2021

Design, implementation, and field-testing of distributed intrusion detection system for smart grid SCADA network

Jeyanth Rajan Babu
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/etd

Recommended Citation
Rajan Babu, Jeyanth, "Design, implementation, and field-testing of distributed intrusion detection system for smart grid SCADA network" (2021). Graduate Theses and Dissertations. 18593.
https://lib.dr.iastate.edu/etd/18593

This Thesis is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
Design, implementation, and field-testing of distributed intrusion detection system for smart grid SCADA network

by

Jeyanth Rajan Babu

A thesis submitted to the graduate faculty in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Engineering (Secure and Reliable Computing)

Program of Study Committee:
Manimaran Govindarasu, Major Professor
Doug Jacobson
Guiping Hu

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this thesis. The Graduate College will ensure this thesis is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University
Ames, Iowa
2021

Copyright © Jeyanth Rajan Babu, 2021. All rights reserved.
DEDICATION

I dedicate my thesis work to my beloved parents, Rajan Babu and Julia, who have been my constant source of inspiration. I dedicate this work to my sister, Jenoshia, for her unwavering support throughout my Master’s degree.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>vi</td>
</tr>
<tr>
<td>NOMENCLATURE</td>
<td>vii</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>viii</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>ix</td>
</tr>
<tr>
<td>CHAPTER 1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Cyber-security for Smart Grids</td>
<td>1</td>
</tr>
<tr>
<td>1.2 State of security of SCADA protocols</td>
<td>2</td>
</tr>
<tr>
<td>1.3 DNP3 SCADA Protocol</td>
<td>3</td>
</tr>
<tr>
<td>1.4 Related Work</td>
<td>6</td>
</tr>
<tr>
<td>1.5 Research gap</td>
<td>7</td>
</tr>
<tr>
<td>1.6 Research overview</td>
<td>8</td>
</tr>
<tr>
<td>CHAPTER 2. DISTRIBUTED INTRUSION DETECTION SYSTEM</td>
<td>9</td>
</tr>
<tr>
<td>2.1 Intrusion Detection System Types</td>
<td>9</td>
</tr>
<tr>
<td>2.2 Distributed Intrusion Detection Architecture</td>
<td>10</td>
</tr>
<tr>
<td>2.3 Security Onion SIEM Tool</td>
<td>10</td>
</tr>
<tr>
<td>2.3.1 IDS pre-requirements</td>
<td>11</td>
</tr>
<tr>
<td>2.3.2 Tools within Security Onion</td>
<td>11</td>
</tr>
<tr>
<td>2.3.3 Distributed Deployment</td>
<td>12</td>
</tr>
<tr>
<td>2.4 Snort IDS</td>
<td>13</td>
</tr>
<tr>
<td>2.4.1 Rule header</td>
<td>14</td>
</tr>
<tr>
<td>2.4.2 Rule actions</td>
<td>14</td>
</tr>
<tr>
<td>CHAPTER 3. PROPOSED D-IDS DESIGN FOR DNP3 SCADA</td>
<td>16</td>
</tr>
<tr>
<td>3.1 Distributed intrusion detection system</td>
<td>16</td>
</tr>
<tr>
<td>3.2 Rule classification</td>
<td>17</td>
</tr>
<tr>
<td>3.2.1 Control command rules</td>
<td>17</td>
</tr>
<tr>
<td>3.2.2 Internal indications (IIN) flag rules</td>
<td>19</td>
</tr>
<tr>
<td>3.2.3 Time threshold rules</td>
<td>19</td>
</tr>
<tr>
<td>3.2.4 Measurement data rules</td>
<td>19</td>
</tr>
<tr>
<td>3.3 Rule design</td>
<td>20</td>
</tr>
<tr>
<td>3.3.1 Control command and IIN rules</td>
<td>20</td>
</tr>
</tbody>
</table>
3.3.2 Measurement data rules ........................................... 21
3.3.3 Time threshold rules ........................................... 24
3.4 DNP3 payload crafting ............................................. 25
  3.4.1 DNP3 CRC checksum ............................................ 25
  3.4.2 DNP3 function codes ............................................ 25
  3.4.3 DNP3 data objects ............................................. 26
  3.4.4 DNP3 payload crafting using Python ......................... 26

CHAPTER 4. TESTBED-BASED IMPLEMENTATION OF DNP3 IDS ........ 27
  4.1 Testbed power system model and network architecture ............. 27
  4.2 Implementation of IDS rules ..................................... 29
  4.3 IDS testing and evaluation ....................................... 30
    4.3.1 IDS rule testing ............................................ 30
    4.3.2 IDS evaluation ............................................. 31
    4.3.3 Sguil alert detection ....................................... 32

CHAPTER 5. FIELD DEPLOYMENT AND TESTING ....................... 34
  5.1 Field deployment IDS architecture ............................... 34
  5.2 Deployment strategy overview ................................... 35
    5.2.1 Phase 0 .................................................... 35
    5.2.2 Phase 1 .................................................... 36
    5.2.3 Phase 2 .................................................... 36
  5.3 Implemented IDS Rules .......................................... 38
  5.4 IDS Testing and Evaluation ..................................... 41

CHAPTER 6. CONCLUSION AND FUTURE WORK ......................... 44
  6.1 Conclusion ..................................................... 44
  6.2 Future work .................................................... 44

BIBLIOGRAPHY .......................................................... 46


LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 3.1</td>
<td>IDS Rule Classification</td>
<td>18</td>
</tr>
<tr>
<td>Table 4.1</td>
<td>Implementation of IDS rules on testbed</td>
<td>33</td>
</tr>
<tr>
<td>Table 5.1</td>
<td>Utility IDS Rule Categories</td>
<td>38</td>
</tr>
<tr>
<td>Table 5.2</td>
<td>Sample analog packet analysis</td>
<td>40</td>
</tr>
<tr>
<td>Table 5.3</td>
<td>Sample analog ruleset</td>
<td>41</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1</td>
<td>DNP3 stack</td>
<td>3</td>
</tr>
<tr>
<td>Figure 1.2</td>
<td>DNP3 Function Codes</td>
<td>4</td>
</tr>
<tr>
<td>Figure 1.3</td>
<td>DNP3 Data Object Groups</td>
<td>5</td>
</tr>
<tr>
<td>Figure 2.1</td>
<td>Snort rule — structure</td>
<td>13</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>byte_test snort keyword</td>
<td>15</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>Testbed D-IDS setup</td>
<td>28</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>IDS Detection Latency</td>
<td>32</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>Testbed — Sguil dashboard</td>
<td>32</td>
</tr>
<tr>
<td>Figure 5.1</td>
<td>IDS Architecture — Local Utility Field Deployment</td>
<td>35</td>
</tr>
<tr>
<td>Figure 5.2</td>
<td>Phase 1 — overview</td>
<td>36</td>
</tr>
<tr>
<td>Figure 5.3</td>
<td>Phase 2 — overview</td>
<td>37</td>
</tr>
<tr>
<td>Figure 5.4</td>
<td>Utility Sguil — IDS Alerts</td>
<td>43</td>
</tr>
</tbody>
</table>
NOMENCLATURE

Smart Grid Terminologies

SCADA Supervisory Control and Data Acquisition

DNP3 Distributed Network Protocol 3

ICS Industrial Control Systems

IED Intelligent Electronic Device

OT Operational Technology

Cyber-security Terminologies

IDS Intrusion Detection System

NIDS Network Intrusion Detection System

SIEM Security Information and Event Management

Other Terminologies

IT Information Technology

SSH Secure Shell Protocol

VM Virtual Machine
ACKNOWLEDGMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis.

First and foremost, my major professor Dr. Manimaran Govindarasu, for his guidance, patience and support throughout this research and the writing of this thesis. His insights and words of encouragement have often inspired me and motivated me to dig deep into my research work.

I would also like to thank my committee members for their efforts and contributions to this work: Dr. Doug Jacobson and Dr. Guiping Hu.

I would like to express my gratitude and thanks to Dr. Gelli Ravikumar, research assistant professor, for his mentorship in research throughout my graduate study.

I would like to thank my research colleagues at the PowerCyber Lab and my friends for their motivation and support in carrying out my thesis work.
With the electric power sector facing a growing trend of cyber-attacks in the past few years, the need for securing the power grid has never been higher, given the impact, any breach in security could cause. This has motivated the need for robust cybersecurity solutions for the grid communication protocols covering major grid operations. Smart grid systems use SCADA communication protocols such as DNP3, IEC 61850, Modbus etc., to communicate between the control center and the substations within the utility. Existing Intrusion Detection Systems (IDS) are focused on Information Technology (IT) rules and few other smart grid protocol-specific Operational Technology (OT) rules, but lack when it comes to rules for DNP3 smart grid communication protocols tailored to the utility. This thesis focuses on developing utility-specific IDS for monitoring the intrusions and anomalies in the DNP3 smart grid communications protocol. The developed solutions have been deployed and tested in a smart grid testbed environment emulating the IEEE 39-bus smart grid network, using Security Onion, an open-source SIEM tool. The IDS solutions have also been field deployed in a local utility’s distribution grid environment where it has been tested against actual smart grid traffic from substations.

