CNN-based anomaly detection for packet payloads of industrial control system

Joo-Yeop Song and Rajib Paul

Department of Software and Computer Engineering,
Ajou University,
16499, South Korea
Email: yubie12@ajou.ac.kr
Email: rajib@ajou.ac.kr

Jeong-Han Yun and Hyoung Chun Kim

The Affiliated Institute of ETRI,
34044, South Korea
Email: dolgam@nsr.re.kr
Email: khche@nsr.re.kr

Young-June Choi*

Department of Software and Computer Engineering,
Ajou University,
16499, South Korea
Email: choiyj@ajou.ac.kr
*Corresponding author

Abstract: Industrial control systems (ICSs) are more vulnerable to cyber threats owing to their network connectivity. The intrusion detection system (IDS) has been deployed to detect sophisticated cyber-attack but the existing IDS uses the packet header information for traffic flow detection. IDS is inefficient to detect packet deformation; therefore, we propose the adoption of packet payload in IDS to respond to a variety of attacks and high performance. Our proposed model detects packet modification and traffic flow by inspecting each packet and sequence of packets. For evaluation, cross verification is conducted to increase the reliability of the statistics.

Keywords: network security; intrusion detection; anomaly detection; convolutional neural network; industrial control system; N-gram method; single packet detection; sequence detection.

Reference to this paper should be made as follows: Song, J-Y., Paul, R., Yun, J-H., Kim, H.C. and Choi, Y-J. (2021) ‘CNN-based anomaly detection for packet payloads of industrial control system’, *Int. J. Sensor Networks*, Vol. 36, No. 1, pp.36–49.

Biographical notes: Joo-Yeop Song has recently completed his Master’s from Ajou University in 2019. His research interests are wireless communications, security, anomaly detection and machine learning.

Rajib Paul is an Assistant Professor at Ajou University, Korea. He received BTech degree in Electronics and Communication Engineering from West Bengal University of Technology, India in 2009. He acquired his Master’s and PhD degrees from the Department of Information and Computer Engineering, Ajou University, in 2012, and 2017, respectively. His research focus includes cognitive radio networks, wireless communications, vehicular networks, machine learning, and device-to-device communications.

Jeong-Han Yun is a Principal Researcher of the Infrastructure Protection Department in the Affiliated Institute of ETRI. He received his BSc, MSc and PhD degrees from Computer Science of KAIST in South Korea. His research interests are security operation management in the CPS environment including intrusion detection using machine learning and dataset generation.

Hyoung Chun Kim is currently a Principal Researcher and a Manager of the Infrastructure Protection Department in the Affiliated Institute of ETRI. His research interests include software security, virtualisation security in cloud computing, secure operating systems, and intrusion detection systems using machine learning in the CPS environment. He received his BSc and MSc degrees in Computer Science from Korea University. He got his PhD degree from the Graduate School of Information Security of Korea University.

Young-June Choi is a Professor at Ajou University, Korea. He received his MS and PhD degrees from the Department of Electrical Engineering and Computer Science, Seoul National University, Korea, in 2002, and 2006, respectively. From 2006 to 2007, he was a Post-Doctoral Researcher at the University of Michigan, Ann Arbor, MI, USA. From 2007 to 2009, he was with NEC Laboratories America, Princeton, NJ, USA, as research staff member. He is an IEEE senior member, an executive director of the Korean Institute of Communications and Information Sciences (KICS), and a Director of the Korean Institute of Information Scientists and Engineers (KIISE).

This paper is a revised and expanded version of a paper entitled 'Generating abnormal industrial control network traffic for intrusion detection system testing' presented at *12th IFIP WG 11.10 International Conference, ICCP 2018*, Arlington, VA, USA, 12–14 March, 2018

1 Introduction

The connectivity of the internet has expanded to cover many industrial domains through Internet-of-Things (IoT) (Brändle and Naedele, 2008); therefore, cyber-security is becoming an important influence for the maintenance and operation of those networks. Especially, the industrial control system (ICS) that has been designed for energy, gas, water, and transportation facility is also connected with the network, even with the internet. These systems are considered being safe and reliable because they have been designed with an emphasis on operationality and security due to the importance of the facility. However, it is reported that ICS networks are also exposed to security threats and the number of reports regarding the threats and vulnerability is increasing (Maglaras et al., 2018; McLaughlin et al., 2016).

To cope with security threats, one of the general approaches is to adopt policies and guidelines such as profiles, accounts, and permissions (Nicol et al., 2008) which define how users access the network or handle the environment. For technical approaches, traditionally data encryption and firewalls have been exploited. Data encryption is a good way to protect the contents of system data, but it requires an encryption process, which can hinder operationality depending on the system environment. However, encryption is not applicable to cover all different types of problems. Firewalls can be installed easily at low cost and require fewer calculations for judgement, which makes them fast to operate. However, firewalls work by setting rules, not by understanding packets. Zero-day attacks or newer types of attacks are not part of the rules and thus they are hard to be detected. Furthermore, defining rules is difficult and requires continuous updating of rules to eliminate vulnerabilities (Garcí-a-Teodoro et al., 2009). Therefore, an additional security system is needed to understand packets and distinguish between normal and abnormal packets.

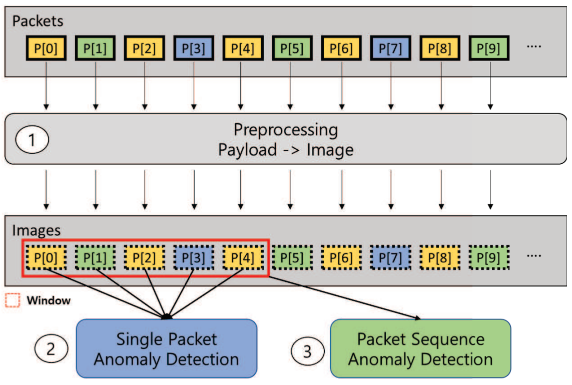
Intrusion detection system (IDS) has been widely used to detect abnormal packets. It can be classified as network-based IDS (NIDS) and host-based IDS (HIDS), depending on the location of the installation. NIDS is installed at specific points in the network, and detects intrusion on multiple hosts, thereby detect attacks looking for targets, scanning vulnerabilities, or entering the system. This approach targets network packets that can be analysed regardless of the host system environment. Therefore, NIDS analyses network traffic flow or packet content and uses snipping to do this. In contrast, HIDS is installed inside a specific host and monitors various states of the host internal system to detect certain behaviours or abnormal situations. HIDS analyses system calls on the host, application logs, file system changes, operational status, etc.

ICS consists of devices for various environments. ICS is comprised of a variety of applications, such as industrial control (IC), industrial process measurement and control (IPCM), supervisory control and data acquisition (SCADA), distributed control (DC), metering, monitoring and diagnostic (MMD), networked electronic control and sensing (NECS), programmable logic controller (PLC), and distributed automation (DA) systems (Cheminod et al., 2013). Further, sensors, PLCs, managed devices, and working devices have different system environments. For sensors or PLCs, system performance may be limited and security technologies may need to consider limited performance. Therefore, NIDS is a realistic solution for ICS of many devices and our focus is on NIDS which can be installed out of each device, because it is not essential to consider the environment and performance of each device.

In literature, IDS has adopted machine learning techniques to learn the behaviour of data traffic. IDS-related studies have mainly used support vector machine (SVM), cluster, and K-nearest neighbour (kNN) techniques. Those studies do not use network traffic as input for model generation, but with special preprocessing such as KDD99 and NSL-KDD (Tavallaee et al., 2009), data of a certain characteristic or packet header information can be extracted (Mahoney and Chan, 2001). Some studies perform byte conversion using payload bytes. Conversion of Z-string, byte distribution information (Wang and Stolfo, 2004; Kind et al., 2009), or partial bytes are exploited to model payload data (Hareesh et al., 2011). Other studies use packet headers, only abnormal parts of the traffic flow, such as DoS and new IP intrusions, for detection. Meanwhile, some studies use payload bytes to detect attacks using payload modification that modifies bytes instead of intact ones. In addition, studies using public data find it difficult to verify that they can function properly even in the network environment they want to install (Khandpur et al., 2017).

