Online Intrusion Detection Design and Implementation for SCADA Networks

by

Hongrui Wang
B.Sc., Southeast University, 2014

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

Online Intrusion Detection Design and Implementation for SCADA Networks

by

Hongrui Wang
B.Sc., Southeast University, 2014

Supervisory Committee

Dr. Xiaodai Dong, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Tao Lu, Co-Supervisor
(Department of Electrical and Computer Engineering)

Dr. Issa Traore, Departmental Member
(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. Xiaodai Dong, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Tao Lu, Co-Supervisor
(Department of Electrical and Computer Engineering)

Dr. Issa Traore, Departmental Member
(Department of Electrical and Computer Engineering)

ABSTRACT

The standardization and interconnection of supervisory control and data acquisition (SCADA) systems has exposed the systems to cyber attacks. To improve the security of the SCADA systems, intrusion detection system (IDS) design is an effective method. However, traditional IDS design in the industrial networks mainly exploits the predefined rules, which needs to be complemented and developed to adapt to the big data scenario. Therefore, this thesis aims to design an anomaly-based novel hierarchical online intrusion detection system (HOIDS) for SCADA networks based on machine learning algorithms theoretically and implement the theoretical idea of the anomaly-based intrusion detection on a testbed. The theoretical design of HOIDS by utilizing the server-client topology while keeping clients distributed for global protection, high detection rate is achieved with minimum network impact. We implement accurate models of normal-abnormal binary detection and multi-attack identification based on logistic regression and quasi-Newton optimization algorithm using the Broyden-Fletcher-Goldfarb-Shanno approach. The detection system is capable of accelerating detection by information gain based feature selection or principle component analysis based dimension reduction. By evaluating our system using the KDD99 dataset and the industrial control system datasets, we demonstrate that our

design is highly scalable, efficient and cost effective for securing SCADA infrastructures. Besides the theoretical IDS design, a testbed is modified and implemented for SCADA network security research. It simulates the working environment of SCADA systems with the functions of data collection and analysis for intrusion detection. The testbed is implemented to be more flexible and extensible compared to the existing related work on the testbeds. In the testbed, Bro network analyzer is introduced to support the research of anomaly-based intrusion detection. The procedures of both signature-based intrusion detection and anomaly-based intrusion detection using Bro analyzer are also presented. Besides, a generic Linux-based host is used as the container of different network functions and a human machine interface (HMI) together with the supervising network is set up to simulate the control center. The testbed does not implement a large number of traffic generation methods, but still provides useful examples of generating normal and abnormal traffic. Besides, the testbed can be modified or expanded in the future work about SCADA network security.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor Professor Xiaodai Dong for her patient and insightful academic guidance and support throughout my study and research in the past two years. She is such a wonderful and great professor that I learnt a lot from her, not only her profound knowledge and rigorous research attitude, but also her excellent personality, which will be beneficial for my whole career and life. I am also very grateful to my Co-Supervisor Professor Tao Lu for his inspired guidance and numerous constructive suggestions during my research. I would also like to thank Professor Issa Traore for his helpful materials and suggestions for my network security research, Professor Dongming Wang and Professor Wei Xu, who guided me in my undergraduate research and encourages me during my graduate study. I would also like to express my gratitude to Professor Wu-Sheng Lu, Professor Lin Cai, Professor Hongchuan Yang for their great courses.

I am also very grateful to our research collaborators Peixue Li and Michael Xie for their help and constructive suggestions. I would also like to express my gratitude to all the members of our research group, whom I have learnt a lot from and spent my happy time with. In particular, I would like to thank Tong Xue, Zheng Xu, Ping Cheng, Yiming Huo, Le Liang, Ming Lei, Farnoosh Talaei, Yongyu Dai, Leyuan Pan, Weiheng Ni, Yuejiao Hui, Lan Xu. I would also like to thank my dear friends Wenyan Yu, Han Xiao, Fei Tong, Hoang Minh Tu, Yue Xu, Ziye Wang, Ming Chen, Qi Chen, Ting Mao, Haiying Chen, Shuai He, Maryam Tanha, Jie Chen, Haoyuan Zhang, Zhe Wei, Yue Li, Jiayi Chen, Yuanzhi Ni and all my friends in Arista Networks for their wonderful accompany.

I acknowledge the Natural Sciences and Engineering Research Council of Canada and the University of Victoria Graduate Awards program for providing financial support for my Master Studies.

Hongrui Wang

DEDICATION

Dedicated to my family and friends.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Historically, industrial control systems (ICSs) are proprietary, making massive attacks against them virtually impossible [48]. The standardization of these systems and increasing adoptions of common communication protocols such as TCP/IP, Modbus and DNP3, and common operating systems such as Windows and LINUX, despite of obvious advantages on performance improvement and cost reduction, have made the was-secretive information easier to access. Consequently, modern ICSs are more vulnerable to attacks [64]. This is of particular concern to the supervisory control and data acquisition (SCADA) system [57], one of the most important ICSs that typically includes large-scale processes in multiple sites to control critical infrastructures such as smart grid, oil and gas pipelines, water and sewage systems, etc.

A typical SCADA system is presented as Fig. 1.1, which is the Fig. 2 from the work [16]. We can see that in a typical SCADA system, there is control centre and multiple remote sites and the control center is connected to Internet with the protection of firewall. The control centre network and the remote site network are also named supervising network and control network [57]. In the supervising network, generally there are computers, web server, human machine interfaces (HMIs), historians, engineering workstations(EWSs), application servers, etc. The supervisory operation of SCADA system is by HMIs. HMIs are responsible for presenting the simulated diagram with real time process states and control values, performing notifications and alarms, and also controlling the process by setting control variables. HMIs obtain all process data from the historians, which is a software application

within HMI responsible for collecting the real time process data. In the control networks, there are programmable logic controllers (PLCs), remote terminal units (RTUs), intelligent electronic devices (IEDs) and some other embedded machines. PLCs are industrial computers, responsible for controlling the manufacturing processes with simple programming. Specifically, PLCs get the information from sensors including states and values, and make changes to control the process based on the preset programming logic. Mostly, PLCs work together with HMIs, which can get the information from PLCs and control PLCs to manage the manufacturing processes. RTUs are microprocessor-based electrical devices, responsible for transmitting the data in a remote system to a RTU master system wirelessly, and receiving the messages from the RTU master system to control the connected sensors or processes. IEDs are also microprocessor-based electrical devices for power equipment, receiving data from sensors and sending commands to control the equipment such as an intelligent circuit breaker able to break the circuit when the current is anomalous. Generally, the communication protocols between the supervising network and control network include Modbus protocol [5, 20], Distributed Network Protocol (DNP3), etc. This thesis focuses on Modbus protocol. Modbus protocol is one of the serial fieldbus protocols used in industrial control system. To accommodate to the TCP/IP network, Modbus/TCP, one of the Modbus variant, has been widely used nowadays. All requests launched by Modbus clients are sent to Modbus servers via TCP/IP network on port 502. In the control system, Modbus clients and servers are often named Modbus masters and slaves, respectively. Modbus protocol is vulnerable to attacks since the implementation of the protocol does not authenticate the messages between the Modbus master and slave devices, which can be exploited to send any control commands to the slaves by any Modbus master.

Breaches of SCADA systems lead to disastrous consequences. In the past, industrial control systems were mostly physically isolated from Internet, however, since the development of Internet, they have been connected to Internet for convenience and efficiency. For example, it was stated in [47] that "Modern control centers have data servers, Human Machine Interface (HMI) stations and other servers to aid the operators in the overall management of the factory network. This SCADA network is usually connected to the outside corporate network and/or the Internet through specialized gateway". The interconnections between the SCADA and external networks such as the Internet and corporate networks further expose SCADA to large-scale cyber security threats. One of the typical examples is the Stuxnet worm [50] discovered

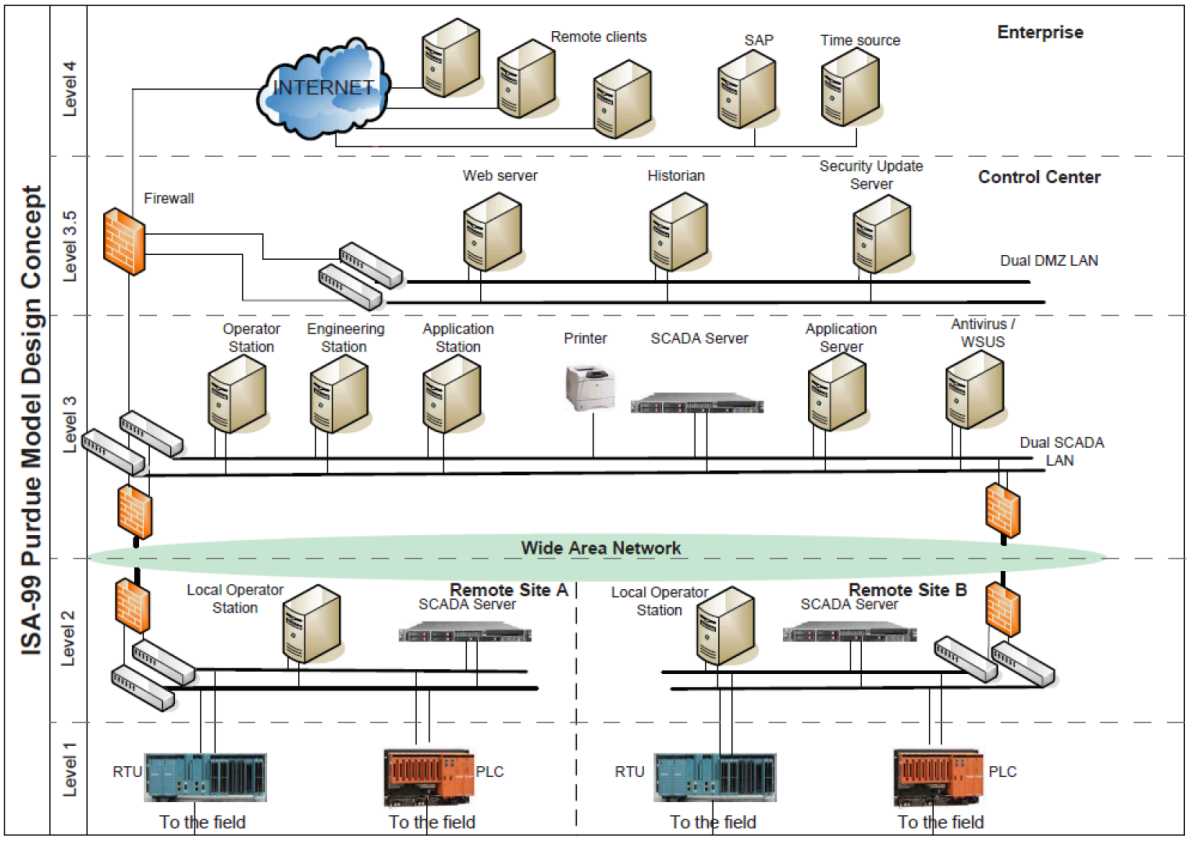

Figure 2 SCADA Network, in accordance with ISA-99

Figure 1.1: HOIDS schemes for SCADA systems(Fig. 2 in [16])

in 2010, which infected more than 100, 000 computers worldwide and damaged almost one-fifth of nuclear centrifuges in Iran by exploiting four zero-day vulnerabilities of Windows systems.

To protect conventional information technology (IT) networks from cyber attacks, implementing intrusion detection systems (IDSs) has been a common practice [29]. Specifically, an IDS is a software application responsible for monitoring the network and reporting or alarming the abnormal network activity to an administrator. IDSs are usually signature-based and/or statistical anomaly-based. Signature-based techniques are highly efficient as they identify the malicious data flow through whitelists or rules generated from the characteristics of known attacks while anomaly-based IDSs spot intrusions by comparing data flow with a principle generated by statistic algorithms [60]. Anomaly-based IDSs are comparatively less efficient as they adopt complex algorithms and consume substantial computing resources. However, a ma-

jor advantage over signature-based IDSs is that they can detect unknown attacks or mutants of known attacks.

Anomaly-based IDSs mostly exploit machine learning classifiers for detecting anomalies. Machine learning classifiers can be classified as supervised learning and unsupervised learning. Unsupervised learning is beyond the scope of the thesis. Hereby, the classifiers discussed in the thesis are supervised ones. Supervised learning often applies the training dataset with multiple features and one class label to train the learning algorithm and obtain the classification principle. Nowadays, broadly used classifiers [26] include Naive Bayes, Logistic Regression, neural networks and Support vector machine(SVM). These four classifiers have their own advantages and often have different performances for different datasets. Here, these classifiers are to be introduced briefly. Naive Bayes is based on the model of simplified Bayesian probability model, which is under the strong assumption of independence among all the features, and constructs the classifier by identifying some class with the highest model probability. The model of Naive Bayes is very simple and often has satisfactory performance for many datasets. But one of its disadvantages is the assumption of independent features, which decreases its popularity to some extent. Logistic Regression is also a probabilistic classifier, but based on logistic function [4] to obtain the classification principle by maximizing the conditional probability of realizing the matching relationship between class labels and data features, which is to be introduced in detail in Chapter 2.2.1. Logistic Regression realizes the classification without the limitations for the correlation between features, and can easily update the principle when incorporating new data. Therefore, it is a popular method in both theoretical research and industrial area. Neural networks classifier is based on a large number of neural units often distributed in multiple layers. Each unit takes inputs, applies a principle to inputs and passes the outputs to the next layer. SVM is based on a hyperplane or sets of hyperplanes in a high-dimensional space to classify data that are hard to be correctly identified in a low-dimensional space. Both neural networks and SVM have high accuracy towards many classification problems, but one disadvantage is time-consuming in both training and detection processes.

In supervised learning, there is often a testing dataset for verifying the performance of the generated classification principle. When there is no testing dataset, validation and cross validation [26] are often useful and reliable methods for estimating the performance of classifying the testing data or real-world data. Specifically, validation is to split the whole training dataset into two parts, the training set and

the validation set, and estimate the classification performance on the validation set using the principle generated by the training set. Cross validation (CV) is to split the training dataset into multiple equal-sized parts, and average the performances of using one part as validation set and the remaining as the training data each time. For example, 10-fold CV, the most widely used validation method, splits the training dataset into 10 equal-sized parts. Validation and cross validation can also be applied to model selection. In terms of classification performance, overfitting is a common issue that needs to be addressed, which refers to a classification principle that models the training set too well, but negatively impacts the performance on the testing data or new data. To limit overfitting, using validation to select a proper model is often used. As a contrast to the concept of overfitting, underfitting refers to a principle that can neither model the training set or the testing set well, often not discussed since it is easy to be identified when bad performance occurs on the training set.

To improve the learning performance, feature selection is often used to simplify the model to be more easily interpreted or to shorten the training and testing time or to limit overfitting to create a more accurate model. The method by calculating the correlation coefficients among all the features is a commonly used way to reduce the redundancy by removing some features with high correlation coefficients. One disadvantage of this method is that there is a requirement of numerical features for the calculation. Another effective feature selection method is based on information gain, which is to be introduced in detail in Chapter 2.2.2. Normally, a feature with high information gain is more important to the classification than those with low information gain. A subset of features with high information gain can accelerate the learning and detection process.

1.2 Related Work

Existing intrusion detection modules for conventional IT networks cannot be directly exploited in SCADA networks due to different network characteristics and system requirements [75]. For example, SCADA systems emphasize real-time processing and many SCADA devices have limited computing abilities. The growing awareness of SCADA security has motivated researches on SCADA-specific IDS. Among them, many SCADA IDSs are signature-based to accommodate the strict real-time constraint and less computationally powerful devices in the networks. Cheung *et al.* [39] developed three techniques for their model-based detection specific to SCADA sys-

tems: protocol-level models, communication-pattern-based detections and a learning-based heuristic detection technique, and further explored the effectiveness by monitoring the networks of Modbus/TCP protocol and detecting the corresponding attacks. Oman *et al.* [58] presented the implementation of a SCADA testbed with the improved ability of overall auditing and detecting abnormal access by monitoring the significant commands and the settings of the SCADA devices. Carcano *et al.* [37] first proposed an intrusion detection method for the SCADA system based on the analysis of multidimensional critical states and state proximity, and implemented the approach on the test scenario of PLCs of the ABB AC800 family and the Modbus over TCP protocol. Ten *et al.* [68] studied an intrusion detection and correlation algorithm specific to SCADA substations networks and an impact evaluation method based on the detected anomalies, and evaluated their design based on the modified IEEE 118-bus system. Goldenberg *et al.* [43] presented an IDS specific for Modbus/TCP systems based on deterministic finite automaton approach, and evaluated the approach on a production Modbus system with high sensitivity and low false positive rate. The deficiency is that they didn't generate malicious attacks for model validation due to the live production system. Yang et al. [71] constructed a rule-based IDS including signature-based and model-based approaches targeting the IEC 60870-5-104 protocol for SCADA networks. In subsequent work, they presented a multi-attribute SCADA-specific intrusion detection system for power networks, comprising access-control whitelists, protocol-based whitelists and behavior-based rules to secure the whole network [72], and tested the approach on a practical grid-connected photovoltaic SCADA system.

On the other hand, we should note that many attacks are either unknown in prior or mutants of their original form. Under these circumstances, anomaly-based IDSs, particularly using machine learning algorithms are advantageous. Yang *et al.* [70] applied an autoassociative kernel regression model along with the statistical probability ratio test and demonstrated the effectiveness of their design on a simulated SCADA system for intrusion detection. Their training dataset consisted of $1,000$ observations, the network traffic statistics of which are from Simple Network Management Protocol. Linda *et al.* [52] proposed an IDS based on two neural network learning algorithms: the Error-Back Propagation and Levenberg-Marquardt, and tested their model using datasets generated by software tools such as Nmap, Nessus and Metasploit. Valdes *et al.* [69] investigated two anomaly detection techniques: pattern-based detection for communication patterns among hosts, and flow-based detection for individual traffic

flows, and showed that their methods are capable of identifying basic attacks against the Modbus servers in their distributed control systems testbed. Zhang *et al.* [74] exploited the support vector machine (SVM) technique and artificial immune system tested on the NSL-KDD dataset to evaluate their distributed intrusion detection system for the multi-layer network architecture of smart grid and related SCADA systems. Maglaras *et al.* [54] presented their SCADA intrusion detection module based on One-Class SVM, training the network data offline with a dataset of $1,570$ packets. Brushi *et al.* [66] explored an estimation-inspection algorithm using logistic regression, and evaluated their design on the testbed of Linux based PLCs, generating a high detection probability with a zero false positive rate.

The growing awareness of SCADA security has motivated researches on improving the security of SCADA systems, not only on theoretical analysis, but also on practical experimental exploration. There are some works introducing the practical setup for their testbeds which simulates SCADA system. The Idaho National Labs (INL) National SCADA Testbed Program [25] set up a large scale electric power grid with firewalls and virtual private networks (VPN) for the network security research of the industrial control system, especially for the cyber security assessments of the control systems in the industry or the government and help them to improve the system security. The testbed designed by [62] combined the network software simulation with real devices such as sensors and actuators. The real devices were used to be attached to the software simulator for observing and studying the hacking effects on the network. The Virtual Power System Testbed (VPST) [31] was set up in University of Illinois at Urbana-Champaign, combining software simulation and real elements as well, for the purpose of being integrated with other testbeds such as Virtual Control System Environment Project (VCSE) [53], which is a project for incorporating all the tools for SCADA system simulation, to study the SCADA security. The European CRitical UTility InfrastructurAL Resilience (CRUTIAL) project [40] [41] established two different testbeds, one of which was to evaluate the protocol IEC 60870-5-104 traffic in electric power grid, and the other one of which was set up by connecting the IED devices to a Matlab/Simulink system for controlling and observing. The paper work in [44] made an overview of the testbeds designed for smart grid security research, in which the necessary components for composing a realistic control system evironment were introduced in detail. The paper work in [42] introduced a laboratory using some real devices from an electric power generation system for exploring the SCADA network security. And Hahn *el al.* [45] implemented a real SCADA testbed

for security research, which provided a learning environment for students as well. The British Columbia Institute of Technology (BCIT) [18] owns a SCADA testbed laboratory named Industrial Instrumentation Process. The laboratory includes several real control systems such as a batch pulp digester, a chemical/blending process and so on for control and security research.

In the work of [73], a tank system, working as a testbed for control systems, was simulated, communicating with the Modbus/TCP protocol. Some attacks and defence toolkits were introduced to emulate the attack and defence in the tank system. The report also showed that PLCs and sensors could be simulated by software to reflect a real industrial production network. Specifically, extended from the tank system in the MBLogic project [8], several components of the testbed system were simulated including the sensors of the tank system, a PLC and a PLC user interface (which was referred as HMI in [73]) on it for the control and monitor of PLC, and the Modbus communication between them. An open source software Honeyd [6] was used to create several fake PLCs by using Nova [6] to configure Honeyd. Some attacking toolkits were discussed and used in the testbed to emulating the hacking behaviours, which included Kali, Nexpose and Samurai. As for the defence of the testbed, an open source software Honeywall [7] including Snort and iptables was used to fulfill the functions of intrusion prevention and firewall. For the network of the testbed system, three types of network were created, which are the external attack network, internal control network and administration network, while the internal network and the external attack network are connected in a bridging mode by Honeywall. Also note that the intrusion detection used in this control system testbed is the signature-based technique.

1.3 Motivation

SCADA systems have the following properties. Firstly, they belong to cyber-physical systems [65], which operate real-time with low tolerance on packets delay. Secondly, frequent patching and updating for SCADA intrusion detection modules are unfavourable due to the inflexibility of the infrastructure and the potential negative impact to the whole work process. Thirdly, a high proportion of SCADA devices have limited computing abilities for implementing sophisticated intrusion detection modules. Fourthly, SCADA networks consist of supervisory and control subnetworks. Each sub-network has different characteristics. The hybrid nature of SCADA net-

works leads to some distinguished characteristics. In particular, the features of field network flows are simpler and more stable, making complex IDS unnecessary. Under these considerations, we design a novel, highly scalable hierarchical online intrusion detection system (HOIDS) for SCADA networks based on machine learning algorithms. HOIDS is uniquely designed to satisfy the real-time requirement in control systems by utilizing an IDS server-client topology where clients distributed at fields perform intrusion detection using the learning principles generated by a central IDS server. This is in sharp contrast to some existing work where IDS is independently implemented at each node in the network and there are no interactions among the IDS modules. By selecting the effective data features based on information gain or reducing the dimension of the feature set, the implementation of IDS clients can be simplified to accommodate the SCADA devices without significant impact on the detection accuracy. HOIDS is also flexible to adjust the detection principles for clients based on practical requirements to improve security.

Besides the theoretical IDS design for SCADA security, to study the SCADA system in a lab environment with most environmental variables controllable, building a testbed that can simulate the operations, physics and network communications in the industrial system is very important and meaningful. Various attempts have been made for practical testbeds simulating SCADA system [18, 25, 31, 40–42, 44, 45, 62]. However, they are either not being open sourced, or not being able to provide the components or simulation details related to our intrusion detection experiments. Consequently, we design and implement a testbed for SCADA network security research. Extended from the control system testbed [73] introduced above, the work in Chapter 4 presented in this thesis aims at a testbed for studying anomaly-based intrusion detection, and with improvements on practicality and extensibility in terms of simulating the SCADA system. Therefore, it makes differences as follows. To reflect a practical SCADA network architecture, the networks involved in the testbed include the control network (for the field devices), which was the internal network in the previous testbed, and the supervising network (for the control center), which is added for simulating a comprehensive SCADA system. An HMI is created in the supervising network to simulate the function of the HMI and historian in the control center, which communicates with PLCs. Note that the HMI in [73] was for controlling one PLC, but the HMI created in the supervising network, a typical representative device in the control center, is able to control all the PLCs, and also for collecting the normal dataflow for studying the anomaly-based intrusion detection. Besides, a generic Linux

system is introduced as a middle box between networks to emulate the network functions, such as routing, IDS, network address translation (NAT) and firewall, through which a higher extensibility for future experiments is achieved. Note that although static routing is used in the settings discussed this thesis, other network functions or approaches can be easily deployed in this Linux-based network middle box, such as the bridge mode. Most importantly, Bro is introduced and discussed to fulfill the IDS functions, of which the script language brings the support to more sophisticated traffic monitoring and analysis, important for the research on anomaly-based intrusion detection. The traffic from the HMI to the PLCs and the attacks to PLCs will go through the middle box, and therefore can be used for normal and abnormal analysis. In another word, based on the control system testbed in [73], which includes the PLC and sensors in the control network, the defence system Honeywall working in the bridge mode with the signature-based intrusion detection tool Snort and the attacking toolkits deployed in the external network, the extension work in Chapter 4 includes adding a supervising network to simulate a comprehensive SCADA system with a new virtual machine HMI in it for simulating the function of the control centre and generating the normal network data, replacing the previous defence system with a new generic Linux middle box installed with anomaly-based and signature-based intrusion detection analyzer Bro, and reconfiguring the routing of all the hosts in the new testbed and configuring the forwarding function of the middle box as a router to simulate the practical SCADA network function. Note that the new testbed reuses the PLC and sensors in the control network, and reuses the attacking toolkits moving into the new supervising network to generate the abnormal data while not including the external network any more. The new virtual machine HMI is created by reconfiguring the MBlogic project on PLC and configuring as the Modbus client of its server PLC. To sum up, the overall features of the testbed introduced in the thesis are as follows: First, the testbed is implemented by software, which can be reconfigured flexibly and enlarged easily according to different research requirements. Second, we adopt an HMI-PLC-Sensors deployment within the networks, which simulates a typical SCADA system communicating using Modbus/TCP protocol. Third, a reasonable SCADA network is implemented, represented by the separated network topology of the whole SCADA network, i.e., the supervising network and control network. Fourth, a combination of useful tools such as Wireshark are supported by the testbed, which are helpful in debugging and in-depth investigations. Last but not least, the system utilizes Bro, a powerful network traffic analyzing tool, which can

be used for the research on both signature-based intrusion detection and anomaly-based intrusion detection. The procedures of signature-based intrusion detection and anomaly-based intrusion detection using Bro analyzer has also been presented in the thesis.