This thesis makes the following contributions: (i) designed a distributed IDS for DNP3, (ii) developed IDS snort rules for utility-specific functions covering the major grid operations, (iii) created DNP3 attack scripts to craft malicious DNP3 payloads which will be used to test the designed IDS solution, (iv) deployed the IDS solutions in a testbed and in a local utility’s distribution grid environment, to test the efficacy of the rules to detect intrusions and anomalies in the system. The test results showed that the IDS was able to detect all the attack cases for the test cases considered and it was observed that the designed IDS latency is within the acceptable communication time limits between the substation and control center.
CHAPTER 1. INTRODUCTION

1.1 Cyber-security for Smart Grids

Electric power sector is one of the four critical infrastructures that are essential for carrying out normal day-to-day activities. The energy consumption has been on the raise exponentially in the last few decades with the ever-increasing population and technology advancements. While advancements have been ongoing in the power sector on improving the energy infrastructure, more emphasis has been placed on securing the communication flow within the power grid, which is now being called as smart grid. The impacts of security lapses/breaches in the utility network could be catastrophic, resulting in damage to life and property. Smart grid Supervisory Control and Data Acquisition (SCADA) communication research have been done in plenty in the last decade or two. Earlier, isolating the smart grid from internet connectivity was thought of as a better strategy. But the cyber-attacks carried out on utilities, such as Stuxnet (1), where a malware was successfully implemented and exploited in an Iranian nuclear facility, despite being isolated from internet, was an eye-opener to implementing secure smart grid communication. Securing the smart grid communications network is of high priority, with the latest cyber-attack on Solarwinds (2) which has plenty of customers on the power and oil sectors, showcasing the need for advancements in cyber-security. The other notable cyber-attacks targeting the Industrial Control Systems (ICS) network were the Duqu (3) and Flame (4) malwares, which serve the purpose of information gathering. Disconnecting the smart grid network from internet would only be a backward step in the current ages of booming IT advancements. With the onset of Distributed Energy Resources (DER) such as solar photo-voltaic cells and electric vehicles (EV) integrated into the smart grid, the concept of prosumers, where end users owning DER can contribute to power generation, have advanced. DER integration have made the need for secure smart grid communication to be a pressing need.
1.2 State of security of SCADA protocols

The typical grid operations include sending control signals to the lower level devices which can be the substation devices or field devices or end user devices, depending on the hierarchical structure of the smart grid network. The control signals sent by the control center could be opening/closing a relay or setting a generation set point. Polling the lower level devices to read the measurement data both binary and analog, is another major grid operation. A few smart grid SCADA communication protocols include DNP3, IEC 61850, Modbus and SEP 2.0. The communication protocol used by the utility depends on the vendor applications/devices used by the utility and also on the region in which the utility is present. DNP3 has been predominantly used in the United States whereas Europe predominantly uses IEC 61850. The SCADA communication protocols carry out the grid operations. While power system operators have visibility over physical electrical parameter values such as voltage or current, they lack visibility over what goes on under the digital communication interface of the power grid which is comprised of the SCADA communication protocols. Information security has been far more advanced for IT communications and web applications. But they lag far behind when it comes to secure ICS communication. Currently there are plenty of smart grid communication protocols, and creating a new standard would only be an addition to the existing collection of smart grid protocols. Also the ICS protocols were created with availability of service in mind and security was never considered. This made the SCADA protocols inherently insecure. The security tools or applications that decode the network traffic and provide visibility over the contents of the SCADA traffic are relatively low, and are not widely available commercially. This is owed mostly to the lack of communication protocol knowledge. Firewalls for SCADA communications suffer from similar issues which is why intrusion detection systems come into picture. While there are plenty of intrusion detection systems (IDS) available depending on the detection methods, the signature-based snort IDS has been chosen as the way to perform intrusion detection in this research work. This is because of the ability of the IDS to cover all the grid operations without false positives. This research focuses on designing IDS for DNP3 SCADA communication protocol.
1.3 DNP3 SCADA Protocol

This section describes the protocol stack of DNP3, functional operations, function codes, IIN, CRC checksums and data objects. DNP3 protocol has 3 layers on top of the TCP/IP layer namely - DNP3 data link layer, DNP3 transport layer, and DNP3 application layer, described in (5). The Figure 1.1 showcases the three DNP3 layers.

Normal DNP3 Master operations include,

1. Send control command signals to the DNP3 secondary devices. The control command signal are sent using function codes such as read, write, operate, and cold restart.

2. Poll the DNP3 secondary device for data point values. Data points hold the electrical parameter data such as voltage, current, frequency or active power.

A single DNP3 frame ranges in size from 0-65535 bytes. If the DNP3 data exceeds the frame size, they are carried forward onto the next frame and reassembled at run-time. DNP3 data link layer is responsible for this process. DNP3 function codes carry out the control command operations. The list of DNP3 function codes are shown in Figure 1.2 from (6).
DNP3 internal indications flags (IIN) are error detection bits present in the DNP3 response packets. They indicate error conditions in the DNP3 secondary devices such as event buffer overflow, device restart, device trouble, parameter error, corrupt config and many more. The IIN flag field is of 2 bytes length, thereby resulting in 16 IIN bits. DNP3 protocol has the CRC checksum feature which detects errors during transmission. DNP3 packets have a CRC checksum field for every 16 bytes with the exception of the DNP3 data link layer which has a CRC checksum at the end of the layer.
The DNP3 data objects group the data points configured within the DNP3 secondary devices. Figure 1.3 from (5) showcases the various DNP3 data object groups. Each data object group is dedicated for a particular data format of data point. And each data object group header is represented by a combination of group number, variation and qualifier, say analog inputs fall under group 30 and are represented as g30v05. The data points are present after the data object group header. The data points could hold binary values, analog values, and even counter values.

**DNP3 group numbers generally fall into categories as follows:**

<table>
<thead>
<tr>
<th>Device Attributes</th>
<th>group number 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary Inputs</td>
<td>group numbers 1 to 9</td>
</tr>
<tr>
<td>Binary Outputs</td>
<td>group numbers 10 to 19</td>
</tr>
<tr>
<td>Counters</td>
<td>group numbers 20 to 29</td>
</tr>
<tr>
<td>Analog Inputs</td>
<td>group numbers 30 to 39</td>
</tr>
<tr>
<td>Analog Outputs</td>
<td>group numbers 40 to 49</td>
</tr>
<tr>
<td>Time</td>
<td>group numbers 50 to 59</td>
</tr>
<tr>
<td>Class</td>
<td>group numbers 60 to 69</td>
</tr>
<tr>
<td>Files</td>
<td>group numbers 70 to 79</td>
</tr>
<tr>
<td>Devices</td>
<td>group numbers 80 to 82</td>
</tr>
<tr>
<td>Data Sets</td>
<td>group numbers 83 to 89</td>
</tr>
<tr>
<td>Applications</td>
<td>group numbers 90 to 99</td>
</tr>
<tr>
<td>Alternate Numerics</td>
<td>group numbers 100 to 109</td>
</tr>
<tr>
<td>Other</td>
<td>group numbers 110 to 119</td>
</tr>
<tr>
<td>Security</td>
<td>group numbers 120 to 129</td>
</tr>
</tbody>
</table>
1.4 Related Work

The DNP3 smart grid protocol is not inherently secure. The paper (7) details the various attacks that could be made on the DNP3 protocol. The paper (8) describes an event buffer flooding attack on the DNP3 protocol. While there is a latest version of DNP3 with secure authentication feature (5), it is still not widely used in practice in utilities. Works from (9), (10) and (11) have showcased man-in-the-middle attacks (MITM) on DNP3 protocol.

The authors of (12) have developed an IDS system for IEC 61850 GOOSE protocol, evaluated on a smart grid testbed. A predefined set of rules have been developed to trigger the IDS alerts. The system reads the log files which are generated when any grid operation such as opening/closing of relay occurs. The paper though does not specify the technique used to read the log files, what actions are captured in the log files or the latency associated with capturing the logs and its subsequent detection.

In (13), the authors have developed a state-based IDS that utilizes a signature-based IDS engine and derived state information of the system, for the DNP3 and Modbus protocols. This paper covers a wide range of potential intrusions on DNP3 and Modbus, by defining an acceptable behavior for the smart grid system through system analysis. If the state derived is classed as unacceptable, an alert is triggered. This paper though fails to showcase the range of grid functions it covers, such as measurement data violations.

In (14), the authors have developed a specification-based IDS for DNP3 using Bro IDS which is now renamed as Zeek IDS. A DNP3 parser has been designed and a protocol validation policy is created. The protocol validation policy defines acceptable behavior based on inter- and intra-packet relations, say a SELECT packet has to be followed by an OPERATE packet. The packets that fail the designed validation policies are flagged as anomalous. This paper however does not mention the range of validation policies developed and how they were inferred.

The authors of (15) have developed an IDS for IEC 61850, but have focused their efforts on mitigating/detecting attacks made on IED. The paper hence describes the attacks such as password
crack using Telnet and FTP, DOS attacks and ARP spoofing, but does not detail the attacks focused on IEC 61850 protocol.

(16) have developed an IDS for IEC 61850 protocol, comprising of access control detection, protocol whitelisting and multi-parameter detection, utilizing knowledge on protocol specifications, logical and physical behavior.

