electronics

Article

Anomaly Detection for SCADA System Security Based on Unsupervised Learning and Function Codes Analysis in the DNP3 Protocol

Mustafa Altaha and Sugwon Hong *

Department of Computer Engineering, Myongji University, Yongin 17058, Korea; mustafaraed@mju.ac.kr
* Correspondence: swhong@mju.ac.kr

Abstract: An Intrusion Detection System (IDS) is a tool used primarily for security monitoring, which is one of the security strategies for Supervisory Control and Data Acquisition (SCADA) systems. Distributed Network Protocol version 3 (DNP3) is the predominant SCADA protocol in the energy sector. In this paper, we have developed an effective and flexible IDS for DNP3 networks, observing that most critical operations in DNP3 systems are utilized based on the function codes in DNP3 application messages, and that exploitation of those function codes enables attackers to manipulate the system operation. Our proposed anomaly-detection method deals with possible attacks that can bypass any rule-based deep packet inspection once attackers take over servers in the system. First, we generated datasets that reflected DNP3 traffic characteristics observed in real-world power grid substations for a reasonably long time. Next, we extracted input features that consisted of the occurrences of function codes per TCP connection, along with TCP characteristics. We then used an unsupervised deep learning model (Autoencoder) to learn the normal behavior of DNP3 traffic based on function code patterns. We called our approach FC-AE-IDS (Function Code Autoencoder IDS). The evaluation of the proposed method was carried out on three different datasets, to prove its accuracy and effectiveness. To evaluate the effectiveness of our proposed method, we performed various experiments that resulted in more than 95% detection accuracy for all considered attack scenarios that are mentioned in this study. We compared our approach to an IDS that is based on traditional features, to show the effectiveness of our approach.

Keywords: Intrusion Detection System (IDS); anomaly-detection; Distributed Network Protocol (DNP3); function code; Supervisory Control and Data Acquisition (SCADA); autoencoder; cybersecurity; machine learning

Citation: Altaha, M.; Hong, S. Anomaly Detection for SCADA System Security Based on Unsupervised Learning and Function Codes Analysis in the DNP3 Protocol. *Electronics* **2022**, *11*, 2184. https://doi.org/10.3390/electronics11142184

Academic Editor: Harald Vranken

Received: 18 May 2022
Accepted: 7 July 2022
Published: 12 July 2022

1. Introduction

A Supervisory Control and Data Acquisition (SCADA) system—an archetypal form of industrial control systems (ICS)—is a process control system that interconnects, monitors, and controls remotely dispersed physical processes. Nations' critical infrastructures, such as power grids, water treatment, gas pipelines, and the like—where safe and reliable operations are of primary concern—rely on the SCADA system. Any cyberattacks on SCADA systems that could cause severe disruption of their reliable operations would wreak devastating economic and societal or, at worst, national security damage. For this reason, cyber security defense mechanisms on SCADA—broadly ICS systems—have been extensively studied and proposed in academic and industrial communities, as well as international standard organizations [1].

Security monitoring in the form of Intrusion Detection Systems (IDS) is one of the viable strategies for SCADA cyber security [2,3]. In particular, predictable patterns of traffic flow and characteristics between fixed nodes in the SCADA system make the IDS solution more attractive compared to typical IT systems. The nuts and bolts of IDS solutions are to discern the normal behavior of target traffic, which is often called "profile", based on which

abnormal behavior can be identified. Hong, et al. [4], surveyed various IDS approaches for the SCADA/ICS system, and classified them into four categories. As [4] noted, a large part of the proposed IDS solution sought to find "rules" that traffic flows in the system manifest, based on the network configuration or the communication protocols deployed and adopted in the target system. A few approaches have endeavored to find rules, going deep down into field device behaviors [5]. Recently, with its success and popularity, deep learning (DL) has drawn much attention from researchers as a promising tool to find the profile of target system traffic behavior [6–10]. The benefit of DL approaches is that they can capture normal behavior by learning target traffic behavior, consequently enhancing the effectiveness of detection performance, and possibly responding to new variants of attacks.

There were two challenges when we adopted DL techniques to develop IDS for the SCADA system. The first challenge was the limitation of training datasets. DL models should be based on learning from a large scale of SCADA network traffic, so as to be sufficiently reliable and capable of reflecting real operations in the system. Gomez, et al. [11], (p. 177462, Table 1) give the summary and comparison of the datasets that have been used in anomaly-detection studies for ICS, as well as traditional networks. ICS-related datasets cover the domains of water treatment, water distribution, gas pipeline, and power grid [12]. These datasets derive from testbeds or simulation, which makes them difficult to regard as reflecting real and reliable traffic characteristics of actual SCADA/ICS systems. In some cases, traffic data generated from real power substations are used [13]. However, the datasets of real systems are rarely open to the public, due to the still-immature status of DL research in this field, compared to other mature areas such as image or speech recognition. The limitation of datasets leads to the difficulty of conducting a proper and comparable evaluation of DL models in the same environment, not to mention the generalization issue, i.e., how selected features and models can be applied to similar environments. The use of the balance dataset was another issue. Depending on an unbalanced dataset, which contains very few abnormal instances, is likely to cause overfitting, consequently degrading the model's performance drastically.

The second challenge was how to simulate attacks. Lee and Hong [14,15] explain the attack vectors and attack trees of recent sophisticated cyberattacks that have actually targeted SCADA/ICS systems. These attacks exploit the vulnerability of software systems, and take over the control of hosts, whether they are servers or device controllers. In this way, the attackers—using the compromised hosts—issue malicious commands to field devices, causing malfunction of the system. In such an attack scenario, it is almost useless to confirm the validity of transmitted messages by checking whether the messages observe the rules or syntax defined by the communication protocols. Currently, standard organizations have proposed SCADA security protocols, which guarantee the authenticity and integrity of communication messages exchanged between hosts in the system [16,17]. However, these schemes have limitations, because attackers can generate malicious messages, still avoiding validity checks. The anomaly- or intrusion-detection system in SCADA should confront new variant attacks which are novel and unknown, in the sense that they can avoid any deep packet inspection based on rules derived from protocols, configurations, or any others deployed in the system.

In this paper, we focused on the SCADA substation based on Distributed Network Protocol version 3 (DNP3). DNP3 is one of the most widely used communication protocols in the power grid SCADA network. The DNP3 application layer specifies the message exchange between a control center and field devices. The function codes in the application message control the operation of the field devices. For this reason, the DNP3 application layer is the part most vulnerable to any possible attacks, and the manipulation of the function codes could constitute the most plausible and fatal attacks in DNP3 systems. Previously, some researchers have focused on function code vulnerability in DNP3 [18]. However, they did not deal with the issues relating to function codes comprehensively, and their solution was confined to syntactical analysis of packet contents, such as CRC and checksum. In this study, we focused on possible attacks by interlopers able to manipulate

the function codes of the message. We generated DNP3 application traffic, and formulated various datasets that reflected the traffic characteristics of DNP3 application messages, which were observed in real-life substations for a long time period [19–21]. Based on the datasets generated in this way, we used the occurrences of each function code per TCP connection as input features.