1.4 Overview of Thesis

The outline of thesis is as follows:

Chapter 1 contains an introduction and related work to the thesis followed by the motivation of the thesis work.

Chapter 2 describes in detail the proposed anomaly-based hierarchical online intrusion detection system for SCADA networks and validates the design by numerical results.

Chapter 3 presents the implementation of a SCADA network testbed, which can be used to implement the anomaly-based intrusion detection system.

Chapter 4 concludes results of the thesis, and discusses the future work.

Chapter 2

Hierarchical Online Intrusion Detection for SCADA Networks

This chapter describes the proposed anomaly-based HOIDS designed for a large-scale SCADA system under critical real-time requirements. The system architecture and operating principles are described in Section 2.1, which presents the IDS server-client distribution architecture and intrusion detection mechanism. The detailed detection models, including normal-abnormal binary detection and multi-attack detection based on logistic regression and quasi-Newton optimization algorithm, the fundamentals of feature selection and dimension reduction to accelerate intrusion detection and the performance measurement methods are illustrated in Section 2.2. The numerical results including the simulations of the SCADA control center network and substation and field networks are presented in Section 2.3. Section 2.4 makes a conclusion for this whole chapter. To verify the design of HOIDS, we choose the KDD Cup 1999 (KDD99) Dataset [23] to simulate the control center network of the SCADA system and the ICS dataset [30] to investigate the substation and field networks in the system. Besides the network simulations for general SCADA system, we evaluate our algorithm by using a power system dataset [24] [59].

2.1 HOIDS System Design

The section describes the detailed design for HOIDS. The system architecture is described in Subsection 2.1.1, which presents the IDS server-client distribution architecture. And the intrusion detection operating principles are illustrated in Subsec-

tion 2.1.2.

2.1.1 HOIDS architecture

Fig. 2.1 illustrates a typical SCADA system consisting of the field networks, substations and control center which is connected to external networks such as the corporate networks and the Internet. In the control center, there are engineering workstations (EWS), human machine interfaces (HMI), energy management systems (EMS), historians, application servers, etc. The network flow in the control center is similar to that in the external IT networks. One reason for this similarity is that backend protocols [48] such as Open Process Communications (OPC) and Inter-Control Centre Protocol (ICCP) work in a client/server manner supported by TCP/IP over Ethernet. Another reason is that control center network is usually connected to corporate business networks, and some might connect to the Internet directly [67]. In the substations, there are EWS, HMI, historians, servers, etc. Field devices consist of intelligent electronic devices (IED), remote terminal units (RTUs), programmable logic controllers (PLC) and many other embedded machines. Generally, the communication protocols used between substations and the field networks are fieldbus protocols. Modbus, DNP3, etc., are widely used, and normally, the networks are isolated from the Internet and other external networks. Protocol gateway (PG) is to translate messages among various protocols.

Due to the hybrid nature of SCADA networks, hierarchical IDS deployment is proposed for different network components as shown in Fig. 2.1. First, the IDS system is composed of IDS agents residing at all components of the SCADA network. Intrusion detection is realized by machine learning algorithms. Second, a client-server model is adopted with the IDS server at the control center, and three levels of IDS clients distributed in the control center, the substation networks and field networks, respectively. The IDS server deployed in the control center is the command centre of the whole system, and all the IDS clients communicate with and are controlled by the IDS server in the control center. Such arrangement is motivated by the fact that field devices are usually simple electronics with constrained computing capability, and the amount of data traffic between the field devices and substations is generally limited. Therefore, it will be impossible for these field devices to host IDS agents that execute full machine learning algorithms, unless external computing modules are added specifically for IDS. The client-server model can greatly alleviate the IDS comput-

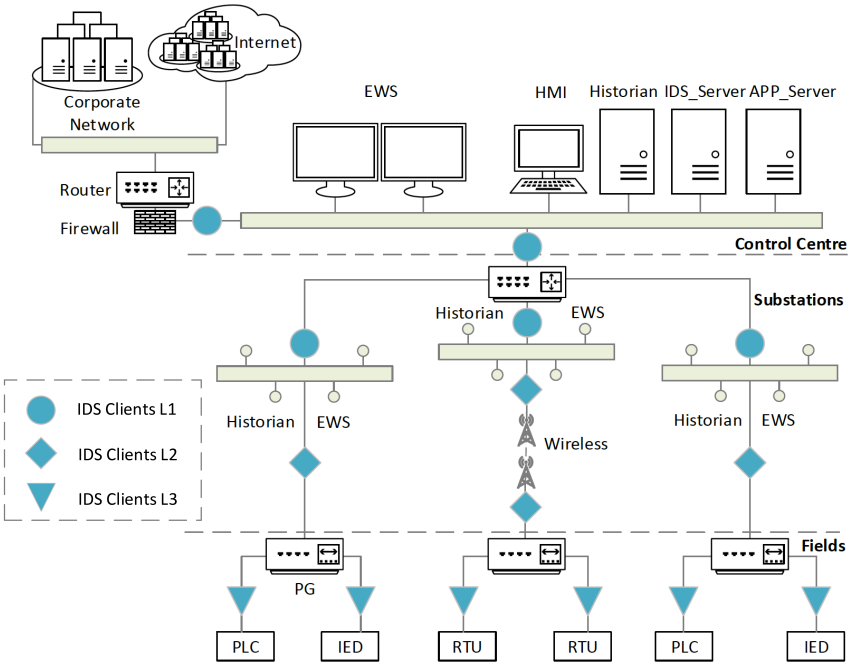

Figure 2.1: HOIDS schemes for SCADA systems

ing needs. It provides flexible IDS complexity for different network components in a SCADA system. Third, the IDS clients can be configured together with signature-based techniques to strengthen security. Furthermore, the IDS server has a holistic view of the whole network and can make adjustment to the detection model of the clients, especially when a certain client detects intrusion, to improve the overall IDS accuracy of all the components in the network.

2.1.2 HOIDS Operating Mechanism

Fig. 2.2 illustrates the mechanism of the IDS server in the control center and the IDS clients distributed at three levels of the network. The IDS server with relatively high computing ability mainly consists of the feature engine and the principle engine. The feature engine is responsible for collecting the features of network flows in different SCADA network components, forming datasets, using the datasets sequentially and periodically to train machine learning algorithms and storing the generated principles for each IDS client to the principle engine. The principle engine is responsible for sending the detection principles to each IDS client sequentially and periodically. The IDS clients deployed at different network spots extract features of the real time traffic

and apply the received principles to analyze current data flows for intrusion detection, launch alarm if anomaly is detected. The clients also constantly send the extracted features and detection results to the IDS server for periodically updating the training set and detection principles online to ensure the accuracy and timeliness of intrusion detection. The separation of training and detection to server and client save considerable computing resources at the client side. Meanwhile such architecture allows sophisticated machine learning algorithms to be adopted and updated easily for the whole network. Different network elements can use different IDS algorithms, both for training at the server side and detection at the client side. Consequently, this design is effective to reduce the global financial budget for securing large-scale SCADA networks. We should also note that most devices in the field networks have limited computing abilities and stringent real-time requirement. Therefore, preprocessing features by effective feature selection techniques and dimension reduction methods will accelerate the detection process and achieve online intrusion detection with minimum impact on the accuracy of intrusion detection (demonstrated in Section 2.3). Furthermore, when an intrusion is detected at a certain IDS client, the hierarchical design has the potential to coordinate IDS detection at different network elements to enhance SCADA security, for example, by adjusting the choice of feature sets for the IDS client and its adjacent clients.

The communication overhead required in the HOIDS design is determined by transmitting the extracted features of captured traffic from IDS clients to the centralized server for training, and by transmitting the detection methods, including the feature extraction method and the principle weights, from the server to the clients. Specifically, the communication overhead is related to two settings: the designated number of features and the frequencies of transmission. Actually both of them are adjustable. The number of features can be changed based on the traffic characteristics and the relationship of the number of features with the detection accuracy will be presented in Chapter 2.3. Technically, a certain number of traffic samples can be used for training machine learning algorithms, so transmission amount is adjustable by controlling the transmission frequency. The adjustment of the above two settings is also constrained by the physical bandwidth of network, in the SCADA systems with limited network capacity. But generally, through properly choosing the settings of above parameters, HOIDS can be applied to an existing SCADA system without violating its capacity. Based on what is discussed above, we may also realize that the settings of feature number and transmission frequency can be formulated as an

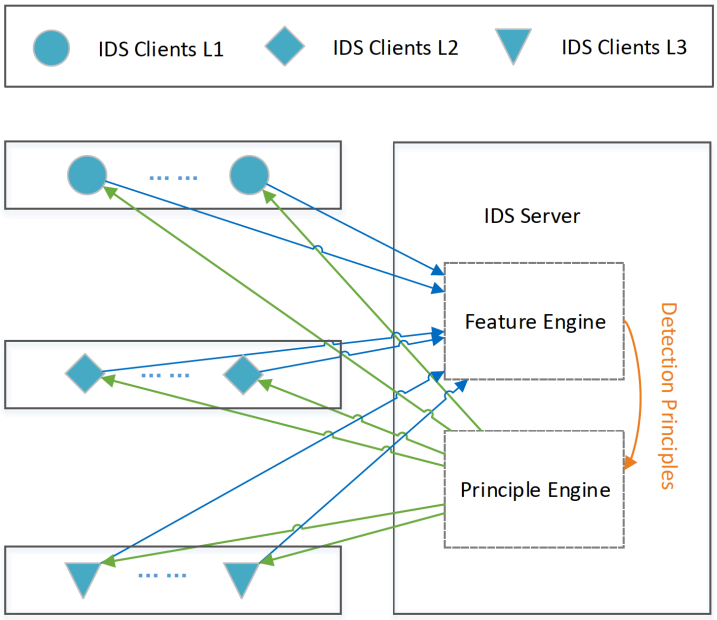

Figure 2.2: Workflow of the HOIDS data transmission

optimization problem that make the tradeoff between communication overhead and detection accuracy, which can be further studied in the future.

Note that the hierarchical design discussed in this scheme mainly represents that the intrusion detection is implemented in three different layers of the SCADA networks. The hierarchies differentiated in the scheme do not interact with each other directly, instead, they are centrally coordinated and controlled the intrusion detection engines in the control center. For example, the operators of control centers are possibly to adjust the intrusion alert threshold of different layers based on the physical equipment capacity and their empirical threat estimation. The main benefits of such a design include both the useful consideration of different traffic characteristics among layers, and the high reliability and simplicity in implementation.

The IDS in different layers of the network can employ machine learning based intrusion detection algorithms of different complexities according to the available resources. In the next section, details of logistic regression based intrusion detection algorithms and feature reduction methods to lower computation complexity are presented.

2.2 Machine Learning Based Intrusion Detection and Detection Acceleration Methods

This section presents the detailed detection models, including normal-abnormal binary detection and multi-attack detection based on logistic regression and quasi-Newton optimization algorithm, presented in Subsection 2.2.1. In Subsection 2.2.2, we present the principles of information gain based feature selection and principle component analysis based dimension reduction to reduce the feature set and accelerate detection. Subsection 2.2.3 describes the performance measurement methods in this Chapter.

2.2.1 Logistic regression [26]

In HOIDS design, we apply machine learning techniques to intrusion detection. Specifically, we use logistic regression to classify the training dataset and generate the detection model. Logistic regression has been a powerful mathematical method for classification problems. Generally, different machine learning algorithms have their own advantages. For HOIDS implementation, logistic regression has more advantages over other techniques. For example, the model of logistic regression can be interpreted clearly as a probability, beneficial for result analysis and model adjustment. As opposed to naive Bayes, another probabilistic algorithm that makes classification under the assumption of independent features, logistic regression can generate the classification principle regardless of the correlation among the training features. Compared to SVM and neural networks, the training time of logistic regression is shorter. SVM may generate many supporting vectors in the detection model, reducing detection efficiency if applied in the HOIDS design. Moreover, logistic regression is able to incorporate new training data easily into the current classification model by using stochastic gradient descent method, which is important for industrial applications. Logistic regression is also efficient to realize multi-classification with low complexity and high accuracy by using the multinomial logistic regression [32] (details in Section 2.3.1), while some other machine learning algorithms rely on the one-against-all approach to achieve multi-classification. Therefore, logistic regression is an appropriate machine learning algorithm that can be applied to industrial network IDS. Next we present the principle of normal-abnormal binary detection and multi-attack detection based on logistic regression.

Normal-abnormal binary detection [51]

In normal-abnormal binary detection, the logistic regression classifier identifies intrusive connections against normal connections. Here, intrusive connections can also be called abnormal connections. Define the feature space of connections as $\mathcal{X} = \mathcal{R}^M$, where M is the number of features for each connection and $\mathcal{R}$ is the set of all real numbers. The classification output space can be expressed as $\gamma = \{+1, -1\}$, where $+1$ is a label representing the abnormal connection and -1 representing the normal connection. The training dataset $\mathcal{D}$ consists of N connections, i.e., $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i), i = 1, 2, ..., N\}$, where $\mathbf{X}$ is a $(M+1) \times N$ matrix representing N connections, and each input connection $\mathbf{x}_i = [x_{i0} \;\; x_{i1} \;\; \cdots \;\; x_{ij} \;\; \cdots \;\; x_{iM}]^T$ has $x_{i0} = 1$, and $\mathbf{y}$ is an N-dimensional label vector with each label $y_i \in \gamma$. The classification model weights can be represented by $\mathbf{w} = [w_0 \;\; w_1 \;\; \cdots \;\; w_j \;\; \cdots \;\; w_M]^T$ where w_j represents the weight of corresponding x_{ij}. In this case, we need to maximize the conditional probability of getting $\mathbf{y}$ given the corresponding $\mathbf{X}$ to generate the classification model, i.e., maximizing

$$P(\mathbf{y}|\mathbf{X}) = \prod_{i=1}^{N} P(y_i|\mathbf{x}_i) \tag{2.1}$$

where

$$P(y_i|\mathbf{x}_i) = \begin{cases} P(y_i = 1|\mathbf{x}_i) & \text{for } y_i = +1 \\ 1 - P(y_i = 1|\mathbf{x}_i) & \text{for } y_i = -1. \end{cases} \tag{2.2}$$

Logistic regression uses the logistic function $\theta(z) = 1/(1 + e^{-z})$ to map the linear combination $z = \mathbf{x}_i^T \mathbf{w}$ to a value between 0 and 1. Let $P(y_i = 1|\mathbf{x}_i) = \theta(\mathbf{x}_i^T \mathbf{w})$, and due to the property of logistic function $\theta(-z) = 1 - \theta(z)$, we have $P(y_i|\mathbf{x}_i) = \theta(y_i \mathbf{x}_i^T \mathbf{w})$. Maximizing the joint conditional probability (2.1) is equivalent to minimizing its scaled negative logarithm, therefore we have the minimization of the objective function F_{bin} for normal-abnormal binary classification

$$F_{bin}(\mathbf{w}) = -\frac{1}{N} \sum_{i=1}^{N} \ln P(y_i|\mathbf{x}_i) = \frac{1}{N} \sum_{i=1}^{N} \ln(1 + e^{-y_i \mathbf{x}_i^T \mathbf{w}}). \tag{2.3}$$

It can be proved easily that the Hessian (a square matrix of second-order partial derivatives of a scalar-valued function) [3] of (2.3) is positive definite as long as $\mathbf{X} \neq \mathbf{0}$ which holds in all practical cases. Consequently, minimizing (2.3) is a global convex optimization problem and can be solved using conventional gradient-based techniques with a proper line search step. Here, we adopt the quasi-Newton

optimization implemented with the Broyden-Fletcher-Goldfarb-Shanno (BFGS) approach and the back-tracking line search method [28], which has been verified as one of the most efficient optimizers for logistic regression [55]. The basic quasi-Newton algorithm is as follows: given initial weights $\mathbf{w}_0$ and the tolerance ϵ, the weights will be updated in the next iteration $\mathbf{w}_{k+1} = \mathbf{w}_k + \boldsymbol{\delta}_k$. And $\boldsymbol{\delta}_k = -\alpha_k \mathbf{S}_k \mathbf{g}_k$ represents the updated step, in which $\mathbf{g}_k$ is the gradient vector of F_{bin} in the k^{th} iteration, α_k is a small positive value decreasing the value of F_{bin} in the k^{th} iteration, and $\mathbf{S}_k$ is an $(M+1) \times (M+1)$ direction matrix. Both α_k and $\mathbf{S}_k$ can be obtained by different methods. Repeat the iteration until convergence, i.e., $\|\boldsymbol{\delta}_k\| < \epsilon$. The BFGS approach is to obtain $\mathbf{S}_k$ iteratively according to

$$\begin{aligned} \mathbf{S}_0 &= \mathbf{I} \\ \mathbf{S}_{k+1} &= \mathbf{S}_k + (1 + \frac{\boldsymbol{\gamma}_k^T \mathbf{S}_k \boldsymbol{\gamma}_k}{\boldsymbol{\gamma}_k^T \boldsymbol{\delta}_k}) \frac{\boldsymbol{\delta}_k \boldsymbol{\delta}_k^T}{\boldsymbol{\gamma}_k^T \boldsymbol{\delta}_k} - \frac{\boldsymbol{\delta}_k \boldsymbol{\gamma}_k^T \mathbf{S}_k + \mathbf{S}_k \boldsymbol{\gamma}_k \boldsymbol{\delta}_k^T}{\boldsymbol{\gamma}_k^T \boldsymbol{\delta}_k} \end{aligned} \tag{2.4}$$

in which $\mathbf{I}$ is an identity matrix of size $(M+1)$, and $\boldsymbol{\gamma}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$. The back-tracking line search is an effective inexact line search method to obtain α_k by finding an α satisfying $F_{bin}(\mathbf{w}_k - \alpha_k \mathbf{S}_k \mathbf{g}_k) \leqslant F_{bin}(\mathbf{w}_k)$, of which details can be referred to [34].

By minimizing (2.3), we can obtain the optimal classification weight vector $\mathbf{w}$ and calculate the probability $P(y_i = 1|\mathbf{x}_i)$. A sample will belong to $y = +1$ if the probability exceeds 0.5, otherwise it will belong to $y = -1$. Since the logistic function is monotonically increasing, compare the value of $\mathbf{x}_i \mathbf{w}$ with 0, and the sample connection will belong to $y = +1$ if the value is positive, otherwise belong to $y = -1$.

Multi-attack detection [51]

As mentioned above, logistic regression can realize multi-classification with low complexity and high accuracy by using the multinomial logistic regression due to the property of the probabilistic model. The representation of the feature space, the number of input data and the number of features are the same as the normal-abnormal binary classification, while the classification output space is $\gamma_{multi} = \{0, 1, ..., K-1\}$, where K represents the number of class types. The classification model weights can be represented by $\mathbf{W} = \{\mathbf{w}_0, \mathbf{w}_1, ..., \mathbf{w}_{k-2}\}$. Note that $\mathbf{W}$ is a $(M+1) \times (K-1)$ matrix, which consists of the classification model weights for $(K-1)$ class types. And the objective function of multi-class logistic regression for minimization can be

denoted as

$$F_{multi}(\mathbf{W}) = -\frac{1}{N}\ln P(\mathbf{y}|\mathbf{X}) = -\frac{1}{N}\sum_{i=1}^{N}\ln P(y_i|\mathbf{x}_i) \tag{2.5}$$

where

$$P(y_i|\mathbf{x}_i)=\begin{cases} e^{\mathbf{x}_i^T\mathbf{w}_{y_i}} \Big/ (1+\sum_{a=0}^{K-2} e^{\mathbf{x}_i^T\mathbf{w}_a}), 0 \le y_i < K-1 \\ 1 \Big/ (1+\sum_{a=0}^{K-2} e^{\mathbf{x}_i^T\mathbf{w}_a}), y_i = K-1 \end{cases} \tag{2.6}$$

and note that $\sum_{y_i=0}^{K-1} P(y_i|\mathbf{x}_i) = 1$. By minimizing the multi-classification objective function (2.5) using the quasi-Newton optimization implemented with the BFGS approach, we can obtain the optimal multi-classification weight matrix $\mathbf{W}$. To classify a new sample connection, calculate all the $P(y_i|\mathbf{x}_i), y_i = 0, 1, ..., K-1$ based on (2.6), and then the sample connection belongs to the class type with the highest probability. In this way, we can use the generated detection principle to classify the testing dataset. Since the one-against-all approach needs to train the binary classification for K times to generate K model weights for all the classes, multinomial logistic regression is more efficient than the one-against-all approach. The training complexity is related to number of variables in the BFGS approaches. In our multi-class detection, the number of variables is $O(KM)$. Given that solving an optimization problem with d variables by BFGS takes $O(d^2)$ time, the training complexity of our approach is $O(K^2M^2)$ per iteration. For the space complexity, it is also $O(K^2M^2)$ determined by the BFGS approach. For the testing, the calculation complexity is $O(KM)$, since we need to calculate $P(y_i|x_i)$ for each of the $(K-1)$ classes, according to 2.6.

2.2.2 Feature selection and dimension reduction

In HOIDS, especially for the field networks, we apply two methods for preprocessing SCADA network data features. The first method is to use information gain to denote the significance of each feature, and select features with high information gain to accelerate intrusion detection. Similar to mutual information, information gain is the reduction in the entropy of labels achieved by partitioning the labels according to a certain feature. For a certain feature, the information from labels will be changed if this feature is not included in the system. As a result, the reduced label entropy is the information the feature brings to the system.

For the training dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i), i = 1, 2, ..., N\}$ with M features

of the input connections and K classifications of labels, the information entropy $H(\mathbf{y})$ of the label $\mathbf{y}$ can be obtained by

$$H(\mathbf{y}) = -\sum_{k=0}^{K-1} P(y_k) \ln P(y_k) \tag{2.7}$$

where $P(y_k) = N_k/N$, and N_k represents the number of samples of class k in the training dataset.

Assume a feature F_m consisting of values $\{f_1, f_2, ...f_S\}$, the information gain IG that feature F_m brings to the system is

$$IG(F_m) = H(\mathbf{y}) - H(\mathbf{y}|F_m) \tag{2.8}$$

in which

$$H(\mathbf{y}|F_m) = \sum_{s=1}^{S} P(f_s) H(\mathbf{y}|f_s) \tag{2.9}$$

$P(f_s) = N_{f_s}/N$, and N_{f_s} denotes the number of samples which have the value f_s in terms of feature F_m. Usually, a feature with high information gain is preferred over those with low information gain. Here, we exploit the concept of information gain to select a feature subset to accelerate the detection model training and online detection process.

The second method is to use singular value decomposition (SVD) to obtain a low-dimensional approximation of the original feature set [28], which is also known as principal component analysis (PCA). We apply PCA to the covariance matrix of the N feature vectors. The covariance matrix can be calculated as

$$\mathbf{C} = \frac{1}{N-1} \sum_{i=1}^{N} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T \tag{2.10}$$

where $\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$ denotes the average vector of the N feature vectors, and $\mathbf{x}_i$ is normalized using zscore [2], the most commonly used method for normalization, to obtain a normalized vector with an average of zero and standard deviation of one. Since $\mathbf{C}$ is at least positive semidefinite, its SVD is the same as its eigen decomposition. We can obtain an approximation of variance matrix $\mathbf{C}$ by considering only the

K largest eigenvalues and the corresponding eigenvectors as

$$\mathbf{C} = \mathbf{USU}^T \approx \mathbf{U}_K\mathbf{S}_K\mathbf{U}_K^T \tag{2.11}$$

in which $\mathbf{U} = [\mathbf{u}_1\ \mathbf{u}_2 \cdots \mathbf{u}_M]$ is an orthogonal matrix, $\mathbf{S} = diag\{\sigma_1, \sigma_2, \cdots, \sigma_M\}$ with non-negative eigenvalues in a descending order, $\mathbf{U}_K$ contains the first K vectors of $\mathbf{U}$, and $\mathbf{S}_K$ is the diagonal matrix containing the first K largest eigenvalues. Compute the projection of the variation $(\mathbf{x}_i - \bar{\mathbf{x}})$ onto the K-dimensional subspace spanned by $\mathbf{U}_K$ as

$$\mathbf{z}_i = \mathbf{U}_K^T(\mathbf{x}_i - \bar{\mathbf{x}}). \tag{2.12}$$

We use $\mathbf{z}_i$ as the new features for each connection. Since $\mathbf{U}_K$ includes K orthogonal vectors, the new features generated by PCA are uncorrelated. The number of principle components K can be chosen by finding the smallest K satisfying $\sum_{i=1}^{K}\sigma_i \Big/ \sum_{i=1}^{M}\sigma_i \geq \rho$, where ρ is the rate of variance retained. This method is specifically effective when the original features are heterogeneous and some may be highly correlated.