The papers (17) has developed signature-based snort IDS for DNP3 protocol, covering many of the grid functions, with an exception of analog measurement violations.

In our work (18), we have developed an IDS for Modbus protocol, established for a DER-integrated network. The IDS system however is limited to Modbus and the protocol specifications and features of DNP3 are very different.

1.5 Research gap

Efforts have been ongoing to bridge the gap between advancements in energy infrastructure and improved smart grid cyber-security. Several solutions have been suggested and research works have been undertaken to improve the security of smart grid SCADA communication protocols. The solutions range from employing multi-agent protection schemes that consider physical grid parametric changes to intrusion and anomaly detection systems. The works from section 1.4 have showcased the various research works undertaken to improve cyber-security of smart grid communication. However there exists a gap in the research activities carried out, as most suggested solutions do not cover all the avenues in which the smart grid protocols could be used. This leads to a gap in the suggested cyber-security solutions, be it IDS, ADS or any specification-based IDS, as they do not cover the major grid operations. Also, most solutions target the substation control systems, focusing on IT attacks such as DOS, Password crack or spoofing. There are plenty of commercially available plug-and-play cyber-security solutions for IT attacks, but very few for smart grid SCADA protocols, which is the major motivation behind carrying out this research work.
1.6 Research overview

The research work has been focused on designing an IDS for DNP3 smart grid communication protocol, covering all its grid operations. A network IDS deployed in a distributed architecture, and detecting network traffic based on signatures for DNP3 protocol has been developed. Distributed architecture of IDS have an IDS Master at the control center of the utility and several sensors at the various substations of the utility. The thesis work covers the attack surfaces exposed by the DNP3 protocol communication between the control center and substations of the utility’s distribution grid. All possible intrusions and anomalies arising from DNP3 SCADA operations have been discussed. Ideally control center sends out control signals to the substations which will then be relayed to the field devices and end devices at user locations. The substations are geographically distributed across the regions, and control center will have to securely communicate with them. This setup opens up possibilities of MITM attacks or physical damage of end devices. All these can be monitored by an IDS system built by modeling system behavior and packing them into rules. The set of rules developed constitute control command signals and measurement poll responses from substation. Any violation from the normal grid operation can be captured by the IDS system as alerts and made visible to the system operator monitoring from the utility control center. The proposed IDS has been designed on a cyber-physical testbed setup emulating a smart grid network, implementing an IEEE-39 bus model. The developed IDS is then tested with power flow simulation applications and then field-tested within a local utility, where the IDS rules have been successfully deployed.

The thesis work has been organized into the following chapters - Chapter 2 describing the intrusion detection system model and technology stack; Chapter 3 elaborating the proposed IDS design, its rule categories, how the rules were designed and DNP3 payload crafting for rule validation; Chapters 4 and 5 describing the case studies of the proposed IDS deployments; and followed by the conclusion and future work chapters.
CHAPTER 2. DISTRIBUTED INTRUSION DETECTION SYSTEM

This chapter explains the various concepts that affect the working of an intrusion detection system. This chapter details the classification of intrusion detection systems based on their detection technique, the distributed intrusion detection architecture implemented, the Security Onion SIEM tool, and the working of snort IDS.

2.1 Intrusion Detection System Types

Intrusion detection systems could be classified into many different types based on their implementation, functionality and detection methods. The IDS discussed here fall under the network-based IDS (NIDS) category that monitor network traffic. The two basic NIDS have been categorized based on their detection methods, and include the following,

1. signature-based IDS

2. anomaly-based IDS

The signature-based IDS works by looking for specific hex codes or ASCII signatures within the network traffic. IDS alerts are then raised when those signatures or combination of signatures are detected. Snort (19) and Suricata (20) are two widely used signature-based IDS. Signature-based IDS cannot detect unknown attacks which is their major drawback.

Anomaly-based IDS work on modeling the good/expected behavior of the system and flag any activity that falls outside the good behavior model to be anomalous. They employ statistical machine learning methods to model the good behavior of the system. More sophisticated IDS implement multi-agent methods to obtain information on the different system operations and correlate them to identify the anomalous behavior. Although anomaly-based IDS can detect unknown attacks, they suffer from plenty of false positives.
In this thesis work, we have used signature-based IDS to perform both intrusion detection and anomaly detection for the DNP3 protocol by modeling all the major grid operations and system behavior as snort rules. Any behavior that falls outside this, are captured by the snort rules and are triggered as alerts. Also the proposed IDS design does not suffer from false positive issues as it is a signature-based IDS utilizing snort, which has been fine tuned to avoid any false positive or false negative.

2.2 Distributed Intrusion Detection Architecture

Distributed intrusion detection system architecture follows a client-server model in which an IDS Master exists as the central node while many IDS sensors deployed at geographically remote locations, act as sensor nodes. The sensors sniff on traffic in the network and trigger alerts based on the rules deployed within them. These alerts are then forwarded to the IDS Master which then visualizes the alerts on a dashboard. The advantage of distributed deployment lies in the fact that the IDS rules corresponding to each sensor can be deployed at the IDS Master and then forwarded to the IDS sensors through a pub-sub technique. Security Onion documentation (21) details the distributed intrusion detection architecture and (6) also describes the implementation of the distributed IDS architecture in a testbed environment. In our case, the D-IDS architecture has been deployed with the IDS Master being located within the control center network and the IDS sensors being deployed at the substation networks.

2.3 Security Onion SIEM Tool

IDS systems are implemented using Security Information and Event Management (SIEM) tools. This research work utilizes Security Onion (22), an open-source SIEM tool, available on Ubuntu platforms, to deploy the IDS solutions. Security Onion provides NIDS, HIDS and ElasticSearch features through tools and packages such as Snort IDS, Suricata IDS, Zeek IDS, Logstash for log management, Kibana for ElasticSearch and alert visualization, OSSEC for HIDS, Sguil for IDS alert rule visualization and many other tools. It is a swiss-army knife that incorporates all the
features mentioned above through the collective functioning and integration of the aforementioned tools.

### 2.3.1 IDS pre-requirements

The IDS requires the following pre-requirements for successful installation and functioning,

1. Hardware requirements: For deployments in production network, the hardware requirements from (23) will have to be present.

2. Network interface: The nodes must have two network interfaces - sniffing and management. Sniffing interface monitors the network traffic while the management interface is responsible for sending alerts/receiving IDS rules.

3. Sniffing interface requirement: The sniffing interface must be either configured in promiscuous mode to view traffic directed to other devices in the network, or port mirroring has to be setup to forward all network traffic to a specific port which can then be monitored by the sensor.

4. Management interface requirement: The management interface should have SSH communication enabled and hence the TCP port 22 will have to be open always. The SSH port can be changed to a custom port by making changes to SSH configuration files module within Security Onion.

5. The UEFI secure boot during installation of virtual machines, hinders functioning of snort IDS, as it disables running of third-party modules such as PFRING. PFRING is crucial for working of snort master and sensor nodes. In order to use snort IDS, UEFI secure boot has to be disabled.

### 2.3.2 Tools within Security Onion

Security Onion being a SIEM tool includes several other tools for network security monitoring (NSM). This section only describes the tools essential for this IDS design.
1. **Snort**: Snort is the signature-based IDS included with Security Onion. Snort is responsible for detecting network traffic signatures which will in turn trigger alerts. Snort includes several keywords, each performing its own function. These keywords and their functions are explained in Section 2.4.

2. **Sguil**: Sguil is the visualization dashboard on IDS Master in which the alerts could be viewed. The dashboard shows the alert message, number of times the alert was triggered, and packet information such as IP and port number, among other information. Sguil has a MySQL database backend that stores the alert information.

3. **Kibana**: Kibana is the analytics tool within Security Onion which will also visualize the alerts and as well provide analytical information regarding the rules. ElasticSearch uses Kibana to view its alerts.

4. **Pulled Pork**: Pulled Pork (24) package within snort carries out the task of establishing the pub-sub rule update model. The pulledpork.conf file in the sensor can be configured to avoid certain rules based on their unique sid and pcre content match. By default, the Pulled Pork is scheduled to execute everyday automatically at 8 am GMT through cron daemon, which can also be disabled by the security operator. The operator can also perform the rule-update to the sensors manually using the rule-update command.

### 2.3.3 Distributed Deployment

Security Onion enables distributed deployments of IDS, as shown in (21). In order to enforce the distributed deployment, forward node configurations of sensor nodes are followed, with storage nodes being an optional configuration in case elastic search features have to be implemented. ElasticSearch enables cross-cluster search and is beyond the scope of this research.

**Master Node**: Master node is the master server that includes components such as Kibana, Sguil, Logstash, Redis and ElasticSearch. All the rules to be deployed at the sensor locations are designed at the Master. These rules are then forwarded to the sensors using PulledPork. The
Master node receives the alerts from the sensor nodes. These alerts are then stored in a MySQL database which is part of Sguil. The alerts are then visualized on Sguil dashboard. Kibana can visualize the alerts as well along with some additional analytical features.

**Forward Node:** The forward nodes carry out the sensory functions utilizing components such as snort, suricata, and Syslog-NG. Forward nodes cannot perform ElasticSearch as a heavy node is required to carry out those functions and is beyond the scope of this research. The forward nodes sniff on network traffic and send the detected alerts to the Master node.