The overall configuration of the proposed detection system is shown in Figure 1. First, packet data is pre-processed into the image form. Detections are performed using pre-processed image data, and detections consist of two methods. Single packet anomaly detection checks the status of individual packets to detect abnormalities in individual packets. If all are normal, the packet sequence anomaly detection will organise the packet previously identified into a single window and use the window as input data to detect abnormalities in the traffic flow. Because the packet was converted to image form during pre-processing, all information need not be parsed when analysing the payload of packets through this method. Users can get hints of normal or abnormal characteristics by looking at the shape in which the hex values of the payload coming from tools such as Wireshark. In other words, one learns the common characteristics between normal packets by looking at the packet form as one image. Packet images showing different singularities are judged to be abnormal packets.

Figure 1 Our anomaly detection system for payloads of ICS network traffic using CNN (see online version for colours)



The network traffic consist of 11 sites, collected from active control systems in critical infrastructure, and collected over a period of 3–29 days per site. In the case of data size, totaling about 18 TB, of which site No. 11 accounts for about 15 TB (Yun et al., 2018). The study was conducted by analysing and utilising traffic data from various sites and various points in time of this network data. The study was conducted by analysing and utilising traffic data from various sites and various points in time of this network data.

We use and analyse network traffic collected from running ICSs of critical infrastructure, where we find that the payload information in ICS typically has a pattern in the payload, unlike the internet traffic. As a means of utilising the entire payload, the detection model learns network packets like an image. To utilise image, we use a convolutional neural network (CNN) that is effective in image processing as a detection model. Through detection models using CNN structure, we propose two types of detection: single-packet detection and sequence-of-packets detection. In the work, we focused on the anomalies related to the falsification of values on packet which can occur critical problems in ICS system because of the vulnerability of PLC. SSSP attacks, SSMP attacks, MSSP attacks, and MSMP attacks (Adepu and Mathur, 2018) are the representative examples. Our proposal is designed to detect anomaly which comes from these value falsification and measurements errors with applying payload in detection model.

To fit in the input of CNN models, we also introduce histogram, padding, and filter methods to pre-process packet payload. To learn the payload information, we generated abnormal data and then conducted performance verification on each of the detection models and further improved the reliability through cross-validation. Our approach to make a model from the payload of packets using CNN is feasible for ICS traffic. Network traffic in a typical environment is transmitted by various protocols such as FTP, HTTP, and VoIP. Also, it is difficult to use payload data directly because it is encrypted over SSL/TLS. Protocols such as DNP3, Modbus for ICS are designed for availability and without encryption, or predetermined message formats. According to our knowledge, this work is the first to apply CNN directly to the raw payload without protocol-specific preprocessing.

The remainder of this paper consists of seven chapters. Section 2 introduces existing techniques and studies related to IDS, and Section 3 analyses the network traffic and introduces machine learning methods. Section 4 explains how to pre-process packet payload and Section 5 describes our abnormal detection models. Section 6 shows how to generate abnormal data and the verification results of the pre-processing method and the detection model and previously introduced in Sections 4 and 5. Section 7 summarises the study.

2 Related work

IDS can be classified into signature-based and anomaly-based detection. Signature-based detection is a way to build a huge database of attack information and to detect matched attacks in the database, thus relying on human expertise and being vulnerable to zero-day attacks. In addition, as attacks become more intelligent, obvious limitations are observed in the signature-based detection. On the other hand, anomaly-based detection is guided by how to define predefined network behaviour that identifies normal or abnormal behaviour. Allowable network behaviour depends on the extent of the learned network data. Thus, anomaly-based detection has been investigated by complementing the limitations of signature-based detection and enabling high levels of security.

In statistical approaches, the activities of network traffic are monitored and profiles are created to represent their probabilistic behaviour such as traffic rate, the number of packets for each protocol, connection speed, and the number of different IP addresses. The earliest statistical approach to matching NIDS and HIDS is the one-parameter model which is modelled with independent Gaussian random variables (Denning and Neumann, 2016). An advantage of such a statistical approach is the accurate detection of malicious activities without requiring prior knowledge of normal activities. The disadvantage is that an attacker can intentionally learn malicious patterns and make setting parameters and metrics difficult to work, which assumes an unrealistic quest process.

One of the most widely used knowledge-based IDS systems is expert systems that are intended to classify audit data according to a set of rules. In terms of rules (specifications) intended to determine reasonable system behaviour, some models may be manually constructed by human experts. The most important benefits of knowledge-based anomaly detection are robustness and flexibility. The main disadvantage is that the development of high-quality knowledge is often difficult and time- and resource-consuming (Sekar et al., 2002).

A major highlighted approach is machine learning techniques that establish a model to classify analysed patterns. They require labelled data to train the model. In many cases, the applicability of the machine learning principle is similar to the applicability of statistical techniques, but machine learning focuses on building models that enhance performance based on previous results, thus providing flexibility, adaptability, and interdependencies. However, machine learning models are heavily dependent on assumptions about acceptable behaviour of the system and consume a lot of resources.

While machine learning techniques vary depending on the amount of data and how they are used, they can generally provide a suitable environment for deploying anomaly-based detection. Although data is difficult to access and analyse statistically, and problems that people find are difficult to specify, they can be characterised and modelled by themselves. The model is also highly scalable and can easily respond to environmental changes. On the other hand, statistical approaches are not suitable for payload-based detection because profiles use information such as the number of packets, connection speed, and the number of IP addresses. The knowledge-based approach is based on expert experience, intuition, and judgement and thus mainly enables to analyse the overall form or part of the feature. Therefore, despite its feasibility, analysing the entire payload is difficult and inefficient, whereas it is easily implementable in machine learning models.

State-of-the-art work using machine learning to implement IDS is found in the literature regarding machine learning techniques and data. Since the features available are already given, existing work has usually used techniques such as SVM, kNN, and cluster for abnormal detection, with public data such as KDD99 and DARPA1998 (Tsai et al., 2015; Ingre and Yadav, 2015; Pervez and Farid, 2014; Ravale et al., 2015) that provides necessary information for model learning. Such public datasets also provide packet header information (IP, Port, payload length) and packet frequency information (Kind et al., 2009; Lin et al., 2015; Javaid et al., 2016; Wang et al., 2010). Most studies utilise partial payloads (the first 100 bytes or some bytes at specific positions) rather than full payloads, or information of the frequency of the bytes (Hareesh et al., 2011). The reason why full payloads are not used is that the machine learning techniques including SVM, kNN, and cluster are affected by the number of parameters. As a result, many studies use specific information and small parameters (Bhuyan et al., 2014).

For detection models that use packet headers as input data, there is a problem that they have the same header information and no difference in input data even though the payload changes (Kind et al., 2009). Furthermore, the methodology of using byte distribution typically performs the task of classifying protocols (e.g., VoIP, HTTP, and FTP) (Zhang et al., 2014) and does not effectively categorise differences of the same protocol type. In this case, it has high accuracy but a high false alarm. Therefore, it is not appropriate to use partial payloads as in the existing studies for anomaly detection.

To resolve the problems mentioned above while using payload-based IDS, an alternative solution is to adopt neural networks. Neural networks are free from the issue of input data size and work effectively to extract and classify features (Nguyen and Armitage, 2008). They use each payload byte as a feature and apply the entire payload for input data. If there are features and patterns in network traffic, a significant shape will be derived when the payload byte value is treated as a single RGB value and the payload is treated as a whole image. We use the images of normal packets to learn a model and allow the detection model to categorise to normal and abnormal packets. Although there are various structures in neural networks, we utilise CNN that is used effectively in image processing. In this work, we use a packet as one image in the process of CNN.