2.2.3 Performance measurement methods

For the detection with a certain feature set, we mainly use recall and precision to measure the performance, which are the critical performance measures for IDS. For the IDS binary classification, true positive (TP), false negative (FN), false positive (FP) and true negative (TN) denote the quantities of intrusions identified as intrusions, intrusions identified as normal, normal connections identified as intrusions and normal connections identified as normal, respectively. Note that for multi-classification, we define TP, FN, FP and TN as the quantities of intrusions identified as correct intrusion types, intrusions identified as normal connections, normal connections identified as intrusions or intrusions incorrectly identified as different intrusion types and normal connections identified as normal, respectively. Recall (r) is defined as $TP/(TP+FN)$, and precision (p) is defined as $TP/(TP + FP)$. For IDS, r is important, while p can not be ignored as well, since intrusions should be detected as many as possible, while alarms are supposed to be real intrusions as many as possible. Specifically, we use 10×10-fold cross-validation (CV), i.e., implementing 10-fold cross validation 10 times on the training dataset with different stochastic orders, to evaluate r and p for each selected feature combination. Compared with 10-fold CV, the most widely used validation procedure, 10×10-fold CV can obtain more reliable performance

estimation since more estimates are always preferred [33]. Estimating the mean of r for 10×10-fold CV are compared using confidence intervals defined by

$$\bar{r} \pm t_{\frac{\alpha}{2}}(n-1)\frac{s}{\sqrt{n}} \tag{2.13}$$

in which $\bar{r}$ and s are the mean and standard deviation of r of the CV samples, $t_{\frac{\alpha}{2}}(n-1)$ is the value of t distribution at $(n-1)$ degrees of freedom for a $(1-\alpha)$ confidence interval, and n is the size of CV samples. Estimation for the mean of p is the same as that for r.

Different practical scenarios may have different requirements for recall and precision. F_β-measure considers both recall and precision to be a performance measure, defined by

$$F_\beta = (1+\beta^2)pr/(\beta^2 p + r) \tag{2.14}$$

in which β is a positive real weight that attaches importance to recall or precision, e.g., F_1-measure is the harmonic mean of precision and recall, $F_{1.5}$-measure puts more emphasis on recall than precision. Here, we use the means of p and r to calculate F_β. In this way, we can choose the model of preprocessing the original features based on the performance estimation of CV.

We also use sensitivity (also called as recall above) and specificity to measure the performance when attacks have a major percentage in the training datasets and the testing datasets. In this case, the attack samples in the training dataset can have a significant impact on algorithm training, which might cause negative effects for classifying normal datasets. Specificity is defined as $TN/(TN+FP)$, which can show the classification performance for normal datasets.

2.3 Numerical results

This section presents the classification results based on the logistic regression algorithm to simulate the intrusion detection for SCADA networks. Note that network data flows have unique characteristics specific to that type of network. For example, the network data flows in a control center are similar to traditional IT networks, and thus the corresponding features can be extracted by the corresponding IDS clients in a similar way to those in IT networks. Therefore, we simulate the network data

in the control center by exploiting the KDD99 dataset [23], the most widely used dataset for evaluating the performance of an IDS designed for IT networks. On the other hand, the network traffic in the substation and field network can be simulated by the ICS dataset [24], [56]. Besides the network simulations for general SCADA system, we evaluate our algorithm by using a power system dataset [24] [59]. All the experiments are implemented by Matlab and run on a server with Intel Xeon 8-core processor E5-2670 and 64 GB RAM.

2.3.1 Simulation of the SCADA control center network

In this section, we present the simulation results of a SCADA control center network using the KDD99 dataset. In our experiment, we exploit the KDD99 (10 percent) training dataset for training and KDD99 testing dataset for testing. The training dataset consists of $494,021$ samples with 41 features (3 nominal and 38 numerical) and 22 different types of attacks (e.g., Back, Land, Neptune and Smurf) that fall into four categories: Denial of service (DOS), Probe, Remote to local (R2L) and User to root (U2R). In the training dataset, there are $97,278$ (19.69%) normal connections, $391,458$ (79.24%) DOS, $4,107$ (0.83%) Probe, $1,126$ (0.23%) R2L and 52 (0.01%) U2R connections. The testing dataset (consisting of $311,029$ samples with 41 features) contains 22 attack types existed in the training dataset and additional 17 different kinds of attacks. In the testing dataset, there are $60,593$ (19.48%) normal connections, $229,853$ (73.90%) DOS, $4,166$ (1.34%) Probe, $16,189$ (5.20%) R2L and 228 (0.07%) U2R connections. The fact that the testing dataset does not have the same probability distribution as the training dataset in terms of additional attacks in the testing dataset, to some extent, makes the detection process close-to-realistic scenarios. All the 41 features, including time- and host-based traffic features, are derived from the characteristics of the network data flow. In our simulation, we exploit 38 numerical features for training and testing.

Using the multinomial logistic regression to classify the training dataset and applying the generated detection principle to the testing dataset, we get the confusion matrix in Table 2.1. Compare these results (recall (r) and precision (p)) with those obtained by the PNrule method [27] (recall (r_{PN}) and precision (p_{PN})), a rule-based classifier applicable to scenarios where different classes have very different distributions in training data. As shown, the overall performance of both methods are consistent. The recalls and precisions calculated for each class by the two methods are

Table 2.1: Classification for the KDD99 testing dataset

	Predicted Class						
Actual Class	Norm	DOS	Probe	R2L	U2R	r	r_{PN}
Norm	59579	790	172	44	8	0.983	0.995
DOS	6380	223451	20	0	2	0.972	0.969
Probe	1064	96	3004	0	2	0.721	0.730
R2L	16137	4	18	25	5	0.002	0.107
U2R	197	4	0	6	21	0.092	0.066
p	0.715	0.996	0.935	0.333	0.553		
p_{PN}	0.730	0.9995	0.925	0.880	0.105		

close. This confirms the validity of multinomial logistic regression. Logistic regression shows higher recall and precision for U2R than PNrule, while PNrule achieves higher recall and precision for R2L. In terms of R2L, logistic regression may benefit from more training samples, since the percentage of all R2L attacks in the training dataset is only 0.23%, while 5.20% in the testing dataset with additional 7 attack types not shown in the training dataset. How to determine whether the training samples are enough is still mostly empirical and without much theoretical foundation in the current research field of machine learning. And for the additional 17 new attacks not shown in the training dataset, the detection rate using multinomial logistic regression for new DOS(6,555), Probe(1,789), R2L(10,196) and U2R(189) attacks are 6.19%, 42.82%,0.08%, 3.17%.

Next, we verify the performance of multinomial logistic regression by comparing it with the one-against-all method. To simplify the simulation, we randomly sample the training dataset. Since the percentages of attack categories of DOS (79.24%), Probe (0.83%), R2L (0.23%) and U2R (0.01%) are uneven, we classify the attacks into 4 categories as Normal (97,278), Smurf (280,790), Neptune (107,201) and other attacks (8,752), which has the percentage of Normal (19.69%), Smurf (56.84%), Neptune (21.70%) and Others (1.77%), as shown in Fig. 2.3. With the same percentage as the KDD99 (10 percent) dataset, we randomly choose 800 Normal, 2,320 Smurf, 800 Neptune and 80 Others to form the new training dataset ($4,000 \times 38$), and 1,000 Normal, 2,900 Smurf, 1,000 Neptune and 100 Others to form the new testing dataset ($5,000 \times 38$) such that the probability distribution of the testing dataset become similar to that of the training dataset. We first use logistic regression one-against-all binary classification to realize the multi-classification. The in-sample error (E_{in}, the

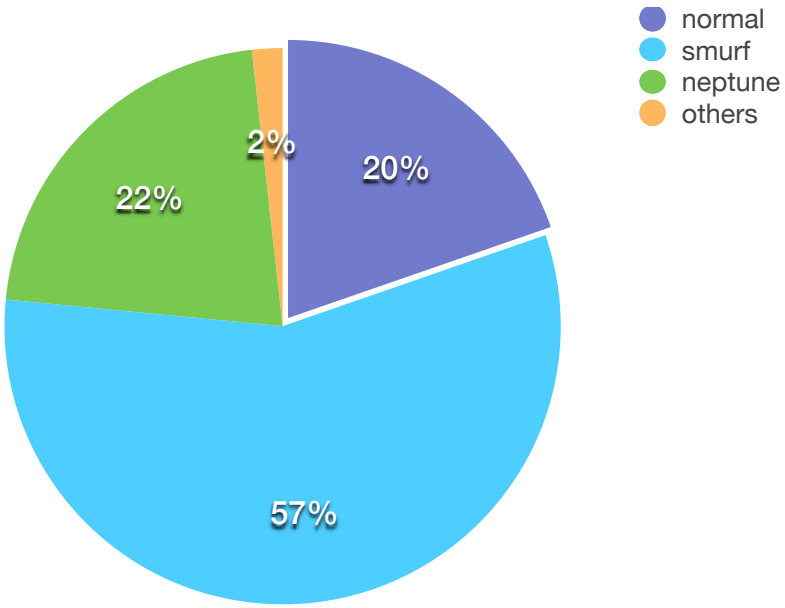

Figure 2.3: Categories in the KDD99 training dataset

ratio of misclassified samples to the total samples in the training dataset) changing with the number of iterations for the new training dataset is shown in Fig. 2.4. After 130 iterations, E_{in}s are 0, 0, 0.001, 0.002 for normal-nonnormal, smurf-nonsmurf, neptune-nonneptune and others-nonothers binary classifications, respectively. The global E_{in} for the training dataset is 0.002. Out of $4,000$ training samples, $3,993$ samples are classified correctly. The out-of-sample error (E_{out}, the ratio of misclassified samples to the total samples in the testing dataset) for the new testing dataset is 0.005. Out of $5,000$ testing samples, $4,974$ samples are classified correctly. Next, we use the multinomial logistic regression method to achieve the multi-classification. The global E_{in} for the training dataset is 0. And E_{out} for the testing dataset is 0.0046. Out of $5,000$ testing samples, $4,977$ samples are classified correctly. The multinomial logistic regression outperforms in terms of efficiency and accuracy.

2.3.2 Simulation of SCADA substation and field networks

To test the SCADA substation and field networks, we exploit the ICS dataset gathered from a gas pipeline system of Mississippi State University's Critical Infrastructure Protection Centre [56]. This dataset has 26 features, including 17 numerical features such as Invalid Function Code (a binary bit indicating the validity of function code), Pump State (a binary bit indicating the state of the pump: on or off) and so on. Here

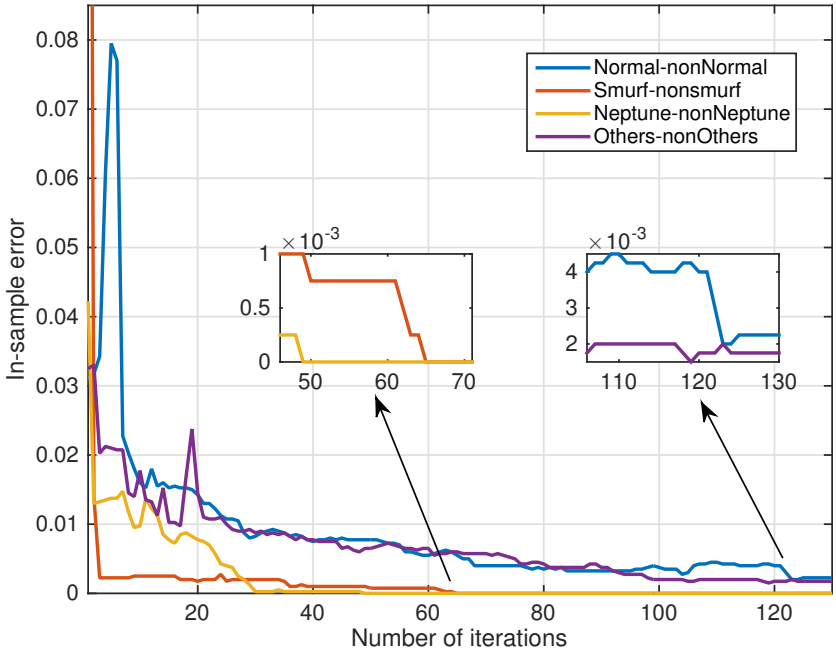

Figure 2.4: One-against-all for KDD99 (Sampled)

we take the Multi-class Command Injection dataset, an important part of the above ICS dataset, as an example to analyze the ICS dataset. The Multi-class Command Injection dataset models the issued commands from the master to control the gas pipeline system, consisting of 28, 086 Good commands (Good), 2 Address Scan attacks (Addr), 9 Function Code Scan attacks (Func), 198 Illegal Setpoint attacks (IllSet) and 49 PID Modification attacks (PID).

We use all 17 numerical features to evaluate the performance of the multinomial logistic regression classifier. To study the impact of features for the classifier, information gains of each feature are shown in Fig 2.5. As mentioned before, normally, the feature with lower information gain has less influence on the classifier. Thus, features are reduced in succession to train the classifier based on their information gain, and the result is shown in Table 2.2. In the table, the elements from left to right in the in-sample error vector represent the misclassified numbers of Good commands, Address Scan attacks, Function Code Scan attacks, Illegal Setpoint attacks and PID Modification attacks, respectively. As seen from the table, when all 17 features are used, the in-sample error is 0, which means all the attacks in the Multi-class Command Injection dataset can be detected by the classifier. When excluding F6, F8,

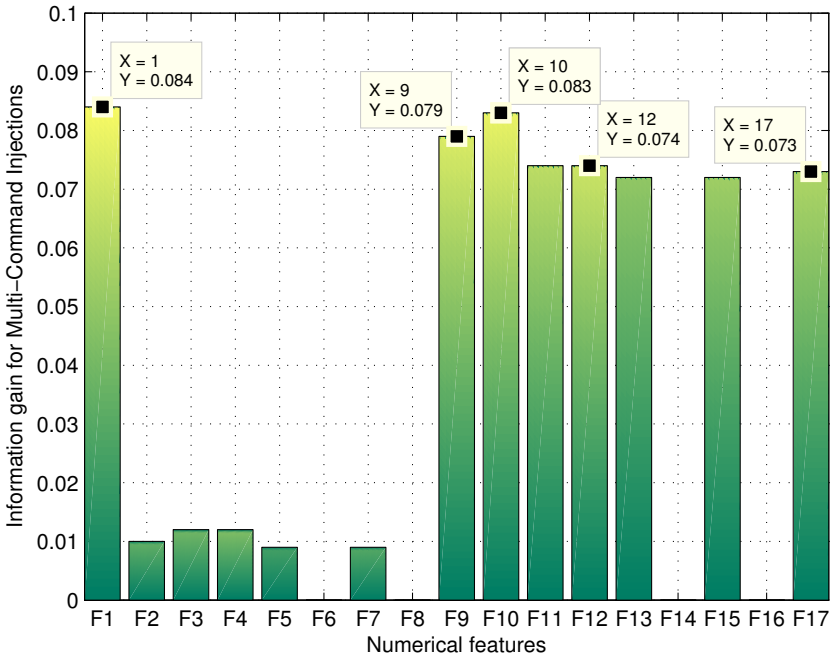

Figure 2.5: Information Gain for Multi-Command Injections (Entropy = 0.084)

F14 and F16, of which information gains are 0, the in-sample error is still 0. Then exclude F2-F5 and F7 with low information gain, the in-sample error ($7.06e-3\%$) is still satisfactory. When only select three features with highest information gain F1, F9, F10, the in-sample error is only $1.06e-2\%$. Therefore, for the substation and field network, we can use reduced feature sets for training and detection to accelerate and simplify the IDS process. For example, if data traffic in a field network component is higher and has more stringent real-time requirement, intrusion detection with only 3 features or 5 features can be a good choice. Or if only polls between masters and slaves exist in a network component in substation or field networks, which means only the read-only input registers of the slaves are operated, intrusion detection with F1, F9 - F12, F17 (6) can be a good choice, since there cannot exist the Illegal Setpoint attacks and thus the in-sample error will be 0.

To further verify the results, we present the 10×10-fold CV performance for the feature reduction sets in Table 2.2, shown as Fig. 2.6. Estimating the mean of recall and precision for 10×10-fold CV are compared using 95% confidence intervals. We can get the same conclusion that reducing features based on information gain not

Table 2.2: Feature reduction for Multiclass Command dataset

No. of features used	Ein-vec	Ein	r/p
F1-17 (17)	[00000]	0.00%	1/1
F1-5, 7, 9-13, 15, 17 (13)	[00000]	0.00%	1/1
F1, 9-13, 15, 17 (8)	[00020]	7.06×10^{-3}%	1/1
F1, 9-12, 17 (6)	[00020]	7.06×10^{-3}%	1/1
F1, 9-12 (5)	[02020]	1.41×10^{-2}%	0.996/1
F1, 9-10 (3)	[02010]	1.06×10^{-2}%	0.996/1

necessarily degrades the classification performance, but speeds the intrusion detection process. We can see that when reducing features based on the information gain, recalls fluctuate in a small range, while precisions present a downward trend. If practical requirement values recall and detection speed more than precision, then 3 features should be a proper choice while precision of 3 features is 0.991, also satisfying. If practical requirement considers both recall and precision, a feature combination with high F_β-measure, e.g., 5 features, would be a good choice. We can set β according to practical requirement, for example, in Fig. 2.6, $\beta = 1.5$, which means that the measure considers recall more than precision, satisfying the intrusion detection scenarios. Another usage of F_β-measure in HOIDS design is when an IDS client detects an intrusion, the IDS server will reconsider the feature selection with higher recall and higher F_β-measure value for the IDS client and the adjacent clients (in the same level and in the adjacent levels) to raise vigilance, which also guarantees a good precision for the incoming detection. However, a traditional way when an intrusion is detected is to lower the detection threshold, which can increase the recall but decrease the precision drastically, thus improving security while burdening the workload.

To further verify the results, we present the 10×10-fold CV performance for the dataset starting from all the 17 features. Estimating the mean of r and p for 10×10-fold CV are also compared using 95% confidence intervals, shown in Fig. 2.7 and Fig. 2.8, respectively. The features are reduced one by one in three different ways: from the features with the lowest IG to the highest IG (blue lines), from the features with the highest IG to the lowest IG (red lines) and reduce the features randomly (yellow lines). The 95% confidence intervals of the last two reducing ways are all less than 0.0025, not presented on the figures for clearer presentation. In Fig. 2.8, precisions have no values when all the connections are classified as normal

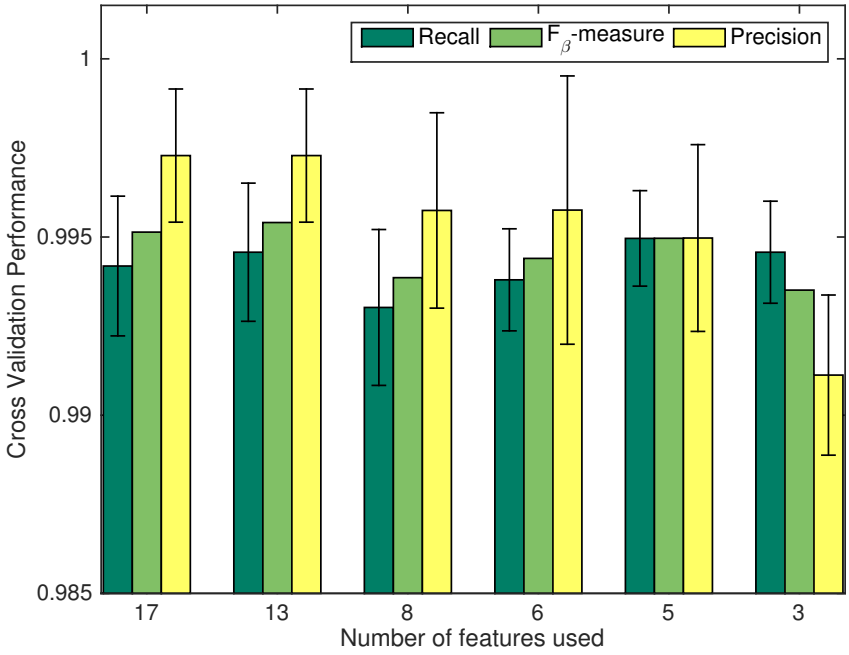

Figure 2.6: Cross validation performance for reduced feature sets

connections, referring to the definition of precision. From Fig. 2.7 and Fig. 2.8, we can see that reducing features randomly or from those with the higher IG can make both recall and precision much worse than the original feature set, while reducing features with lower IG can keep good performance even when only 4 or 3 features left in the feature set. Therefore, we can conclude that reducing the features with lower IG while keeping a few features with higher IG is an effective way to select feature subsets, able to accelerate training and detection while keeping high recalls and precisions. Dimension reduction based on PCA is also applied to the ICS dataset, shown as the purple lines in Fig. 2.7 and Fig. 2.8. The number of principle components is reduced from 7, since the first 7 eigenvalues keep about 100% variance. The 10×10-fold CV performance is obtained by using the new features generated by projection. We can see that when the number of new features is 6, the 10×10-fold CV performance is better than reducing the original features with lower IG. Note that the original features of the ICS dataset do not contain any traffic features and can be extracted quickly. Therefore, dimension reduction based on PCA is especially effective here. The performance of false alarm rate is shown in Fig. 2.9. With the number of features

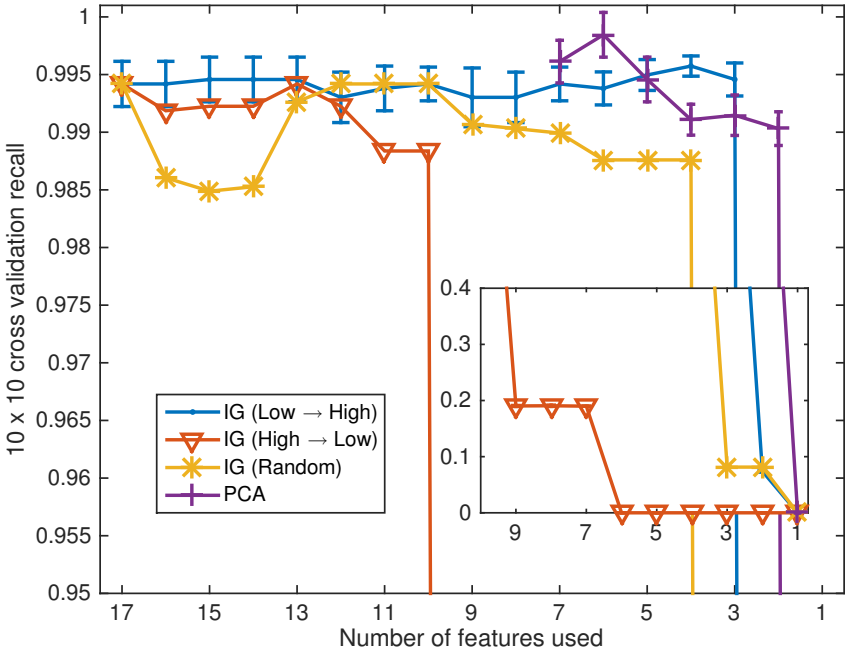

Figure 2.7: Recall performance with cross validation for all the features

decreasing, the false alarm rate for IG and PCA does not show much degradation, which validates the proposed feature reduction is effective in terms of still keeping a low false alarm rate.