### 2.4 Snort IDS

Snort IDS detects signatures from network traffic based on snort rules. Snort rules are basically divided into two parts - the rule header and rule action, shown in Figure 2.1.
2.4.1 Rule header

The rule header detects the fields from the TCP/IP layer such as protocol used, source and destination IP addresses, and source and destination port numbers. The rule headers contain action to be performed on the packet such as alert, drop and log. Typical IDS use alert action predominantly to perform passive alerting on detection of the signatures in the rule. More aggressive IDS can be made to behave as IPS and drop packets. The protocol field detects what protocol the payload utilizes. These could be TCP, UDP, or ICMP. The IP and port fields are detected from the network packets. The rule header also contains the traffic flow direction field which indicates which direction the traffic is flowing, from the source to destination or vice versa.

2.4.2 Rule actions

Rule actions specify how the signature matching takes place. Several snort keywords that perform these actions are explained below,

1. msg: Content of message to display on the alert.

2. content: Allows for signature matching which could be hex codes (|FF|) or ASCII characters.

3. depth: Indicates how far to look for a certain signature. Depth is not relative to previous content match. It starts from start of protocol application layer. Depth, offset, within and distance keywords must be used in conjunction with content keyword.

4. offset: Offset indicates where to start looking for a content match.

5. within: It ensures that there are as many bytes as mentioned in the within keyword, from the previous content match. Depth performs an absolute match from the start of the payload whereas within performs a relative match from previous content match.

6. distance: Distance ignores as many bytes indicated from the previous content match.
7. byte_test: It is used to compare a byte field against a value depending on the comparison operator used. It includes the fields such as number of bytes to test, operator, byte value, offset, value format, and little or big endian, as shown in Figure 2.2.

```
byte_test:<bytes to convert>, [!]<operator>, <value>, <offset> \
[, relative][, <endian>][, string, <number type>][, dce] \
[, bitmask <bitmask_value>];
```

- bytes = 1 - 10
- operator = '<' | '=' | '>' | '<=' | '>=' | '&' | '!='
- value = 0 - 4294967295
- offset = -65535 to 65535
- bitmask_value = 1 to 4 byte hexadecimal value

Figure 2.2 byte_test snort keyword

A snort rule containing (byte_test: 1, >=, 0xfa, 4, relative, hex, little;) matches on a single byte value greater than or equal to 250 (decimal equivalent of 0xfa) at an offset of 4 relative to previous content match, with the value being processed in little endian format (LSB and MSB are reversed).

8. pcre: It matches on perl regular expressions.

For example, a sample snort rule given below.

```
alert tcp $src_ip $src_port ->$dst_ip $dst_port (msg:"Sample rule"; content: "Hi"; offset:4; depth:6; content: "user"; distance:2; within:6; sid:9500000;)
```

This snort rule will fire on a payload having the text "Hi" as the 4th byte from the start (indicated by offset) but within 6th byte (indicated by depth), and then having the text "user" after 2 bytes but within 6 bytes from "Hi".
CHAPTER 3. PROPOSED D-IDS DESIGN FOR DNP3 SCADA

This chapter describes the IDS design proposed for the smart grid DNP3 communication network. The IDS architecture follows a distributed deployment model from (6), in which the IDS sensors are deployed at various locations and in most cases, these locations are remote. These sensors then communicate with the IDS Master. Distributed deployment has been briefed in sections 2.2 and 2.3.3. The proposed IDS design includes IDS snort rules for various DNP3 functions such as control commands, IIN flags, and analog measurement data. This section also describes creating DNP3 attack scripts from Python for various DNP3 functions. The proposed IDS is capable of covering all the functions of DNP3 smart grid communications without any false positives which was previously not possible. Though the control command and IIN-based rules could be developed, as shown in works of (6) and (25), analog data violations could never be detected. The proposed IDS presents a way to monitor DNP3 communications as a whole, covering all its functions and grid operations.

3.1 Distributed intrusion detection system

D-IDS architecture: The Distributed IDS architecture is a deployment model from (21), with forward nodes and local storage configurations. This model has the IDS sensors deployed at the substation send alerts to a central IDS Master at the control center. The IDS Master sends IDS rules to the sensors through a pub-sub model. The rules can be configured to be published to the sensors either periodically through PulledPork or through a need-basis using rule-update command. The sensors and the master are called nodes. Each node has two network interfaces - management interface and sniffing interface. The management interface is the communication interface between the sensors and the master. It is important to note that the sensors do not communicate with each other but only to the master. SSH tunnels are used to setup secure communication channels
between the master and the sensors. The sniffing interface is the network interface in which the
sensors sniff the network traffic. Usually this interface is the substation network in the utility that
do not have access to the internet. The IDS Master is present at the control center while the IDS
sensors are deployed at the substation networks. The sniffing interface of the sensor is configured
to be in promiscuous mode, giving the sensors an ability to monitor traffic directed to other devices
within the substation network such as the substation RTU.

**IDS using rule-based techniques:** The signature-based snort IDS was used as the IDS en-
gine. Snort has plenty of keywords, each with its own functionality, shown in 2.4. The idea behind
this IDS design is to detect violations, both intrusions and anomalies, within the DNP3 communi-
cation. Few examples of intrusions include unauthorized DNP3 control command injections from
an external adversary such as unauthorized select-operate, cold restart and other function code
executions. Examples for anomalies include frequency violation, voltage violations, repeat device
restart IIN responses, repeat operate or direct operate function code execution even from the con-
trol center. IDS rules have been developed to cover all these functional categories and have been
tested using specially crafted DNP3 packets using Python which will be explained in the upcoming
sections.

### 3.2 Rule classification

This section describes the various categories of IDS rules that have been developed. Leaving out
the IT rules such as reconnaissance attempts, file operations and DOS attempts, the rule categories
detail the DNP3 rules. These include control command rules, IIN rules, time threshold rules and
measurement data rules. The Table 3.1 describes the rule categories.

#### 3.2.1 Control command rules

Control command rules target the DNP3 control command functions established through func-
tion codes. DNP3 function codes could be found in detail in (5). Few of those function codes
include read, response, select, operate, direct operate warm restart, cold restart, enable unsolicited
Table 3.1  IDS Rule Classification

<table>
<thead>
<tr>
<th>Rule Category</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Command Rules</td>
<td>DNP3 FC</td>
<td>Snort rules to detect DNP3 control command function codes such as Operate, Direct Operate, Enable Unsolicited, Disable Unsolicited, Warm and Cold Restart.</td>
</tr>
<tr>
<td>IIN Rules</td>
<td>DNP3 IIN</td>
<td>Snort rules to trigger alerts on presence of IIN bits set in DNP3 Response packets from substation RTU, such as Corrupt Config, Event Buffer Overflow, Device Restart and Device Trouble.</td>
</tr>
<tr>
<td>Time Threshold Rules</td>
<td>DNP3 FC and IIN</td>
<td>Snort rules to trigger on detection of repeat anomalous conditions within a pre-defined time frame. Rules include Repeat Device Restarts, Repeat Operate and Warm/Cold restarts.</td>
</tr>
<tr>
<td>Measurement Data Rules</td>
<td>DNP3 DO</td>
<td>Snort rules to detect anomalies in the analog measurement data reported by DNP3 secondary devices by checking on boundary conditions. Rules include Phase and Line voltage anomalies, over and under frequency violation, and active power reversal.</td>
</tr>
</tbody>
</table>

response and disable unsolicited response. The control center issues control commands to the DNP3 secondary devices at the lower levels using the function codes. The control commands could include write time-and-date data to synchronize the secondary devices with the primary, poll the data point values categorized by classes, or issue commands such as operate to open/close relays. These control commands from the master are critical to the operations of the smart grid. For example, cold restart can perform a complete restart of all the applications in the DNP3 secondary device and reset the DNP3 software. This category of snort rules alert when these critical control commands are issued by the master or by an external adversary in the smart grid network. The idea behind the detection of these critical functions is to provide visibility to the system operator that such a function is being executed on the remote DNP3 secondary devices. The criticality of the control commands are classified based on the impact caused by them. Developing a rule to trigger on frequent control commands issued by the control center will not make sense as it only increases the number of alerts generated, and such visibility will not be desired by the operator. In
such cases of control commands, only the commands issued by an external adversary are flagged as alerts. The 3.3 section details how this is achieved.

### 3.2.2 Internal indications (IIN) flag rules

IIN abbreviated as internal indications, are error detection bits in the DNP3 response packets. If the DNP3 secondary devices do not have any error condition, the IIN bits are all reset to zero. There are 16 bits in IIN, each indicating an error condition or a critical message to the primary. Some of the IIN bits include end device restart, device trouble, substation in local control, event buffer overflow and many more. Not all IIN bits are critical and must be flagged. This category of snort rules alert when any of the critical IIN bits such as device restart or event buffer overflow are set. This gives more visibility to the operator regarding the error condition at substations located remotely. Few of the flagged critical IIN bits include device restart, device trouble, data points in end device in local control mode, corrupt config, and event buffer overflow.