We considered the anomaly-detection method to deal with possible attack scenarios where attackers could manipulate DNP3 application messages at will. For this purpose, we applied the unsupervised deep learning model to detect anomalies. The autoencoder network is one of the most commonly used as an unsupervised DL model for intrusion-detection [22–24]. Recently the Generative Adversarial Network (GAN) was also widely applied for this purpose [25,26]. In this paper, we adopted an autoencoder network that had a simple internal architecture, since it could reconstruct input features sufficiently to meet the intended purpose of this study [24]. To summarize, our contributions in this paper are as follows:

- A framework to generate datasets that represent real-world traffic in power grid substations, based on [19], by using modified OpenDNP3;
- A method to generate attack datasets, such that the adversarial (injection, dropping or modification) traffic is embedded within the background traffic by using a combination of malicious attacks when a host is hijacked;
- We employed an extractor by which DNP3 instances were processed, to pull out data features that represented the behavior of the DNP3 network, discovering important feature representations;
- We employed an AE-IDS model to discover important feature representations from the training data, and generated a model to detect normal and abnormal behaviors. Cooperation between the proposed model and IDS deep learning models was evaluated by our own dataset, which we created using the OpenDNP3 library, which could reflect the real network traffic;
- Finally, analysis of accuracy, F1, precision, and recall values for the detection of intrusion into our model FC-AE-IDS (Function Code Autoencoder IDS), compared to an IDS model based on the traditional input features, to show the effectiveness of our model.

The rest of this paper is organized as follows: Section 2 describes the background of the DNP3 protocol; in Section 3, we explain the traffic characteristics of the dataset that we generated in this study, the input features, and the autoencoder model; in Section 4, we present how to generate datasets, and an analysis and performance evaluation of the proposed method; finally, we discuss our conclusion and future work in Section 5.

2. Background to DNP3

2.1. DNP3 Protocol

DNP3 defines the rules according to which SCADA devices (outstations) and control stations (master) communicate data and control commands [16]. Using the familiar power grid parlance, the master is typically a central control center, and the outstation is a remote terminal unit (RTU) or intelligent electronic device (IED).

DNP3 is a layered protocol, as shown in Figure 1. DNP3 application messages can be delivered over serial links or the TCP/IP protocol. DNP3 consists of its own three layers (functions). DNP transport (often called pseudo-transport) and data link layers involve mostly fragmentation of application messages, along with some other functions such as flow control or error control. They are designed for the purpose of communication over the serial bus, which was the main communication environment in the old days. Currently, most SCADA systems run over the TCP/IP network environment. When DNP3 operates over TCP/IP, the master station acts as a TCP client connected to multiple outstations that play the role of TCP servers.

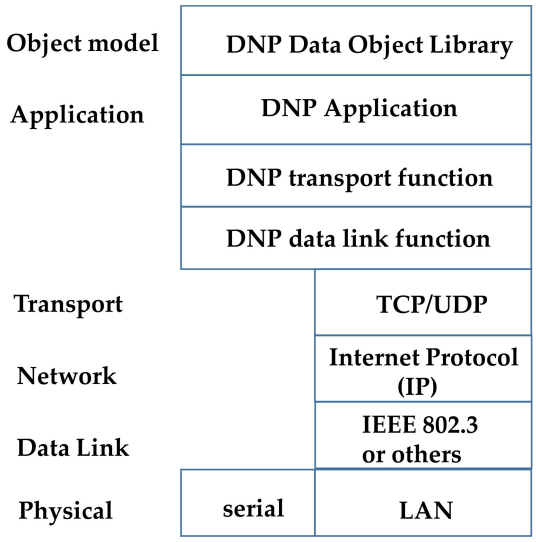

Figure 1. DNP3 protocol stack.

DNP3 application messages are classified into two categories: request and response messages. The master requests data—such as binary inputs, analog inputs, count inputs, and configuration data—by sending read request messages to outstations. The master also issues control commands that close or trip a circuit breaker, or send some analog output values to outstations. Not only do the outstations respond to the master's request, but they can also send messages autonomously when certain events happen, such as data point changes (unsolicited response).

The function codes in the DNP3 application message, which is shown in Figure 2, specify which operation should be performed on the outstations. The types of operation are shown in Table 1. Using function codes, a master can perform all necessary operations in order to monitor and control the outstations.

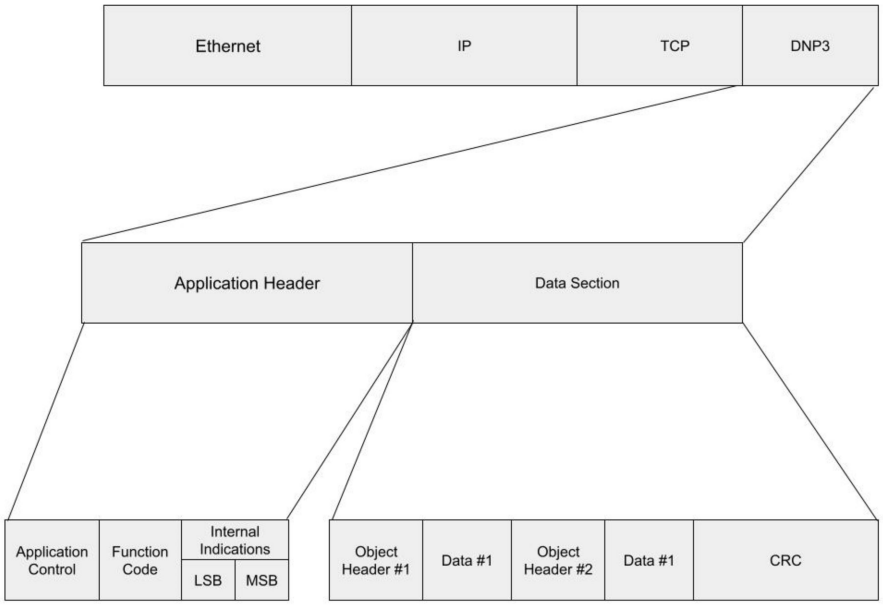

Figure 2. DNP3 application message.

Table 1. DNP3 function codes used in the study.

Code	Function	Description
0	Confirm	Confirms the receipt of an application layer fragment.
1	Read	Requests the specified object from an outstation.
2	Write	Stores specified object in an outstation.
3	Select	Selects output points but does not produce any action.
4	Operate	Produces output actions based on the selected points.
5	Direct Operate	Selects and operates the specified output.
7, 8,9,10,11,12	Freeze and Clear	Specified objects to freeze.
13	Cold Restart	Performs the desired reset sequence.
15,16	Initialize	Initializes the specified data to power up the initial values.
17	Stop Application	Stops the specified application.
20,21	Enable/Disable Unsolicited	Enables/Disables spontaneous reporting of the specified data object.
23	Time	The value to adjust time in an outstation.
129	Response	Response to a request message.

2.2. DNP3 Attacks

DNP3 attacks can happen in all DNP layers [27–29]. An extensive summary of all possible attacks in each layer is shown in [29]. This paper classifies most attacks into three categories with 21 attack instances. Those attacks range from passive reconnaissance to execution of malicious operations on field devices while masking their actions. Among all possible threats, the deadliest form of attack is the one that corrupts internal devices, causing abnormal status and consequent malfunction. In most recent ICS attack instances, this kind of attack took place when intruders hijacked internal nodes by utilizing the software vulnerability of the system. In those cases, the attackers were able to possess the system privilege to modify or fabricate DNP3 messages, eventually disrupting the system operation at will. In terms of manipulating DNP3 application messages, the DNP3 application layer is the most vulnerable part. In particular, as most critical operations in DNP3 systems can be performed based on function codes, exploitation of the function codes opens the gate to manipulation of the system operation.

Table 2 shows a summary of the most feasible attacks utilizing function codes. Paper [19] explains possible scenarios of attacks that utilize the function codes. The 'Write' function code, which is used for storing data in the outstation, can be misused to overflow or corrupt an outstation's memory. 'Freeze' and 'Clear' function codes are used for freezing (holding) or clearing (removing) values in the buffer (or register) of an outstation. 'Cold Restart' and 'Warm Restart' codes perform complete or partial reset of all hardware and software in the outstation, consequently shutting down and restarting outstations. The 'Initialize' function code can misplace the application state as specified by the data in the request message; it can therefore be misused for reinitializing an outstation with arbitrary states. The 'Stop' code halts the running of the application specified by the data in the request, thus making the outstation unresponsive to commands from the master. The 'Unsolicited Response' code can be used to overflow an outstation's buffer, possibly enabling DoS attacks.