In terms of the simulation results above, we started with all 17 numerical features to evaluate the performance of the multinomial logistic regression classifier. After further analyzing the dataset by calculating the correlation coefficients among all the features, we find that F6 PID Rate, F8 Pipeline PSI, F14 delta PID Rate and F16 delta Pipeline PSI are all constant, F3 PID Cycle Time, F4 PID Deadband, F5 PID Gain, F7 PID Reset are highly correlated, and F11 delta PID Cycle Time, F12 delta PID Deadband, F13 delta PID Gain, F15 delta PID Reset are highly correlated. The correlation coefficients among all the features are presented in the Appendix A.1, A.2. The high correlation among the feature set can also be verified by observing the performance change when reducing the features from the ones with the highest features to the lowest features, as shown in Fig. 2.10, Fig. 2.11 and Fig. 2.12. We can see from these three figures, especially from Fig. 2.10 and Fig. 2.11 that there are 3 stages during the dynamics of the performances. Therefore, 7 features are left

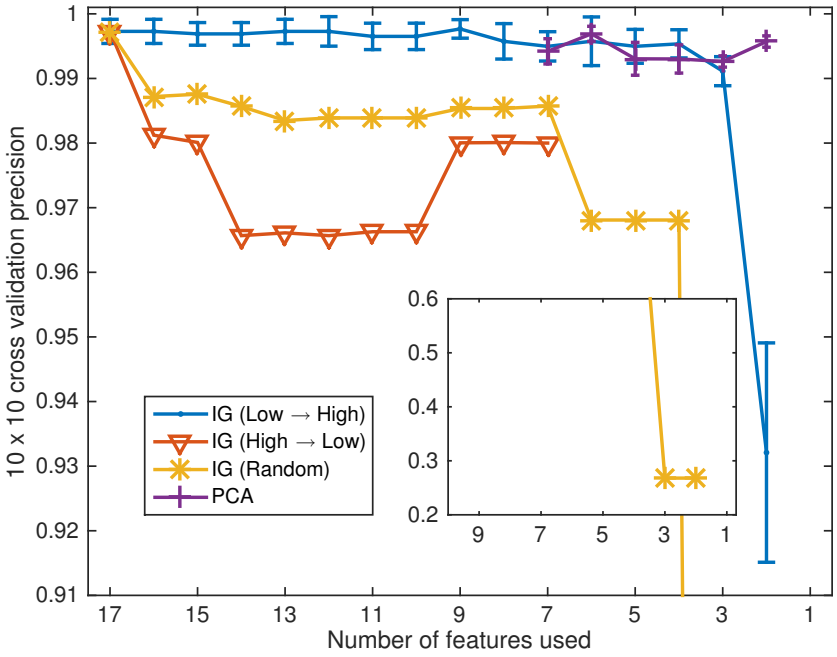

Figure 2.8: Precision performance with cross validation for all the features

in the dataset for analysis after removing the constant features and highly correlated features. The results for false alarm rate are shown in Fig. 2.13. We can observe the changing pattern of false alarm rate with the varying number of features is consistent with that of precision, which is shown in Fig. 2.12. Also, it is reasonable to maintain only 7 features as proposed above, because of the the low false alarm rate shown under such a setting.

Next, we are to present the 10 × 10-fold CV performance for the dataset with 7 features. The recall and precision for 10 × 10-fold CV are presented using 95% confidence intervals in Fig. 2.14 and Fig. 2.15, respectively. As mentioned before, normally, the feature with lower information gain has less influence on the classifier. The features are reduced one by one in three different ways: from the features with lower IG to higher IG (blue lines), from the features with higher IG to lower IG (red lines) and from a random order (yellow lines). The 95% confidence intervals of the last two reducing ways are all less than 0.003, not presented on the figures for clearer presentation. In Fig. 2.15, precisions have no meaning when all the connections are classified as good connections, because $TP + FP$ is equal to 0, referring to

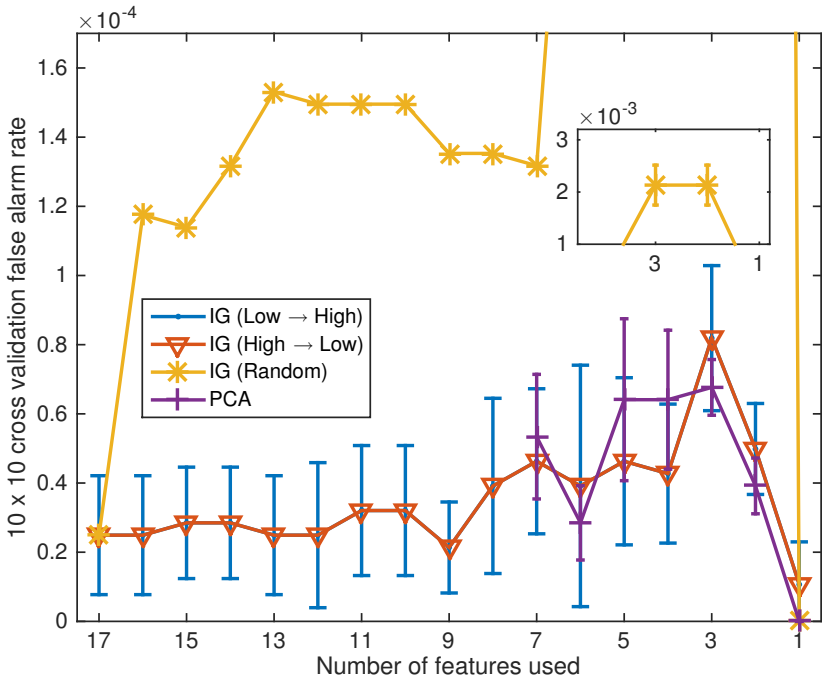

Figure 2.9: False alarm rate with cross validation for all the features

the definition of precision. From Fig. 2.14 and Fig. 2.15, we can see that reducing features randomly or from those with higher IG can make both recall and precision much worse than the original feature set, while reducing features with lower IG can keep good performance even when only 4 or 3 features are left in the feature set. Note that although the recalls of red and yellow lines when 4 features are left are marginally above the blue line with overlapping confidence intervals, their precisions are notably less than the blue line. Therefore, we can conclude that reducing the features with lower IG while keeping a few features with higher IG is an effective way to select feature subsets, able to accelerate training and detection while keeping high recalls and precisions. Dimension reduction based on PCA is also applied to the ICS dataset, shown as the purple lines in Fig. 2.14 and Fig. 2.15. The number of principle components is reduced from 7, since the first 7 eigenvalues keep about 100% variance. The order of reducing new features is from the ones with lower eigenvalues. The 10×10-fold CV performance is obtained by using the new features generated by projection. We can see that when the number of new features is 6, the 10×10-fold CV performance is better than reducing the original features with lower IG. Even

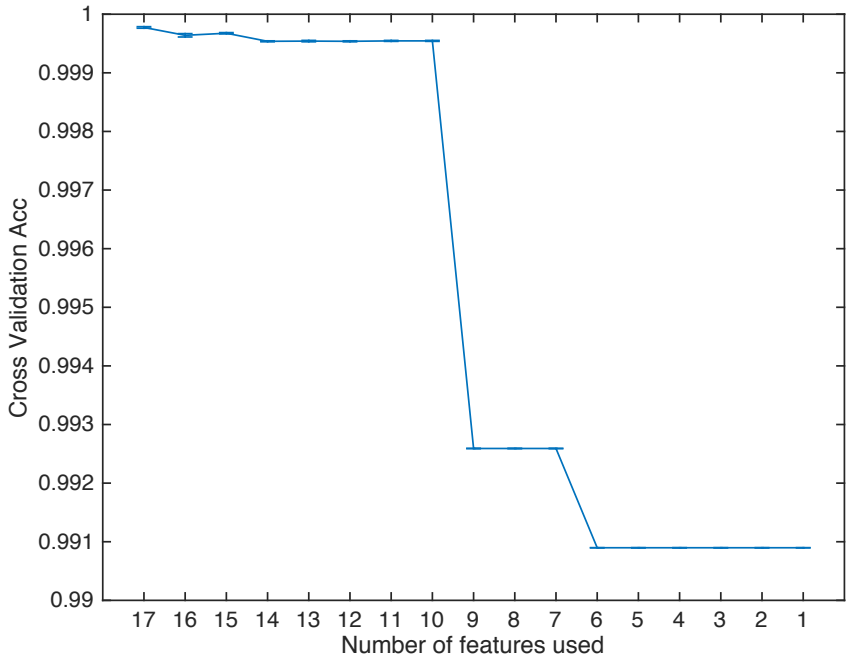

Figure 2.10: Accuracy performance dynamics with cross validation

when the number of new features is reduced to 2, the recall does not show a significant decline while having an increased precision. In Fig. 2.16, the results about false alarm rate are shown. It apparently shows when there is no static relationship between the number of features utilized and the false alarm rate achieved. Meanwhile, we can observe that the false alarm rate of IG (Low to High) with the decreasing number of features, shows a trend similar to that of PCA. Therefore, for the control networks, we can use reduced feature sets by PCA or IG criterion to accelerate and simplify the IDS process without much degradation of the detection performance.

Furthermore, recalls and precisions for each class are presented in Fig. 2.17. The yellow lines are the results of using the 7 features, comparable with the best work among 6 machine learning algorithms including Naive Bayes [49], Random Forests [35], OneR [46], J48 [63], NNge [36] and SVM [61] in [30], which used 12 features in the dataset and 2 of them are exploited in our dataset. The blue lines are obtained by using 6 new features generated by PCA, which improves the recall and precision of detecting the Address Scan attack to around 0.1. Although the number of this attack is only 2 in the dataset, it is still meaningful to improve the performance

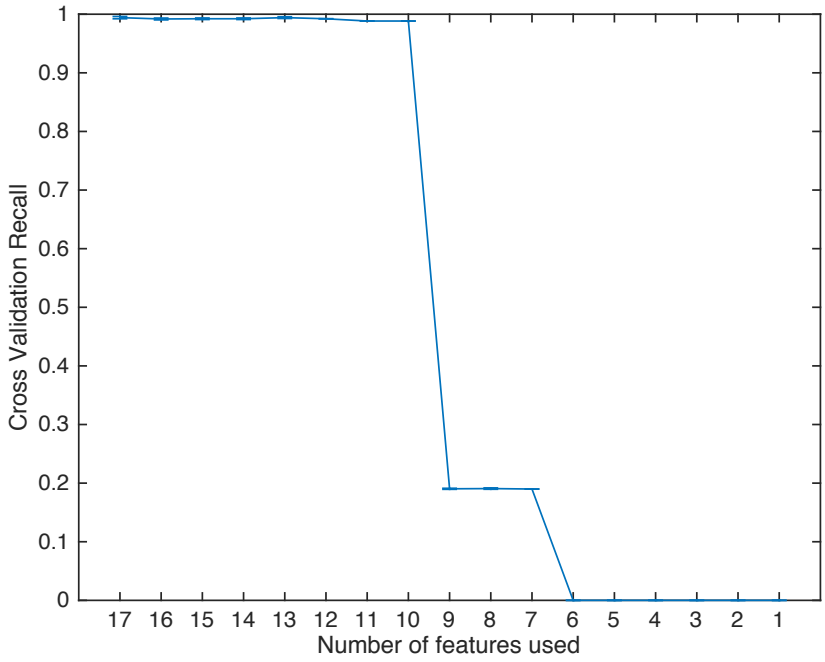

Figure 2.11: Recall performance dynamics with cross validation

of detecting them. Moreover, we process the Address feature among nominal features and add it into our preprocessed dataset with 7 independent features, making a total of 8 features. The results are presented with the red lines. We can see that the recall and precision of detecting the Address Scan attack get improved significantly to 0.700 and 0.567, respectively, which has obvious advantage over the best work among 6 machine learning algorithms in [30]. Through the analysis above, we can know that the multinomial logistic regression using selected 7 features is comparable with the best detection algorithm in [30]. What's more, the multinomial logistic regression using 6 new features generated by PCA and using an additional Address feature can improve the detection performance significantly for class Addr, while keep the detection performance for the other four classes. As for the false alarm rate, the results are shown in Fig. 2.18. The only difference among methods in terms of false alarm rate lies on the Good case, which is because some attacks are predicted as normal connections, while no normal connection is predicted as attack.

By receiver operating characteristics curve or ROC curve, the performance of classification with varied threshold settings can be verified. Therefore, the schemes dis-

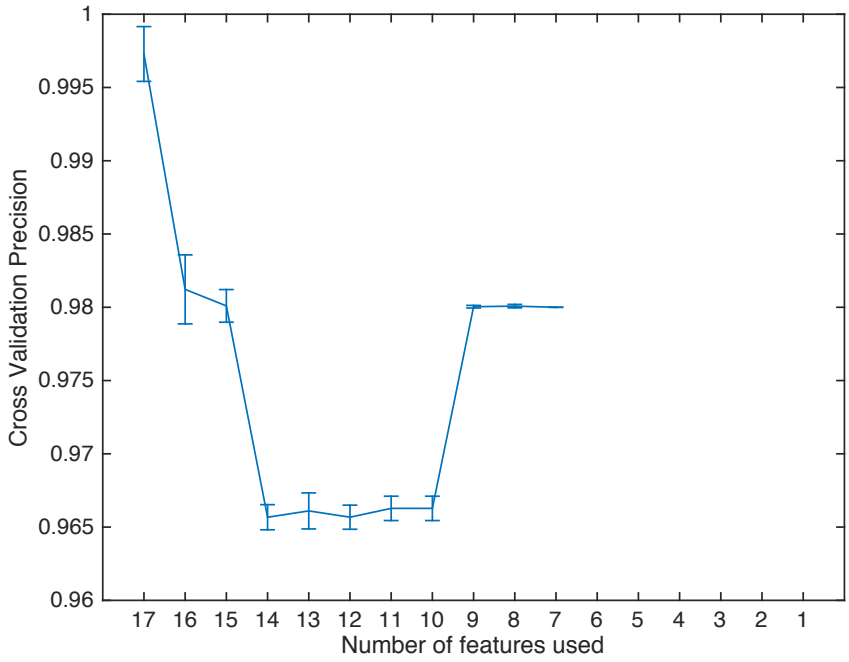

Figure 2.12: Precision performance dynamics with cross validation

cussed above are further compared based on the ROC curve and the results are shown in Fig. 2.19, Fig. 2.20 and Fig. 2.21. As the formulated problem is fundamentally a multi-class classification, each one of the 5 plotted lines in each figure represents the ROC result of one class versus the others. Due to the characteristic of the dataset, most classes show a very high true positive rate with a very low false positive rate. The main difference is on the decision of Addr class. On that category, among the schemes, the one with 8 features is the best and it is shown in Fig. 2.21. This justifies that the added feature has a positive effect. Being consistent with the experiments above, PCA with 6 features is better than the setting of 7 original features.

From all the above results, we can see that the design of HOIDS for SCADA systems is effective and feasible. The practical implementation of IDS clients distributed at global SCADA networks can be achieved by devices with low computing ability such as field-programmable gate array (FPGA), realizing online detection while substantially reducing financial budget. As can be seen another advantage of the HOIDS implementation is the high flexibility and portability. When meeting different requirements for SCADA networks protection, what needs to be done is to determine the

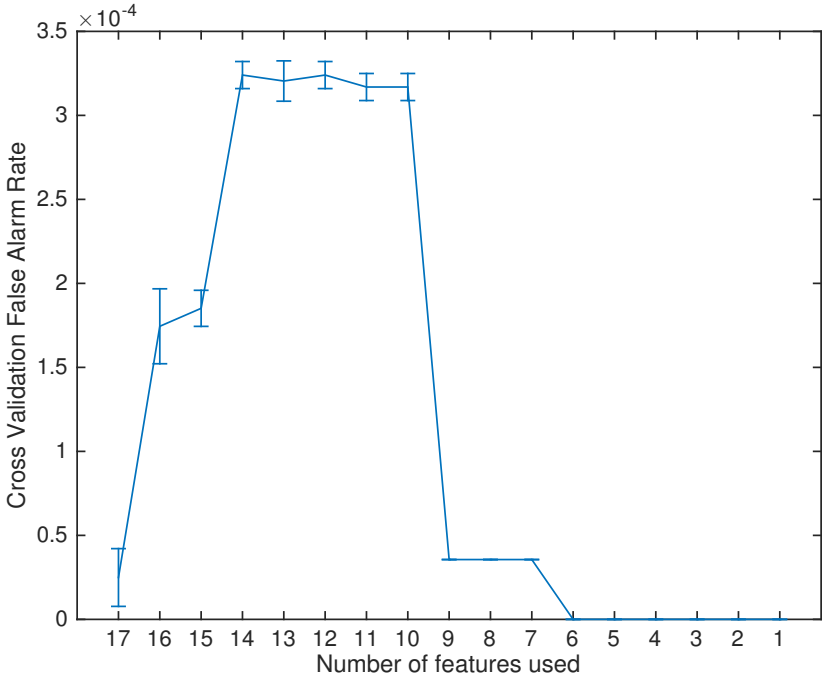

Figure 2.13: False alarm rate dynamics with cross validation

protection strategy and then modify the algorithms on the IDS server, since the most critical parts for the hierarchical online detection accuracy are principle generation and feature selection, both of which are only operated on the IDS server. For the IDS clients deployed on a large scale, there is no need to make major modifications. We choose logistic regression here, and other machine learning algorithms can also be considered according to practical scenarios. Also, we can choose different machine learning algorithms for different levels of SCADA network data flow, or exploit multiple algorithms simultaneously, which is known as meta learning. Likewise, features can be selected based on information gain, correlation coefficients, PCA or other feature selection methods based on the property of the certain dataset. What's more, IDS clients can be configured with signature-based techniques together to strengthen security. Therefore, the flexibility and portability of HOIDS implementation make it highly practical and beneficial to financial budget saving.

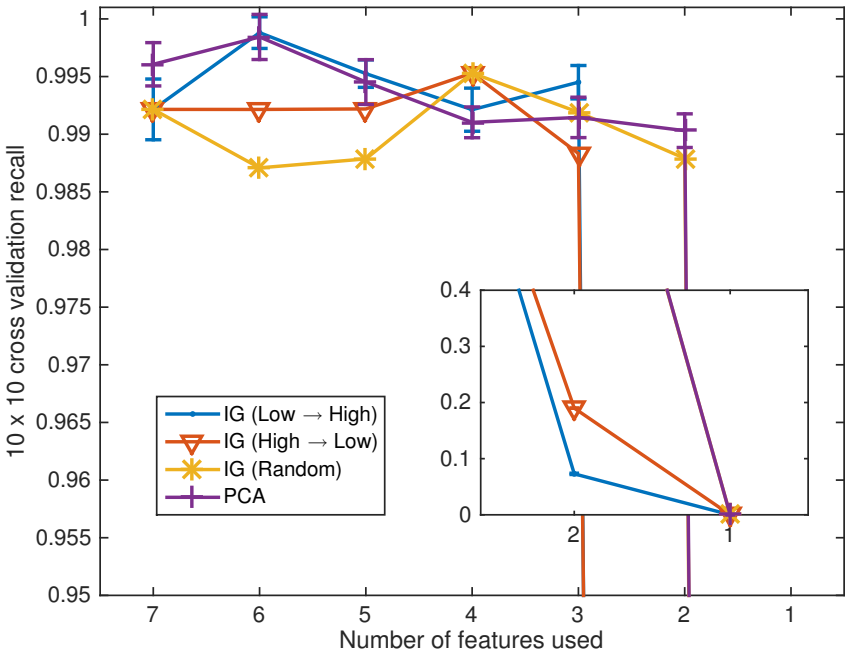

Figure 2.14: Recall performance with cross validation for optimized feature sets

2.3.3 Simulation using power system datasets

Section 2.3.1 and section 2.3.2 have presented the simulation results for general SCADA networks. This section is to evaluate the multinomial logistic regression using power system datasets [24] [59]. The datasets consist of fifteen sampled data files, which were sampled from one original dataset at one percent randomly. Details of the power system testbed for generating the datasets can be referred to the description file in [24]. There are 37 types of scenarios including short-circuit fault, remote tripping command injection, data injection and some other scenarios, which are correspondingly divided into three classes natural events (8), no events (1) and attack events (28). Here we use multinomial logistic regression to identify these three classes. For each sample in the dataset, they have 128 features including 116 types of measurements from 4 different phasor measurement units (PMU) and 12 features for the control panel logs, Snort alerts and relay logs.

Since the data files are randomly sampled from the original datasets, we use the first 5 data files for training, and another 5 date files (Data6, Data7, Data8, Data9, Data10) for testing. After preprocessing and scaling the datasets, we exploit

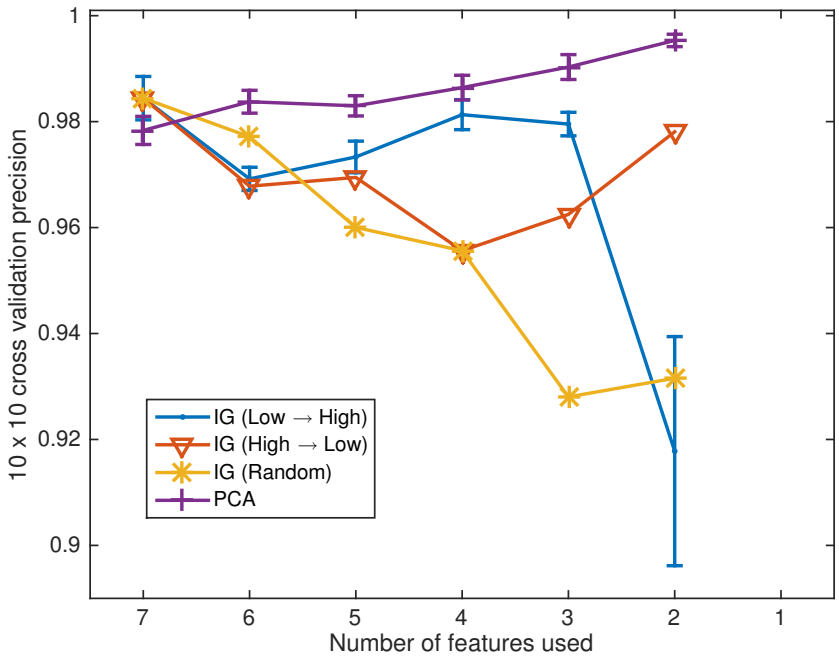

Figure 2.15: Precision performance with cross validation for optimized feature sets

the multinomial logistic regression using the 52 new features generated by PCA, by which the rate of over 99% variance of the whole dataset can be retained. We compare the training and testing results using the multinomial logistic regression (MLR) with those using LIBSVM algorithm [38], shown as Table 2.3. In the Table 2.3, we present in-sample performance by classification accuracy (Acc), sensitivity (Sensi), specificity (Speci) and training time, of which the unit measurement of time is second (s). The reason using sensitivity and specificity here is because attack events in the datasets have a major percentage in both the training datasets and the testing datasets, which is over 60%. In this case, the attack samples in the training dataset can have a significant impact on algorithm training, which might cause negative effects for classifying normal datasets. Specificity is an effective measurement which can show the classification performance for normal data, and sensitivity can present the detection performance for attacks. Here, the calculation of accuracy follows the same merit for the multi-classification used in section 2.3.1 and 2.3.2. And we should note that the calculations of sensitivity and specificity here use the merit of binary classification by treating natural and no events together as normal traffic, since the

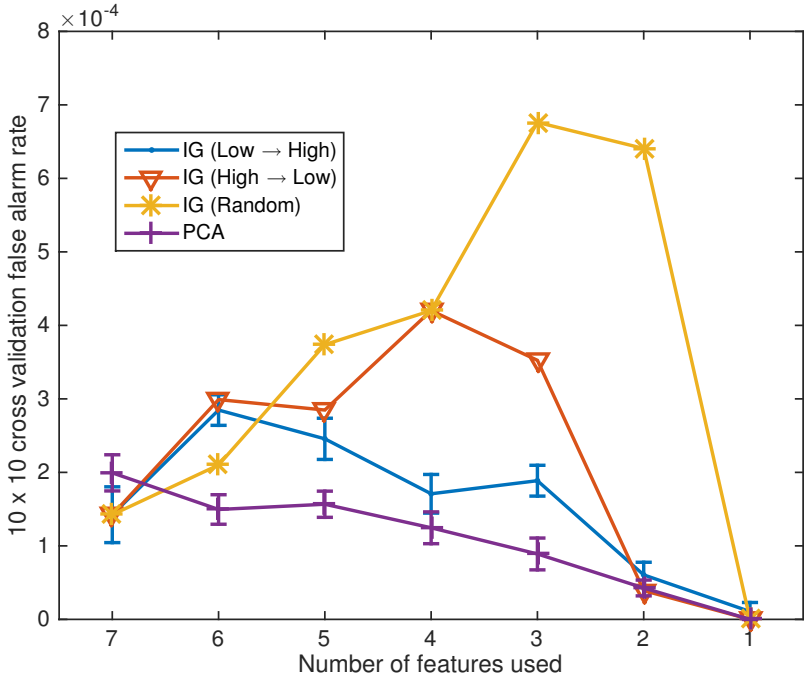

Figure 2.16: False alarm rate with cross validation for optimized feature sets

algorithm LR has better ability to identify natural traffic and no events while the algorithm LIBSVM identifies most of the natural traffic and no events as attacks, and we would not see the obvious difference by calculating the sensitivity and specificity using the multi-classification merit. From the in-sample performance in Table 2.3, we can know that the classification performances for MLR and LIBSVM are comparable. For both of the algorithms, their sensitivities are quite high comparing to the low specificity, which means both of the algorithms have accurate ability in detecting the attacks in the training dataset, while they are relatively weak in identifying the normal events in the training dataset. Also we can notice that the training time for MLR is around 2.5 hours, while LIBSVM is around only 3 minutes. The time obtained here are measured with the average CPU time by running 20 times of each algorithm. However, we should note that the average testing time for MLR is only 0.5580 seconds, while the testing time for LIBSVM is 28.32 seconds, which is too long to be accepted in terms of the latency requirement in a real-time control system.

