults protected by relay R1’s zone 2 and relay R2’s zone 1. Relay R1 should trip after 20 cycles and R2 should trip instantly for group 3 faults. From Figure 5.6, group 3 faults occur between 80-90% of the line.

Observed trip times in group 2 tended to increase as the fault approached the zone 1 to zone 2 boundary points. To compensate for this observed behavior the SLG fault paths were grouped by fault location per the following groups; (10-23%, 24-29%, 30-35%, 36-40%, 41-60%, 61-65%, 66-70%, 71-80%, 81-90%). Additionally, it was observed that trip times partially correlated to the system load. As a result, the SLG fault paths were grouped by fault location and load. Four load ranges were used; (200-249, 250-399, 300-349, 350-399 MW). This grouping subdivided the SLG fault paths into 9*4 = 36 sub-groups.

5.4.3 Mining common paths

The mining paths step produced 5000 paths from 5000 instances of the 25 scenarios. The mining common path algorithm produced 477 common paths. The minimum and maximum number of common paths for a single scenario was 4 and 53 respectively. The
15 SLG fault scenarios had 421 common paths spread among them. The remaining 10 scenarios had 56 common paths. The large number of common paths for the SGL faults is due to the large variation in relay trip times as fault location and system load varies.

Common paths can be mapped into two-dimensional coordinates with the Y-axis indicating the state identification code (state ID) and the X-axis indicating normalized timestamps. An edge between two vertices represents the temporal transition between two states. Each vertex is marked with state information. Note that only necessary features are displayed to save space. Figure 5.7 shows common paths for two scenarios, a fault in the 36-40% fault location of line L1 and a fault replay attack on line L1. The fault and fault replay paths both start at the system normal state. For real faults, the PMU will measure high current when a fault is present while for a fake fault the attacker injects high current measurement to the PMU. This makes the second state of the both common paths high current detected at relay R1, i.e. IR1 = High. However, these paths differ immediately because for the fault replay, the attacker has to inject relay trip commands to relay R1 and R2 at the same time. As such, the second state for the fault replay attack has the trip commands to R1 and R2 detected by Snort, i.e. SNT = (R1, R2) in Figure 5.7.
Figure 5.7 2-D coordinates documenting fault versus fault replay attack common paths

Figure 5.8 shows common paths for line maintenance and command injection attack scenarios. The primary difference between the two scenarios is the command to open relays R1 and R2 originates from the control panel computer for the line maintenance scenario. This causes the control panel log to include a trip command message. The common path for the line maintenance scenario includes a state noting the detection of control panel log events (i.e. CP = (R1, R2)) and states showing Snort detecting remote trip command network packets (i.e. SNT = (R1, R2)). The common path for command injection includes the Snort alert but excludes the control panel log state.
Figure 5.8 2-D coordinates documenting line maintenance versus command injection attack common paths

Figure 5.7 and Figure 5.8 demonstrate that common paths contain the critical states for different scenarios. The primary contribution of the mining common paths algorithm is the ability to automatically create unique paths for each scenario type from data sets which measure behavior associated with the scenarios.

5.4.4 Evaluation

Three approaches were used to evaluate the IDS. First, the IDS was used to classify 5,000 instances of scenarios from the test data set described in section IV of this chapter. Confusion matrices are provided to show IDS accuracy. A detailed review of the algorithms ability to classify SLG faults by fault location is also provided. Second, training and testing was repeated with sets of 4 scenarios missing from the data set. This test was used to demonstrate the IDS’s ability to detect zero day attacks and unknown
scenarios. Finally, IDS cost and performance was measured by measuring the amount of processing time and memory required during training and evaluation.

Table 5.2  Confusion matrix for scenarios Q1-Q13

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
<th>Q7</th>
<th>Q8</th>
<th>Q9</th>
<th>Q10</th>
<th>Q11</th>
<th>Q12</th>
<th>Q13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>505</td>
<td>0</td>
<td>31</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Q2</td>
<td>0</td>
<td>502</td>
<td>0</td>
<td>34</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Q3</td>
<td>10</td>
<td>0</td>
<td>301</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Q4</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>321</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>Q5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>130</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Q6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>108</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Q7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>67</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Q8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Q9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>99</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Q10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>57</td>
<td>0</td>
</tr>
<tr>
<td>Q11</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>127</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Q12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>104</td>
<td>0</td>
</tr>
<tr>
<td>Q13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>179</td>
<td>0</td>
</tr>
<tr>
<td>Oth.</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Unk</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>35</td>
<td>32</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Unc</td>
<td>0</td>
<td>2</td>
<td>8</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.3  Confusion matrix for scenarios Q14-Q25