Table 2. DNP3 scenarios of attacks utilizing function codes.

Function	Attack Scenario
Write	An action could make an outstation's memory corrupt or overflow.
Freeze/Clear	An action could lead an outstation to malfunction and crash.
Cold Restart	An action could shut down and restart an outstation, and also cause a Denial-of-Service (DoS) condition.
Initialize	An action could cause an outstation to reinitialize itself and result in arbitrary states.
Stop	An action could make an outstation unresponsive to any master commands.
Unsolicited	This makes an outstation overflow its buffer, enabling DoS.

3. The Proposed Anomaly-Detection Method (FC-AE-IDS)

3.1. Traffic Characterization

The goal of our study was to discover the normal pattern or behavior of DNP application layer traffic in the real SCADA system, and to detect any possible attacks based on this observation. Thus, the first step of our approach was to generate DNP3 traffic that could characterize normal patterns of DNP3 message exchange in the real-world DNP3 system operation environment.

Irvene, et al. [19], (p. 53, Figure 7) showed traffic characterization of DNP3 application layer data captured from live operating power substation networks for a reasonably long time. The datasets shown in the paper were taken from four medium-voltage distribution substations non-consecutively over the course of two-and-a-half years. Formby, et al. [20], also investigated TCP characteristics based on the real traffic captured from real power grid networks. Their observations mainly focused on the TCP behaviors of the DNP3 devices that could be distinguished from those in the traditional network.

Initially, we generate DNP3 traffic, and obtained a normal dataset that preserved the function codes frequency similar to what is shown in [19]. In addition to normal traffic generation, we also generated DNP3 attack messages manipulating the DNP3 function codes. The attacks were divided into six types: injection, dropping, modification, cold code, initialized code, and stop code. The first three attack types were to inject new malign messages, remove legitimate messages, and alter legitimate messages. The last three attack types specially injected illegitimate messages with Cold Restart, Initialize Application, and Stop Application function codes, with the intended actions explained in Table 2. Details of the traffic generation are explained in the next section.

3.2. Input Features and Autoencoder Network

Feature extraction consists of feature construction and feature selection. The window-based feature extraction technique is one way of selecting attribute vectors that reflect the time-series nature of packet streams [30]. The window is a certain timeframe within which constructed features are analyzed. The pattern or behavior within one single window corresponds to one data instance in deep learning models. A difficulty of applying the window-based feature extraction is determining the optimal size of the window.

In this paper, we took the approach of selecting features based on TCP connections, as shown in Figure 3. We set up TCP connections, and observed the function codes frequency on each connection. The idea behind this approach was that it could capture the behavior of DNP3 application layer messages, along with TCP behavior [20]. A single connection corresponded to one instance, which lasted one minute. The whole cycle per day consisted of 1440 instances, which contained more than 100,000 DNP3 application messages. The DNP3 input features extracted per instance are shown in Table 3. In addition, two TCP/IP features, which measured the transferred bytes from destination to source, and the opposite, were also included in the input features. Thus, if there was any increase or decrease in function codes, it would trigger a change in these features. The other 12 DNP3 features have records of how many DNP3 function codes were sent per instance.

We adopted an autoencoder network as an unsupervised learning model, which consisted of an input layer, hidden encoder/decoder layers, and an output layer. The autoencoder network architecture in the study, as shown in Figure 4, had an input layer of 14 nodes (features), three hidden layers, and an output layer consisting also of 14 nodes that reconstructed the input features. The Rectified Linear Unit (ReLU) was used in the hidden layer, and the sigmoid function was used at the output layer.

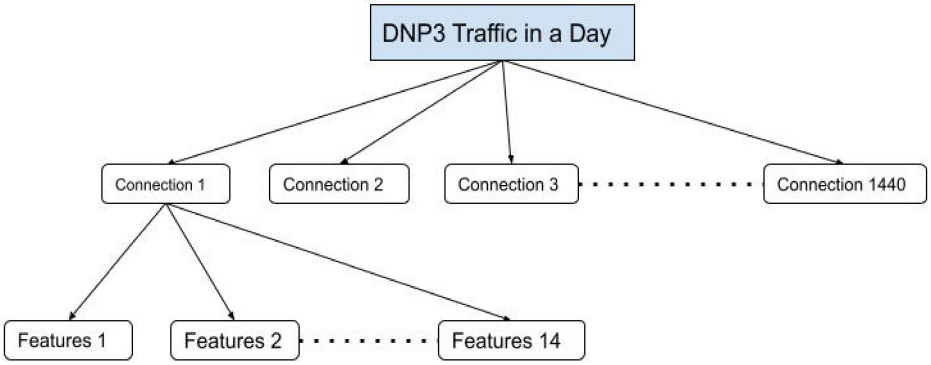

Figure 3. Two-level feature extraction.

Table 3. List of DNP3 input features.

Feature	Function
DNP3 payload length	The total length of the DNP3 payload contained within the connection.
Read Counter	Count Read codes
Write Counter	Count Write codes
Select Counter	Count Select codes
Operate Counter	Count Operate codes
Freeze Counter	Count Freeze codes
Cold Restart Counter	Count Cold Restart codes
En/Dis code Counter	Count Enable/Disable unsolicited
Initialize Counter	Count Initialize Application
Stop Application Counter	Count Stop Application
Record Time Counter	Count Record Current Time
Response Counter	Count Response

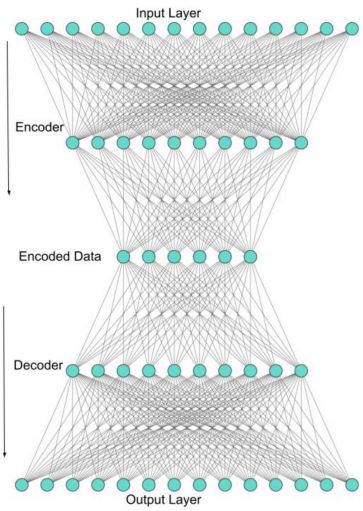

Figure 4. The autoencoder network structure.

4. Experiment

4.1. Dataset Generation

Dataset generation for the experiment utilized two Linux host systems; one operated as a master, and the other operated as an outstation, both of them running OpenDNP3 [31]. We modified the OpenDNP3 library for data visualizations, in order to better observe trends in the application layer function codes and packet sizes. The analysis was performed by Wireshark [32]. The captured PCAP file from the DNP3 network was converted into CSV by a program written in Python using the PyShark library, where we were able to

choose specific fields from the DNP3 traffic. The screenshot of Figure 5 shows an example of TCP and DNP3 packets in a connection by Wireshark. We also generated attack packets, based on the assumption that an attacker had already gained access to the DNP3 master to execute all possible attack scenarios. Four datasets of network traffic were generated over the course of four days: normal dataset and three datasets mixed with attack packets. Figure 6 shows the Pearson correlation between the 14 input features.

```
> Frame 119: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface -, id 0
> Ethernet II, Src: PcsCompu_73:d4:bd (08:00:27:73:d4:bd), Dst: PcsCompu_8d:35:08 (08:00:27:8d:35:08)
> Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.2
> Transmission Control Protocol, Src Port: 49723, Dst Port: 20000, Seq: 126, Ack: 148, Len: 24
v Distributed Network Protocol 3.0
  > Data Link Layer, Len: 17, From: 1, To: 10, DIR, PRM, Unconfirmed User Data
  > Transport Control: 0xe4, Final, First(FIR, FIN, Sequence 36)
  > Data Chunks
  > [1 DNP 3.0 AL Fragment (11 bytes): #119(11)]
  v Application Layer: (FIR, FIN, Sequence 5, Cold Restart)
    > Application Control: 0xc5, First, Final(FIR, FIN, Sequence 5)
      Function Code: Cold Restart (0x0d)
```

Figure 5. An example of DNP3/TCP packets captured using Wireshark.