Note that the implementation method, such as the programming language and functions used, can affect the calculation time taken by the proposed method and

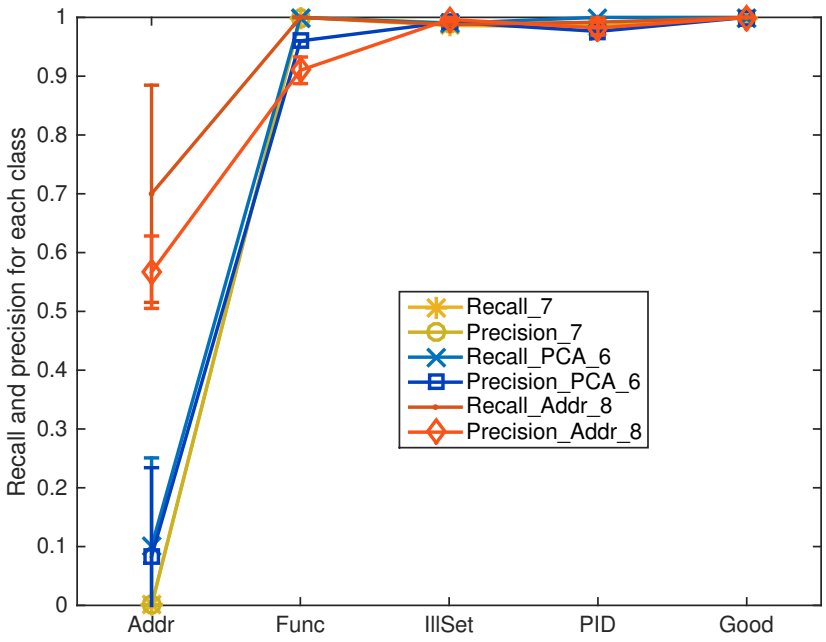

Figure 2.17: Performance of each class for multi-command injections

SVM. We have discussed the analytical time complexity of the proposed method in the above section. For SVM, the article [38] showed that for 2-class training, it takes at least $O(N)$ time per iterator, where N is the number of samples, while empirically the number of iterations may be higher than the number of samples; for the testing in 2-class SVM, its complexity is linear to the number of support vectors and the number of features. For K-classification, libSVM maintains $K \times (K-1)/2$ classifiers, each for one pair of the K classes. In testing, the data vector should be tested on all these classifiers, which somehow explains why libSVM is slower than the proposed method in terms of the time taken in testing.

Table 2.3: Performance Evaluation for Power System Datasets

Algorithms	In-sample performance				Testing Time (s)
	Acc	Sensi	Speci	Training Time (s)	
MLR	0.7169	0.9770	0.0915	$8.9090e+03$	0.5580
LIBSVM	0.7254	0.9888	0.0878	194.1010	28.32

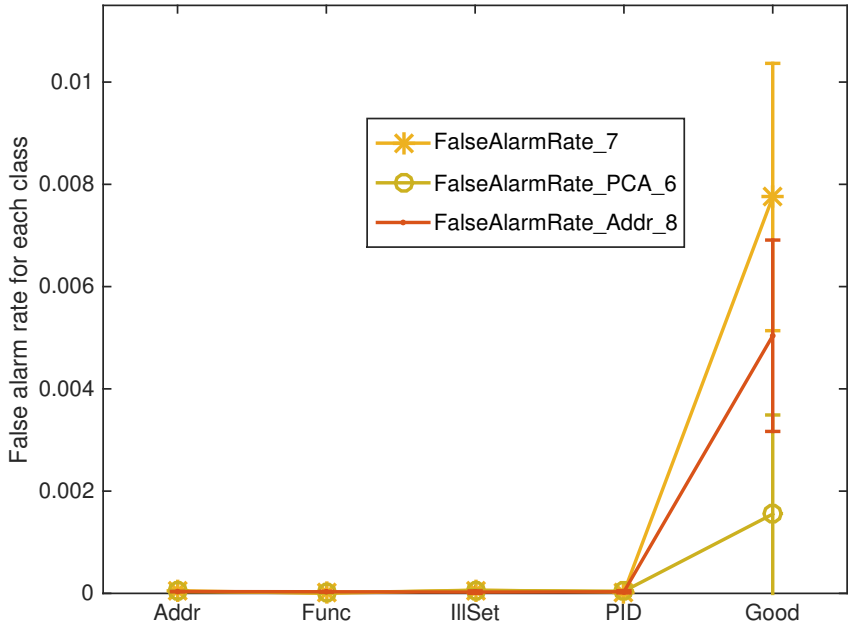

Figure 2.18: False alarm rate of each class for multi-command injections

Furthermore, we present the sensitivities and specificities of all the testing datasets we used in Fig. 2.22. The first testing datasets are the training datasets, of which the performance is denoted as in-sample ones. The other testing datasets Data6, Data7, Data8, Data9 and Data10, which have no common samples as the training datasets, are used to evaluate the out-sample classification performance. From the figure, we can find that LIBSVM has quite high sensitivities, of which the average and standard deviation are 0.9855 (corresponding to, on average, 98.55% of the attacks can be identified) and 0.0044, separately, in detecting attacks, while it has quite low specificities in identifying the normal events, of which the average and standard deviation are 0.0515 (corresponding to, on average, only 5.15% of the normal events can be identified) and 0.0093, separately. However, logistic regression has performances in terms of sensitivities and specificities as follows. The average and standard deviation of sensitivities using LR are 0.4804 (corresponding to, on average, 48.04% of the attacks can be identified) and 0.0158, separately. The average and standard deviation of specificities using LR are 0.5605 (corresponding to, on average, 56.05% of the normal events can be identified) and 0.0611, separately. We can see that the classi-

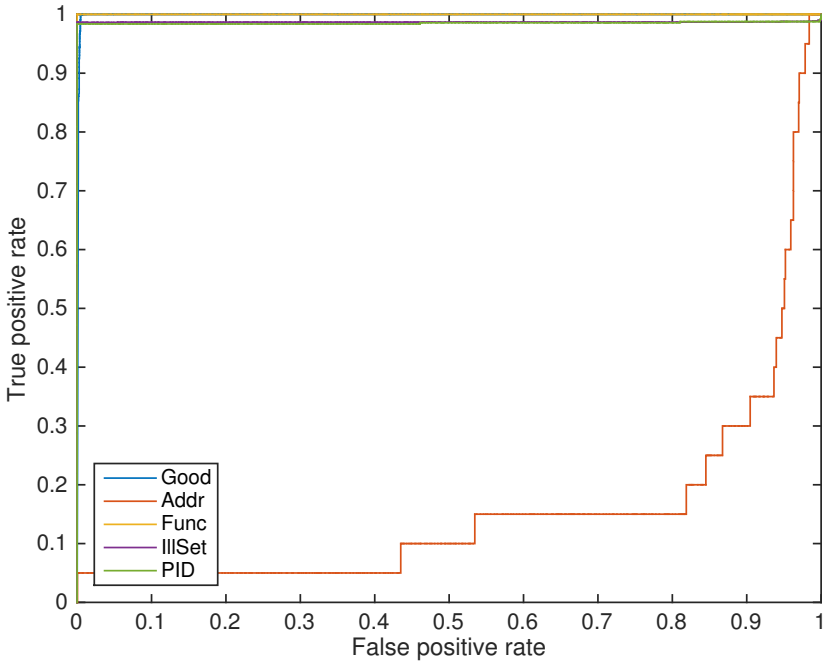

Figure 2.19: ROC curve for 7 original features

fication performances of both algorithms on the power system dataset are not that good. One reason could be that only these two machine learning algorithms are used here, while other algorithms might have better performance. Another reason could be that the power system dataset published so far is for studying the features, not necessarily perfectly classifiable. The learning curve using the power system training dataset is presented in Fig. 2.23. From the figure, we can know that adding more training dataset in the learning system is unnecessary to improve the performance. The detection performance might get improved when the dataset generator provides more useful features.

Compared with work [74], which proposed a distributed IDS for Smart Grid (SGDIDS) by deploying the analyzing modules, our IDS has obvious advantages. In [74], they described the network of Smart Grid as a 3-layer network topology, including home area networks(HAN), neighbourhood area networks (NAN) and wide area networks (WAN). Intrusion detection was realized by deploying an analyzing module(AM) at each node, using machine learning methods. The mechanism of SGDIDS was that if the node could not get the classification result, it would transmit its

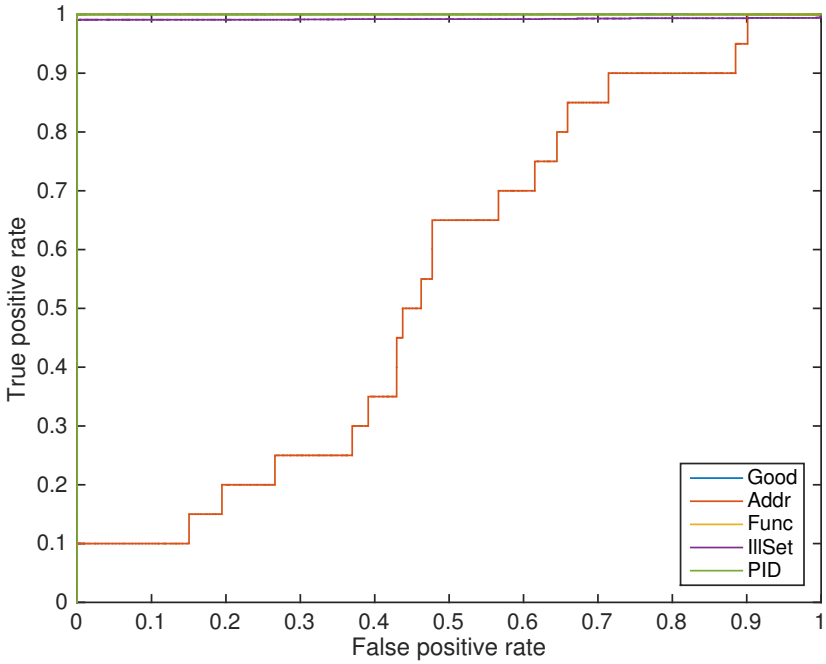

Figure 2.20: ROC curve for PCA of 6 features

dataset to a higher layer with better computing ability using the shortest routing path, which would be slow and unrealistic for the Smart Grid networks. It seemed that the distributed IDSs in the SGDIDS with the communication ability to the adjacent ones could reduce the communication overhead in the network. However, if the network state is good and the communication overhead can be controlled to satisfy the network capacity, this design would be unnecessary. Besides, it would be hard to implement and maintain as the AMs need to be coordinated since communication among them. Also, it would need a higher cost for these AMs as they are responsible for training in the design, requiring a higher computing ability. For HOIDS, each IDS client with limited computing ability can process the detection easily, with the efficient algorithms training in the control center, which is efficient, reliable and feasible.

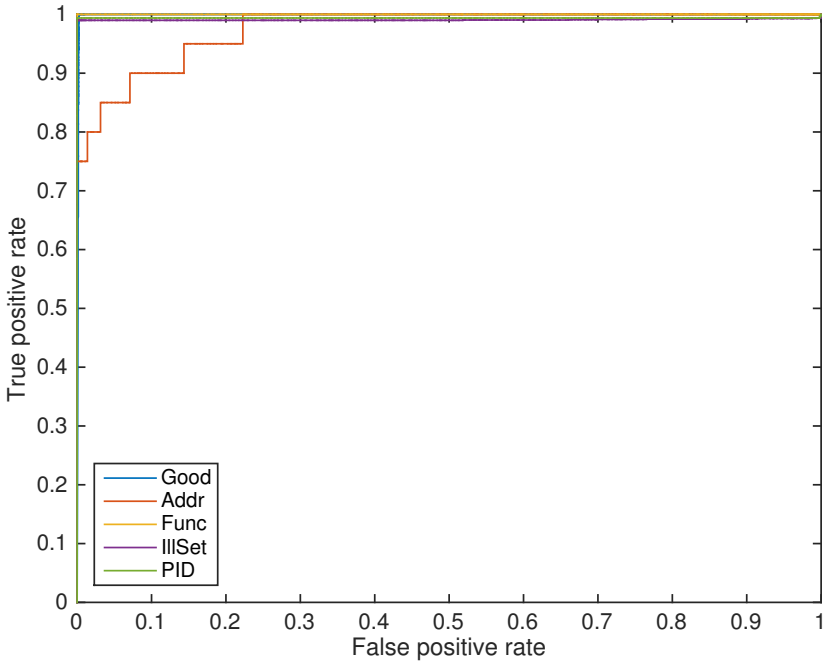

Figure 2.21: ROC curve for 8 features (with address added)

2.4 Conclusion

This chapter has proposed a novel hierarchical online IDS based on logistic regression and quasi-Newton optimization algorithm. The design of the IDS server in the control center and IDS clients distributed in the control center, substation and field networks can secure all cyber assets within the SCADA systems intelligently, while at the same time reducing the financial budget for the large scale systems. Furthermore, the design of intrusion detection with smaller sets of features based on feature selection techniques and dimension reduction approaches can accelerate the detection process to satisfy the real-time requirements of SCADA systems and facilitate the implementation of the hardware devices.

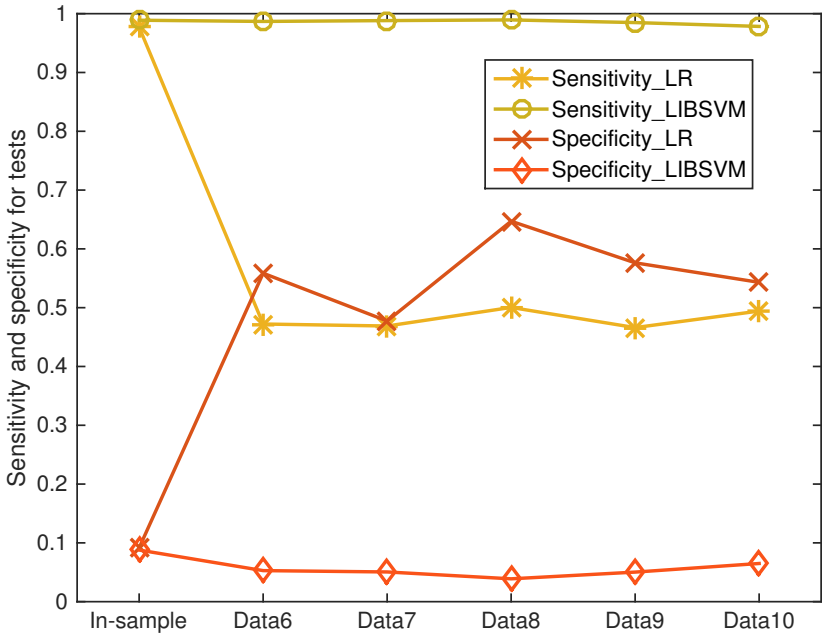

Figure 2.22: Performance for Evaluating Different Testing Datasets

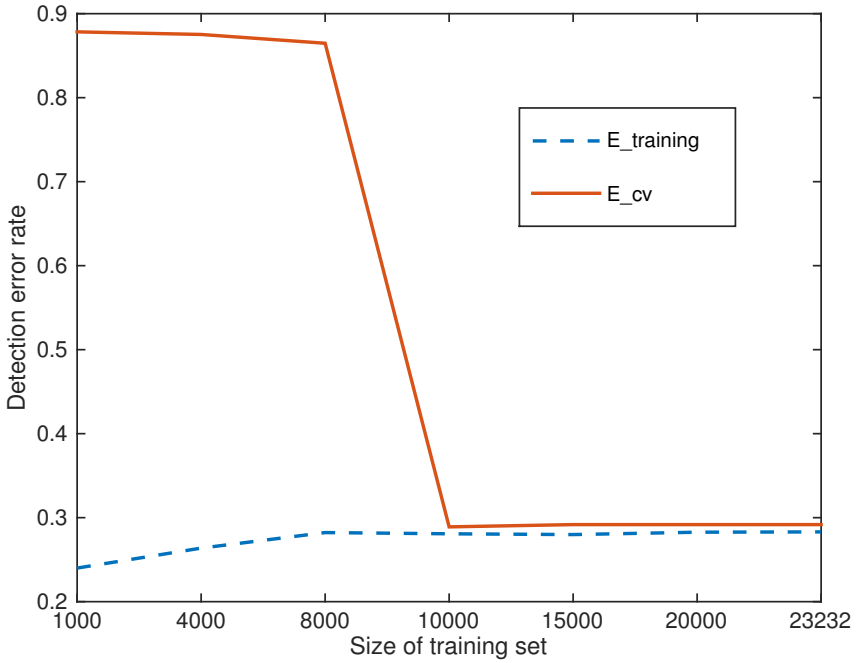

Figure 2.23: Learning curve for power system training dataset

Chapter 3

SCADA Network Testbed Implementation

Nowadays, the growing awareness of SCADA security has motivated researches on protecting SCADA systems, not only on theoretical analysis, but also on practical experiments. To explore the SCADA network systems, we can set up a testbed for SCADA network security research. Simulating the working environment of SCADA networks, the testbed helps to implement and verify the idea of anomaly-based SCADA IDS proposed in Chapter 2. Through the testbed, we can record the network data flows, extract significant features, exploit detection rules based on machine learning algorithms and realize online detection. The testbed discussed in this chapter is implemented by software, being flexible and adjustable. It simulates a typical tank system communicating with Modbus protocol, one of the most common control protocol in the control system.

The work done in this chapter extended the earlier work [73] with multiple important revisions introduced to the system. Specifically, to meet our research target of anomaly-based intrusion detection system for SCADA network, the following changes are made: a supervising network with an HMI representing the system in the control center replacing the external network in [73], a generic Linux-based middle box for network functions, and a new powerful intrusion detection system for both anomaly-based and signature-based analysis. With these modifications, the testbed is a more comprehensive and reasonable software-based implementation of SCADA network testbed, which can contribute to more advanced research of intrusion detection in SCADA networks including anomaly-based detection using flexible and effective

feature extraction approaches.

The outline of this chapter is organized as the following three sections: Section 3.1 presents the SCADA testbed architecture, including design principles, the overall network topology, the HMI-PLC-Sensors deployment and the fundamentals of the new intrusion detection framework. Section 3.2 introduces testbed working procedures, in terms of handling both normal and abnormal network dataflows. Section 3.3 gives the testbed configuration details, including the details for the virtual machines and the network configuration.

3.1 Testbed Architecture

3.1.1 Testbed design

The design principles of the testbed are as follows: first, for representing a practical SCADA system, the testbed is more to simulate both the functions and security threats of existing SCADA systems than to show a paradigm of making the system more secure; second, for the tractability, the components in the SCADA system is implemented briefly and in a representative way, instead of with all details; third, for the extensibility, a generic Linux system is used as the container to simulate the networking functions, such as routing, firewall and IDS.

In a typical SCADA system, the networks involved are Internet, control center network (or termed as supervising network), and field networks (or termed as control networks). The testbed currently focuses on simulating the supervising network and control network. Thus, the main tasks of implementation lie on the simulation of entities or devices in these two types of networks as well as their connections. The architecture of the proposed testbed is illustrated in Fig. 3.1, which consists of the two types of networks discussed above, and the admin network for the simulation control. The testbed implementation is built based on the previous version in [73], with changes in two nodes while the rest of the testbed nodes almost keep the same as [73] except some network configuration modifications: one is the Bro hosting router node replacing the Snort hosting node, and one is the added HMI node in the supervising network. Control networks is connected with supervising network by a device termed as Router and implemented based on a generic Linux system. Simulated Internet (or termed as external network) could be added to the testbed, and connected to the supervising network via a network device same as the one currently used between the

supervising network and control network. It is not implemented in the testbed so far as technically it will not make a difference on the implementation of Router in the testbed, beyond being replicated to the intersection between the supervising network and external network. Note that it can be easily extended in the future, when we need to differentiate the handling of threats from Internet and from the control center.

In terms of the entities simulated, it is different from the ones proposed in [73] at least in two aspects: first, a human machine interface (HMI) was created in the supervising network which represents the job of control center; second, the middle box between the supervising network and control network is changed to a generic linux-based system with the intrusion detection tool Bro [10] installed, substituting Snort [14]. The first change makes the testbed closer to the real systems, and the second brings the support to more sophisticated and advanced intrusion detection design and provides a higher extensibility for network function simulation.

3.1.2 Testbed network topology

As shown in Fig. 3.1, the control network is within the IP range 10.0.0.0/24 and used to represent the field networks, which contains programmable logic controller (PLC) PLC_Master, and multiple virtual PLCs and sensors Mod_Slave. The design of virtual PLCs is based on Honeyd technique using Nova configuration [6], which can be referred to [73]. PLCs and Sensors are communicating with Modbus protocol [20], one of the most common control protocol in the industrial networks currently. In terms of Modbus protocol, Mod_Slave is configured as the server of its client PLC_Master. PLC_Master and Mod_Slave simulate a typical industrial tank system, similar with the tank system in [56].

The supervising network is within the IP range 192.168.100.0/24 and simulates the control center. A human machine interface (HMI) is added to the supervising network of the testbed system as illustrated in the figure, being one of the most important contributions that I made to the testbed implementation, since it is used for generating the normal data and simulating the function of SCADA supervising network. In practical SCADA systems, people working in the control center use some computer interfaces to input control commands and collect information, termed as HMI. HMI can represent such a component of the system as well as the security threats introduced by it. Note that although each individual PLC can have its own HMI service for the remote access, termed as PLC_HMI, to control the PLC directly

through it is less effective in large scale systems. Differently, the HMI introduced here is a representative of functions related to centralized control and management used in the control center. Specfically, due to the largely different computational and storage capability between PLCs and control center, HMI together with the information system such as historians in the control center can aggregate the information from distributed field locations and allow operators to efficiently apply a centralized control. Therefore, it is simulated in the testbed.

For the currently implemented functions of HMI, it not only can present the state of the PLC_Master on the web console through reading, but also can control the pump speed by writing. The appearance of web interface of HMI in Fig. 3.2 is like the Fig. 8 in [73], which is similar to the demo figure on MBlogic website [1]. However, we should note although the appearance of the web real-time monitor on HMI is like the Fig. 8 in [73], the principles of monitor presentation are different. Fig. 8 in [73] gets the monitor data by reading its own system address, while the web real-time monitor on HMI in the thesis inquires the data by Modbus communication between HMI host and PLC_Master host. Similar to the practical SCADA systems, the HMI here supervises the PLC and is able to control the PLC. They communicate with each other using Modbus protocol as well. Normally in the supervising network or in the control center, there also exist other components, such as engineering workstations (EWS), historians, application servers and so on, which can be extended to in the future.

The attackers in the testbed are deployed in the supervising network, forming a reasonable hacking scenario. Note that moving them to the Internet is also supported by the current testbed architecture. There are Kali [13], Nexpose [21] and Samurai [22] attacking toolkits here to launch the attacks to the tank system networks, including scan attacks, command injection attacks, denial of service attacks and so on. Currently, most attacks supported in our testbed experiments are launched by Kali and Modpoll tool [19]. Kali Linux is a Debian-based open-source software, containing more than 600 hacking tools. Modpoll tool is a command line based communicating tool based on Modbus protocol, and can also be used for hacking the networks using the Modbus protocol. Besides, all the hosts in the testbed are deployed with virtual machines, which can be installed easily on a general computer. They are all set on the administration network which is 172.16.1.0/24 for being remotely operated and for the connection to the Internet.

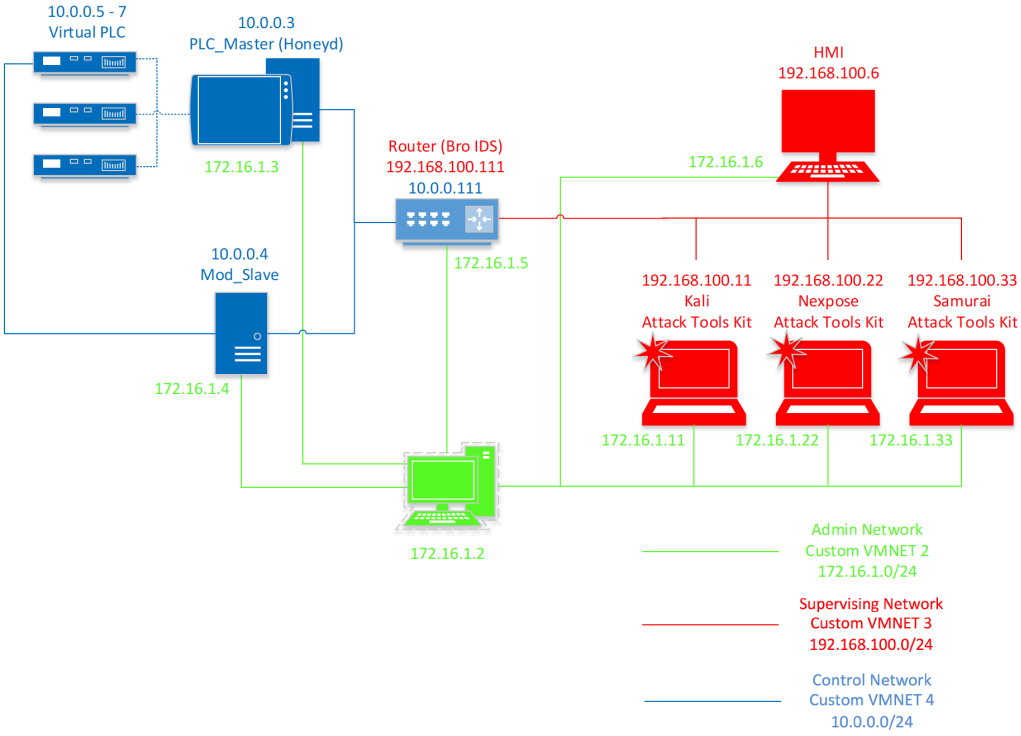

Figure 3.1: SCADA testbed architecture and network topology

3.1.3 Network function simulation

A generic Linux system is set up between the supervising network and the control network for the simulation of network functions including the support of anomaly-based intrusion detection. This component is illustrated as Router in Fig. 3.1. Using the generic Linux, we can implement different network functions such as routing, firewall, IDS, Network Address Translation (NAT), etc. In the testbed, the Router implements several functions simultaneously: first, it achieves the routing and forwarding function as it makes the two networks connected; second, it fulfils the IDS function, which means that it monitors and analyzes the traffic for intrusion detection. Note that although these functions may have different implementation in the real system, e.g., implemented by separated devices, our current design of Router can achieve our design purpose of the testbed, i.e., to simulate the network functions in a SCADA system.