<table>
<thead>
<tr>
<th>Q14</th>
<th>Q15</th>
<th>Q16</th>
<th>Q17</th>
<th>Q18</th>
<th>Q19</th>
<th>Q20</th>
<th>Q21</th>
<th>Q22</th>
<th>Q23</th>
<th>Q24</th>
<th>Q25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q14</td>
<td>220</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Q15</td>
<td>0</td>
<td>208</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Q16</td>
<td>0</td>
<td>0</td>
<td>162</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Q17</td>
<td>0</td>
<td>0</td>
<td>40</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Q18</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>73</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Q19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>33</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Q20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>45</td>
<td>0</td>
</tr>
<tr>
<td>Q21</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>424</td>
<td>0</td>
</tr>
<tr>
<td>Q22</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>413</td>
<td>0</td>
</tr>
<tr>
<td>Q23</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>122</td>
</tr>
<tr>
<td>Q24</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>112</td>
</tr>
<tr>
<td>Q25</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>114</td>
</tr>
<tr>
<td>Oth.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Unk</td>
<td>10</td>
<td>12</td>
<td>23</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>6</td>
<td>33</td>
<td>16</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>Unc</td>
<td>37</td>
<td>58</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 5.2 and Table 5.3 provide confusion matrices for the 25 tested scenarios. The confusion matrices were separated into two tables to allow them to fit in the column width of this chapter. The row labeled “Oth” represents scenarios Q14-Q25 in Table 1 and Q1-Q13 in Table 5.2. The row labeled “Unk” provides the number of instances which were unclassified due to no matching common path. Finally, the row labeled “Unc” provides the number of instances with uncertain classification due to matching more than one common path from more than one scenario.

In total, 90.4% of the tested instances were correctly classified and 2.7% of the instances were misclassified. 4.7% of instances were classified as unknown and 2.2% were classified as uncertain. All of the cases of uncertain classification were related to SLG fault instances which matched common path for more than one fault scenario.

Table 5.4 Confusion matrix for sub-groups in scenario Q1

<table>
<thead>
<tr>
<th></th>
<th>10-23%</th>
<th>23-29%</th>
<th>30-35%</th>
<th>36-40%</th>
<th>41-60%</th>
<th>61-65%</th>
<th>65-70%</th>
<th>71-80%</th>
<th>81-90%</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-23%</td>
<td>191</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>23-29%</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30-35%</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>36-40%</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>2</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>41-60%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>41</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>61-65%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>65-70%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>3</td>
<td>14</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>71-80%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>38</td>
<td>18</td>
</tr>
<tr>
<td>81-90%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>135</td>
</tr>
<tr>
<td>Unk. Fault</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Unk</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.4 displays a confusion matrix of classifications of the sub-groups for scenario Q1 which is a SLG fault on line L1. As previously mentioned, the SLG fault paths were grouped by fault location and system load. To save space the groups in the
 confusion matrix were combined to just show the fault location grouping. The row labeled “Unk. Fault” indicates unknown faults, i.e. instances classified as more than one fault. The row labeled “Unk” indicates unknown classification, i.e. instances which could not be classified due to not matching a common path.

The overall accuracy rate for the groups shown in the Table 5.4 was 84.6%. The majority of misclassification occurred when SLG fault groups were classified as members of a neighboring or nearby fault group. The 16 cases of unknown faults all occurred because both the (30-35%) and (36-40%) shared a common path. The intent of the grouping of SLG faults was not to classify SLG faults by a specific fault location. However, Table 5.4 demonstrates the mining common path algorithm’s strength of finding unique paths for even similar scenarios.

Training and classification processing time and memory usage were measured using an Ubuntu Linux Virtual Machine with 3.5GHZ CPU and 2GB memory. Training required 0.33 seconds per scenario instance and 34 MB memory. Classification of test cases required 0.85 seconds per scenario instance to complete and 26.2 MB of memory.