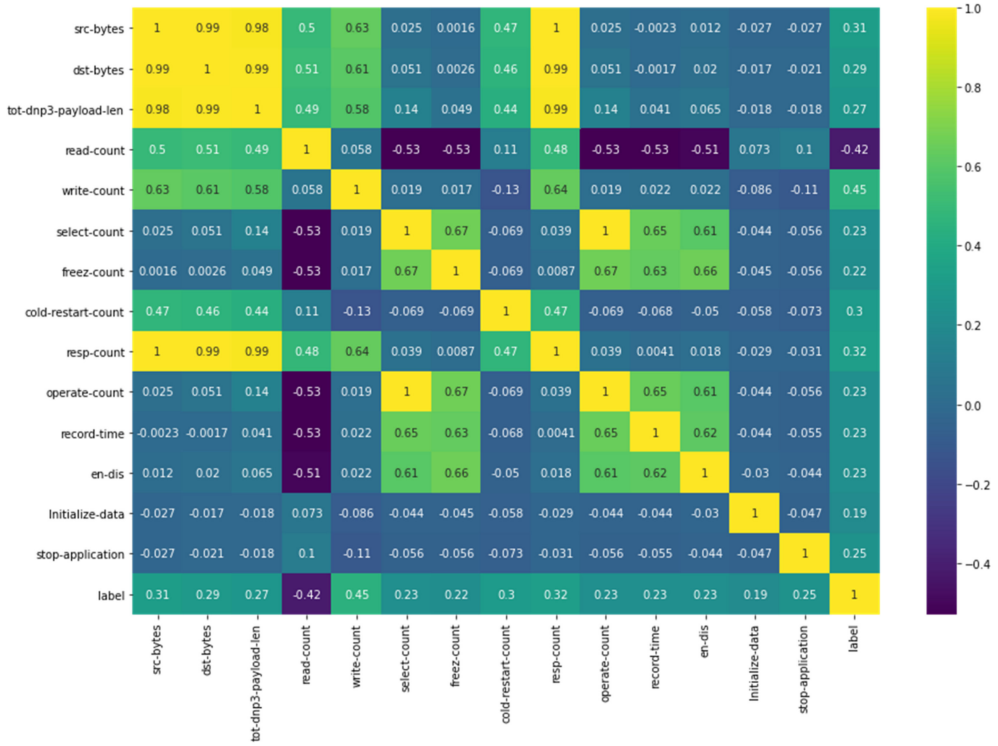

Figure 6. Correlation between input features.

4.1.1. Normal Dataset

As explained in Section 3.1, we generated the packets that followed the traffic pattern observed in a real-life substation operation. Figure 7 shows a breakdown of all the function codes seen in the normal dataset. As shown in this figure, the majority of the function codes were Read and Response; Stop, Restart, or Initialize function codes rarely took place. The Write, Select/Operate, and Freeze/Clear function codes made up less than 5% of the total application layer traffic.

Figure 7. Function codes distributions of normal dataset.

4.1.2. Dataset D1

The attack packets were classified into six categories depending on the attack types, as shown in Figure 8. As each name implies, the goal of the injection attack was to send new illegitimate packets, and the goal of the removing attack was to delete legitimate packets. The goal of the alter attack was to send packets with modified function codes. Among the injections, when injected packets had 'Cold', 'Initialize', or 'Stop' function codes, the attack types were specified separately. These function codes can impact system availability, which is considered the top priority security issue in the industrial control system. The D1 dataset contained almost 5% attack packets and 95.4% normal packets. The chart in Figure 8 shows the proportions of normal and abnormal packets of each attack type.

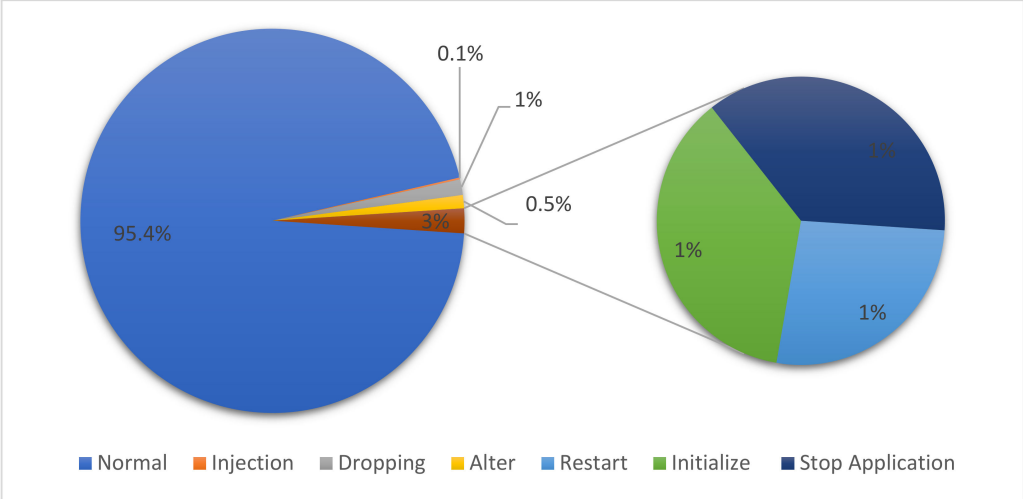

Figure 8. Function codes distributions for dataset D1.

4.1.3. Dataset D2

In dataset D2, we increased the attack percentage up to 25%, while the other 75% maintained normal DNP3 traffic. The increment in function codes was as follows: injection function code at 5%; the drop and altering function codes were both at 6%; cold restart code was around 4%; stop and initialize code was up to 2%. Figure 9 below shows the proportions of DNP3 traffic in dataset D2.

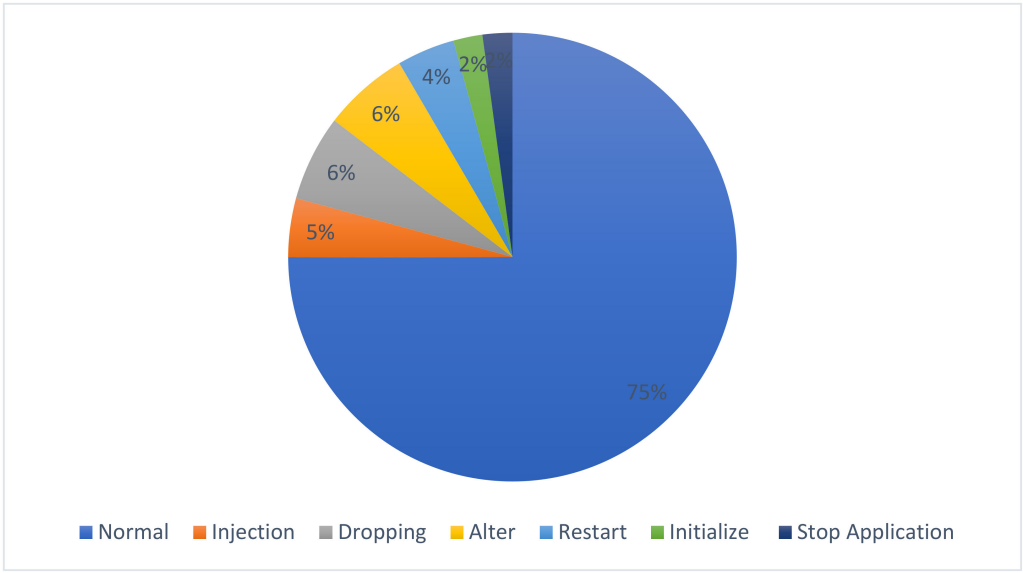

Figure 9. Function code distributions for dataset D2.