In the Router, some tools can be installed such as the network traffic analyzer

Wireshark [15] and Bro [10] for traffic learning. Bro can be set to supervise some certain network interfaces. Compared with the Snort-based open-source software deployed in the previous version, the Router deployed here is a general-purpose Linux host, through which, people can easily and flexibly add new networking features and functions to the testbed for different experiments. Using our current implementations as examples, the routing between the supervising network and control network is set up so that they are connected, and the Linux-based intrusion detection framework Bro is installed which can monitor the traffic between the supervising network and control network and the traffic inside each of them.

For the routing function, we simply implement it by the static routing function in the Linux kernel, where the ip command is to used add routing rules. Currently the static routing based on Linux kernel is enough to connect the two networks in the testbed. Note that based on the generic Linux system, we can implement routing with different tools or for extra features. For example, it is common that we can implement NAT based on iptables so that private ip addresses can be used in the control network, and not exposed to the other networks. The IDS function implemented by Bro in the Router can identify the malicious traffic. The network card can be set to monitor the traffic in promiscuous mode, in this way, the malicious intra-network traffic can also be detected.

As mentioned above, this generic Linux-based middle box can provide a better extensibility. Therefore, we can easily add the firewall function into the testbed, although it is not implemented currently. For example, upon a malicious behaviour detected by the IDS, we can easily take the action of blocking certain IP address by adding some iptable rules to the Linux middle box.

3.1.4 HMI-PLC-Sensors deployment

The tank system is originally designed by MBLogic [8]. MBLogic is a collection of software packages to simulate typical industrial processes. Here we exploit and make configurations to the MBLogic Demo tank system and get the HMI-PLC-Sensors tank system.

PLC_Master installed with MBLogic is configured as a client communicating with Mod_Slave configured as a server installed with MBLogic. HMI in the external network, installed with MBLogic is configured as a client communicating with PLC_Master. Speaking in another way, two pairs of client/server are formed, which

are PLC_Master/Mod_Slave and HMI/PLC_Master. The configurations are made on the config files in the mblogic including mbclient.config, mbhmi.config, mblogic.config, mbserver.config and so on. HMI, PLC_Master and Mod_Slave are communicating with Modbus protocol, which uses the function code to perform the command action. The function code can be referred to Appendix A.3.

PLC_Master reads the tank level sensors from Mod_Slave (function code is 3) and writes the pump speed value to the pump speed valve of Mod_Slave (function code is 16), as shown in Fig. 3.3. Note that Fig. 3.3 is almost the same as Fig. 9 in [73], but shows more details about reading and writing registers for continuous integration and corrects 4 errors of address presentation in the previous work. HMI presents the state of the PLC_Master including the tank level values and the pump speed presentation on the web console through reading (function codes are both 3), but also can control the pump speed by writing using Modpoll tool [19], as shown in Fig. 3.4. In Fig. 3.3 and Fig. 3.4, the Logic Addresses are the addresses for the PLC or sensors. Actually, they are corresponding to system addresses. Therefore, operations to the system addresses can make effect in the configuration. In Fig. 3.3 and Fig. 3.4, H, L, HH, LL are the thresholds limiting the capacity of water in each tank to protect the tanks. The tank system is supposed to generate alarms when the water level is over HH (95) or below LL (5), and to launch warning signals when the water level is between H (80) and HH (95), or between L (20) and LL (5).

As mentioned above, the configurations are made in mbclient.config, mbhmi.config, mblogic.config, mbserver.config and other files. We give the mbclient.config file of the HMI module added in the external network as below:

[PumpSpeed_TankLevels]
repeattime = 100
fault_coil = 1330
protocol = modbustcp
readholdingreg = function=3, uid=1, memaddr=32210, qty=1, remoteaddr=32210
readholdingreg1= function=3, uid=1, memaddr=42210, qty=1, remoteaddr=42210
readholdingreg2= function=3, uid=1, memaddr=42211, qty=1, remoteaddr=42211
type = tcpclient
fault_holdingreg = 1330
fault_inp = 1330
retrytime = 5000

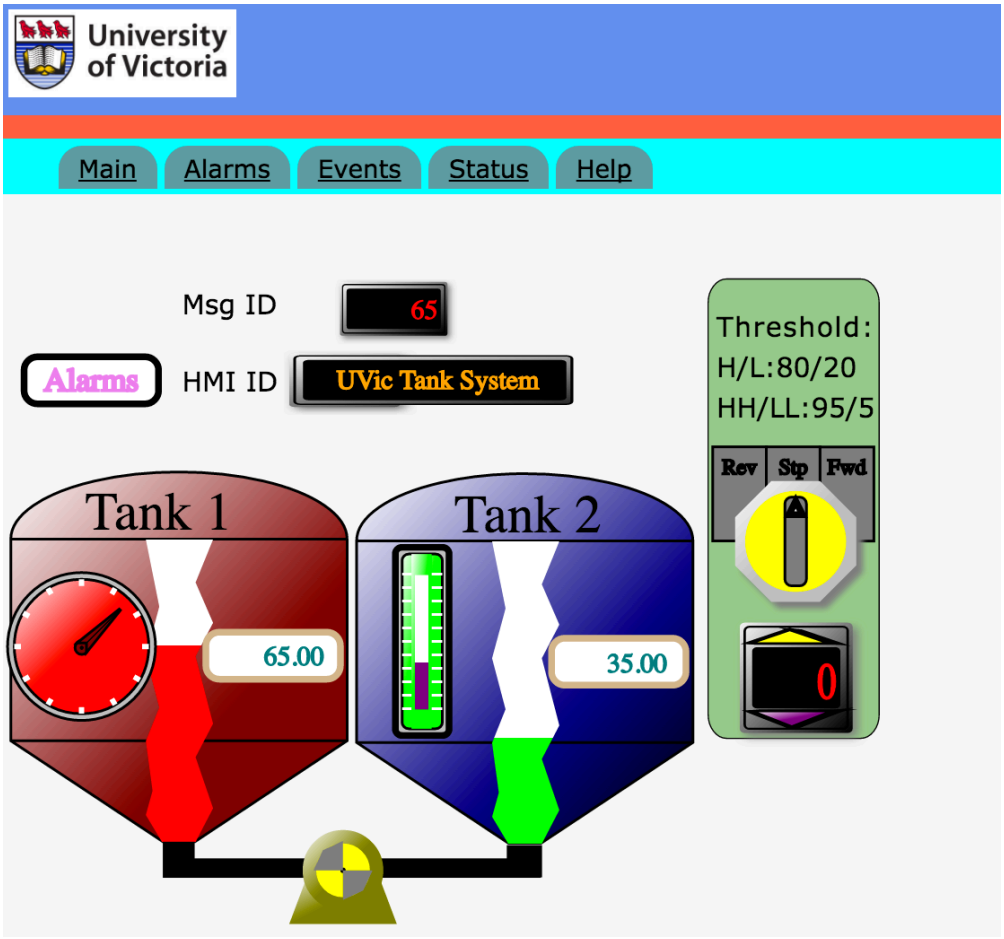

Figure 3.2: The web real-time monitor on HMI of SCADA tank system(after [73])

host = *10.0.0.3*
fault_inpreg = *1330*
action = *poll*
cmdtime = *100*
port = *502*
fault_reset = 65282

The above mbclient.config file of HMI creates 3 client agents on HMI communicating with their Modbus Server (the port for Modbus protocol is 502), i.e., PLC, to read the contents of the pump speed and the water levels of two tanks by specifying the system addresses on their Modbus server PLC, of which the IP is 10.0.0.3. The contents of the pump speed and the water levels are stored in the holding registers.

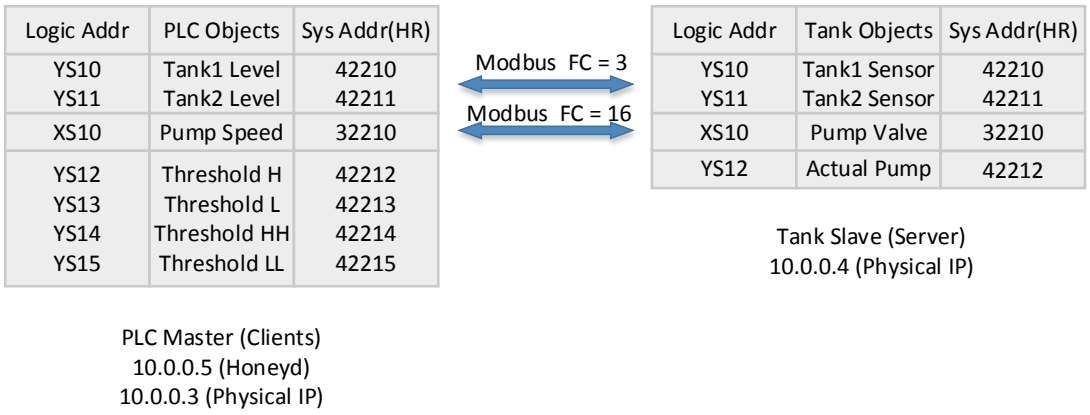

Figure 3.3: PLC-Sensor communication configuration (after [73])

There are 4 data types in total in terms of Modbus protocol, including discrete inputs, coils, input registers and holding registers [20]. Holding registers are read-write 16 bit integers. Usually, they are used to denote analogue outputs or internal numbers which can both be read and written by the user. The settings of mbclient.config files for PLC and Sensors are in the same way as HMI.

We should note that some figures in these two subsections above are presented in a style which is adapted from [73] with some differences, such as Fig. 3.1, Fig. 3.2 and Fig. 3.3, for the continuous integration. Differences between the current configurations and the previous ones are obviously shown when comparing the works.

3.1.5 Fundamentals of intrusion detection using Bro

To cope with the various demands and algorithms in intrusion detection, the framework and tool Bro [10] is introduced to the proposed testbed. Bro is an open-source, unix-based network intrusion detection system (IDS), developed at Lawrence Berkley National Lab in 1998. The fundamentals of Bro is shown as Fig. 3.5. It can capture and filter the network packets by libpcap tool. Then Bro puts these packets together to form corresponding events, i.e., actions performed by certain packets, e.g., http request or reply. Next, these events will be observed or compared by the policy script

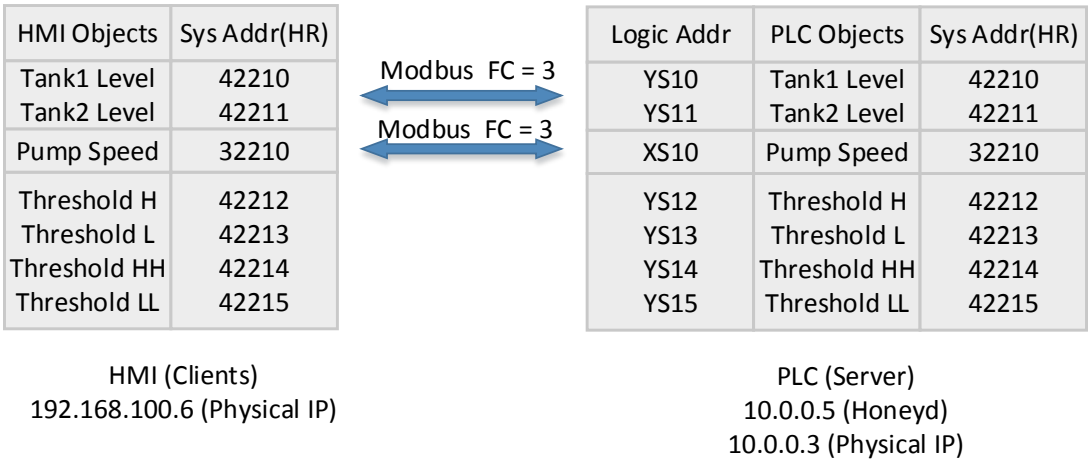

Figure 3.4: HMI-PLC communication configuration

interpreter, and Bro will generate notice.log files if it finds some packets wired or abnormal.

For the Bro configuration, it is necessary to set three files: /nsm/bro/etc/broctl.cfg, /nsm/bro/etc/networks.cfg, /nsm/bro/etc/node.cfg. For example, to assign a specific network interface for Bro to observe and make records. In the testbed, the interface eth1 192.168.100.111 of Bro is being observed. We can use the information summarized by Bro's Event Engine as features and use them to generate more features to depict the normal connections and abnormal connections. When preparing enough dataset, we can use them to train machine learning algorithms and get the detection principles. Then we implement the principles to Bro's Policy Script Interpreter and realize the online detection. In the process of operating Bro, it is necessary to learn the Bro script, the scripting language of Bro. Using Bro script to extend and customize Bro's analyzing function, e.g., extract network features and design detection principles, is the primary way, efficient and flexible in terms of network analysis development. Bro scripts can be learned from [17]. Bro is designed for the networks with high speed of 80Mbps. Although Bro is not designed as multithreaded, it can be extended the workload across multiple processor cores when the limitation of a single core has been reached. This is the feature of Bro to build a larger detection or

analysis system, called Bro Cluster, shown as Fig. 3.6.

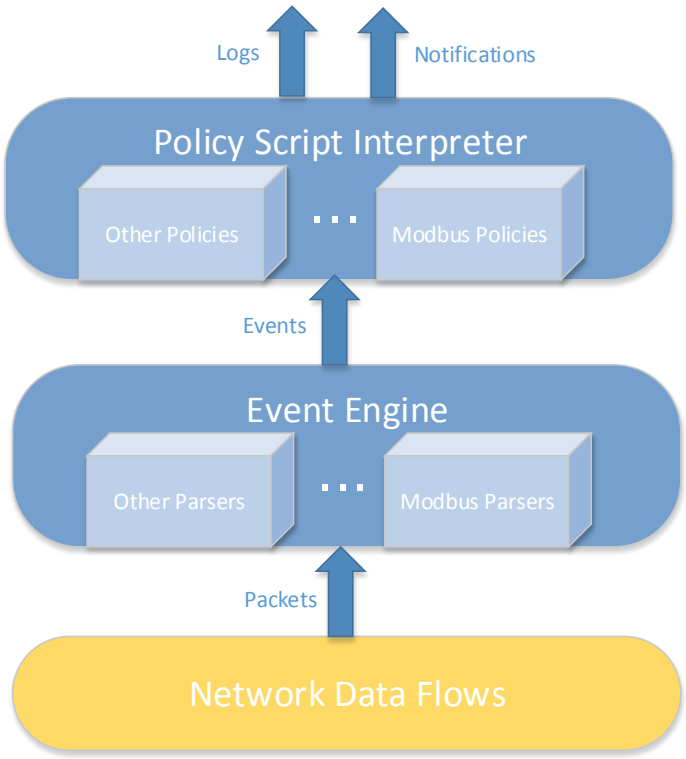

Figure 3.5: The working mechanism of Bro IDS

Another intrusion detection system Snort [14] also can be deployed in the testbed for intrusion detection, but between Snort and Bro, they each has some own advantages. In terms of the network analysis ability, Bro is more powerful. The pros and cons of Bro are compared with Snort, listed below as Fig. 3.7. Snort and Bro are both open source softwares. Currently IDSs in the industrial system use Snort to protect the networks. Bro has been used for network research for over 10 years due to its powerful analysis for the network data flow and feature extraction. As shown in Fig. 3.7, Bro is designed for high speed network and has a quicker dataflow processing speed than Snort. For the attack detection rules, Snort has many predesigned downloadable rules, only for signature-based intrusion detection. Bro not only has lots of predesigned detection rules, but also can be designed by researchers based on different scenario requirements using its own scripting language Bro script, which is more sophisticated than Snort, and suitable for both signature-based and anomaly-

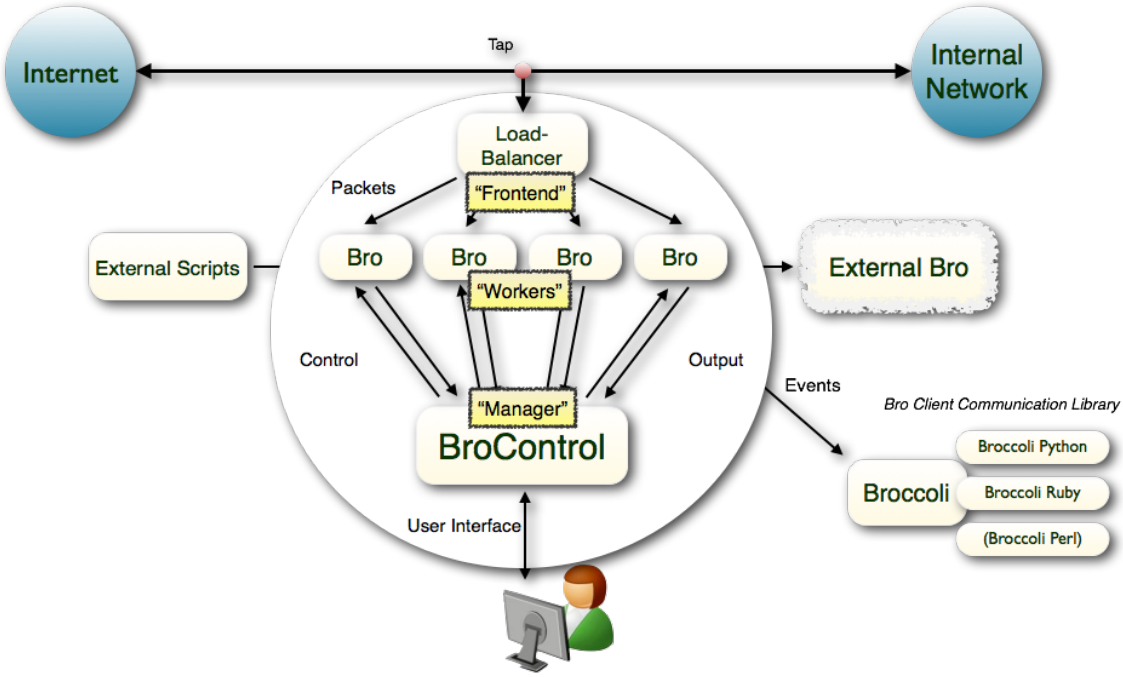

Figure 3.6: The fundamentals of Bro Cluster [11]

based intrusion detection. In terms of flexibility, Bro is completely customizable since researchers can write their codes for extracting specific new features and new detection rules while Snort has limited customization. The advantages of Snort are that its installation is easier than Bro, and it supports multiple operating system (OS) such as Windows, Mac OS and Linux, while so far Bro can only support the Unix-based OS. What's more, Snort has lots of documents available, while the documents of Bro are limited, compared to Snort. During the research, we can use them based on different requirements.

3.2 Testbed Working Procedures

The testbed simulates a HMI-PLC-Sensors tank system. The pump can topple over the water from one tank to another by setting the pump speed properly. Positive pump speed value is moving the water from the left tank to the right one, and negative speed is moving the water in the opposite direction, referring to the demo figure on MBlogic website. PLC is to monitor the states of the Sensors including the

"Bros" and cons		
Item	Snort	Bro
Speed	Medium	High speed network
Signatures	Downloadable rules	Sophisticated (pre + design)
Flexibility	Limited customization	Completely customizable
Installation	Easy	Difficult
OS Compatibility	Multiple OS support	Unix-based
Documentation	LOTS available	Very limited

Figure 3.7: Comparison between Snort IDS and Bro IDS

water level values of two tanks and the pump speed, and able to control the pump speed of the tanks by writing the value into the system address of the register holding the value of pump speed. HMI can inquire all these information about Sensors by reading from the system addresses of registers in PLC to monitor the whole tank system. Since HMI is located in the supervising network, it has to communicate with the control network through the Router, on which we installed the Bro network analyzer on it. The attacking toolkit of Kali is also in the supervising network. It can issue the attacks to the tank system. Similarly, the attacking dataflow has to pass the Router to hack the tank system, which can be caught and recorded by Bro analyzer. If Bro detects weird dataflow or attacks, it will generate the warning information or alarming information to the notice.log file. This section is to illuminate the working procedure of the testbed for SCADA network security study.

3.2.1 Generating normal and abnormal network dataflow

The design aim of this testbed is to simulate the SCADA system network and to implement the anomaly intrusion detection based on machine learning algorithms. Therefore, it is necessary to have the attacking network dataflow, as well as the normal network dataflow in SCADA system, which can be extracted to be data flow features to train the machine learning algorithms to generate the detection principle. This subsection is to introduce the techniques used in the testbed for capturing the

normal network data flow and abnormal data flow generated by launching attacks from Kali.

To capture and collect the network data flow including the normal and abnormal data flow, both Wireshark software and tcpdump tool can be used effectively. Here, we first introduce the way using Tcpdump, in which we used in our testbed. Tcpdump tool can be used in the command line. To assign the supervised network interface and the pcap file to be written is necessary for the command. For example, to capture the dataflow from the eth1 and write the data into capture_data_example.pcap file, the command and presentation information are as below:

root@ubuntu: # tcpdump -i eth1 -w capture_data_example.pcap
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes
2036 packets captured
2040 packets received by filter
4 packets dropped by kernel.

From the command and information above, we can know that the tcpdump tool is listening on the eth1 network interface card, and it has captured 65535 bytes, which are included in 2036 packets.

Actually, Wireshark can be used in the similar way as Tcpdump, since their fundamentals of capturing the network dataflow is almost the same. The advantage of Wireshark is able to present the parsed network data on a graph use interface, as is shown in Fig. 3.8. In the figure of Wireshark analyzing interface, the first part is the presentation of the packet information. We can see that the 14th packet is one of the three Transmission Control Protocol (TCP) handshake packets between the two hosts Kali 192.168.100.11 and PLC 10.0.0.3. And the 17th packet is the query packet using Modbus/TCP protocol with Function code 17 from Kali to acquire the device information of PLC, which is a malicious connection to get access to the Slave ID of PLC, according to Appendix A.3. The second part of the information shown in the Wireshark user interface is the summary of a certain packet from the perspective of four-layer TCP/IP protocol architecture [9], i.e., Network interface layer, Internet layer, Transport layer and Application layer. In terms of the 17th packet in the figure, we can see that it is using Modbus/TCP protocol in the application layer, TCP in the Transport layer and IP in the Internet layer based on the Ethernet technique in the Network interface layer. And each layer protocol can be expanded in more

detail. For example, there are details of the packet content such as the transaction identifier, protocol identifier, length and unit identifier in terms of the Modbus/TCP application protocol. The third part shown on the user interface is the raw network data in unit of bytes, which can also be parsed to the cleartext by Wireshark if the protocol being used is not encrypted.

An example of normal network data analysis presented by Wireshark is shown in Fig. 3.9. We can see that the 4877th packet is sending a query message from HMI (192.168.100.2) to PLC (10.0.0.3), and the 4878th packet is responding the value in the register from PLC to HMI. The 4880th packet and the 4881th packet are doing the similar query and responding. We can also note that the 4886th packet is writing a value from HMI to PLC and the 4888th is responding to HMI about the writing. The other detailed parsed information can also be seen on the interface, just as the abnormal data example introduced above.