A study conducted by viewing sequence of packet size without payload (Papadogiannaki et al., 2018) showed that the sequence of packets could reflect the unique characteristics of service/protocol.

2.1 Data sharing architecture

There are three major types of data sharing architectures:

- Centralised, multiple datasets hosted at a single location in a common schema. For example, the Cancer Genome Atlas (TCGA) (https://portal.gdc.cancer.gov/) Data Portal manages genome related data; National Biomedical Imaging Archive (https://imaging.nci.nih.gov/ncia/login.jsf) manages DICOM based medical research data.
- Federated, with a virtual view of physically separate datasets. For example, Cancer Biomedical Informatics Grid (caBIG®) (https://biospecimens.cancer.gov/relatedinitiatives/overview/caBig.asp) is a grid based data federation infrastructure that supports a CQL query language across distributed data sources.
- Distributed, physically and virtually separate datasets.

Centralised approach is often limited to common data types. Biomedical research, however, generates complex data and often new data types. Distributed approach is often difficult to retrieve, interpret and aggregate results, and lacks data consistency between research sites. While caGrid is becoming widely used in biomedical research community, caGrid itself has a complex infrastructure and the effort is significant.

Considering the high dynamic nature of biomedical research and the need of cost effective data sharing, we develop a hybrid architecture which combines the benefits of the centralised approach and the distributed approach. In this approach, a data sharing central server is provided for multiple distributed data sources, and stores only published metadata (not raw data or images with large sizes) from distributed data sources. Users can have flexible management of data sharing through publishing or unpublishing data with a simple operation. The metadata contain context information of original data sources (including raw data) which are still managed at distributed research sites. The central server provides an integrated view of all shared data, and shared data can be easily aggregated and retrieved from the central server. This architecture not only is lightweight, but also provides support of consistent data sharing through collaboratively managing schema and semantically tagging of data.

3 Network traffic and machine learning

This research uses network traffic collected from the ICS network in critical infrastructure. First, we analyse the network traffic to derive any features that could be used for our model.

3.1 Traffic feature

The traffic data has been collected for a period of time and stored in PCAP format that has a unique information in terms of length, the number of packets, the number of IP-IP pairs, and payload length. Its overall characteristics are summarised in Table 1.

Table 1 Information of a single PCAP

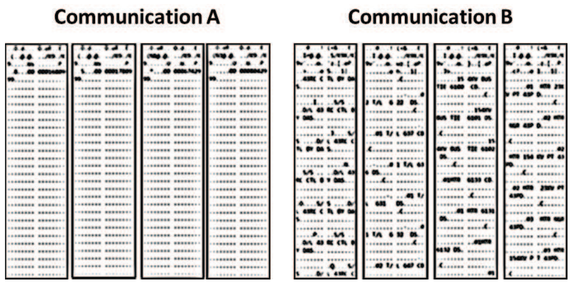
Parameter	*Value*
Size	700MB∼ 1.2GB
Traffic length	40Min ∼ 1 h
Number of packet	1Mil ∼ 3Mil
Number of IP-IP (one-way)	200 ∼ 300
Payload length	0 ∼ 1500 bytes

It contains about 1–3 million packets with a size of about 1 GB for an hour-long traffic. There are 200 to 300 IPs for one-way, which means if a link between $IP_1 \leftrightarrow IP_2$ is present, the two directions $IP_1 \rightarrow IP_2,\ IP_2 \rightarrow IP_1$ are separately counted. The payload length varies from 0 to 1500 across PCAPs, but usually has a regular length over a certain link, $IP_n \rightarrow IP_k$. Packets contain a set of certain lengths, such as $(12, 500, 600)$, $(570, 600, 1400)$, etc, owing to the nature of ICS.

When using only payloads of a single packet, a packet of zero length is excluded because there is no data available for learning. However, when using multiple packets together, the zero-length packets are taken into account for the trailing relationship.

The shape of IP payloads in each communication link is slightly different but has a unique pattern as shown in Figure 2. The payload of our dataset exhibits regular gaps, fixed shapes, repetition of specific values, change in byte values at specific locations, and no length greater than 1,500. For communication A in Figure 2, the average payload length of packets is about 500 bytes, and except for the first few bytes, the rest of the bytes are all composed of zeros, which can be a good feature in building machine learning models. The average payload length of packets in communication B in Figure 2 is around 1,200 bytes and although it does not show a particular pattern in the payload configuration, this may use information such as zero repetition after entering a certain length. The regularity of these iterations is likely to be used for machine learning. In addition to these two examples, there are various forms of communication. Using such features enables one to discover patterns and characteristics through neural networks, which might be otherwise difficult to design models and set rules one by one.

Figure 2 Packet payloads extracted from control system



The traffic used in the study also has some communication paths that do not have any features such as empty payloads. These parts seem to be working like normal network traffic and file transfer. However, traffic does not suddenly disappear or change when it has a characteristic pattern.

3.2 Machine learning for detection

Detection systems can be largely classified into signature-based and anomaly-based. Unlike signature-based detection that builds huge databases and find matched attacks, anomaly-based detection finds patterns on their own through learning. Therefore, anomaly-based detection is mainly used in machine learning methods and various methods are available for machine learning, including SVM, kNN, artificial neural network (ANN), etc. If one byte is viewed as a unit that contains a certain feature, it has as many as 1500 input data. SVM and kNN are generally not suitable for utilising these payloads because they operate with a few features that are already separated. ANNs are slightly different from such methods as SVM and kNN. The biggest advantage of using ANNs is that they can learn from the data they use and create the approximate function they want. Even if the byte value itself is used as a feature, it can create a model through an ANN.

Based on ANNs, deep neural networks such as CNN, recurrent neural network (RNN), and reinforcement learning have been evolved. An appropriate model among them is determined by the characteristics of the data and the purpose of the model. In this study, we choose CNN models for the dataset because the payload data collected in an ICS can be analysed as an image that has been proven to work effectively with CNN.

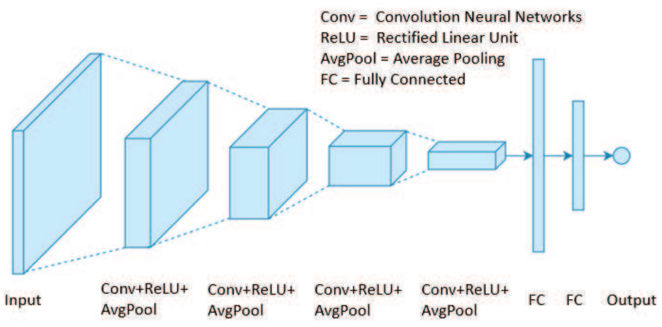
Learning models are created using the Tensorflow library for the CNN structure. The CNN structure consists of a repeated set of a convolutional layer, an activation function, and a pooling layer as well as a fully connected layer (Yamashita et al., 2018). The learning model was constructed from these layers. The input size of the model is determined by data and preprocessing methods. The output of the model is divided into two classes, normal and abnormal. A simple learning model is shown in Figure 3.

CNN maintains the shape of the I/O data for each layer, effectively recognises the characteristics of adjacent images while maintaining spatial information, and has the characteristics of images extracted through the pooling layer that can be gathered and enhanced. These parts can be an appropriate learning model, as payload characteristics can be maintained through layers, messages at certain locations can be highlighted through maintaining spatial information, and these are reinforced through the pooling layer.