Tenfold cross-validation was used to evaluate the detection accuracy of zero-day attack scenarios as shown in Table 5.4. For each round of testing four scenarios are randomly selected to be excluded from training but present in the testing data set. The average detection accuracy for zero-day attack scenarios was 73.43%. However, there are cases where the detection rate for zero day attack is low. For example, in Round 3, the zero day detection rate was 50.5%. Analysis of this case showed that scenario Q6 (command injection to trip relay R1 and R2) was always misclassified as scenario Q3.
(fault replay attack on Line L1). This is reasonable because the fault replay attack includes injected trip commands targeting relays R1 and R2.

Table 5.5 Detection accuracy for 4 random zero-day attacks 10x validation

<table>
<thead>
<tr>
<th>Round</th>
<th>Excluded Scenarios</th>
<th>Z.D. Acc. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Q3, Q11, Q18, Q22</td>
<td>76.3</td>
</tr>
<tr>
<td>2</td>
<td>Q2, Q8, Q12, Q23</td>
<td>67.3</td>
</tr>
<tr>
<td>3</td>
<td>Q6, Q11, Q16, Q17</td>
<td>50.5</td>
</tr>
<tr>
<td>4</td>
<td>Q1, Q5, Q8, Q10</td>
<td>73.3</td>
</tr>
<tr>
<td>5</td>
<td>Q1, Q9, Q19, Q21</td>
<td>91.8</td>
</tr>
<tr>
<td>6</td>
<td>Q5, Q13, Q20, Q23</td>
<td>64.7</td>
</tr>
<tr>
<td>7</td>
<td>Q5, Q10, Q15, Q16</td>
<td>63.8</td>
</tr>
<tr>
<td>8</td>
<td>Q12, Q13, Q19, Q24</td>
<td>70.7</td>
</tr>
<tr>
<td>9</td>
<td>Q2, Q7, Q9, Q17</td>
<td>76.3</td>
</tr>
<tr>
<td>10</td>
<td>Q9, Q10, Q16, Q19</td>
<td>99.8</td>
</tr>
</tbody>
</table>

5.5 Conclusion

The IDS described in this chapter provides stateful monitoring of an electric transmission distance protection system by leveraging a fusion of synchrophasor data and information from relay, network security logs, and energy management system logs.

The IDS is trained using a mining common paths algorithm. Common paths are hybrid signatures and specifications which described patterns of system behavior associated with power system events. The algorithm provides a time-domain data
analysis approach to overcome transients present in the measurements. This is done by
mining shared states out of a group of observed paths. Common paths are used to
describe system responses to power system disturbances, control actions, and cyber-
attacks.

The IDS matches monitored system state traversal to common paths to make
classification decisions. Classification is specific to each trained scenario rather than
simply an indication of normal or abnormal activity.

In this work the IDS was trained and evaluated for a 3-bus 2-line transmission
system which implements a 2 zone distance protection scheme. Twenty five scenarios
consisting of SLG faults, control actions, and cyber-attacks were implemented on a
hardware-in-the-loop test bed. Scenarios were run in a loop 10,000 times with
randomized system parameters to create a dataset for IDS training and evaluation. The
IDS correctly classified 90.4\% of tested scenario instances. Evaluation also included a
tenfold cross-validation to evaluate the detection accuracy of zero-day attack scenarios.
The average detection accuracy for zero-day attack scenarios was 73.43\%. The
performance of the proposed IDS has outscored that in [43] in average detection accuracy
and accuracy of zero-day attacks.
CHAPTER VI
CONCLUSION AND FUTURE WORKS

6.1 Conclusion

Synchrophasor systems are an emerging technology. Prior to installation of a synchrophasor system a set of cyber security requirements must be developed, new devices must undergo vulnerability testing, and proper security controls must be designed to protect the synchrophasor system from unauthorized access.

In this dissertation we described the process used to develop a set of cyber security requirements in the design stage of a synchrophasor project. A set of cyber security rules was derived from review of the NISTIR 7628 Guidelines for Smart Grid Cyber Security, DHS Security Procurement Language for Control Systems, and from utility internal requirements. Next, the dissertation discussed a cybersecurity vulnerability analysis and testing process. The testing process included network congestion and protocol mutation testing of multiple phasor measurement units and phasor data concentrators. The testing section provides limited results due to confidentiality agreements and ethical reporting requirements. The testing section also discussed shortcomings of the fuzzing tool used and described a network fuzzing framework that is capable of fuzzing server to client interactions, client to server packet contents, and system state. Next, the dissertation discussed the process of reviewing synchrophasor system components against the drafted cyber security requirements. Each requirement
was discussed in the context of the synchrophasor system and recommendations were provided for meeting requirements. This dissertation also provides discussion on writing SNORT intrusion detection rules based upon the results of cyber security testing.