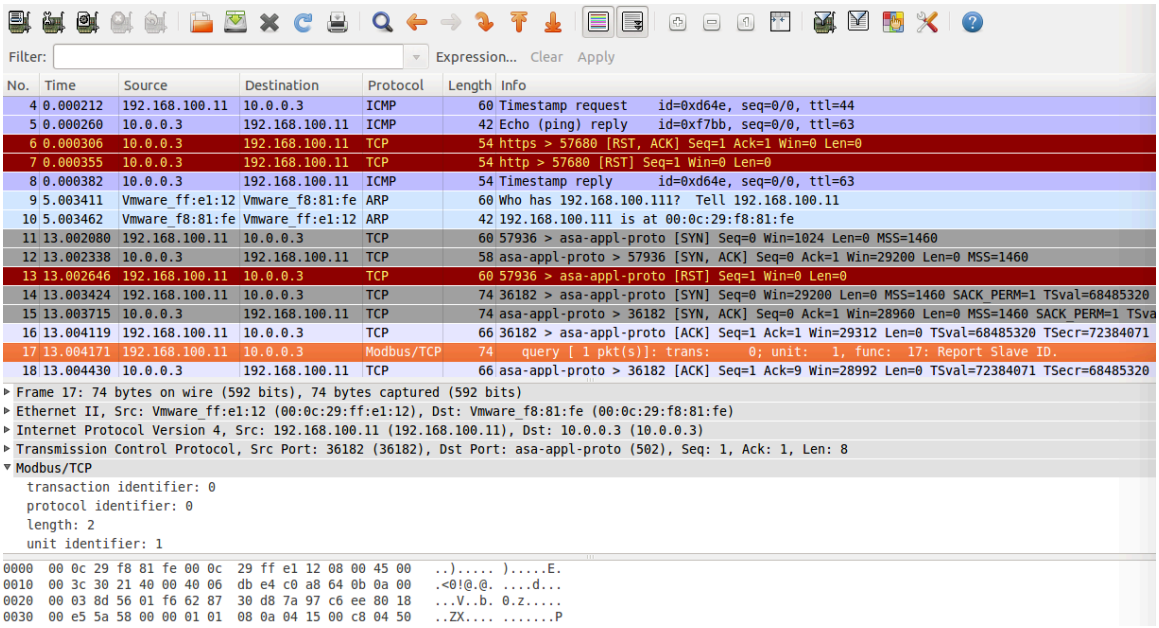

Figure 3.8: Wireshark network data analysis example (Scan attack)

For normal and abnormal network dataflow, the procedures of capturing the packets are both the same. The unique difference is that the normal data should be generated from the normal Modbus communication between PLC and HMI, while the abnormal data should be generated from the abnormal hacking from Kali to the control network such as PLC and Mod_Slave. The abnormal data can be generated

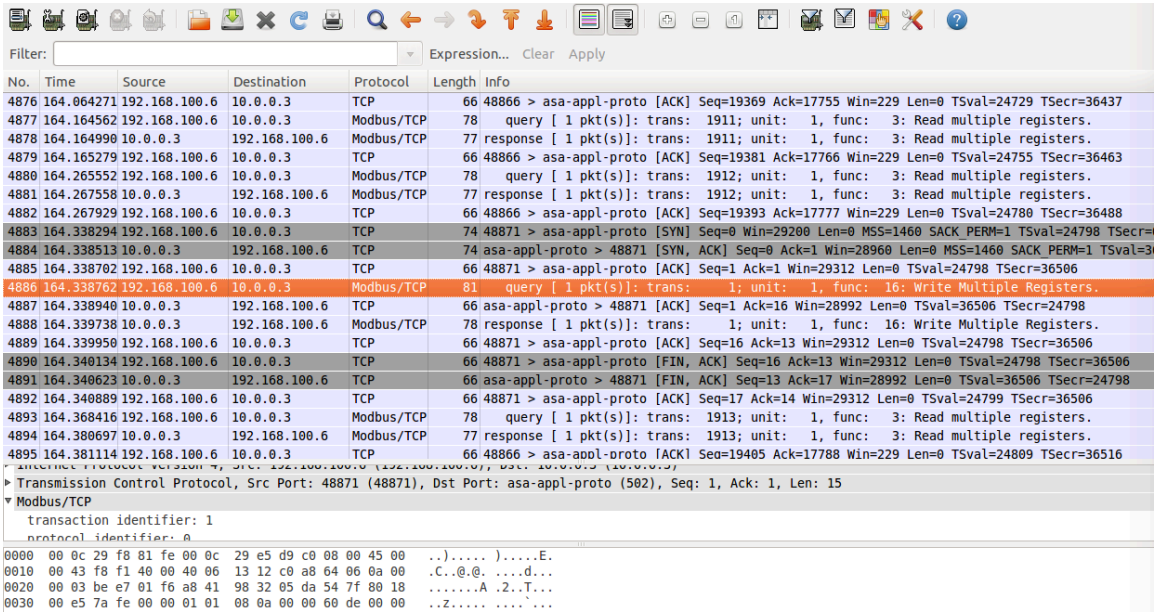

Figure 3.9: Wireshark network data analysis example (Normal data)

from different attacking tools in Kali, such as Namp, modpoll, Spoofing and so on.

3.2.2 Intrusion detection by Bro

When captured the data flow, one way to analyze the pcap file is to use Wireshark to see the basic parsing details, as introduced in the previous subsection 3.2.1. Another way is to use Bro to read, analyze and operate on the file using the Bro script to obtain the specific features or information required. The fantastic advantage of Bro is to detect the abnormal data flows using its own rules which can be designed flexibly using Bro script. Below we would design some Bro scripts to detect some intrusions. Note that although the examples shown below are based on the pcap files of pre-captured traffic for the illustration purpose, the same scripts can also be used to process live traffic on the network interfaces by using the command line utility to start a Bro daemon.

Specifically, two attacking and defence examples are designed to verify the practicality of Bro for the SCADA testbed. The first one is about a scan attack using Nmap from Kali to obtain the detailed network information of the IP range 10.0.0.0/24. The hacking command and information obtained are shown as below:

root@kali: # nmap -T5 10.0.0.0/24
Starting Nmap 6.47 (http://nmap.org) at 2016-09-05 23:16 PDT
Nmap scan report for 10.0.0.3
Host is up (0.0028s latency).
Not shown: 995 closed ports
PORT STATE SERVICE
21/tcp open ftp
22/tcp open ssh
80/tcp open http
6565/tcp filtered unknown
8080/tcp open http-proxy
Nmap scan report for 10.0.0.4
Host is up (0.0014s latency).
Not shown: 997 closed ports
PORT STATE SERVICE
22/tcp open ssh
8081/tcp open blackice-icecap
8082/tcp open blackice-alerts
Nmap scan report for 10.0.0.5
Host is up (0.0018s latency).
Not shown: 996 filtered ports
PORT STATE SERVICE
21/tcp open ftp
23/tcp open telnet
80/tcp open http
111/tcp open rpcbind
(...similar information for 10.0.0.6 - 10.0.0.8)
Nmap scan report for 10.0.0.111
Host is up (0.00059s latency).
Not shown: 999 closed ports
PORT STATE SERVICE
22/tcp open ssh
Nmap done: 256 IP addresses (7 hosts up) scanned in 22.43 seconds

We can see that this Nmap scan attack is one of the smart but horrible attacks.

From the information it obtained, we know the exact hosts active in the IP range which are 10.0.0.3 - 10.0.0.8 and 10.0.0.111, the exact open ports of each host for different protocol services such as ftp port 21, ssh port 22, http port 80, an unknown port 6565 and http-proxy port 8080 open on the host 10.0.0.3. Once the attackers know the open port information, it will be easy for them to use for attacking these services. Therefore, it is a secure way to be able to detect these kind of Scan attacks before the hackers launch a real attack.

The detecting principles for the attacks should be designed specifically to that certain attack according to its characteristics. In most cases, the scan attacks are supposed to attempt to connect to the hosts for a certain times, which is more than the amount of attempts a normal connection tries. Therefore, we can exploit this characteristic to design the detection principle for the Scan attacks. The following code in Fig. 3.10 is a part of Bro exited script to detect scan attacks. The code shows that it counts the failed connection attempts per source address. If the amount of the failed attempts is equal to SOME_THRESHOLD, it will generate a NOTICE, which generates a notice.log file in the current directory.

```
global attempts: table[addr] of count &default=0;
event connection_rejected(c: connection)
{
  local source = c$id$orig_h;              #Get source address.
  local n = ++attempts[source];             #Increase counter.
  if ( n == SOME_THRESHOLD )               # Check for threshold.
    NOTICE(...); #Alarm.
  }
}
```

Figure 3.10: Task: Count failed connection attempts per source address

When launching the scan attack at Kali using Nmap on the SCADA testbed and using the scan attack Bro scripts to detect the traffic on the Router Net, a notice.log file would be generated with features and information as below:

```
#notice.log
#separator x09
#set_separator ,
#empty_field (empty)
```

#unset_field
#path notice
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p fuid file_mime_type
file_desc proto note msg sub src dst p n peer_descr actions suppress_for
dropped remote_location.country_code remote_location.region
remote_location.city remote_location.latitude remote_location.longitude
#types time string addr port addr port string string
string enum enum string string addr addr port count string set [enum] interval
bool string string string double double
1471644241.318993 - - - - - - -
tcp Scan::Address_Scan 192.168.100.11 scanned at least 25 unique hosts
on ports 443*/tcp in 0m0s local 192.168.100.11 -*
443 - *bro Notice::ACTION_LOG* 3600.000000
F - - - - -
1471644251.336538 - - - - - - -
Scan::Port_Scan 192.168.100.11 scanned at least 15 unique ports of host
10.0.0.111 in 0m0s local 192.168.100.11 10.0.0.111 - -
bro Notice::ACTION_LOG 3600.000000 *F - - - - -*

The above notice.log file concludes the main information of the Scan attacks including reporting the Address_Scan attack and reporting the Port_Scan attack from the host 192.168.100.11, which actually is from Kali. The fields terms are the features extracted for the attacks, such as the uid, id.orig_h, id.orig_p, actions and so on. Since all the networks in the testbed are virtual simulated networks, not realistic networks, some of the feature items cannot obtain the proper contents, which have been replaced with "-".

Another example of attacking and defending is about one type of command injection attack. In the testbed, HMI can exploit Modpoll to control the pump speed. Assume that it is the normal operation only when the pump speed value is set to be 0 from supervising network, which is the operation of HMI. Therefore, any operation from the supervising network of which the pump speed is set more than zero will be considered as abnormal operations. Based on the hacking scenario assumption, we can launch an command injection attack on Kali by the command using Modpoll tool:

modpoll -0 -r 32210 10.0.0.4 10

The Modpoll command above is trying to set the pump speed of Sensors to be 10 directly, which is supposed to cause damages to the tank system. Specific to this type of command injection attack, a simple version of the principle interpreter can be written to focus on the operating address and the corresponding values, as shown in Fig 3.11. The code is to detect whether the first written value to registers of which the addresses are from 32210 (pump speed) is positive value. Actually, the event of modbus_write_multiple_registers_request is summarized by the event engine of Bro, which can be searched and exploited on the website of Bro [12]. In terms of Modbus protocol, Bro has already included an protocol interpreter. For each Modbus connection, Bro can make clear of the content of the communication and make records. Some simple information has been presented in the Modbus.log files. Other information can be easily accessible by printing out the variables in these events. After sending the Modpoll command above, a notice.log file would be generated with features and information as below:

#notice.log
#separator x09
#set_separator ,
#empty_field (empty)
#unset_field
#path notice
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p fuid file_mime_type
file_desc proto note msg sub src dst p n peer_descr actions suppress_for
dropped remote_location.country_code remote_location.region
remote_location.city remote_location.latitude remote_location.longitude
#types time string addr port addr port string string
string enum enum string string addr addr port count string set [enum] interval
bool string string string double double
1471816187.552404 - - - - - - -
Modbus::Att_CommandInjection - - 192.168.100.11 -
- - bro Notice::ACTION_LOG 3600.000000
F - - - - -

The two attacking and defencing examples discussed above are basic representative,

which can be followed as by generating other types of attacks and defencing them by designing Bro scripts. Some other attacks can also be generated by Nmap in Kali, so We can also explore more attacks in the future based on the research requirements.

```
event modbus_write_multiple_registers_request(
c: connection,
headers: ModbusHeaders,
start_address: count,
registers: ModbusRegisters){
  if ( start_address == 32210 && registers[0] > 0) {
    Notice ( $note=MODBUS::Att_CommandInjection,
             $src=c$id$orig_h
             #$msg=...,
             #$identifier=... );
  }
}
```

Figure 3.11: Task: Find the positive value written to the register with address 32210 (pump speed)

3.2.3 Implementing machine learning in Bro

The intrusion detection using Bro introduced in the previous section exploits the specific characteristics of the known attacks, which belongs to signature-based intrusion detection. Bro can also implement machine learning methods flexibly by designing Bro scripts, which belongs to anomaly-based intrusion detection. To test and verify the intrusion detection based on machine learning through the testbed, two of the machine learning stages need to be implemented in the testbed: data collection and intrusion detection based on a trained model. For data collection, we need to obtain a dataset of packets captured by tcpdump, or further a dataset of vectors with extracted features. The collected data are used to train a model by the machine learning algorithm. To apply the training result, the trained model needs to be deployed to the IDS framework where it takes effect in terms of identifying new traffic. Below we briefly show how to achieve the data collection (i.e., feature extraction) and intrusion detection under the framework of Bro. Note that the types of features used in data collection and intrusion detection are assumed to be same, without loss of generality.

Feature extraction is an essential component of data collection. The built-in

scripts in Bro can extract multiple kinds of features for each connection. Currently, Bro can generate 21 default basic features for each connection, summarized in /nsm/bro/logs directory, as shown in Fig. 3.12, including content features such as uid, id.orig_h, id.orig_p, service and so on, and time-based features such as ts, duration, resp_bytes, resp_packets and so on. Bro can also generate basic features for a certain protocol. For example, Bro is able to generate 8 default basic features for the connections communicating with Modbus protocol including function code content feature. For specific research purposes, it is necessary to design specific features for each connection such as for anomaly intrusion detection study. In terms of generating features, it is useful to refer to the feature work of KDD99 dataset [23], of which all the features are generated and extracted by designing Bro scripts. KDD99 dataset has 41 features including basic features of individual TCP connections, similar as the basic features in Fig. 3.12 such as protocol_type, flag, service and so on, content features within a connection suggested by domain knowledge such as num_failed_logins, num_hot_indicators, su_attempted, num_compromised and so on, and traffic features computed using a two-second time window such as num_count_same_host, SYN_error_rate, REJ_error_rate and so on.

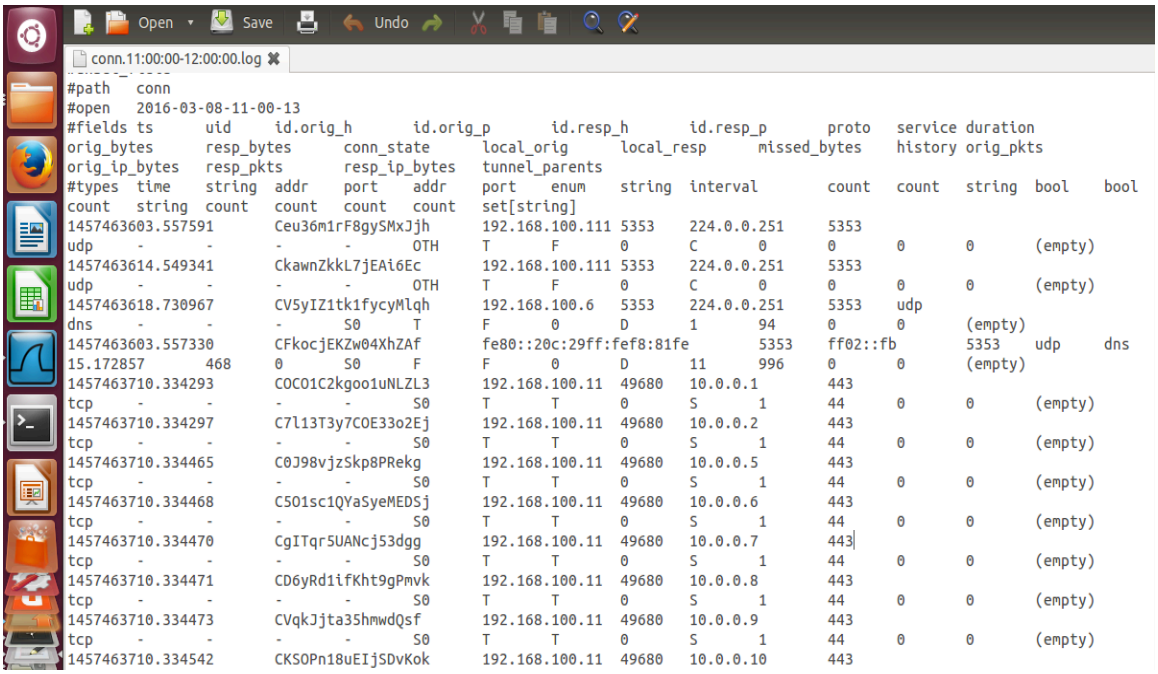

Figure 3.12: Example of connection features

A set of features specific to the Modbus protocol, which plays an important role in the SCADA network, can be designed and implemented as following. We can extract a set of features based on the hooks provided in the protocol analyzer of Bro. In the file src/analyzer/protocol/modbus/events.bif, Bro defines a set of event functions related to the Modbus requesting types as shown in Fig. 3.13. We can implement and add some scripts to the Bro framework as user-defined reactions to these events. When the events listed in Fig. 3.13 are detected by the Bro protocol analyzer, we can execute some customized code to achieve the steps of feature collection and intrusion detection in machine learning. Such functions should be implemented by event functions, just as shown in Fig. 3.11.

---

event modbus_read_coils_request%(c: connection, headers: ModbusHeaders, start_address: count, quantity: count%);
event modbus_read_discrete_inputs_request%(c: connection, headers: ModbusHeaders, start_address: count, quantity: count%);
event modbus_read_holding_registers_request%(c: connection, headers: ModbusHeaders, start_address: count, quantity: count%);
event modbus_read_input_registers_request%(c: connection, headers: ModbusHeaders, start_address: count, quantity: count%);
event modbus_write_single_coil_request%(c: connection, headers: ModbusHeaders, address: count, value: bool%);
event modbus_write_single_register_request%(c: connection, headers: ModbusHeaders, address: count, value: count%);
event modbus_write_multiple_coils_request%(c: connection, headers: ModbusHeaders, start_address: count, coils: ModbusCoils%);
event modbus_write_multiple_registers_request%(c: connection, headers: ModbusHeaders, start_address: count, registers: ModbusRegisters%);
event modbus_read_file_record_request%(c: connection, headers: ModbusHeaders%);
event modbus_write_file_record_request%(c: connection, headers: ModbusHeaders%);
event modbus_mask_write_register_request%(c: connection, headers: ModbusHeaders, address: count, and_mask: count, or_mask: count%);
event modbus_read_write_multiple_registers_request%(c: connection, headers: ModbusHeaders, read_start_address: count, read_quantity: count, write_start_address: count, write_registers: ModbusRegisters%);
event modbus_read_fifo_queue_request%(c: connection, headers: ModbusHeaders, start_address: count%);

Figure 3.13: Requesting event functions implemented by Bro analyzer for the modbus protocol

For feature collection, we should add some code to react to each of the listed events. The added code will extract the features by processing the parameters and the name of the event functions. Upon an event, some features will be extracted and inserted to a log file as a new entry. The features extracted are in 4 categories. The first category is designed as the type of request, i.e., the event type. As we can observe that all these event functions share the same parameters about connection and ModbusHeaders, so the second and third category are designed as related to the connection information and the Modbus header. The fourth category is for the other parameters in the listed functions. Although different functions may have different number of parameters, we can always predefine the set of parameters extracted, so that the feature vector length is determined. When filling a feature vector, if an event function does not include some chosen parameters, we can set the default value of these omitted item as zero in the vector. Following this procedure, we can obtain a log file of the generated vectors, which is actually the dataset collected.

The implemented script in Bro would extract features for each connection, and output them in a file notice.log shown as follows. These extracted features are directly applied in determining whether the connection is an intrusion or not. Besides, the output file is used by the training algorithm to determine the weights.

#factor.log
#separator x09
#set_separator ,
#empty_field (empty)
#unset_field
#path factor
#open
#fields a_class b_orig_p b_resp_p b_orig_size b_orig_num_pkts
b_orig_num_bytes b_resp_size b_resp_num_pkts b_resp_num_bytes b_d
uration c_tid c_pid c_len c_uid c_func d_address d_value
#types count count count count count count count count
double count count count count count count int
16 54253 502 15 2 112 0 1 60 0.00056 1 0 9 1 16 32210 5
#close

The intrusion detection are also implemented based on the event functions. Specif-

ically, in each event function, we have already generated a feature vector following the method in data collection above. This feature vector should be multiplied to the weight vector from the training result. Depending on the resultant value of the multiplication, IDS can choose between generating an alert notice or not based on a threshold.

The weight vector is read from an input file by the program. The following file weight.input is an example for that. The values shown in the file are set for illustration, and they are supposed to be determined by the training algorithm. Due to the current limitation of traffic generation in the testbed, the number of collected samples are not large enough to support a meaningful training, so the weights used here are not from real training.

#weight.input
#separator x09
#set_separator ,
#empty_field (empty)
#unset_field
#path factor
#open
#fields index a_class b_orig_p b_resp_p b_orig_size b_orig_num_pkts
b_orig_num_bytes b_resp_size b_resp_num_pkts b_resp_num_bytes b_d
uration c_tid c_pid c_len c_uid c_func d_address d_value
0 1 5 5 2 3 1 0 -1 3 1 0 2 1 3 0 18
#close

The implemented script in Bro also makes decisions about intrusion detection. Specifically, for each connection, the vector of extracted features is multiplied with the vector of input weights, and if the result is above than a threshold, a notification is shown in the output file notice.log as follows. In summary, the explanation about how to use Bro to implement the feature collection and making decisions, which are needed in the implementation of most machine learning methods, has been presented.

#notice.log
#separator x09
#set_separator ,

#empty_field (empty)
#unset_field
#path notice
#open
*#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_*
p fuid file_mime_type file_desc proto note msg sub
src dst p n peer_descr actions suppress_for dropped
remote_location.country_code remote_location.region
remote_location.city remote_location.latitude remote_location.longitude
#types time string addr port addr port string string string string
enum enum string string addr addr port count string set [enum]
interval bool string string string double double
1471216187.552404 - - - - - - -
Modbus::Att_Anomaly - - *192.168.100.11* -
- - *bro Notice::ACTION_LOG* 3600.000000
F - - - - -
1471216187.552404 - - - - - - -
Modbus::Att_Signature - - *192.168.100.11* -
- - *bro Notice::ACTION_LOG* 3600.000000
F - - - - -
#close

3.3 Testbed configuration details

For readers interested in reproducing the testbed system, the configuration details are presented below. In the SCADA testbed system, referred to Fig. 3.1, there are control network within the IP range 10.0.0.0/24, which contains PLC_Master with IP 10.0.0.3, several virtual PLCs with IPs 10.0.0.5 - 10.0.0.7, and sensors Mod_Slave with IP 10.0.0.4. They are communicating using Modbus protocol. Mod_Slave is configured as the server of its client PLC_Master, which can also be referred as the slave of its master PLC in the perspective of control system. The supervising network within the IP range 192.168.100.0/24 contains HMI with IP 192.168.100.6. And the PLC is the server of the client HMI. HMI, PLC_Master and Mod_Slave compose the

tank system. The network configurations for the hosts HMI, PLC and sensors are similar. But the configurations for the client roles and server roles should be different, which are to be introduced in the following part. The supervising network can access to the control network through the Router with one network interface card in the control network with IP 10.0.0.3 and another network interface card in the supervising network with IP 192.168.100.111, on which has installed Wireshark and Bro for traffic analyzing. Bro has been set to supervise the interface 192.168.100.111. The network configuration for the Router and the installation of the softwares on it will be introduced later. The attackers in the testbed, Kali with IP 192.168.100.11, Nexpose with IP 192.168.100.22 and Samurai 192.168.100.33, are deployed in the supervising network, to form a reasonable hacking scenario. The network configurations for the hosts Kali, Nexpose and Samurai are similar. So far, most attacks in the testbed are launched from Kali. Therefore, we only present the configuration details for Kali. All the hosts in the testbed are deployed with virtual machines. They are all set on the administration network with the IP range of 172.16.1.0/24 for being remotely operated and the connection to the Internet.

3.3.1 HMI-PLC-Sensors configuration details

The detailed configurations for Sensors Mod_Slave is as Table. 3.1. For the network configuration, it is necessary to modify the file /etc/network/interfaces to configure the interface cards and the file /etc/rc.local to add IP routing guide information as the table. Note that since Mod_Slave is the server of PLC, we should always start the TCP/IP server first, i.e., Mod_Slave, and then start the TCP/IP client PLC. Another thing requiring attention is that if ./mod_slave.sh cannot be started successfully, kill the related processes with its PID number.

The detailed configurations for PLC_Master is as Table. 3.2. Similar as Mod_Slave, it is supposed to modify the file /etc/network/interfaces to configure the interface cards and the file /etc/rc.local to add IP routing guide information as the table. Note that again we should start the TCP/IP client PLC after its server Mod_Slave. Similarly, if ./mb_logic.sh cannot be started successfully, kill the related processes with its PID number.

The detailed configurations for HMI is as Table. 3.3. For the network configuration, we need to modify the file /etc/network/interfaces to configure the interface cards and the file /etc/rc.local to add IP routing guide information as the table. Note

that since PLC is the server of HMI, remember to always start the server PLC first, then the client HMI. Similarly, if ./mblogichmi.sh cannot be started successfully, kill the related processes with its PID number. Another thing requiring notification is that HMI can not only acquire the information of the tank system from PLC, but also able to control the tank system by exploiting Modpoll to set PLC pump speed.