4.1.4. Dataset D3

Dataset D3 was constructed on the assumption that almost half of all communications were attack data. The other near-half of transmitted function codes were normal ones. The percentages of attack data increased by double; the biggest increase was in injection attacks, because more scope was presented for attackers to choose among all attack scenarios. The increase in injection attacks reached 18% of the total traffic, while dropping, alter, and cold attacks were just 8%, and initialize and stop attacks amounted to 4%, as shown in Figure 10.

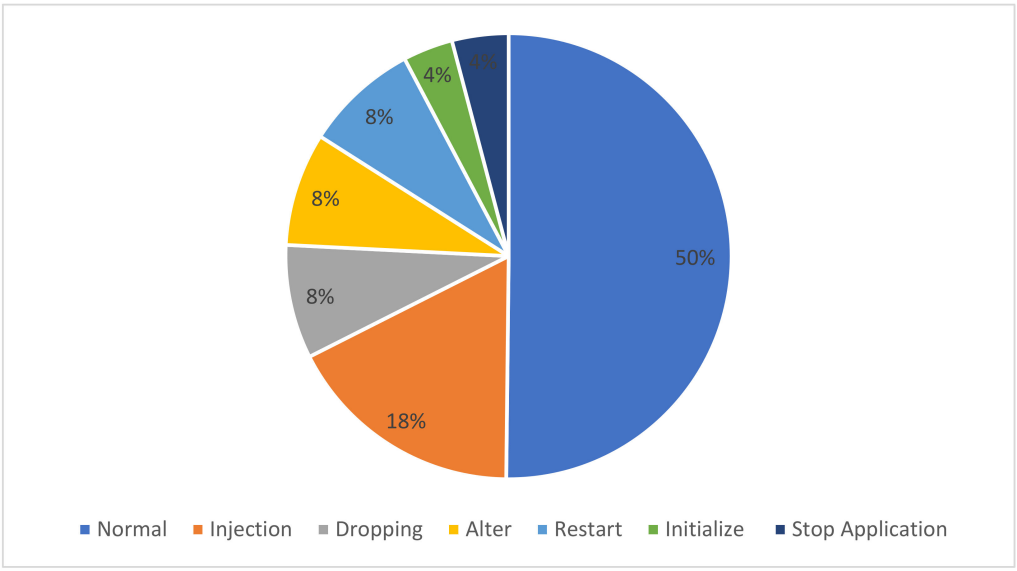

Figure 10. Function code distributions for dataset D3.

Figure 11 shows the distribution of attack types in datasets D1, D2, and D3. Dataset D1 had: 1360 normal instances; 14 injection attack instances; 19 dropping and 16 altering instances; 9 injected cold restart instances; and 11 initialized and stop application instances. This gave a total of 1440 instances, and the attack percentage amounted to slightly over 5% of the total traffic. Datasets D2 and D3 had 1079 and 722 normal instances, respectively, together with 360 and 717 abnormal instances, which marked around 25% and 50% attack rates for the total traffic. Figure 12 also shows the function codes frequency of four datasets. It verifies that the datasets most closely followed the function codes frequency of the normal dataset, which traced the traffic characteristics of DNP3 application layer data observed from live operating power substation networks for a reasonably long time [19], (Figure 7).

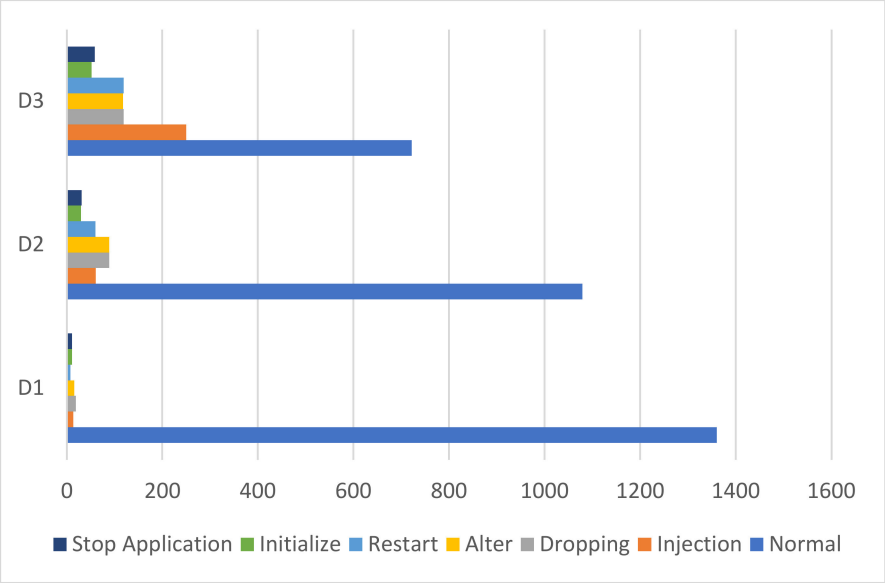

Figure 11. Attack distributions of datasets D1, D2, D3.

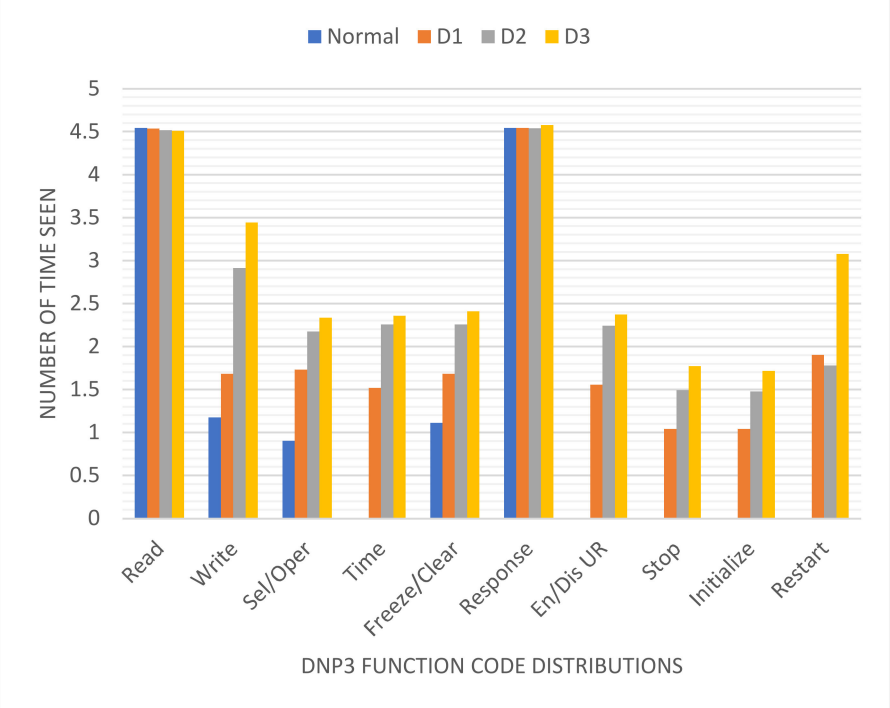

Figure 12. Function codes distributions of Normal, D1, D2, D3.

4.2. Performance Evaluation

The Keras programming environment, with TensorFlow at the backend, was used to design the model. Designing an efficient deep learning model is a challenging task. First, we carried out the hyperparameter optimization of the autoencoder model. The proposed model was then trained by normal D1, D2, and D3 datasets.