4 Preprocessing

To use machine learning, packet payloads must be tailored to the input type of machine learning models. This process is called preprocessing. Machine learning requires extracting the required data from the collected network data (i.e., PCAP in our dataset) and matching the input length, but packet payloads have various lengths. To handle it, the SCAPY library can be utilised to extract necessary data by reading PCAPs or to transform them into text and import the data separately. And it is necessary to change packets of different lengths to a fixed length.

Figure 3 CNN layer (see online version for colours)



For example, the packet payload data have a length of P_{len} and the input data format of the model is x, then the input size should satisfy $x = 1 \times P_{len}$. However, payload length P_{len} varies and therefore cannot always be matched to the size of x.

4.1 *Basic preprocessing methods*

In this study, we propose three methods, histogram-based, padding-based, and filter-based preprocessing methods to maintain a constant length of input.

4.1.1 *Histogram-based preprocessing*

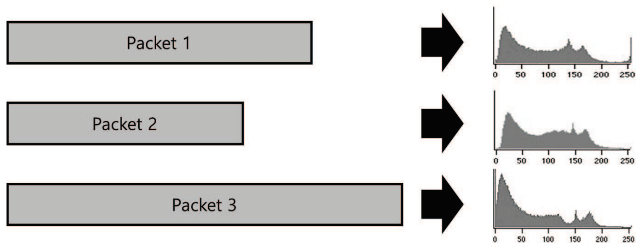
Histogram-based preprocessing uses the distribution of packet sizes and has been used as one of the common methods in existing studies that utilise payloads. It uses byte frequency information of packet payloads, thus creating a byte histogram that has an index, for example from 0 to 255, and a data length fixed to 256 bytes. Therefore, regardless of the payload length, it is possible to make a fixed input size. Figure 4 expresses an example of histogram-based preprocessing where the HEX value of packet (i.e., payload) is accumulated to obtain a feature of the model.

For the example in Figure 4, the procedure of this preprocessing method is illustrated as follows:

- obtain the frequency of byte values of packet payloads and store them in 256 length array
- then convert an array of 256 lengths containing frequencies of byte values to the input size of 16×16 for the model.

Because the generated histogram only shows the byte frequency, it causes a problem that some position information of a certain payload that could be useful for protocol classification is ignored. To solve this problem, we devise padding-based and filter-based preprocessing.

Figure 4 Histogram-based preprocessing: histogram generation



4.1.2 *Padding-based preprocessing*

Padding-based preprocessing identifies the maximum length of data and determines the input size of the model to match the maximum length. For some data, if the size is shorter than the maximum length, the rest payload is padded with zeros. For instance the TCP protocol with the maximum segment size (MSS), the maximum length cannot be bigger than 1500 bytes, so the length of input data is set to 1500 where the part other than of the real payload, [maximum length - real packet length], is padded with zeros. In this paper, we adopted the maximum length of 1500 based on our observation of the dataset.

Figure 5 Padding-based preprocessing: basic padding and middle padding

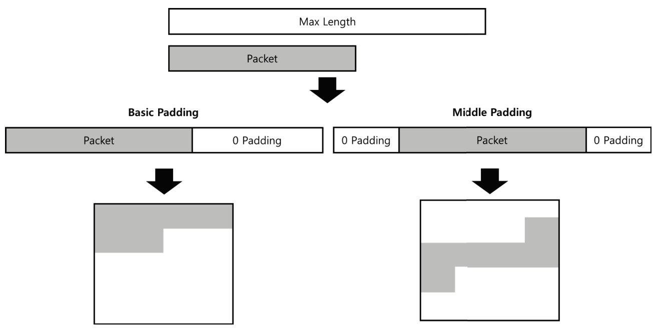


Figure 5 shows how data is converted through padding-based preprocessing. The basic padding method begins to populate the packet data from index 0 and the rest of the padding values are filled with zeros. Using the basic padding will result in the actual portion of the data being pushed to the top left, which may affect the performance of learning. We also devise a middle padding method where the actual data can be centred by placing padding on both sides so that the actual data can be centred. The procedure of padding-based preprocessing is illustrated as follows:

- Add zeros for [maximum length - real payload length]. For basic padding, add zeros from the end of the packet payload. For middle padding, fill zeros for the part except the centred payload.
- Change the byte information so that the converted data has a form of $x \times y$ for the model input.

If most payloads have small lengths compared to the maximum length, this preprocessing may not give good input features for learning a model.

4.1.3 Filter-based preprocessing

Filter-based preprocessing is intended to yield a fixed length without padding, by reflecting the byte location information of the packet payload. If the filter length is N, the payload that is shorter than N is increased to accommodate to the size N and the payload that is longer than N is decreased to accommodate to the size N.

Figure 6 shows the process of converting packets of length 3 to length 4. The minimum common multiple of 3 and 4 is 12 and thus a 12-length array is created where the number of bytes to be copied is 4. When creating filter bytes, the number of bytes to select is 3, so 3 bytes are used in a 12-length array. A total of four bundles are then selected, and then each bundle is averaged. Finally, a filter byte array of four lengths is produced. The procedure of filter-based preprocessing is illustrated as follows:

1 Find the minimum common multiple of the filter length and the length of the current packet payload.

2 Create a byte array by copying each byte of $\frac{minimum\ common\ multiple}{current\ packet\ \ length}$ to current packet payload.

3 Copy each byte of the current packet payload during $\frac{minimum\ common\ mutiple}{current\ packet\ \ length}$ to create a byte array.

4 Select $\frac{minimum\ common\ mutiple}{Filter\ length}$ bytes in the copied byte array and average them to create filter bytes.

5 Repeat step 3 to create the remaining bytes.

6 Convert the bytes so that the filter byte has $x \times y$ for the model input.

The above process allows the creation of filter bytes to be based on byte location information or actual byte values. Filter bytes 1 and 2 differ from the original value as it is mixed but are based on the byte value at that location. It is possible to reflect as much information as possible in the process of generating data that fits the filter length.

4.1.4 Example

We now describe an example of the three preprocessing methods for the same data in Figure 7 where image processing is expressed in monochrome for the data size of 16×16, 40×40, and 28×28. Because the histogram shows the frequency, (0,0) on the coordinate is the frequency of the value 0, and (16, 16) is the frequency of the value 256. The bright part represents the byte value that appears frequently and the black part represents few or no byte value. Image coordinates for padding and filter methods represent payload indexes. The brightness varies according to the byte value. For padding-based preprocessing, we can see some patterns appearing repeatedly at regular intervals, and the bottom part appears black due to padding. For filter-based preprocessing, the coordinates are not fully matched to the index but are expressed at a certain rate and repetition patterns are observed, similar to padding-based preprocessing. The locality of byte values can be utilised because regular patterns are repeated when data are imaged by considering the byte position of the payload.

Figure 6 Filter-based preprocessing

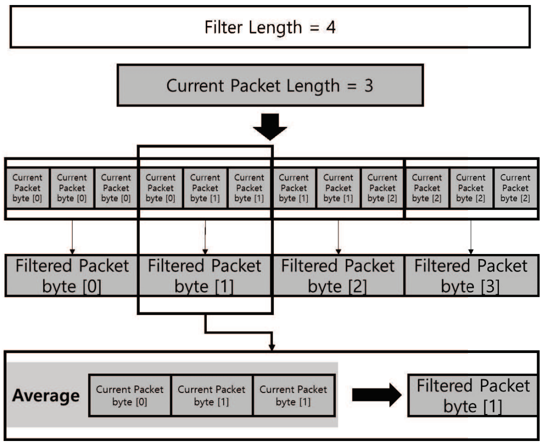
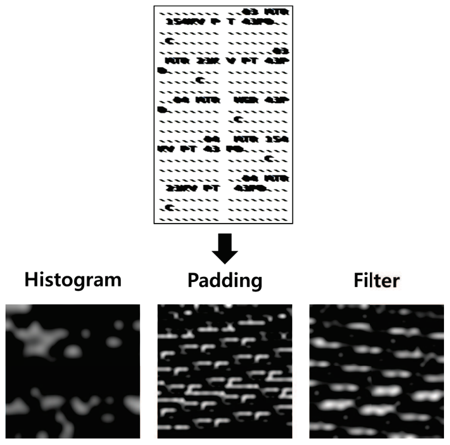