3.3.2 Router configuration details

The detailed configurations for the Router virtual machine is as Table. 3.4. For the network configuration, we need to modify the file /etc/network/interfaces to configure the interface cards, the file /etc/sysctl.conf to open IP routing and forwarding function and the file /etc/rc.local to start the Bro network analyzer as the table. Note that for Bro's normal function on the Router, there are three more files (i.e, /nsm/bro/etc/node.cfg, /nsm/bro/etc/broctl.cfg and /nsm/bro/etc/networks.cfg) to be configured to specify the supervised network interface, the email recipient when notices generated and the network IP ranges to be supervised, of which the configuration details are as below:

#(/nsm/bro/etc/node.cfg)
 # This is a complete standalone configuration.
[bro]
type=standalone
host=localhost
interface=eth1
#eth1:192.168.100.111

#(/nsm/bro/etc/broctl.cfg)
 # Can change the recipient address for all emails
sent out by Bro and BroControl.

#(/nsm/bro/etc/networks.cfg)
192.168.100.0/24 Private IP space

3.3.3 Attacker configuration details

The detailed configurations for Kali virtual machine is shown in Table. 3.5. Similarly for the network configuration, it is necessary to modify the file /etc/network/interfaces to configure the interface cards and the file /etc/rc.local to add IP routing guide information as shown in the table. Note that although there already exist multiple types of attacking tools installed in Kali, Modpoll is not included and should be download from its own website.

We should note that tables in this section present some common basis network configurations which is adapted from [73] with some differences, such as Ethernet information and IP addresses in Table. 3.1, Table. 3.2 and Table. 3.5. Differences between the current configurations and the previous ones are obviously shown when comparing the works. For example, the routing information for the new routing system and the account information are all unique in the current system. We give some simple summaries about the old configurations and present the new extensions here for the continuous integration.

3.4 Conclusion

This chapter presents the fundamentals, working procedure and the configuration details for the testbed, which is designed for SCADA network security research and implementing the idea of HOIDS in Chapter 2 on a practical SCADA testbed. The testbed simulates a classic tank system of HMI-PLC-Sensors deployment with virtual machines, which can be modified easily and expanded conveniently to a larger SCADA system. The distributed network topology of the tank system and the hacking host with the control network and supervising network, make the testbed a reasonable simulated SCADA control networks. With the Bro network analyzer supervising the network data on the Router, the testbed is able to record, analyze and extract features for normal and abnormal data flows, and can detect the abnormal ones through the design of Bro script based on the characteristics of the attacks, which is beneficial to the intrusion detection study using machine learning techniques, which would be the future work. When the feature dataset with enough normal and abnormal data samples, we may use it to train the machine learning algorithms introduced in Chapter 2 and apply the generated detection principle to Bro. Therefore, the idea of anomaly intrusion detection gets implemented. Besides, the whole testbed can also

combine the signature-based intrusion detection and anomaly intrusion detection for more secure protection.

Table 3.1: Mod_Slave configuration details

Virtual machine name	Mod_Slave
Interface and IP address	Eth0: 10.0.0.4/24 (control network) Eth1: 172.16.1.4/24 (administration network)
Network information	(/etc/network/interfaces) auto lo iface lo inet loopback #The primary network interface auto eth0 iface eth0 inet static address 10.0.0.4 netmask 255.255.255.0 network 10.0.0.0 broadcast 10.0.0.255 auto eth1 iface eth1 inet static address 172.16.1.4 netmask 255.255.255.0 (/etc/rc.local) # Add IP Routing ip route add 192.168.100.0/24 via 10.0.0.111
Installed software	MBLogic (https://sourceforge.net/projects/mblogic/)
Account and password	Mod_Slave: root/elwa350 HMI (http://172.16.1.4:8082/hmidemo.xhtml) MBLogic Status (http://172.16.1.4:8081/index.html)
Run procedure	(Under root account) cd ∼/mblogic (to start MBLogic sensors) ./mod_slave.sh

Table 3.2: PLC_Master configuration details

Virtual machine name	PLC_Master
Interface and IP address	Eth0: 10.0.0.3/24 (control network) Eth1: 172.16.1.3/24 (administration network)
Network information	(/etc/network/interfaces) auto lo iface lo inet loopback auto eth0 iface eth0 inet static address 10.0.0.3 netmask 255.255.255.0 auto eth1 iface eth1 inet static address 172.16.1.3 netmask 255.255.255.0 (/etc/rc.local) # Nova IP routing add ip route add 192.168.100.0/24 via 10.0.0.111
Installed software	MBLogic (https://sourceforge.net/projects/mblogic/) Nova (https://github.com/DataSoft/Nova)
Account and password	PLC_Master: root/elwa350 HMI (http://172.16.1.3:8082/hmidemo.xhtml) MBLogic Status (http://172.16.1.3:8081/index.html) Nova web console (https://172.16.1.3:8080/): root/root
Run procedure	(Under root account) quasar (to start Nova) Then log in Nova web console, click “Start Haystack” and “Start Packet Classifier” on the top right corner. Then virtual PLCs are running. cd ∼/mblogic (to start MBLogic PLC) ./mblogic.sh

Table 3.3: HMI configuration details

Virtual machine name	HMI
Interface and IP address	Eth0: 172.16.1.6/24 (administration network) Eth1: 192.168.100.6/24 (supervising network)
Network information	(/etc/network/interfaces) auto lo iface lo inet loopback #The primary network interface auto eth0 iface eth0 inet static address 172.16.1.6 netmask 255.255.255.0 gateway 172.16.1.2 auto eth1 iface eth1 inet static address 192.168.100.6 netmask 255.255.255.0 (/etc/rc.local) # Add IP Routing ip route add 10.0.0.0/24 via 192.168.100.111
Installed software	MBLogic (https://sourceforge.net/projects/mblogic/) modpoll (http://www.modbusdriver.com/modpoll.html)
Account and password	HMI: abc/abc, root/elwa350 HMI (http://172.16.1.6:8082/hmidemo.xhtml) MBLogic Status (http://172.16.1.6:8081/index.html)
Run procedure	(Under root account) cd ∼/mblogic (to start MBLogic HMI) ./mblogichmi.sh

Table 3.4: Router Net configuration details

Virtual machine name	Router
Interface and IP address	Eth0: 172.16.1.3/24 (administration network) Eth1: 192.168.100.111/24 (supervising network) Eth2: 10.0.0.3/24 (control network)

Network information	(/etc/network/interfaces)
	auto lo
	iface lo inet loopback
	
	auto eth0
	iface eth0 inet static
	address 172.16.1.5
	netmask 255.255.255.0
	gateway 172.16.1.2
	
	auto eth1
	iface eth1 inet static
	address 192.168.100.111
	netmask 255.255.255.0
	
	auto eth2
	iface eth2 inet static
	address 10.0.0.111
	netmask 255.255.255.0
	
	(/etc/sysctl.conf)
	# Enable packet forwarding for IPv4
	net.ipv4.ip_forward=1
	
	(/etc/rc.local)
	/nsm/bro/bin/broctl start
	exit 0

Installed software	Bro IDS (https://www.bro.org/documentation/index.html) Wireshark (https://www.wireshark.org/)
Account and password	Net: net/net, root/elwa350
Run procedure	Bro is already started at the starting procedure. Wireshark can be started using command in the terminal.

Table 3.5: Kali configuration details

Virtual machine name	Kali
Interface and IP address	Eth0: 192.168.100.11/24 (supervising network) Eth1: 172.16.1.11/24 (administration network)
Network information	(/etc/network/interfaces) # The loopback network interface auto lo iface lo inet loopback #The primary network interface allow-hotplug eth0 iface eth0 inet static address 192.168.100.6 netmask 255.255.255.0 auto eth1 iface eth1 inet static address 172.16.1.11 netmask 255.255.255.0 (/etc/rc.local) # Add IP Routing ip route add 10.0.0.0/24 via 192.168.100.111
Installed software	Kali (http://www.kali.org/) modpoll (http://www.modbusdriver.com/modpoll.html)
Account and password	Kali: root/elwa350
Run procedure	There are many different kinds of attacking tools in Applications -> Kali Linux.

Chapter 4

Conclusions and Future Work

To further improve the security of the SCADA systems, anomaly-based intrusion detection system design can be as an efficient complement to the traditional signature-based IDS in the industrial networks to embrace the big data era. This thesis presents a novel hierarchical online intrusion detection system (HOIDS) for SCADA networks based on machine learning algorithms theoretically and establishes a testbed for implementing the idea of anomaly-based intrusion detection.

For the theoretical design, the IDS server deployed in the control centre and the IDS clients distributed in the whole SCADA system including the control centre network, substation networks and the field networks can well protect the system. Accurate models of normal-abnormal binary detection and multi-attack identification using logistic regression and quasi-Newton optimization are implemented for the generation of the detection principle. Furthermore, the detection system can be accelerated by feature selection techniques and dimension reduction approaches, while keeping good detection accuracy. With the server-clients distributed topology and the effective machine learning algorithms, high intrusion detection rate can be achieved with minimum impact to the whole system. The whole intelligent design can secure the system globally and even in a faster secure way while able to reduce the financial budget for the large scale systems and facilitate the implementation of the hardware devices.

For the testbed implementation, a classic tank system of HMI-PLC-Sensors deployment with virtual machines is established for SCADA network security research and implementing the theoretical idea of anomaly-based intrusion detection system on a simulated SCADA testbed scenario. The testbed simulates a reasonable SCADA network by deploying the control network and supervising network, and with hackers

in the supervising network launching different types of attacks to the tank system. With the Bro network analyzer installed on the router, the testbed can record and analyze the network data, extract features for normal and abnormal data flows, and detect the abnormal ones by designing Bro scripts according to the attacking characteristics. In the future work, more datasets should be collected by implementing and introducing more traffic generation methods to the testbed, including both normal and abnormal data samples. And after that, they could be exploited to train the machine learning algorithms introduced theoretically and apply the generated detection principle to Bro for intrusion detection. The whole design including both the theoretical proposition and testbed implementation can combine the signature-based intrusion detection and anomaly-based intrusion detection together for more secure protection.

In the future, there are still many interesting areas to be explored further under the discussed topic. People may find that the connection between the SCADA networks and public networks makes the former vulnerable to attack, but it is still inevitable for various communication purposes. Therefore, novel security infrastructures and algorithms that can efficiently and accurately classify traffic flows including the Internet network data and control protocol data to multiple detailed classes are necessary to be studied. When studying intrusion detections, people should also follow the development of machine learning algorithms and approaches, e.g., new hardwares have made some complex algorithms feasible in the production environment. An efficient, practical and reasonable testbed environment is important to verify new methods in intrusion detection. For example, we can extend the feature extraction in the testbed to provide more features that can be used in data training. How to automatically execute the training process when a certain number of new data samples are available can be done in the future. The network architecture can be simulated in a more practical way if the details of a real example industrial system are available. Further, more attacking tools and normal traffic generator can be introduced to the testbed to collect a meaningful dataset for SCADA IDS research.

Appendix A

Additional Information

Table A.1: Correlation coefficients among the features of Multi-command Injections (a)

	1	2	3	4	5	6	7	8
1	1	0.8325	0.8328	0.8328	0.8328	NaN	0.8328	NaN
2	0.8325	1	0.9964	0.9964	0.9964	NaN	0.9964	NaN
3	0.8328	0.9964	1	0.9999	1	NaN	1	NaN
4	0.8328	0.9964	0.9999	1	1	NaN	1	NaN
5	0.8328	0.9964	1	1	1	NaN	1	NaN
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	0.8328	0.9964	1	1	1	NaN	1	NaN
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	0.3309	0.6848	0.6667	0.6667	0.6667	NaN	0.6667	NaN
10	-0.0088	0.0016	-0.0075	-0.0075	-0.0075	NaN	-0.0075	NaN
11	0.8411	0.9952	0.9954	0.9954	0.9954	NaN	0.9954	NaN
12	0.8411	0.9952	0.9954	0.9954	0.9954	NaN	0.9954	NaN
13	0.8411	0.9952	0.9954	0.9954	0.9954	NaN	0.9954	NaN
14	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
15	0.8411	0.9952	0.9954	0.9954	0.9954	NaN	0.9954	NaN
16	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
17	0.6788	0.9404	0.9409	0.9409	0.9409	NaN	0.9409	NaN

Table A.2: Correlation coefficients among the features of Multi-command Injections (b)

	9	10	11	12	13	14	15	16	17
1	0.3309	-0.0088	0.8411	0.8411	0.8411	NaN	0.8411	NaN	0.6788
2	0.6848	0.0016	0.9952	0.9952	0.9952	NaN	0.9952	NaN	0.9404
3	0.6667	-0.0075	0.9954	0.9954	0.9954	NaN	0.9954	NaN	0.9409
4	0.6667	-0.0075	0.9954	0.9954	0.9954	NaN	0.9954	NaN	0.9409
5	0.6667	-0.0075	0.9954	0.9954	0.9954	NaN	0.9954	NaN	0.9409
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	0.6667	-0.0075	0.9954	0.9954	0.9954	NaN	0.9954	NaN	0.9409
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	1	0	0.6210	0.6210	0.6210	NaN	0.6210	NaN	0.6567
10	0	1	-0.0079	-0.0079	-0.0079	NaN	-0.0079	NaN	-0.0070
11	0.6210	-0.0079	1	1	0.9999	NaN	1	NaN	0.9453
12	0.6210	-0.0079	1	1	0.9999	NaN	1	NaN	0.9453
13	0.6210	-0.0079	0.9999	0.9999	1	NaN	0.9999	NaN	0.9453
14	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
15	0.6210	-0.0079	1	1	0.9999	NaN	1	NaN	0.9453
16	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
17	0.6567	-0.0070	0.9453	0.9453	0.9453	NaN	0.9453	NaN	1

Table A.3: Modbus protocol function code

Function Type	Function Name	Function Coe
Data access: Bit	**Read Discrete Inputs**	2
	Read Colls	1
	Write Single Coll	5
	Write Multiple Colls	15
Data access: 16-bit	**Read Input Register**	4
	Read Holding Registers	3
	Write Single Register	6
	Write Multiple Registers	16
	Read/Write Multiple Registers	23
	Mask Write Register	22
	Read FIFO Queue	24
Data access: File	Read File Record	20
	Write File Record	21
Diagonostics	Read Exception Status	7
	Diagnostic	8
	Get Com Event Counter	11
	Get Com Event Log	12
	Report Slave ID	17
	Read Device Identification	43
Other	Encapsulated Interface Transport	44

Bibliography

[1] http://mblogic.sourceforge.net/techdemo/hmidemo/hmidemo.xhtml. *MBlogic Web information.*

[2] https://cn.mathworks.com/help/stats/zscore.html#btikeav. *zscore information.*

[3] https://en.wikipedia.org/wiki/hessian_matrix. *Hessian information.*

[4] https://en.wikipedia.org/wiki/logistic_function. *Logistic function information.*

[5] https://en.wikipedia.org/wiki/modbus. *Modbus introduction.*

[6] https://github.com/datasoft. *Honeyd and Nova infomation.*

[7] https://projects.honeynet.org/honeywall/. *Honeywall infomation.*

[8] https://sourceforge.net/projects/mblogic/. *MBlogic information.*

[9] https://technet.microsoft.com/en-us/library/cc958821.aspx. *TCP/IP protocol architecture information.*

[10] https://www.bro.org/documentation/index.html. *Bro information.*

[11] https://www.bro.org/documentation/slides/index.html. *Bro slides learning information.*

[12] https://www.bro.org/sphinx/scripts/base/bif/plugins/bro_modbus.events.bif.bro.html. *Bro Events information.*

[13] https://www.kali.org/. *Kali information.*

[14] https://www.snort.org/. *Snort information.*

[15] https://www.wireshark.org/. *Wireshark information.*

[16] https://www.yokogawa.com/us/technical-library/resources/application-notes/scada-cyber-security/. *SCADA cyber security.*

[17] http://try.bro.org/#/trybro?example=hello. *Bro script learning.*

[18] http://www.bcit.ca/appliedresearch/tc/facilities/industrial.shtml. *Industrial Instrumentation Process Lab.*

[19] http://www.modbusdriver.com/modpoll.html. *Modpoll information.*

[20] http://www.modbus.org/. *Modbus information.*

[21] http://www.rapid7.com/products/nexpose/. *Nexpose information.*

[22] http://www.samuraistfu.org/home. *Samurai information.*

[23] `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`.

[24] `https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets`.

[25] Common cyber security vulnerabilities observed in control system assessments by the inl nstb program. *Idaho National Laboratory (INL)*, 2008.

[26] Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. Learning from data. *AMLBook*, 2012.

[27] R. Agarwal and J. Mahesh V. Pnrule: A new framework for learning classi er models in data mining (a case-study in network intrusion detection). In *Proceedings of First SIAM Conference on Data Mining*, 2000.

[28] Andreas Antoniou and Wu-Sheng Lu. *Practical optimization: algorithms and engineering applications.* Springer Science & Business Media, 2007.

[29] Stefan Axelsson. Intrusion detection systems: A survey and taxonomy. Chalmers University of Technology, Goteborg, Sweden, 2000.

[30] Justin M Beaver, Raymond C Borges-Hink, Mark Buckner, et al. An evaluation of machine learning methods to detect malicious scada communications. In *12th International Conference on Machine Learning and Applications (ICMLA)*, volume 2, pages 54–59, 2013.

[31] David C Bergman, Dong (Kevin) Jin, David M Nicol, and Tim Yardley. The virtual power system testbed and inter-testbed integration. In *CSET*, 2009.

[32] Dankmar Böhning. Multinomial logistic regression algorithm. *Annals of the Institute of Statistical Mathematics*, 44(1):197–200, 1992.

[33] Remco R Bouckaert. Choosing between two learning algorithms based on calibrated tests. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 51–58, 2003.

[34] Stephen Boyd and Lieven Vandenberghe. Convex optimization. *Cambridge university press*, 2004.

[35] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[36] Martin Brent. Instance-based learning: Nearest neighbor with generalization. *Hamilton, New Zealand*, 1995.

[37] Andrea Carcano, Alessio Coletta, Michele Guglielmi, Marcelo Masera, A Trombetta, et al. A multidimensional critical state analysis for detecting intrusions in scada systems. *IEEE Transactions on Industrial Informatics*, 7(2):179–186, 2011.

[38] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[39] Steven Cheung, Bruno Dutertre, Martin Fong, Ulf Lindqvist, Keith Skinner, and Alfonso Valdes. Using model-based intrusion detection for scada networks. In *Proceedings of the SCADA security scientific symposium*, volume 46, pages 1–12, 2007.

[40] Giovanna Dondossola, Geert Deconinck, Fabrizio Garrone, and Hakem Beitollahi. Testbeds for assessing critical scenarios in power control systems. In *International Workshop on Critical Information Infrastructures Security*, pages 223–234. Springer, 2008.

[41] Giovanna Dondossola, G Garrone, Judit Szanto, Geert Deconinck, Tom Loix, and Hakem Beitollahi. Ict resilience of power control systems: Experimental

results from the crutial testbeds. In *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, pages 554–559. IEEE, 2009.

[42] Igor Nai Fovino, Marcelo Masera, Luca Guidi, and Giorgio Carpi. An experimental platform for assessing scada vulnerabilities and countermeasures in power plants. In *3rd International Conference on Human System Interaction*, pages 679–686. IEEE, 2010.

[43] Niv Goldenberg and Avishai Wool. Accurate modeling of modbus/tcp for intrusion detection in scada systems. *International Journal of Critical Infrastructure Protection*, 6(2):63–75, 2013.

[44] Adam Hahn, Aditya Ashok, Siddharth Sridhar, and Manimaran Govindarasu. Cyber-physical security testbeds: Architecture, application, and evaluation for smart grid. *IEEE Transactions on Smart Grid*, 4(2):847–855, 2013.

[45] Adam Hahn, Ben Kregel, Manimaran Govindarasu, Justin Fitzpatrick, Rafi Adnan, Siddharth Sridhar, and Michael Higdon. Development of the powercyber scada security testbed. In *Proceedings of the sixth annual workshop on cyber security and information intelligence research*, page 21. ACM, 2010.

[46] Robert C Holte. Very simple classification rules perform well on most commonly used datasets. *Machine learning*, 11(1):63–90, 1993.

[47] Vinay M Igure, Sean A Laughter, and Ronald D Williams. Security issues in scada networks. *Computers & Security*, 25(7):498–506, 2006.

[48] Eric D Knapp and Joel Thomas Langill. Industrial network security: Securing critical infrastructure networks for smart grid, scada, and other industrial control systems. *Syngress*, 2014.

[49] Pat Langley, Wayne Iba, and Kevin Thompson. An analysis of bayesian classifiers. In *Aaai*, volume 90, pages 223–228, 1992.

[50] Ralph Langner. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, 9(3):49–51, 2011.

[51] Saskia Le Cessie and Johannes C Van Houwelingen. Ridge estimators in logistic regression. *Applied statistics*, pages 191–201, 1992.

[52] Ondrej Linda, Todd Vollmer, and Milos Manic. Neural network based intrusion detection system for critical infrastructures. In *International Joint Conference onNeural Networks (IJCNN)*, pages 1827–1834, 2009.

[53] T C Service M J McDonald, G N Conrad and R H Cassidy. Cyber effects analysis using vcse. *Sandia National Laborotories*, SAND2008-5954, Sep 2008.

[54] Leandros Maglaras, Jianmin Jiang, et al. Intrusion detection in scada systems using machine learning techniques. In *Science and Information Conference (SAI)*, pages 626–631, 2014.

[55] Thomas P Minka. A comparison of numerical optimizers for logistic regression. *Unpublished draft*, 2003.

[56] Thomas Morris, Rayford Vaughn, and Yoginder S Dandass. A testbed for scada control system cybersecurity research and pedagogy. In *Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research*, page 27, 2011.

[57] Andrew Nicholson, Stuart Webber, Shaun Dyer, Tanuja Patel, and Helge Janicke. Scada security in the light of cyber-warfare. *Computers & Security*, 31(4):418–436, 2012.

[58] Paul Oman and Matthew Phillips. Intrusion detection and event monitoring in scada networks. *Critical Infrastructure Protection*, pages 161–173, 2008.

[59] Shengyi Pan, Thomas Morris, and Uttam Adhikari. Developing a hybrid intrusion detection system using data mining for power systems. *IEEE Transactions on Smart Grid*, 6(6):3104–3113, 2015.

[60] Animesh Patcha and Jung-Min Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks*, 51(12):3448–3470, 2007.

[61] John Platt et al. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

[62] Carlos Queiroz, Abdun Mahmood, Jiankun Hu, Zahir Tari, and Xinghuo Yu. Building a scada security testbed. In *Network and System Security, 2009. NSS'09. Third International Conference on*, pages 357–364. IEEE, 2009.

[63] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[64] Robert Radvanovsky and Jacob Brodsky. Handbook of scada/control systems security. *CRC Press*, 2013.

[65] Ragunathan Raj Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems: the next computing revolution. In *Proceedings of the 47th Design Automation Conference*, pages 731–736, 2010.

[66] Julian Rrushi and Kyoung-Don Kang. Detecting anomalies in process control networks. *Critical Infrastructure Protection III*, pages 151–165, 2009.

[67] Keith Stouffer, Joe Falco, and Karen Scarfone. Guide to industrial control systems (ics) security. *NIST special publication*, 800(82):16–16, 2011.

[68] Chee-Wooi Ten, Junho Hong, and Chen-Ching Liu. Anomaly detection for cybersecurity of the substations. *IEEE Transactions on Smart Grid*, 2(4):865–873, 2011.

[69] Alfonso Valdes and Steven Cheung. Communication pattern anomaly detection in process control systems. In *IEEE Conference on Technologies for Homeland Security (HST)*, pages 22–29, 2009.

[70] Dayu Yang, Alexander Usynin, and J Wesley Hines. Anomaly-based intrusion detection for scada systems. In *5th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies (NPIC & HMIT)*, pages 12–16, 2006.

[71] Yi Yang, Keiran McLaughlin, Tim Littler, Sakir Sezer, and HF Wang. Rule-based intrusion detection system for scada networks. In *Renewable Power Generation Conference (RPG)*, pages 1–4, 2013.

[72] Yi Yang, Keiran McLaughlin, Sakir Sezer, Tim Littler, Eul Gyu Im, Bernardi Pranggono, and HF Wang. Multiattribute scada-specific intrusion detection system for power networks. *IEEE Transactions on Power Delivery*, 29(3):1092–1102, 2014.

[73] Liao Zhang. https://arxiv.org/abs/1701.05323. *An Implementation of SCADA Network Security Testbed*, 2015.

[74] Yichi Zhang, Lingfeng Wang, Weiqing Sun, Robert C Green, Mansoor Alam, et al. Distributed intrusion detection system in a multi-layer network architecture of smart grids. *IEEE Transactions on Smart Grid*, 2(4):796–808, 2011.

[75] Bonnie Zhu and Shankar Sastry. Scada-specific intrusion detection/prevention systems: a survey and taxonomy. In *Proceedings of the 1st Workshop on Secure Control Systems (SCS)*, 2010.