### 3.2.3 Time threshold rules

Time threshold rules are triggered when the anomalous packets are found more than once in a short time frame. Alerts under this category include repeat device restarts, repeat operate command issued, and repeat event buffer overflow. The time frame is determined after DNP3 request - response packet analysis of the system, computing the inter-arrival rate of packets. The time taken for successive response packets to arrive can help estimate the inter-arrival rate of response packets. This gives us an idea of how many response packets can be observed within a time frame, say 15 seconds, under normal behavior of the smart grid. This metric helps us design the time threshold rules for DNP3.

### 3.2.4 Measurement data rules

This category of rules focus on measurement data responses that the DNP3 secondary devices send in response to poll requests from the primary. Though measurement data points could hold
binary as well as analog data, these rules focus only on analog measurements, as binary data can only indicate status information, say if a relay is open or closed. Measurement data rules work based on boundary conditions, having an upper and a lower limit. If the concerned data point holds frequency values, the ideal range of frequency would be 59-61 Hertz. Any frequency values beyond this range would be considered as an error or a fault condition within the smart grid network. Although the data point could theoretically hold values beyond the range, these behaviors are generally an anomaly under normal system operation. For instance, it would be impossible for the system frequency to be 40 Hertz. The ideal range for the various data points could be determined after consultation with the power system engineers. The data points that need to be observed have IDS rules targeting them and they alert the system operator when any of those violations are observed within the smart grid network.

3.3 Rule design

This section describes how the snort rules for the various rule categories from Table 3.1 are designed. An important fact to remembered while developing snort rules is that the snort engine will have to know the exact position of the hex codes for performing any content matching or any sort of value comparisons. If the position of a field within a payload is changing dynamically, snort engine cannot make the adjustments automatically. The IDS rules will have to be designed in a way in which the dynamically changing positions of fields can be correctly located. This issue is addressed in the Measurement data rule design sub-section 3.3.2, where the position of data objects will vary from packet to packet. This section explains the usage of the different snort keywords to arrive at the snort rules.

3.3.1 Control command and IIN rules

The snort rules for control command and IIN categories are rather straightforward. These two rule categories are bunched together as they can be designed using similar methods. They can be
designed in two ways, using the dnp3preprocessor of snort or using snort keywords such as content, distance, depth and within. Only the rule action part of snort rules are explained here.

### 3.3.1 Without preprocessor

The DNP3 link layer header (05 64) will have to be matched first, as any DNP3 packet starts with that hex code. Then the `content` keyword matches on the DNP3 function code in combination with `distance` keyword which indicates how many bytes will have to be present between the previous header match and the function code. In case of IIN, the DNP3 response function code |81| will have to be matched, followed by the relevant value of IIN fields.

### 3.3.1.2 Using preprocessor

The `dnp3_func` snort keyword filters the relevant DNP3 function codes, say dnp3_func: cold_restart. This alerts when any DNP3 packet containing cold_restart function code is observed in the network. For IIN-based rules, the keyword `dnp3_ind` is used. The usage is as follows dnp3_ind: device_restart. This would alert when any DNP3 response packet with device restart IIN bit set is observed.

### 3.3.2 Measurement data rules

The design of measurement data rules can be classified into two - packets without input change event and packets with input change event. DNP3 data objects within the DNP3 packet typically include binary input, binary output, analog input and analog output. The presence of the data objects differ for different devices. Input change event data objects indicate how many data points have changed from the previous poll operation. The presence of input change event data object is usually at the start of the DNP3 data objects. It alters the position of the subsequent DNP3 data object groups in the packet. For a DNP3 secondary device with 20 data points, the reported input change events could be anything between 1 to 20. This alters the position at which the other data objects such as analog input start. Snort needs to find the data object header signature of the
targeted data point within the packet, and then compare the analog values using *byte_test* keyword. The position occupied by a data point within the packet is dependant upon,

1. The number of data points configured within the device

2. The datatype of the data point, say 16-bit signed integer or 32-bit floating point.

3. The presence of CRC checksums for every 16 bytes of data

Once the data point is correctly located, the *byte_test* keyword can compare the data value against the boundary conditions which are going to be the upper and lower limits that the data point can take. Therefore basically two rules are required to detect violations for a single data point, given that no input change event object is present in the packet and the position of the data point is not going to change.

**Without input change event:** For DNP3 packets without input change event data object (g32v02), the position of measurement data object header, say, analog input data object, is not going to change. This is the case for testbed and test-lab networks. The DNP3 data object group, variation and qualifier together form the analog data snort signature. The analog measurement snort signature for analog input is |1e 02 01 00|. For designing the measurement data rules for this category, the following have to be done,

1. Match the DNP3 start header (|05 64|)

2. Match the DNP3 response function code (|81|)

3. Match the DNP3 data object header, |1e 02 01 00| for analog input

4. Compare data point value using *byte_test*, say data point 3 within analog input data object

Each content keyword is accompanied with distance, offset or within keywords to match the contents at different positions. A sample snort rule is shown below.
With input change event: Utility distribution grid environment have plenty of data points whose values keep changing more frequently. DNP3 protocol includes an input change event data object (g32v02) which lists all the data points that have changed from the previous poll request from the control center DNP3 primary. This dynamic positional change is compensated by use of distance and within snort keywords. It is imperative to first detect the position of the analog data signature. The position is going to vary for input change event count 1 to input change event counts 20. By padding the distance keyword with position of signature when input change event count is 1 and padding the within keyword with the position of signature when the input change event count is 20, the range of bytes that snort looks for the analog data signature can be extended, thereby snort can be made to detect the signature within the range of bytes specified above. Once snort finds the last instance of the signature, it moves on to compare the data point values using byte_test. To simplify things, the distance keyword holds the position value when only one input change event is present. The within keyword holds the position value for the maximum number of change events. This rule design effectively increases the range of positions to look out for the DNP3 data object header signature within the DNP3 payload. A sample snort rule is given below.

```
alert tcp $DNP3_SLAVE $DNP3_PORTS ->$DNP3_MASTER any (msg: "Voltage violation >1.04pu @ Richview Substation"; content: "\[05 64]\"; depth: 2; content: "\[81\"; distance: 10; content: "\[1e 02 01 00\"; distance: 2; byte_test: 2, >, 0xc842, 4, relative; sid: 96500125;)
```

Need for many analog rules: The presence of CRC checksums and the data object header signatures repeating in a cyclic manner for different number of input change event fields, enables us
to group the change event numbers that have similar patterns. The pcre keyword helps us do that
matching over a range of input change event values having similar patterns. Here the pcre matches
for input change event counts 2, 6, 9, 10, 15, and 18. With the distance and within keywords set
to 72 and 183, the snort IDS engine is going to look out for the signature |1e 02 01 00| between
positions 72 and 183 from the previous content matching for input change event counts which is
done through pcre. The analog data signature cannot be maintained the same as the CRC check-
sum maybe present within it in certain cases. In such cases, the signature will have to be broken
down to discard the CRC checksum. Each differing signature and changing positions of distance
and within keywords will have to be captured with a separate rule, giving rise to a scenario in
which many rules will have to be written for a single analog violation, in order to eliminate false
positives and to correctly capture all the change event cases. Without pcre, 2n number of analog
rules will have to be for n number of data points in a substation DNP3 secondary device.

3.3.3 Time threshold rules

Time threshold rules utilize the threshold snort keyword, which in turn has the type, track_by,
count and seconds fields. The packets can be tracked by source or destination. The rule is triggered
when the count of the packets exceed the limit mentioned in the rule within the time frame men-
tioned through seconds field. This rule is effective in determining violations such as repeat device
restarts or repeat trip operation. A sample snort rule in this category is shown below.

```snort
alert tcp $DNP3_SLAVE $DNP3_PORTS ->$DNP3_MASTER any (msg: "Test-
Lab Repeat Device Restarts"; dnp3_ind: device_restart; threshold: type both,
track by_src, count 3, seconds 5; sid: 96500169;)
```

This rule triggers when at least 3 IIN device restart response packets are detected within 5
seconds from the DNP3 slave device.
3.4 DNP3 payload crafting

This section describes the technique behind creating DNP3 payloads for various functionalities using Python scripts. These scripts will then be used to validate the IDS rules by injecting them from a Kali machine deployed outside the smart grid network. These DNP3 payloads are designed to target the DNP3 secondary devices at the substation RTU or DNP3 primary at the control center, prompting them to perform certain functions such as select-operate on substation relays or report analog data violations to the DNP3 primary.

3.4.1 DNP3 CRC checksum

DNP3 CRC checksums are the error detection bits and are 2 bytes long. The checksums are used to detect errors in transmission but cannot guarantee security against cyber-attacks. These checksums can be computed using (26) packages on Python or using online CRC calculator from (27) for every 16 bytes of data within DNP3 packet. This section does not go in-depth into the mathematics behind DNP3-CRC calculations. These checksums are computed for every 16 bytes with an exception of the DNP3 data link layer for which the CRC is computed once, at the end of the layer. For a valid DNP3 packet to be crafted, correct CRC checksums are very important. Packets with incorrect CRC checksums will be outrightly discarded by the DNP3 slave or master.

3.4.2 DNP3 function codes

DNP3 function codes are used to perform control command operations on DNP3 secondary devices. The function code is a part of the DNP3 header, present within the DNP3 application layer. This position of function code within a DNP3 packet is always fixed. Each function code is identified by an unique hex code, each performing its own operation. The DNP3 data objects associated with a function code also changes depending on the function code used. For instance, a select/operate function code DNP3 packet has the information of data point (say, IED-1) it wishes to perform the trip or close operation on. The DNP3 data object header will contain the DNP3
object group, variation, qualifier, start and end bits, and the operation code for trip or close. Care must be taken to compute the CRC checksum bytes for every 16 bytes of the DNP3 packet.