While fixing the identified vulnerabilities in information infrastructure is imperative to a secure power system, it is likely that successful intrusions will still occur. The ability to detect intrusions is necessary to mitigate the negative effects from successful attacks. This dissertation proposed a data mining algorithm called the mining common path algorithm to learn patterns from data for different power system scenarios. The mining common paths algorithm creates common paths which represent a set of critical states that a system will step through in temporal order for a power system scenario. The algorithm provides a time-domain data analysis approach to overcome transients present in the measurements. This is done by mining shared states out of a group of observed paths. The resulting classifier matches monitored system state traversal to common paths to make classification decisions. In effect, the mining common path algorithm creates a set of temporal signatures for a system which describe scenarios. This approach presented is applicable to many types of industrial control systems where control algorithms are well known and cyber attack impacts can be examined on a system test bed.

Multiple case studies were performed to validate the mining common paths algorithm and the IDS which used common paths created by the mining common paths algorithm.

First, to demonstrate the ability of the mining common paths algorithm to learn common paths from complex power system data, a case study was performed to
demonstrate the ability to learn common paths for and distinguish SLG ground faults and
cmd injection attacks which remotely trip relays on a 3-bus 2-line transmission
system. The experiment applied the mining common paths algorithm to data with 2
classes; faults and command injection. This experiment demonstrated the ability of the
mining common paths algorithm to find common paths for both types of scenarios, power
system events and cyber-attacks. Furthermore, the experiment demonstrated the
precisions of the mining common paths algorithm by showing that faults grouped by
location and system load could be accurately distinguished from one another using
common paths created by the algorithm.

Second, an IDS prototype was built that employs common paths to detect power
system disturbances, control actions, and cyber-attacks on a larger scale. The IDS used
stateful monitoring of the electric transmission distance protection system by leveraging a
fusion of synchrophasor data and information from relay, network security logs, and
energy management system logs. The IDS matches monitored system state traversal to
common paths to make classification decisions. Classification was specific to each
trained scenario rather than simply an indication of normal or abnormal activity.

The IDS was trained an evaluated for a 3-bus 2-line transmission system which
implemented a 2 zone distance protection scheme. Twenty five scenarios consisting of
SLG faults, control actions, and cyber-attacks were implemented on a hardware-in-the-
loop test bed. Scenarios were run in a loop 10,000 times with randomized system
parameters to create a dataset for IDS training and evaluation. The IDS correctly
classified 90.4% of tested scenario instances. Evaluation also included a tenfold cross-
validation to evaluate the detection accuracy of zero-day attack scenarios. The average detection accuracy for zero-day attack scenarios was 73.43%.

Table 6.1 summarizes how the intrusion detection system proposed in this dissertation fulfills the seven requirements introduced in Chapter I.

Table 6.1  Fulfillment of seven requirements for the proposed IDS

<table>
<thead>
<tr>
<th>Req. #</th>
<th>Req. Description</th>
<th>How the proposed IDS meets requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The IDS should provide stateful monitoring.</td>
<td>This requirement is met by monitoring the system according to a state machine in 2-D graph. The state machine contains a number of common paths which represent the patterns of power system scenarios and cyber-attacks. Common paths are mined from data collected across the system.</td>
</tr>
<tr>
<td>2</td>
<td>The IDS should be able to detect power system disturbances, normal control operations and cyber-attacks</td>
<td>25 scenarios were created for power system disturbances, normal control operations and cyber-attacks. The IDS creates a set of unique common paths for each scenario and makes classification according to these common paths.</td>
</tr>
<tr>
<td>Req. #</td>
<td>Req. Description</td>
<td>How the proposed IDS meets requirements</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>3</td>
<td>The IDS should be able to detect zero day attacks</td>
<td>Test cases that have patterns that do not match any of the common paths trained in the IDS are classified as zero day attacks. The experiment in Section 5.5 showed that the average detection rate for four unknown scenarios is 73.43%. This is higher than what was reported in [43].</td>
</tr>
<tr>
<td>4</td>
<td>The IDS should be able to process large amount of data</td>
<td>This is done by a data formatting process where a large amount of synchrophasor data and logs are aggregated and processed into a special form called path. A path is a compact representation of large volume of data as it only consists of a sequence of system states. The mining common paths algorithm is further used to shrink the larger group of paths into a representative set of common paths which represent normal variation and serve as a set of signatures for each scenario.</td>
</tr>
</tbody>
</table>
Table 6.1 (Continued)