The data generation and evaluation were carried out over three datasets, which were captured over 3 days respectively. As mentioned in the previous sections, datasets D1, D2, and D3 contained attack instances, which were dispersed randomly in the normal dataset. Each attack was executed to last approximately 60 s (one instance). The modification or injection attacks were configured using 1 to 10 DNP3 packets during one instance, to provide a reasonable impact on DNP3 traffic. In this experiment, we calculated the Mean Squared Error (MSE) for a normal dataset through the autoencoder network. The MSE was evaluated by the following equation:

$$\mathrm{MSE}(x', x) = \frac{1}{N} \cdot \sum_{n=1}^{N} \left(x'_n - x_n\right)^2 \quad (1)$$

where x represented an actual output, and x' represented a predicted output. The maximum MSE of the normal traffic was used as a criterion to determine whether instances were normal or abnormal.

Figure 13 shows the MSE of the normal and attack traffic of the three datasets, D1 through D3. In this figure, the *y* axis represents MSE, and the *x* axis is the instance. The six graphs show how many errors were in the normal traffic, and how much deviation happened in the attack traffic. Sporadic spikes in data traffic implied that they may have deviated from normal patterns.

Figure 13a shows the MSE of dataset D1 without any attacks, which had a minimum error of around 0.000938, an average MSE of around 0.001067, and a maximum MSE of about 0.004550. Figure 13b shows the results of the attack packets in dataset D1. We can see higher errors in this graph, as expected, having an average MSE of 0.096794 and a maximum of around 0.303068. However, there were some instances of a minimum MSE of 0.001329, which could be considered as normal traffic, depending on the threshold values for classifying normal or abnormal. If we use the average MSE normal traffic, these instances were above the threshold, resulting in being considered abnormal instances. Figure 13c–f shows the results for datasets D2 and D3. As they had more attack instances, they had more spikes in the attack data. Otherwise, they had similar patterns as that of dataset D1, and we can apply the same interpretation to these datasets.

To validate our method, we compared the results with those obtained when the model used the input features that the traditional IDS technique usually works with. We extracted the input features—which were mostly related to TCP characteristics, which had 41 attributes as shown in [33], (p. 450, Table VI)—from the same datasets we generated. We used those features for the AE-IDS model as an input layer consisting of 41 nodes, and an output layer consisting of 41 nodes with some hidden layers. We then evaluated the performance of the IDS that used traditional features (TCP features).

Figure 14a shows the MSE of dataset D1 without any attacks, which had a minimum error of around 0.003211, an average MSE of around 0.008258, and a maximum MSE of about 0.039162. Figure 14b shows the results of the attack on dataset D1. We can see higher errors in this graph, as expected, having an average MSE of 0.062872 and a maximum of around 0.173274. From the results, we can see from the MSE metric, even with a 5% attack, that the traditional features do not give much information about DNP3 attacks. Figure 14c–f shows the results for dataset D2 and D3. From the observation of these graphs, we can see that the traditional features of AE-IDS have difficulties in distinguishing normal instances from abnormal ones.

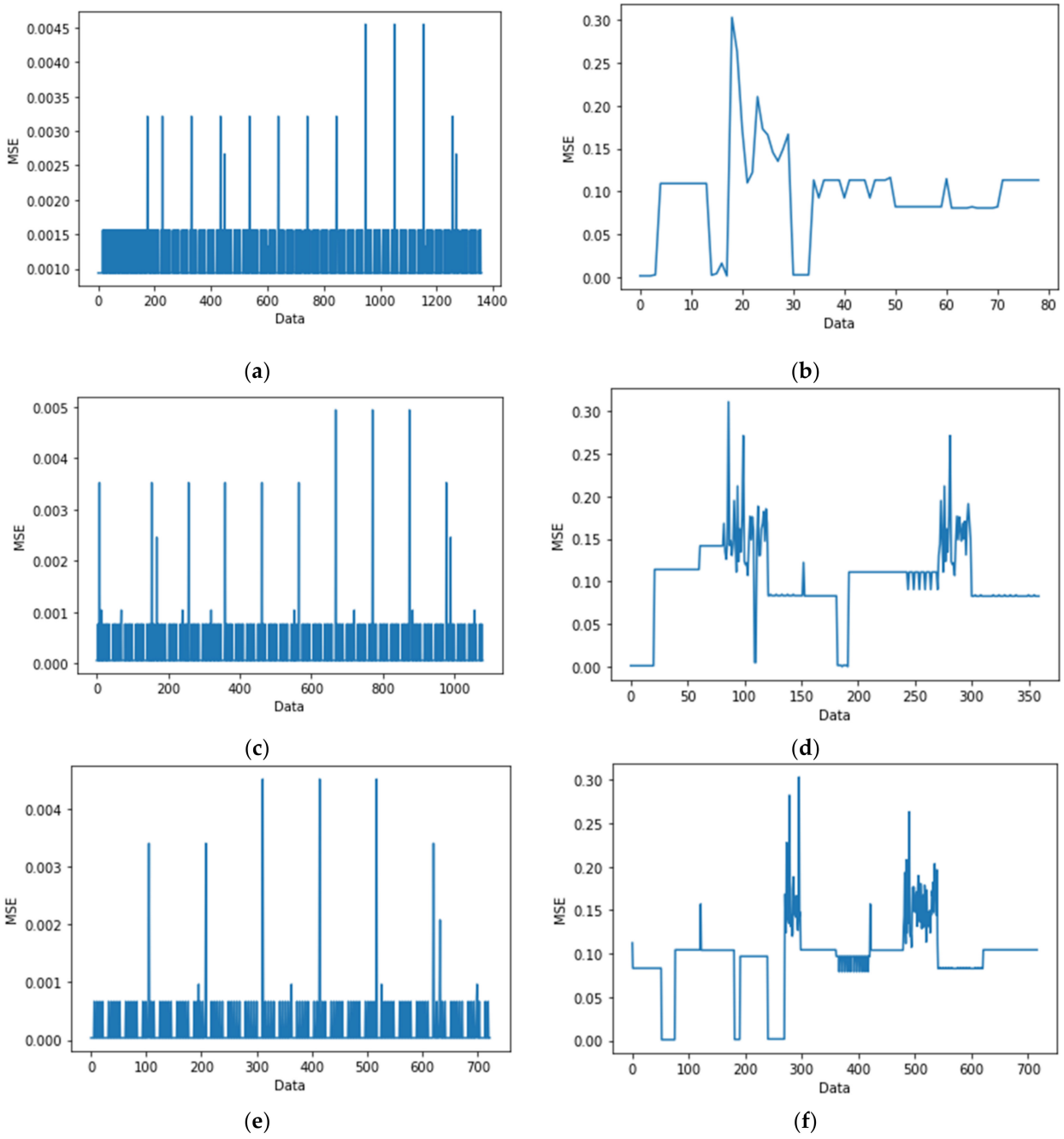

Figure 13. Results from FC-AE-IDS: (**a**) MSE of normal traffic for dataset D1; (**b**) MSE of attack traffic for dataset D1; (**c**) MSE of normal traffic for dataset D2; (**d**) MSE of attack traffic for dataset D2; (**e**) MSE of normal traffic for dataset D3; (**f**) MSE of attack traffic for dataset D3.

From a comparative perspective, to evaluate the effectiveness of our method, we computed the following performance metrics for the three datasets: accuracy, recall, precision, and F1. In these metrics, True Positive (TP) was the number of attacks classified correctly as attacks; True Negative (TN) was the number of normal events rightly classified as normal; False Positive (FP) was the number of normal events misclassified as attacks; and False Negative (FN) was the number of attacks misclassified as normal.