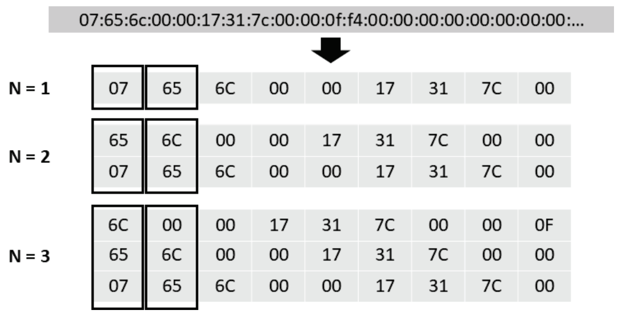
Figure 7 Images as the result of preprocessing of a payload



4.2 Extension of preprocessing using N-gram

N-gram is a method of extracting N consecutive elements from a string. This method can be further applied to data generated through the preprocess described earlier. For example, for a result of padding preprocessing, N-gram generates input data that considers N bytes together as shown in the example in Figure 8. When $N = 1$, its result is the same as the original data; thus, N is greater than or equal to 2 for this method. The larger N is, the more information can be provided by the form of data that can take into account the back-to-back relationship. If N is large, the input data is also large, so it cannot be expanded indefinitely, and unnecessary information can be bundled together. Therefore, proper sizing, i.e., finding suitable N, is required. In the example of Figure 8, when N is 1, the byte information is used individually, but when $N = 2$, data are coupled as 07-65, 65-6C, 6C-00, and so on. Such consecutive data are used effectively in the process of feature extraction and search.

Figure 8 Example of applying N-gram packet sequence


07:65:6c:00:00:17:31:7c:00:00:0f:f4:00:00:00:00:00:00:00:00:...

N = 1	07	65	6C	00	00	17	31	7C	00
N = 2	65	6C	00	00	17	31	7C	00	00
	07	65	6C	00	00	17	31	7C	00
N = 3	6C	00	00	17	31	7C	00	00	0F
	65	6C	00	00	17	31	7C	00	00
	07	65	6C	00	00	17	31	7C	00

5 CNN-based anomaly detection model

For anomaly detection by using CNN, we propose two detection models according to the type of input data: single-packet detection and sequence detection models. The single packet detection model detects abnormalities in payloads in a single packet unit and the sequence detection model detects abnormal sequences of packets. Here we also explain how to generate abnormal data so that learning can be done for each detection model.

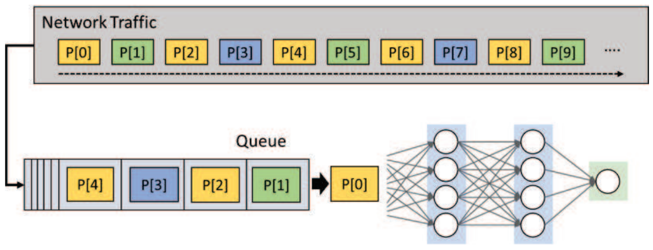
5.1 Single packet detection model

The procedure of learning the payload of each packet to generate the model is shown in Figure 9. The detection model operates in units of a packet and performs a process to classify whether the payload of a packet is normal or abnormal. The detection model attempts to detect when the payload deviates from normal rules.

Injection attacks attempt to attack by sending or modifying specific commands. The payload produced by the attack will not be a command previously used or will exhibit a form of data that is out of the rule set. If a model is created by learning the normal rules through machine learning, a category for normal data is created. Therefore, if this category is well designed, the model itself can be effectively classified for abnormalities without a particular design.

The layer of the single packet detection model consisted of $INPUT \rightarrow [CONV + RELU + POOL] \times\ 3 \rightarrow FC$. The input layer depends on the pre-treatment method. The padding method is $40 \times 40 \times 1$ and the histogram is $16 \times 16 \times 1$. Although the filter can be resized, it is set to $40 \times 40 \times 1$ to configure the same environment when comparing padding method and performance. The $[CONV + RELU + POOL]$ layer consists of three layers and consists of $20 \times 20 \times 32$, $10 \times 10 \times 64$, $5 \times 5 \times 128$. For $POOL$, the mean value was used. FC is 200 in size and its final output is two-class, normal and abnormal.

Figure 9 The single packet detection model (see online version for colours)



The window size needs to be set to match the traffic environment, so it is needed to check the size of window properly. If the window size is too small for a message, it will be difficult to design a delicate model. If there is a pattern, a sufficient set of packets of that pattern should be used for learning. However, as only a fraction of time is usable in practice, a model different from the actual pattern may be created. Conversely, if the model has a large window, it can be used to learn patterns to the extent that they are not correct. And because the amount of learning data is so large, it can take a long time to learn or cause memory problems.

While analysing the traffic, we merge three or four data packets to a bundle and inspect patterns of packets including two signal packets which tell beginning and ending of the bundle. So the window size varies from five to six that contains a bundle as well as the signal packets. For windows size 4, signal packets are difficult to contain properly and for size 6 it takes a long time to process, but they show performance similar to size 5. Therefore, we select window size of 5 for experimentation.

The sequence detection model uses five packets as a window and 40*40*1 packets in sequential order to form $40 \times\ 40 \times\ 5$ and uses them as input. Therefore, the INPUT size is $40 \times 40 \times\ 5$. $[CONV + RELU + POOL]$ and FC layers were used in the same structure as the single packet detection model.

6 Experimental results

We verified the single packet and sequence detection models through our experimentation. The single-packet detection model tests for normal and abnormal classification performance based on the abnormal rate. Further, the N-gram method is applied to evaluate its effect on performance. The sequence detection model tests for normal and abnormal classification performance according to the method of abnormal generation.

One of the keys to using the machine learning model is to estimate the performance of the model that responds to the new dataset, which is verified via cross-validation. Network traffic is sorted in the form of PCAP files in chronological order. Suppose test sets 1, 2, and 3, each represented by PCAP1, PCAP2, and PCAP3, appear in this order. One of the test sets is used as train data, while the other ones are used as test data, and this is repeated by choosing one test set as train data, as illustrated in Figure 10. Tests at different intervals increase the reliability of the statistics, showing that they can operate sufficiently over the entire interval, not just at specific intervals. We need to choose an IP pair link with sufficient data to proceed with the experiment. So we looked at the IP pair link information, the number of packets, and the average payload length contained in one PCAP.

Figure 10 Cross-validation using PCAP dataset (see online version for colours)

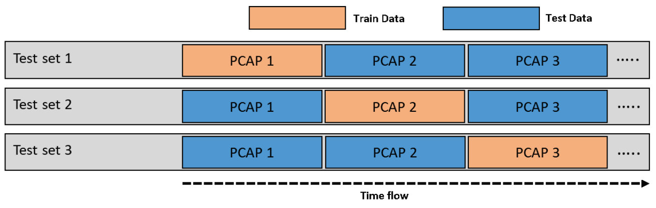


Table 2 shows part of communication links, each with the number of transmitted packets and their average length. Some links show more than 100,000 transmitted packets while some other links do 100 packets only. Also, the average packet length of some links is more than 100, while that of other links is close to zero. Under these various conditions, we choose some links that have enough data for experiments. Since the single packet detection uses payload data, their packet length should be long enough. Therefore, we choose 25 IP pair links that have more than 10,000 packets with non-zero packet length.

We chose different locations for the previously described traffic information, sites, or collection periods. The information shown in the table for the selected location is extracted and the data selection criteria are established and used for learning or testing. Because the feature of the whole traffic did not change significantly, a similar result could be obtained from the use of data from different locations as a whole was chosen and learning or testing was repeated.