3.4.3 DNP3 data objects

DNP3 data objects include several data objects each with its headers and set of data points. Few of the data objects include 16-bit binary input, 32-bit analog input, analog input change event and many more. More information on the DNP3 data objects could be found on (5) and in section 1.3. The DNP3 data object header includes object group, variation, qualifier, start and stop bits. The data object values contain a set of data points associated with the data object. These point values hold the binary/analog measurement values and can be of 1-bit, 16-bit or 32-bit depending on the data type stored. Proper format of the data objects is necessary for successfully crafting a DNP3 payload.

3.4.4 DNP3 payload crafting using Python

For successfully crafting a DNP3 packet using Python, information on fields such as packet length, source, destination, function code, DNP3 data object headers, point values and correct CRC checksums are necessary. PyCRC Python package found on (26) could be used to calculate DNP3 CRC on Python. This package can be installed using pip package manager. The crafted packets can be injected from the adversary using sockets package which is an in-built Python package. More information on the sockets package could be found on (28).
CHAPTER 4. TESTBED-BASED IMPLEMENTATION OF DNP3 IDS

This chapter details the IDS solutions implemented on an emulated test bed network implementing an IEEE 39 bus model of smart grid. The IDS rules implemented covered the various functional categories of DNP3 protocol with a few additions of IT-based attacks. IT attacks include IT reconnaissance attempts such as ping and NMAP port scan, and file operations such as file upload and file download. The DNP3 functional categories cover control command functions, IIN response flags, and analog measurement responses.

This chapter describes the power system model implemented in the testbed and the network architecture deployed, the various IDS rule categories, the IDS testing and the evaluations made from alert visualization.

4.1 Testbed power system model and network architecture

The emulated test bed implements an IEEE 39 bus model from (29), having 1 control center and 3 substations, resembling a smart grid utility distribution grid environment. The end device of the substations was developed with eTerra EMS platform by GE to include many remote locations with different data point configurations such as voltage, current and frequency.
A distributed IDS architecture as shown in Figure 4.1 was deployed with Security Onion SIEM tool. The IDS architecture deployed an IDS Master and 3 IDS sensors covering the 3 substations. All the IDS sensors had a management interface and a sniffing interface. The management interface is used to send IDS alerts to the IDS Master using SSH. The sniffing interface sniffs on the network traffic and is in promiscuous mode.
4.2 Implementation of IDS rules

The implemented rule categories include IT reconnaissance rules, IT file operations such as upload and download, and DNP3 operational and measurement rules. The Table 4.1 showcases the implementation of the various rule categories described above.

**Rule Category C.1:** This category triggers when the sensor detects ICMP ping packets sent from an adversary (not white-listed). This action could be a potential reconnaissance attempt to detect the reachability of the DNP3 secondary device.

**Rule Category C.2** This category triggers when the sensor senses port scan activity targeting the DNP3 secondary device. Port scan is a reconnaissance attempt from an adversary, characterized by sending TCP SYN packets to sweep through all the ports in a device ranging from 0-65535. The ports that return an RST flag are classified as closed ports.

**Rule Category C.3** This category triggers when file upload operation is detected. A count of 20 TCP packets that sends more than 1400 bytes within 5 seconds are bound to cause a file transfer operation. This behavior has been observed across multiple file transfers. This category utilizes the time threshold snort feature.

**Rule Category C.4** This category flags on file upload. It is the same as 4.2 except for a change in the direction of data transfer, from the DNP3 secondary to external network.

**Rule Category C.5** This category triggers when the sensor encounters DNP3 control commands such as Operate, Warm restart, Cold restart, Direct operate and many more. These actions could be performed from a legitimate DNP3 primary or from an adversary.

**Rule Category C.6** This category of rules trigger when the DNP3 response packets from the DNP3 secondary contain internal indications flags (IIN). The presence of IIN bits indicate that there is an error condition at the DNP3 secondary level.

**Rule Category C.7** The analog measurement data violations could be detected by this category of rules. DNP3 data objects have several data points, each of which correspond to an electrical parameter such as voltage, current or frequency. This set of rules involve matching the response function code (\(81\)), then matching on DNP3 object header(\(|1e\ 05|\) for g30v05), and then finally...
comparing the data point value using byte_test. Few analog measurement violations include voltage violation, tieline power flow violation, generation output violation and system frequency violations. The testbed setup did not face the issue of having input change event modifications explained in the previous chapter under section 3.3.2. Therefore two rules are enough to detect the analog data violations for each data point.

4.3 IDS testing and evaluation

The IDS rule-sets categorized above were subject to tests through DNP3 payloads crafted on Kali-Linux using Python, and through reconnaissance attempts and file operations from Kali, which is placed outside the smart grid network, acting as an adversary.

4.3.1 IDS rule testing

Testing IT attacks: The IDS rules were tested with attacks from the command terminal on Kali, targeting the DNP3 secondary at the substation network. Attacks such as ping, NMAP port scan, file upload and download were carried out. In order to perform file upload and download, establishing persistence using a Metasploit shell is required. A reverse TCP exploit successfully establishes a shell on the target which is persisted. Files were then uploaded and downloaded using this shell. This activity is particularly dangerous as the adversary can exploit this to inject malware into the target and as well as extort confidential information from the target. These activities trigger the developed IDS rules from categories C.1 to C.4.

Testing DNP3 rules: The DNP3 rules were tested using specially crafted DNP3 payloads from Kali using Python. These scripts included control command functions, IIN flag responses and analog data violations. Control command DNP3 packets will have to target the DNP3 secondary while the IIN-violations and analog data violation DNP3 packets will have to target the control center. IIN and analog measurement violations are anomalies within the smart grid system, which are meant to detected at the substation network level. An adversary will gain no benefit if he tries to inject IIN and analog response packets. Therefore for the purpose of testing the rules, the IIN
and analog measurement packets are injected at the substation DNP3 secondary. The test cases included Python scripts for,

1. Control command functions such as Operate - Trip relay, Operate - Close relay, Direct Operate - Trip relay, Direct Operate - Close relay, Warm Restart, Cold Restart, Enable Unsolicited Class 1, 2, 3 and 123, Disable Unsolicited Class 1, 2, 3 and 123.

2. IIN-based violations such as event buffer overflow, device restart, device trouble, data point in local control, and config corrupt.

3. Analog data violations such as tieline power flow violations, system frequency violations, generation set point violations and voltage violations.

The testing process was divided into two categories - detection accuracy test and detection latency test. **IDS detection accuracy test** involved injecting all the crafted DNP3 payloads from Kali and evaluating if the IDS was able to correctly detect all the attacks. **IDS detection latency test** was also called the stress test. In this test, the DNP3 attack packets were injecting on a repeat basis - ranging from 10 to 1000 packets. The time taken by the IDS for generating the alerts was evaluated. The stress test was conducted for DNP3 unauthorized trip attack and DNP3 analog voltage violation attack.

### 4.3.2 IDS evaluation

The IDS system was subject to detection accuracy test and detection latency test. The IDS was evaluated based on the tests, and the following inferences could be made,

**IDS detection accuracy**: The IDS was tested for its detection accuracy. The IDS was able to successfully detect all the attacks, thereby having an accuracy of 100%.

**IDS detection latency**: Detection latency could be defined as the time taken for the IDS to generate alerts, once after the attacks were detected by the IDS sensors. The Figure 4.2 shows the IDS detection latency which is within reasonable limits (maximum of 3.78 ms), as the DNP3 polling time is 1 second.
4.3.3 Sguil alert detection

The IDS alerts that were generated across the various substation sensors were collected and visualized on the Sguil dashboard at the IDS Master, shown in Figure 4.3.
Table 4.1 Implementation of IDS rules on testbed