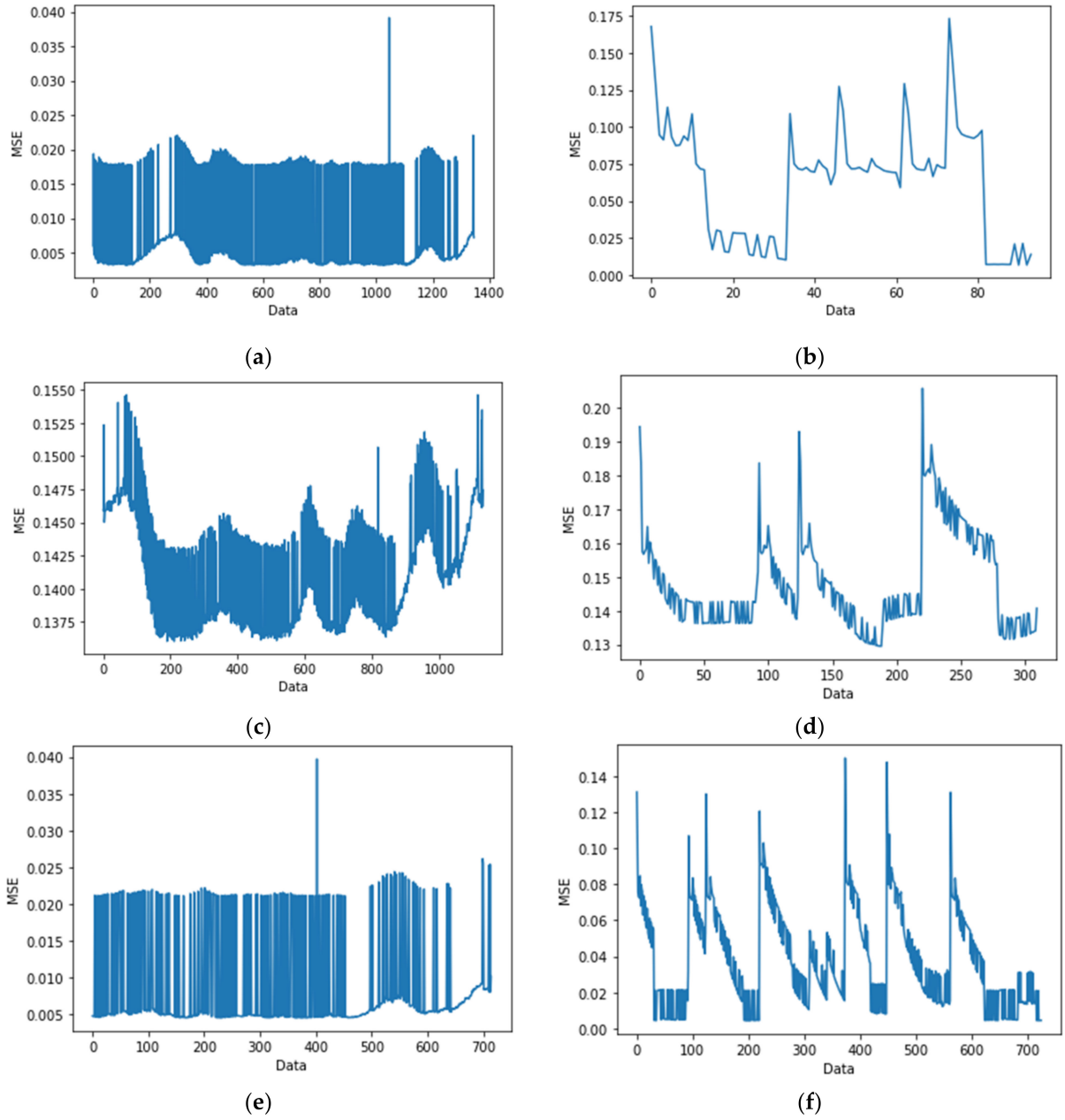

Figure 14. Results from the traditional AE-IDS: (**a**) MSE of normal traffic for dataset D1; (**b**) MSE of attack traffic for dataset D1; (**c**) MSE of normal traffic for dataset D2; (**d**) MSE of attack traffic for dataset D2; (**e**) MSE of normal traffic for dataset D3; (**f**) MSE of attack traffic for dataset D3.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FN} + \text{FP}} \tag{2}$$

$$\text{F1 score} = \frac{2 * \text{TP}}{2 * \text{TP} + \text{FP} + \text{FN}} \tag{3}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5}$$

As shown in Table 4, the performance of FC-AE-IDS was better than the traditional IDS (Trad-AE-IDS), which is based on 41 attributes as shown in [33]. In terms of accuracy,

FC-AE-IDS accuracy was approximately 99% for dataset D1, while it was 97.57% and 95.35% for datasets D2 and D3. This meant that there was a 95–97% chance of detecting any anomalous traffic inside the network. As can be seen, the Trad-AE-IDS accuracy of the trained data was 97.71% of accuracy for the D1 dataset, which only had 5% of attacks, while the recall and precision were 65% and 98%, respectively. The low recall rate meant that the Trad-AE-IDS classifier had a high number of false negatives (undetected attacks). This meant the model did not do well. The reason that precision was high was that the threshold was previously set too high. That D2 and D3 results fell off, meant that the Trad-AE-IDS features were not effective for those types of attacks. As a result, FC-AE-IDS features had better classification capability than Trad-AE-IDS features. Furthermore, this result implied that if the unsupervised learning model was used with imbalanced datasets like dataset D1, it might lead to biased results. The results show that our approach was very effective for various traffic with different attack ratios. In particular, the method produced a very accurate detection rate even for dataset D1, which had very low attack instances.

Table 4. Comparison of performance between different algorithms for different datasets.

Algorithm	Dataset	Accuracy	Recall	Precision	F1
Trad-AE-IDS	D1	97.71	65.96	98.41	78.98
	D2	84.72	29.35	98.91	45.27
	D3	70.97	42.48	99.68	59.57
FC-AE-IDS	D1	99.03	86.08	95.77	90.67
	D2	97.57	91.39	99.09	95.50
	D3	95.35	91.10	99.54	95.12

5. Conclusions

When an intruder hijacks internal nodes in the SCADA system by utilizing the software vulnerability of the system servers, they can possess the system privilege to modify or fabricate application messages, eventually disrupting the system operation at will. These highly sophisticated attacks are the ones that commonly happened in the recent SCADA/ICS attack incidents. One of the contributions of this paper is to address how to cope with this kind of attack, which can avoid normal rule-based inspections derived from deployed protocols and/or internal configurations. As most critical operations in DNP3 systems are performed based on the function codes, attackers exploit the function codes and manipulate the system operation. The proposed anomaly-detection method uses the function codes frequency per TCP connection as input features, along with the TCP characteristics, to find normal behaviors of DNP3 traffic. The other novelty of this work is that we made and generated several datasets that reflected DNP3 traffic characteristics that have been observed in real-world power substations, according to [19]; and, after generating the dataset, we were able to obtain the correct information about normal patterns of the DNP3 traffic. Therefore, the function code analysis could be weighted to help with classification. Finally, we compared our model FC-AE-IDS to traditional IDS, and the proposed model outperformed the traditional method in terms of accuracy, recall, and F1 score, proving its efficiency for SCADA security. This gives more validity to the proposed method based on function code analysis, showing that it is better for classification, rather than just depending on the TCP connection features.

As further study, we may extend the input features to improve the effectiveness of detection. We will also further explore how sparsity constraints are imposed on the autoencoder, and how sparse AE-IDS can be designed to further improve intrusion-detection effectiveness. We may also extend the unsupervised model further, to ones like GAN, to improve the model's accuracy.