6.1 Generating abnormal ICS traffic based on normal traffic

Since traffic was collected from ICS in actual normal operation, abnormal traffic cannot be acquired. As a result, abnormal traffic generation was required based on normal traffic collected. Abnormal traffic performed two tasks: creation of packet content and creation of traffic flow.

6.1.1 Abnormal single packet

This study generates data that expresses abnormalities and explains how to generate abnormal data. We tried to express an injection track that was closely related to payloads, and to do so we tried to transform a certain part of the payload. For this method, we randomly change a certain ratio of the packet payload to generate the data for the test and the results show a modified message, a modified command, and a message that was out of the rules.

Figure 11 shows an example of abnormal data that has altered part of the normal data payload. This abnormal data is exploited so that the model can learn and operate properly. Continuing learning will create a baseline for normal data in the model. Generated baseline allows the detection model to classify whether or not the payload of a packet is a normal pattern.

Table 2 The number of packets and average packet length in each link

IP-IP	*Number of packet*	*Average packet length*
47.91.57.10-47.75.57.15	16,014	0.82
47.75.57.235-47.75.57.15	164,307	503.74
47.75.57.11-47.75.57.15	282,911	103.35
47.75.57.10-47.75.57.15	1,335	6.40
47.75.57.174-47.75.57.15	2,668	16.00
47.75.57.187-47.75.57.15	15,122	7.96
47.75.57.166-47.75.57.15	1,030	42.71
47.75.57.106-47.75.57.15	4,107	619.60
47.93.57.15-47.75.57.15	15,347	0.86
47.73.57.15-47.75.57.15	9,324	0.70
47.87.57.15-47.75.57.15	5,995	1.08
47.95.57.10-47.75.57.15	20,046	0.66
47.89.57.10-47.75.57.15	12,918	0.78
29.209.235.55-47.75.57.15	15,180	150.86
47.21.57.15-47.75.57.15	705	218.58
29.21.255.225-47.75.57.15	7,633	5.35
47.21.57.179-47.75.57.15	123	32.00
29.209.235.207-47.75.57.15	917	4.98
29.21.255.222-47.75.57.15	7,359	5.99
47.75.57.135-47.75.57.15	178	5.00
47.75.57.130-47.75.57.15	178	5.00
47.75.57.131-47.75.57.15	178	5.00
47.75.57.175-47.75.57.15	421	199.83
47.75.57.171-47.75.57.15	120	21.00

6.1.2 Abnormal packet sequence

The sequence detection model also requires abnormal test data to evaluate the learned model, similar to the single packet detection model. Unlike a single packet, abnormal test data is generated in several sequence variations: packet sequence shuffles regardless of a window, packet sequence shuffle within a window, packet replacement with an in-window packet, and packet replacement with an out-of-window. In Figure 12, the number N in $P[N]$ denotes the ordered packet sequence and the packet colour a characteristic, which means that the same coloured packets have similar features.

Figure 12 shows 4 types that we generate abnormal packet sequences based on normal traffic for performance testing. Type 1 of Figure 12 shuffles packets randomly across the entire PCAP to generate abnormal traffic, thus generating an entirely random order. Then, this variation creates significant changes to the original sequence.

Figure 11 An example of a generated abnormal payload for evaluation of the single packet detection model (see online version for colours)

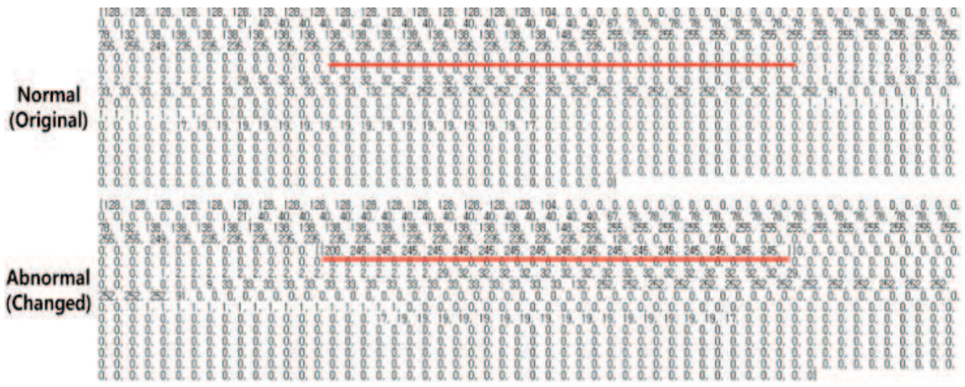
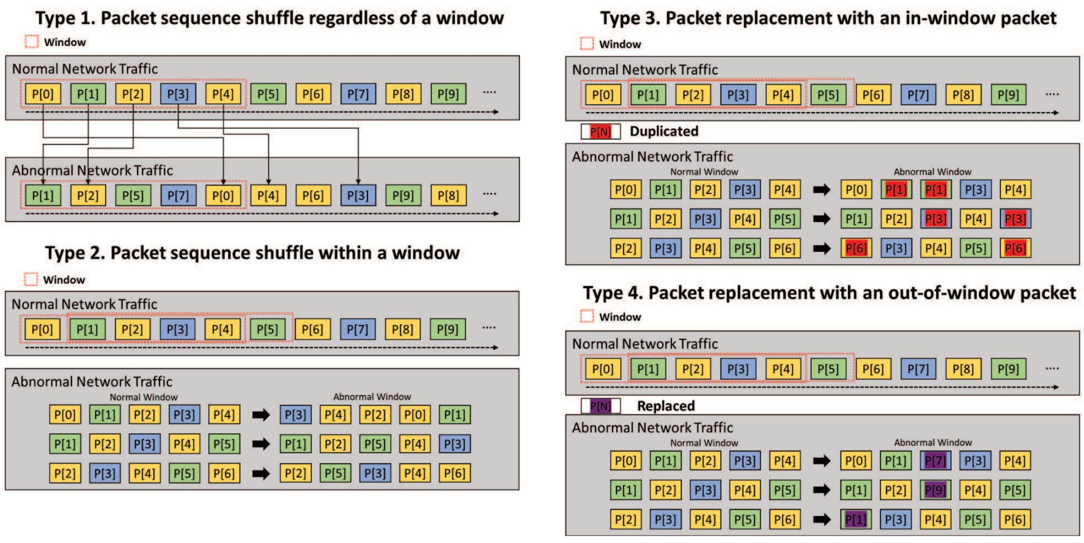


Figure 12 Four methods to generate abnormal packet sequences for evaluation of the sequence detection model (see online version for colours)


Type 1. Packet sequence shuffle regardless of a window
Window
Normal Network Traffic
Abnormal Network Traffic
Type 2. Packet sequence shuffle within a window
Normal Window
Abnormal Window
Type 3. Packet replacement with an in-window packet
Duplicated
Type 4. Packet replacement with an out-of-window packet
Replaced

Type 2 of Figure 12 also shuffles packets within a window, which produces an abnormal sequence that is likely to look similar to the normal sequence but does not exist in reality. Suppose there is a certain sequence of packets in each communication segment. The sequence can be broken if someone interrupts and injects another packet while sending a message. This attack can be detected by using this method. The model can also be used as a traffic check to see if the sequence in the window can distinguish the changed traffic.

Type 3 of Figure 12 replaces a packet with another redundant packet within a window to mimic a replay attack. It selects one packet in a window and replaces that packet at a different packet location by duplicating some packets within a window. This expresses the replay attack that sends a normal packet at a different time. It can be used to check whether or not a sequence can detect changed traffic.

Type 4 of Figure 12 also considers packet replacement where a packet is replaced with another packet regardless of the window. This case can create more serious threats because the number of different types is more diverse than the packet replacement within a window. It can be used as a traffic check to detect if a sequence is changed due to a replaced packet at a different time point.