<table>
<thead>
<tr>
<th>Rule Category</th>
<th>Protocol Classification</th>
<th>Sample Snort Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.1</td>
<td>IT reconnaissance</td>
<td>alert icmp !$WHITELISTED any -&gt;$DNP3_SECONDARY any (msg: &quot;Ping detected on DNP3 secondary&quot;; sid:96500100;)</td>
</tr>
<tr>
<td>C.2</td>
<td>IT reconnaissance</td>
<td>alert tcp any any -&gt;$DNP3_SECONDARY !$TCP_PORTS (msg: &quot;NMAP port scan detected on DNP3 secondary&quot;; flags:S; threshold: type both, track by_src, count 500, seconds 30; sid:96500105;)</td>
</tr>
<tr>
<td>C.3</td>
<td>IT file operation</td>
<td>alert tcp any any -&gt;$DNP3_SECONDARY any (msg: &quot;File upload into DNP3 secondary&quot;; dsize:&gt;1400; threshold: type both, track by_src, count 20, seconds 5; sid:96500108;)</td>
</tr>
<tr>
<td>C.4</td>
<td>IT file operation</td>
<td>alert tcp $DNP3_SECONDARY any -&gt;any any (msg: &quot;File download from DNP3 secondary&quot;; dsize:&gt;1400; threshold: type both, track by_src, count 20, seconds 5; sid:96500109;)</td>
</tr>
<tr>
<td>C.5</td>
<td>DNP3 function code</td>
<td>alert tcp !$DNP3_MASTER any -&gt;$DNP3_SECONDARY $DNP3_PORTS (msg: &quot;Testbed DNP3 Unauthorised Operate FC&quot;; content: &quot;</td>
</tr>
<tr>
<td>C.6</td>
<td>DNP3 IIIN</td>
<td>alert tcp $DNP3_SECONDARY $DNP3_PORTS -&gt;$DNP3_MASTER any (msg: &quot;Testbed Event buffer overflow in DNP3 secondary&quot;; dnp3_ind: event_buffer_overflow; sid:96500113;) alert tcp $DNP3_SECONDARY $DNP3_PORTS -&gt;$DNP3_MASTER any (msg: &quot;Testbed device restart in DNP3 secondary&quot;; dnp3_ind: device_restart; sid:96500114;)</td>
</tr>
<tr>
<td>C.7</td>
<td>DNP3 measurement data</td>
<td>alert tcp $DNP3_SECONDARY $DNP3_PORTS -&gt;$DNP3_MASTER any (msg: &quot;Voltage violation &gt;1.04pu @ Richview Substation&quot;; content: &quot;</td>
</tr>
</tbody>
</table>
CHAPTER 5. FIELD DEPLOYMENT AND TESTING

The IDS solutions described in section 3.3 were deployed at a local utility as a part of the commitment to DOE-CEDS research project. The designed IDS rules were field tested for its accuracy, effectiveness and sensitivity. The IDS solutions deployed were signature-based snort rules, targeting the different functional categories of DNP3 SCADA communication within an utility distribution smart grid network. The functional categories of DNP3 communication protocol included control command functions, error indication responses and measurement data responses.

This section describes the IDS architecture that was utilized to make the rule deployment, the different sequential phases in which the technology transfer activity was carried out, the different rule categories of the IDS rules, and finally the testing made and evaluations performed.

5.1 Field deployment IDS architecture

A distributed, network-based IDS architecture shown in 5.1 was developed and deployed in the utility network, following up the work from (17). The IDS Master was deployed within the control center. There were 5 sensor nodes that were deployed at various locations where substations are present to detect alerts on the substation network traffic. Of the 5 sensor nodes, one was dedicated for testing purposes on a lab-network that had a DNP3 device installed to simulate the actual network traffic.

The DNP3 secondary devices collect and store the data on electrical parameters from field devices at the lower level. The data is collected periodically depending on the poll rate. The DNP3 Master relays poll the DNP3 secondary and send control command signals to them. The DNP3 secondary have specific data points configured for storing specific electrical parameters, say frequency, line voltage or line current. Utilities have their DNP3 secondary configured with plenty
of data points. DNP3 secondary in the test-lab network will only contain a sample of the data points that the relays in the production network of the utility have.

Figure 5.1  IDS Architecture — Local Utility Field Deployment

5.2  Deployment strategy overview

This section details the different phases under which the deployment of IDS solutions were carried out. The entire field deployment could be broken down into three phases. These phases are explained below:

5.2.1  Phase 0

Phase 0 is also called the installation phase. In this phase, the utility network architecture was reviewed, with further analysis performed on the nature of DNP3 traffic, the bandwidth required and virtual machine configurations that will need to be setup. All the prerequisites for a successful functioning of the IDS was checked. These prerequisites are mentioned in Section 2.3.1.
There were 5 substations and 1 control center within the utility network. Therefore Security Onion SIEM tool had to be installed in all the 6 machines. Phase 0 involved installing of Security Onion SIEM tool and then configuring the VMs for IDS master and IDS sensor deployments. The sniffing networks had promiscuous mode setup which enabled sniffing the network traffic. A management interface was setup on the sensor machines which enabled them to send alerts to the Master. The next phases detail the deployment of IDS rules and the tests carried out.

5.2.2 Phase 1

Phase 1, also called the coverage test phase, targeted the test-lab network sensor. This phase involved creating rules for the control command, IIN rules, and time threshold rules for the test-lab sensor. DNP3 payloads were crafted using Python scripts to test every rule deployed. These payloads were then injected from a kali-Linux machine in the test-lab network, targeting the test-lab DNP3 relay. These attacks were detected by the sensor and the alerts generated were visualized on the Sguil dashboard in IDS Master. An overview of Phase 1 is shown in 5.2.

![Phase 1 - overview](image)

5.2.3 Phase 2

Phase 2 deployments, also called the measurement data test, targeted the analog measurement data points within the utility. This phase involved creating measurement data rules for the various
data points in the lab-network and the substation network. Phase 2, shown in 5.3 was carried out in two sub-phases, phase 2.1 targeted the analog rules in the test-lab sensor and phase 2.2 targeted the analog test rules at the substations. In this phase, the rules for the substation networks were not tested through DNP3 script injections from a kali machine as it was a live network serving thousands of households. Only the test-lab network was subject to DNP3 payload scripts with data points exceeding the analog boundary conditions, causing the corresponding alerts to be triggered.

A key observation that was made between test-lab and substation deployments were in the measurement data rules. As explained in Section 3.3.2, the utility substation is subject to frequent changes and hence has the input change event DNP3 data object present in its DNP3 response packets to the control center.
5.3 Implemented IDS Rules

This section details the different categories of IDS rules deployed at the utility. The rules could be classified into control command rules, internal indications (IIN) flag rules, time threshold rules and measurement data rules. These categories are tabulated in Table 5.1.

Table 5.1 Utility IDS Rule Categories

<table>
<thead>
<tr>
<th>Rule Category</th>
<th>Sample Snort Rule</th>
</tr>
</thead>
</table>
| C.1 | alert tcp !$DNP3_MASTER any ->$DNP3_SECONDARY $DNP3_PORTS (msg: "Test-Lab DNP3 Unauthorised Operate FC Trip Operation"; content: "|05 64|"; depth: 2; content: "|04|"; distance: 10; content: "|81|"; distance: 7; sid:96500179;)

| | alert tcp !$DNP3_MASTER any ->$DNP3_SECONDARY $DNP3_PORTS (msg: "Test-Lab DNP3 Unauthorised Disable Unsolicited Class 1"; dnp3_func: disable_unsolicited; dnp3_obj:60,2; sid:96500154;)

| C.2 | alert tcp $DNP3_SECONDARY $DNP3_PORTS ->$DNP3_MASTER any (msg: "Test-Lab Event buffer overflow in end device"; dnp3_ind: event_buffer_overflow; sid: 96500163;)

| C.3 | alert tcp $DNP3_SECONDARY $DNP3_PORTS ->$DNP3_MASTER any (msg: "Test-Lab Repeat Device Restarts"; dnp3_ind: device_restart; threshold: type both, track by src, count 3, seconds 5; sid: 96500169;)

| C.4 | alert tcp $DNP3_SLAVE $DNP3_PORTS ->$DNP3_MASTER any (msg:"VAB<67kV at substation–2"; content: "|05 64|"; content: "|81|"; distance: 10; within: 1; content: "|20 02|"; distance: 2; within: 2; pcre: "/[S\s]{1}?/([x02]|x06|\x09|\x0c|\x0f|\x12)/iAR"; content: "|1e 02 01 00|"; distance: 72; within: 183; byte_test: 2,<,0x1a2c,12,relative,little; sid: 96500321;)

**Rule Category C.1:** Control Command Rules target the DNP3 control command functions such as Read, Write, Select, Operate, Cold Restart, Warm Restart and many more. These rules
include function codes that perform critical operation and the execution of them should be visible to the system operator. Their functioning is similar to the control command rules explained in previous sections.

**Rule Category C.2** IIN-based Rules target the IIN-bits (IIN - Internal Indications) setup in the DNP3 response packets.

**Rule Category C.3** Time Threshold Rules trigger based on packet counts within the targeted time frame. These rules intend to capture conditions such as too many event buffer overflows within 15 minutes, indicating that such behaviour of the data point is abnormal. These rules utilize the threshold snort function which keeps track of each packet by its source IP.

**Rule Category C.4** Measurement data rules trigger based on boundary conditions of the analog threshold values. These boundary conditions were set after consultations with the utility power system engineer. For instance, the line voltage AB had a boundary condition of 67kV as the lower limit and 72kV as the upper limit. These rules were established for several other data points such as line voltages BC and CA, phase voltage A, B and C, frequency, active power, and reactive power.

**Active power flow reversal**: Another set of rules for active power flow reversal had to be designed in a slightly modified way. Since the power flow is bidirectional, the change in directions were represented by change in signs. A negative active power flow indicates a fault condition in the power system. This fault condition is triggered as alerts to give visibility to the operator. The datatype of active power data point is 32-bit signed integer. The range of positive integers in a 32-bit signed integer is between 0-32767 and the range of negative integers is between 32768 to 65535. So `byte_test` is compared against values greater than 32767 to detect negative active power flow.

This category of rules were designed for response packets with and without input change event data objects.

**Without input change event data object**: As discussed in Section 3.3.2, the analog data violation rules for response packets without input change event data object will have two rules
defining an analog data point, one for comparing lower boundary condition and another for comparing upper boundary condition. These rules can accurately detect the violations in analog data values.