Author Contributions: Methodology, software, validation, formal analysis, investigation, resources, data curation, writing—original draft preparation, visualization, M.A.; methodology, formal analysis, investigation, review and editing, supervision, project administration, funding acquisition, S.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the South Korean government (MSIT) (No. 2021R1F1A1062412).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hong, S. Cyber security strategies for substation automation systems and their implications. *Int. J. Smart Grid Clean Energy* **2019**, *8*, 747–756. [CrossRef]
2. Mitchell, R.; Chen, I.-R. A survey of intrusion detection techniques for cyber-physical systems. *ACM Comput. Surv.* **2014**, *46*, 4. [CrossRef]
3. Hu, Y.; Yang, A.; Li, H.; Sun, Y.; Sun, L. A survey of intrusion detection on industrial control systems. *Int. J. Distrib. Sens. Netw.* **2018**, *14*, 1–13. [CrossRef]
4. Hong, S.; Lee, J.-M.; Altaha, M.; Aslam, M. Security Monitoring and Network Management for the Power Control Network. *Int. J. Electr. Electron. Eng. Telecommun.* **2020**, *9*, 356–363. [CrossRef]
5. Lin, H.; Slagell, A.; Kalbarczyk, Z.T.; Sauer, P.W.; Iyer, R.K. Runtime Semantic Security Analysis to Detect and Mitigate Control-Related Attacks in Power Grids. *IEEE Trans. Smart Grid* **2018**, *9*, 163–178. [CrossRef]
6. Drewek-Ossowicka, A.; Pietrołaj, M.; Rumiński, J. A survey of neural networks usage for intrusion detection systems. *J. Ambient Intell. Humaniz. Comput.* **2021**, *12*, 497–514. [CrossRef]
7. Luo, Y.; Xiao, Y.; Cheng, L.; Peng, G.; Yao, D. Deep Learning-Based Anomaly Detection in Cyber-Physical Systems: Progress and Opportunities. *ACM Comput. Surv.* **2020**, *54*, 1–36. [CrossRef]
8. Aleesa, A.M.; Zaidan, B.B.; Zaidan, A.A.; Sahar, N.M. Review of intrusion detection systems based on deep learning techniques: Coherent taxonomy, challenges, motivations, recommendations, substantial analysis and future direction. *Neural Comput. Appl.* **2019**, *32*, 9827–9858. [CrossRef]
9. Liu, H.; Lang, B. Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey. *Appl. Sci.* **2019**, *9*, 4396. [CrossRef]
10. Xin, Y.; Kong, L.; Liu, Z.; Chen, Y.; Li, Y.; Zhu, H.; Gao, M.; Hou, H.; Wang, C. Machin Learning and Deep Learning Methods for Cybersecurity. *IEEE Access* **2018**, *6*, 35365–35381. [CrossRef]
11. Gomez, A.L.P. On the Generation of Anomaly Detection Datasets in Industrial Control Systems. *IEEE Access* **2019**, *4*, 177460–177473. [CrossRef]
12. Adepu, S.; Kandasamy, N.K.; Mathur, A. *EPIC: An Electric Power Testbed for Research and Training in Cyber Physical Sys-tems Security, Computer Security*; Springer International Publishing: Berlin/Heidelberg, Germany, 2019; pp. 37–52.
13. Kwon, S.; Yoo, H.; Shon, T. IEEE 1815.1-Based Power System Security with Bidirectional RNN-Based Network Anomalous Attack Detection for Cyber-Physical System. *IEEE Access* **2020**, *8*, 77572–77586. [CrossRef]
14. Lee, J.-M.; Hong, S. Keeping Host Sanity for Security of the SCADA Systems. *IEEE Access* **2020**, *8*, 62954–62968. [CrossRef]
15. Lee, J.-M.; Hong, S. Host-Oriented Approach to Cyber Security for the SCADA Systems. In Proceedings of the 2020 6th IEEE Congress on Information Science and Technology, Agadir-Essaouira, Morocco, 5–12 June 2021.
16. *IEEE Std 1815-2012*; IEEE Standard for Electric Power Systems Communications—Distributed Network Protocol (DNP3). IEEE Power and Energy Society: New York, NY, USA, 2012.
17. *IEC TC57 WG15*; Power Systems Management and Associated Information Exchange—Data and Communications Security—Part 3: Communication Network and System Security—Profiles Including TCP/IP, IEC 61850-3. IEC: Geneva, Switzerland, 2014.
18. Singh, C.; Nivangune, A.; Patwardhan, M. Function code based vulnerability analysis of DNP3. In Proceedings of the 2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Bangalore, India, 6–9 November 2016.
19. Irvene, C.; Shekari, T.; Formby, D.; Beyah, R. If I Knew Then What I Know Now; On Reevaluating DNP3 Security using Power Substation Traffic. In Proceedings of the 5th Annual Industrial Control System Security (ICSS) Workshop, San Juan, PR, USA, 10 December 2019; pp. 48–59.
20. Formby, D.; Walid, A.; Beyah, R. A Case Study in Power Substation Network Dynamics. In Proceedings of the ACM on Measurement and Analysis of Computing Systems, Online, 13 June 2017; Volume 1, pp. 1–24.
21. Jung, S.S.; Formby, D.; Day, C.; Beyah, R. A first look at machine-to-machine power grid network traffic. In Proceedings of the 2014 IEEE International Conference on Smart Grid Communications (SmartGridComm), Venice, Italy, 3–6 November 2014; pp. 884–889. [CrossRef]
22. Wang, C.; Wang, B.; Liu, H.; Qu, H. Anomaly Detection for Industrial Control System Based on Autoencoder Neural Network. *Wirel. Commun. Mob. Comput.* **2020**, *2020*, 8897926. [CrossRef]
23. Farahnakian, F.; Heikkonen, J. A Deep Auto-Encoder based Approach for Intrusion Detection System. In Proceedings of the 2018 20th International Conference on Advanced Communication Technology (ICACT), Chuncheon, Korea, 11–14 February 2018.

24. Altaha, M.; Lee, J.-M.; Aslam, M.; Hong, S. An Autoencoder-Based Network Intrusion Detection System for the SCADA System. *J. Commun.* **2021**, *16*, 210–216. [CrossRef]
25. Grammatikis, P.R.; Sarigiannidis, P.; Efatathopoulos, G.; Panaousis, E. ARIES: A Novel Multivariate Intrusion Detection System for Smart Grid. *Sensors* **2020**, *20*, 5305. [CrossRef] [PubMed]
26. Shahriar, M.D.; Haque, N.I.; Kahman, M.A.; Alonso, M. G-IDS: Generative Adversarial Networks Assisted Intrusion Detection Systems. *arXiv* **2020**, arXiv:200676v1.
27. Rodofile, N.R.; Radke, K.; Foo, E. Framework for SCADA cyber-attack dataset creation. In Proceedings of the Australasian Computer Science Week Multiconference, Online, 31 January 2017.
28. Radoglou-Grammatikis, P.; Sarigiannidis, P.; Efstathopoulos, G.; Karypidis, P.-A.; Sarigiannidis, A. DIDEROT: An intrusion detection and prevention system for DNP3-based SCADA systems. In Proceedings of the 15th International Conference on Availability, Reliability and Security, Virtual Event, Dublin, Ireland, 25–28 August 2020. [CrossRef]
29. East, A.; Butts, J.; Papa, M.; Shenoi, S. *A Taxonomy of Attacks on the DNP3 Protocol, Critical Infrastructure Protection III*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 67–81.
30. Linda, O.; Vollmer, T.; Manic, M. Neural Network Based Intrusion Detection System for Critical Infrastructures. In Proceedings of the 2009 International Joint Conference on Neural Networks, Atlanta, GA, USA, 14–19 June 2009; pp. 1827–1834.
31. Available online: https://dnp3.github.io (accessed on 6 July 2022).
32. Available online: https://www.wireshark.org (accessed on 6 July 2022).
33. Dhanabal, L.; Shantharajah, S.P. A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. *Int. J. Adv. Res. Comput. Commun. Eng.* **2015**, *4*, 446–452.