6.2 *Performance of single packet detection model*

We evaluated the performance of models according to each preprocessing method and then evaluated their application to N-gram.

6.2.1 *Comparison of basic preprocessing methods*

We first compared the performance of the three preprocessing methods under the same condition. A total of 25 IP sections were tested individually. The biggest difference among padding, filter, and histogram-based preprocessing is whether the original is retained to take into account the position information of the byte values.

Figure 13 shows the results of the comparison. The accuracy of the padding and filter methods is between 94% and 95%, while the histogram method shows the accuracy of 85%. This gap comes from byte position information in the padding and filter methods.

Table 3 shows the result of each IP pair link. Padding and filter-based preprocessing methods show similar performance in most cases, and the histogram-based one shows a large difference in normal detection, although they have similar detection ratios in abnormal detection. Darker background colour represents data with relatively higher accuracy. The padding and filter-based method for indices 19, 20, and 21 show near 99% of accuracy in normal classification, but the histogram-based method show only 30% of accuracy. The data in indices 19, 20, and 21 have a byte value at a specific location. Therefore, it is confirmed that location information could give a significant impact on classification performance.

Figure 13 The performance of the single packet detection model with the three preprocessing methods (see online version for colours)

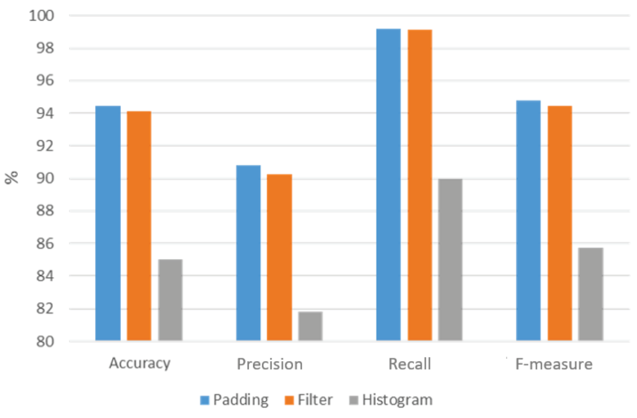
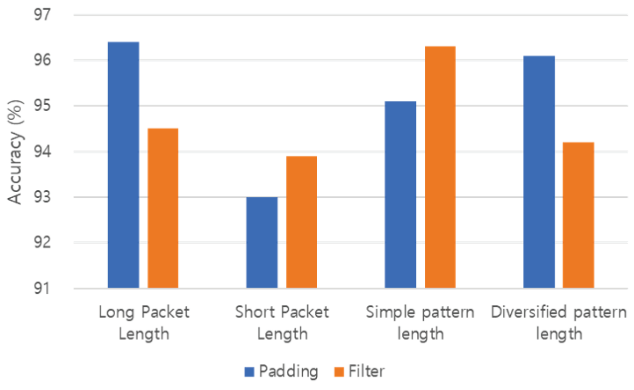


Table 3 The result of the single packet detection model for each IP pair link (see online version for colours)

	Detection Accuracy					
	Padding		Filter		Histogram	
Index	Abnormal	Normal	Abnormal	Normal	Abnormal	Normal
1	0.90	0.97	0.92	1.00	0.91	1.00
2	0.86	1.00	0.88	1.00	0.85	0.81
3	0.79	1.00	0.83	1.00	0.71	0.99
4	0.95	0.99	0.98	0.98	0.87	0.95
5	0.99	1.00	0.98	1.00	0.98	1.00
6	1.00	1.00	0.99	1.00	0.99	1.00
7	1.00	1.00	1.00	1.00	1.00	1.00
8	0.96	1.00	0.96	0.99	0.80	0.70
9	0.95	1.00	0.91	1.00	0.94	1.00
10	0.89	1.00	0.86	1.00	0.82	0.81
11	0.91	1.00	0.91	1.00	0.87	1.00
12	0.84	1.00	0.87	1.00	0.83	0.84
13	0.92	1.00	0.92	1.00	0.99	1.00
14	0.86	1.00	0.88	1.00	0.78	0.80
15	0.82	1.00	0.83	1.00	0.56	1.00
16	0.28	0.50	0.08	0.47	0.12	0.36
17	0.31	0.50	0.08	0.49	0.07	0.45
18	0.82	1.00	0.82	0.99	0.71	0.99
19	0.94	1.00	0.95	0.96	0.73	0.28
20	0.89	0.97	0.93	0.97	0.84	0.28
21	0.88	1.00	0.86	0.98	0.85	0.32
22	0.92	1.00	0.91	0.95	0.90	0.90
23	0.11	0.44	0.13	0.44	0.25	0.34
24	0.77	0.90	0.81	0.90	0.56	0.98
25	0.04	0.44	0.06	0.48	0.10	0.46

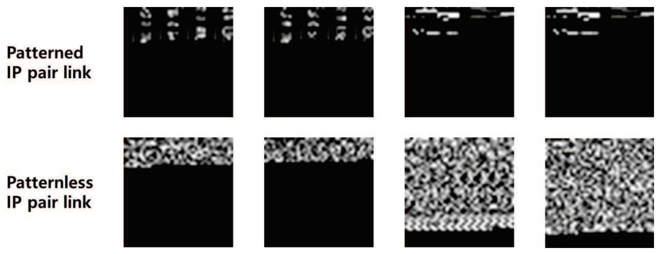
Padding and filter-based preprocessing show similar performance on average, but some differences are witnessed. First, there is a difference in the packet length. The average length for best accuracy is 237 and 127, each for padding and filter-based preprocessing, respectively. Second, communication patterns are different. Padding-based preprocessing shows better results for data of different lengths and filter-based one does better results for data of similar length. Figure 14 shows a comparison between padding and filter methods. Information about the variation in packet length appears in the pattern and average packet length can be used to determine how to preprocess. In comparison with average packet length, packets that are large than average or short than average, are denoted as long packet and short packet respectively.

Figure 14 Accuracy comparison of the single packet detection between padding and filter preprocessing (see online version for colours)



We also witnessed low accuracy in some IP pair links, e.g., 16, 17, 23, and 25, for all three methods, where no features are found. The results of imaged packets with no pattern are shown in Figure 15. It is difficult to find any feature in this case and it is confirmed that the actual length or message of bytes is diverse and does not have any features. This seems to be the result of file data, the transmission of general network traffic, which has nothing to do with typical ICS traffic.

Figure 15 Examples of payload-preprocessing results



6.2.2 *Result of N-gram extension*

The test results of various communication segments have shown that preprocessing can work effectively. Further, we evaluated the impact of N-gram on padding and filter methods that maintain the localisation information of bytes. The same experimental data as before are used for all N values. That is, the other conditions are all the same, but N values are different.

Figure 16 shows the average accuracy when N-gram is used with padding-based preprocessing for the data from test results obtained in different sections. Although the results vary for each communication section, the average accuracy increases as N increases. However, when N is 5, its performance seems to be worse than the case $N = 4$. This is because unnecessary data has been considered, thus causing interference. The results of recall, precision, and f-measure when N-gram is applied to padding-based preprocessing. In order to further refine, each result is also divided to normal (true) and abnormal (false). In general, recall and precision are inversely proportional, showing opposite trends for true and false cases. As for normal cases, the accuracy is already high regardless of N. In contrast, improvements are more clearly seen as N increases in abnormal cases. For normal cases, sufficient accuracy is achieved regardless of N. Therefore, the results are displayed with a focus on the abnormal detection accuracy according to N. As the average result shows in the preceding figures, $N > 1$ generally shows higher accuracy than $N = 1$.