**With input change event data object**: The analog data violation rules in this category will need to have more than 2 rules for a data point as the analog data signature changes for different input change event counts in the response packet. In order to avoid missing out on any anomalous analog data point and to avoid reporting false positives, analog rules for all possible number of input change events will have to be developed. This results in at least 5-10 IDS rules to be designed for a single analog data point. The table 5.2 below showcases the analysis performed to determine the analog data signature and the values to be padded to distance and within keywords.

**Table 5.2 Sample analog packet analysis**

<table>
<thead>
<tr>
<th>#Change events</th>
<th>DO header position</th>
<th>#CRC until DO</th>
<th>Vab position</th>
<th>Distance</th>
<th>Analog Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>72</td>
<td>4</td>
<td>96</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>77</td>
<td>4</td>
<td>101</td>
<td>13</td>
<td>1e 02 01</td>
</tr>
<tr>
<td>3</td>
<td>84</td>
<td>5</td>
<td>108</td>
<td>12</td>
<td>1e 02 01 00</td>
</tr>
<tr>
<td>4</td>
<td>89</td>
<td>5</td>
<td>113</td>
<td>12</td>
<td>1e 02 01 00</td>
</tr>
<tr>
<td>5</td>
<td>94</td>
<td>5</td>
<td>118</td>
<td>12</td>
<td>1e 02 01 00</td>
</tr>
<tr>
<td>6</td>
<td>101</td>
<td>6</td>
<td>123</td>
<td></td>
<td>1e 02 01 00</td>
</tr>
<tr>
<td>7</td>
<td>106</td>
<td>6</td>
<td>130</td>
<td>12</td>
<td>1e 02 01 00</td>
</tr>
<tr>
<td>8</td>
<td>111</td>
<td>6</td>
<td>135</td>
<td>12</td>
<td>1e 02 01 00</td>
</tr>
<tr>
<td>9</td>
<td>118</td>
<td>7</td>
<td>140</td>
<td>10</td>
<td>1e 02 01 00</td>
</tr>
<tr>
<td>10</td>
<td>123</td>
<td>7</td>
<td>147</td>
<td>12</td>
<td>1e 02 01 00</td>
</tr>
<tr>
<td>11</td>
<td>128</td>
<td>7</td>
<td>152</td>
<td>12</td>
<td>1e 02 01 00</td>
</tr>
<tr>
<td>12</td>
<td>133</td>
<td>7</td>
<td>157</td>
<td>9</td>
<td>1e</td>
</tr>
<tr>
<td>13</td>
<td>140</td>
<td>8</td>
<td>164</td>
<td>12</td>
<td>1e 02 01 00</td>
</tr>
<tr>
<td>14</td>
<td>145</td>
<td>8</td>
<td>169</td>
<td>12</td>
<td>1e 02 01 00</td>
</tr>
<tr>
<td>15</td>
<td>150</td>
<td>8</td>
<td>174</td>
<td>14</td>
<td>1e 02</td>
</tr>
<tr>
<td>16</td>
<td>157</td>
<td>9</td>
<td>181</td>
<td>12</td>
<td>1e 02 01 00</td>
</tr>
<tr>
<td>17</td>
<td>162</td>
<td>9</td>
<td>186</td>
<td>12</td>
<td>1e 02 01 00</td>
</tr>
<tr>
<td>18</td>
<td>167</td>
<td>6</td>
<td>191</td>
<td>13</td>
<td>1e 02 01</td>
</tr>
<tr>
<td>19</td>
<td>174</td>
<td>7</td>
<td>198</td>
<td>10</td>
<td>1e 02 01 00</td>
</tr>
<tr>
<td>20</td>
<td>179</td>
<td>7</td>
<td>203</td>
<td>12</td>
<td>1e 02 01 00</td>
</tr>
</tbody>
</table>

Based on the Table 5.2, the analog rules for the data point VAB could be designed. The Table 5.3 illustrates how the rules for different input change event counts are grouped together, and how the varying signatures are captured as separate rules. The table only contains a sample set of rules for VAB <67kV. The first row of the table is an example of grouping rules based on similar analog data signature. The IDS rule for input change event 2 shows how differing analog data signatures in a payload have to be captured with a new rule. The last row is an example of creating a separate rule because of the presence of CRC checksum.

<table>
<thead>
<tr>
<th>#Change events</th>
<th>Analog data rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not 2, 6, 9, 10, 12, 15 and 18</td>
<td>alert tcp $DNP3_SECONDARY $DNP3_PORTS -&gt;$DNP3_MASTER any (msg:&quot;VAB&lt;67kV&quot;; content: &quot;</td>
</tr>
<tr>
<td>2</td>
<td>alert tcp $DNP3_SECONDARY $DNP3_PORTS -&gt;$DNP3_MASTER any (msg:&quot;VAB&lt;67kV&quot;; content: &quot;</td>
</tr>
<tr>
<td>9</td>
<td>alert tcp $DNP3_SECONDARY $DNP3_PORTS -&gt;$DNP3_MASTER any (msg:&quot;VAB&lt;67kV&quot;; content: &quot;</td>
</tr>
</tbody>
</table>

### 5.4 IDS Testing and Evaluation

The IDS rules that were deployed in the utility in three phases were tested for its accuracy. The testing was carried out separately for the test-lab sensor and the utility production network.

**IDS testing on test-lab network:** For the test-lab network, DNP3 attack scripts were created to carry out the testing. DNP3 attack scripts were developed on Python for the control command, IIN-based, time threshold and measurement data categories. The control command, IIN-based and
time threshold testing was similar to the testbed IDS testing explained in the previous chapter. The attack scripts were targeted towards the DNP3 test-relay which was configured with very few analog data points. The DNP3 responses from test-relay did not contain any input change event data object, and hence the DNP3 analog data packets for testing were crafted accordingly. All the attack scripts were injected and the alerts were observed on Sguil dashboard in the IDS Master.

**IDS testing on utility substation network:** The testing on utility substation network cannot be carried out in similar fashion to that of the test-lab network, as the injection of scripts can break the system. So, in order to perform the testing, the IDS rules were designed in a way to trigger on legitimate analog data traffic by adjusting the boundary conditions of the data point values in the rules. The DNP3 traffic in the network was observed carefully and the alerts generated were correlated with the packets observed to check for false positives and detection accuracy.

**IDS Evaluation:** The IDS was evaluated for accuracy. Accuracy is defined as the ratio of the sum of true positives and true negatives to the sum of all the packets. No false positives were found for the rules. This is particularly impressive for the analog measurement rules as any anomaly detection system utilizing machine learning algorithms are bound to have plenty of false positives. The accuracy of the IDS was evaluated to be at 100%.

**Sguil alert visualization:** The Sguil dashboard in the IDS Master shows all the alerts that the IDS system generates, and is shown in Figure 5.4. The IP and port information are masked in the Figure.
Figure 5.4 Utility Sguil — IDS Alerts

The substation networks did not trigger any measurement data rule as there were no anomalous analog data found at the time of capture. Overall an IDS for DNP3 that covered all the functional aspects of DNP3 and as well the analog measurement data that flows through it, was deployed.

Acknowledgement: This research is funded in part by US DOE Grant # DE- EE0008773, and US DOE Grant # DE-OE0000830.
CHAPTER 6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

The research work has demonstrated how the designed IDS system can monitor the grid operations carried out using DNP3 protocol. The efforts have also been validated by deployments both on the testbed and within a local utility, where it has shown to work flawlessly using a distributed deployment architecture. The IDS rules were designed to detect intrusions as well as anomalies in the grid operations. The IDS rules could detect grid operations such as DNP3 control commands, DNP3 error response messages indicating an error condition within the system, and DNP3 measurement data responses from the DNP3 substation devices in response to poll requests from the control center. The analog measurement violations detected covered line and phase voltage violations, frequency violations, active power reversal violations and line current violations. DNP3 attack packets were crafted using Python scripts for all the rule categories. The IDS rules were tested against the attack scripts and evaluations were made, based on the detection and the alert generated on the Sguil dashboard. None of the attack packets were misclassified, leaving no false positives and having 100% detection accuracy. The IDS evaluation done on the testbed setup has shown to have IDS detection latency as low as 4 microseconds approximately, which is well within the acceptable limits.

6.2 Future work

Future additions to this work would be focused on automating the generation of rules for changes in the data point configuration within the substation DNP3 secondary devices. Current IDS design methods would require a complete redesign of the snort rules to compute the values to be padded to the distance and within keywords. Snort engine framework had its own limitations, especially when computing the position of the signatures for analog data. The analysis part to arrive at the
correct position and identifying the specific signature to match, required plenty of manual effort and time, which will have to be repeated in case the DNP3 slave data point configurations change. This could be effectively minimized by developing an application that can read through the packet data or use a parser to extract DNP3 packet information, and then use the information to trigger alerts based on violations. This approach would make the rule design dynamic and would make for a holistic functioning of the IDS. Further efforts will have to be directed towards developing standard industry-grade test cases for DNP3 to evaluate the developed IDS. Anomaly detection techniques can be developed to detect anomalies in the analog measurement data to detect the abrupt dynamic changes in analog data within the defined boundary conditions.
BIBLIOGRAPHY