<table>
<thead>
<tr>
<th>Req. #</th>
<th>Req. Description</th>
<th>How the proposed IDS meets requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>The Development of IDS should have low cost.</td>
<td>The IDS construction process from data formatting to generating common paths can be automated by software. Although expert knowledge is still needed for feature quantization and may be used for path grouping, this is huge progress with comparison to [27] where rules are created manually using expertise.</td>
</tr>
<tr>
<td>6</td>
<td>The IDS should be able to withstand continuous changes to system configuration and load level changes.</td>
<td>Results show the IDS can detect changes in configuration which result from operators taking a line out of service or a relay operating after detecting a fault by learning these patterns from collected data. New changes in configuration will result in new scenarios. Since the IDS learns patterns from data, new scenarios will be simulated in the test bed and relevant data will be collected to train the IDS. The IDS overcomes changes in load level by grouping paths according to load level ranges before the common paths learning process.</td>
</tr>
</tbody>
</table>
Table 6.1 (Continued)

<table>
<thead>
<tr>
<th>Req. #</th>
<th>Req. Description</th>
<th>How the proposed IDS meets requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>The IDS should have high detection accuracy and a low false positive rate.</td>
<td>The evaluation results in Section 5.5 has shown that for classifying 25 scenarios 90.4% of the tested instances were correctly classified and 2.7% of the instances were misclassified. 4.7% of instances were classified as unknown and 2.2% were classified as uncertain.</td>
</tr>
</tbody>
</table>

6.2 Future Works

This dissertation enables multiple possible future. First, the vulnerability assessment process presented in this work can be applied to other emerging cyber-physical systems such as advanced metering infrastructure (AMI) and other industrial control systems. As to different cyber-physical systems, different sets of penetration tests can be developed based on the unique structure of the target system and the control and protection algorithms available in the system in order to study the attack consequences on corresponding systems. The fuzzing framework proposed in this work can also be extended to test other industrial protocols such as MODBUS and ANSI C12.22 for AMI. While penetration testing is a practical method to assess a target system’s security features, the other method of performing vulnerability assessment is through theoretical methodologies such as Bayesian network and attack tree formulation. The theoretical method can be applied to cyber-physical systems to derive quantitative vulnerability measures in order to determine how likely a vulnerability can be exploited.
Second, we have shown that the proposed IDS works for a 3-bus and 2-line transmission system implemented distance protection scheme. However, a realistic power system usually contains thousands of buses and transmission lines. Training the IDS for such a large system will raise other questions, for example, is data from all transmission line required for training? Are all features needed? While PMU are not available for each transmission line, what will be the optimal locations to place PMU in order to collect the data needed for training the IDS?

Third, an immediate next step for the IDS will be to implement the IDS in real-time. This requires the IDS to be able to process a continuous stream of synchrophasor data. In this dissertation, for proof of concept the proposed IDS is trained and tested with off-line data. To achieve real-time implementation, two methods can be used. In the first method, the synchrophasor data stream can be first buffered in a historian. Then the common paths of different scenarios can be learned from historian data off-line using mining common paths algorithm but the detection can be implemented in real time to match the system states to the common paths. The second method is to implement the learning process in real-time as well. In this case, the continuous data stream may be windowed to learn paths. The mining common path algorithm might need to be updated so that the known patterns can be refined based on the new path learned from the coming data in a window. The potential problems associated with this method are related to the definition of the size for a window and the start of the window. The start of the window determines the start state of a path. In this dissertation, a path starts from the system normal state. But in a more complex environment where the occurrence of scenarios might overlap, the start of a window may not be easily found. Also, other data mining
methods such as stream data mining may be worth investigating to learn patterns from a continuous data stream for the purpose of intrusion detection.

Forth, the future plan for intrusion detection will include using pattern recognition techniques to detect anomalies [76][77][78][79]. In addition, the causal event graph theory can be used to develop the intrusion detection for power system.
REFERENCES