Figure 16 The impact of N-gram on the padding preprocessing for performance of the single packet detection (see online version for colours)

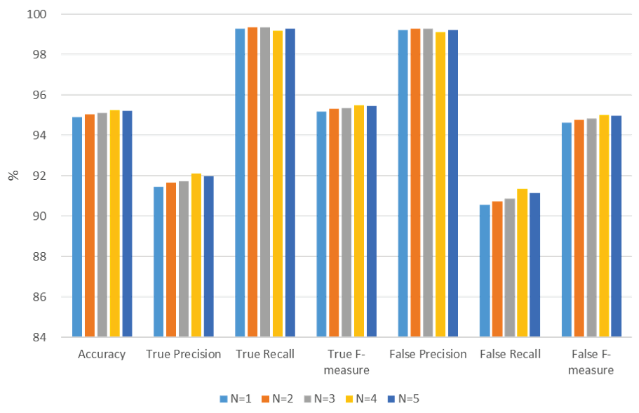
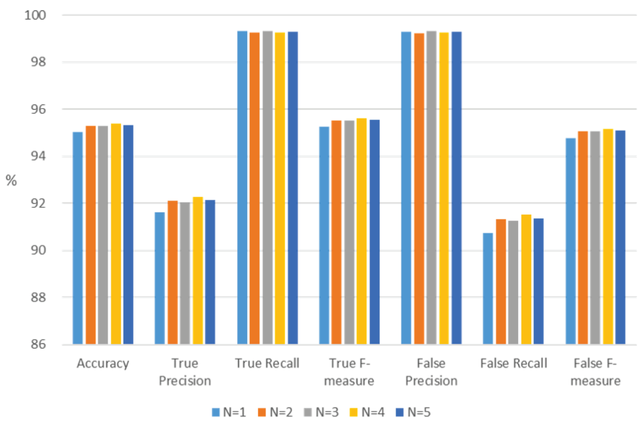


Figure 17 The impact of N-gram on the filter preprocessing for performance of the single packet detection (see online version for colours)



We conducted an experiment by applying N-gram to the filter method in the same way as we did with the padding method. Figure 17 show the impact of N-gram on filter preprocessing. Like the accuracy in padding-based preprocessing, the average accuracy keeps increasing from $N = 1$ to $N = 4$ but reduces when $N = 5$. The other performances are also similar to those of padding preprocessing.

The results thus far show that $N > 1$ is always better than $N = 1$. The case, $N = 4$, shows the highest accuracy. For data where location information of payload is meaningful and that affect other values, N-gram is very effective on the results. Although this study found that the case, $N = 4$, achieves the highest accuracy, the result would be different if more complex data or learning from an in-depth model is needed. In general, as N increases, the size of the data that is generated increases, so more memory, time, and resources are consumed. It is necessary to specify the size, N, that is appropriate for the learning environment.

6.3 *Performance of sequence detection model*

Sequence learning uses a sequence of normal network traffic. To generate sequence data, use the padding method to attach packets in order. This process produces sequence input data that combines packets. We experimented with four types of abnormal sequences. For normal traffic, a constant sequence, regardless of the time point, would be well classified. In case of abnormal traffic, each of the abnormal types is tested. Because each abnormal type has different sequence characteristics, we were able to test them by type and check information about vulnerable patterns based on accuracy to determine which features are well detected. Based on these results, a set of criteria for organising learning data will be presented to enable classification of specific patterns.

Figure 18 shows nearly 99% of accuracy for normal cases. In other words, there is regularity and persistence in the normal sequence. These characteristics can be used to easily generate criteria for normal sequences in the model. The average accuracy of the abnormal case is about 90%, which varies depending on the type of sequence abnormalities. Types 1 and 2 represent packet sequence shuffle, each in the full range and the window range, respectively. Because both types are in a different order, there are many differences between normal and abnormal sequences; therefore, their model has an average accuracy of 96% in both types. Types 3 and 4 replace a particular packet with a different packet, each within a window and out of a window, respectively. These two types modify the sequence for the abnormal sequence to look like normal sequence, thus yielding no significant difference in normal and abnormal sequences. Their average accuracy is around 85%. Since the model is constructed based on Type 1, the accuracy is relatively lower than Types 1 and 2.

Table 4 shows the average accuracy of sequence detection for each IP-pair link when each type of abnormal sequence is used. For normal sequences, nearly all IP-pair links show high accuracy. If patterns exist for different links, the criteria for normal sequences can be easily generated. Both types 1 and 2 show similar results, since the variation in the order completely alters the normal pattern regardless of the range. In Types 3 and 4, some IP pair links differ from each other. At links indices as 12, 13, 14, and 15, Type 3 achieves relatively low accuracy of 70% but Type 4 achieves 95%. In contrast, at links of indices 1, 2, 7, and 11, Type 3 achieves 90% and Type 4 does 80% of accuracy. These differences are found in the characteristics of the traffic. The accuracy of Type 3 is degraded if multiple communication patterns exist, and that of Type 4 is degraded if the communication pattern is monotonous. Generally, there are not enough datasets that respond to other characteristics. To obtain high accuracy for both types, the amount of dataset should be sufficient.

Figure 18 The average accuracy of the packet sequence detection (see online version for colours)

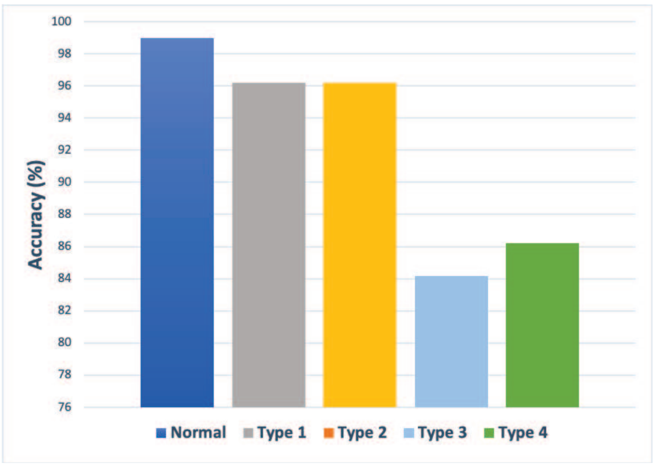
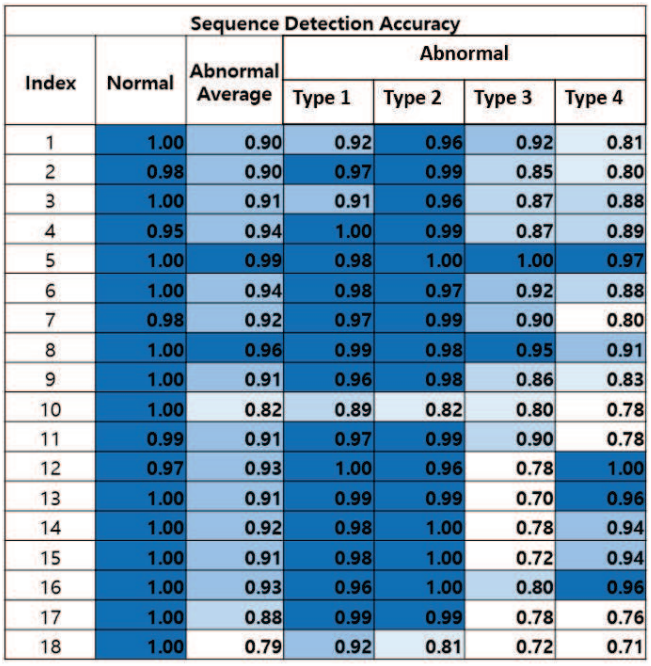


Table 4 The accuracy of the sequence detection for each IP-pair link (see online version for colours)



Sequence Detection Accuracy						
Index	Normal	Abnormal Average	Abnormal			
			Type 1	Type 2	Type 3	Type 4
1	1.00	0.90	0.92	0.96	0.92	0.81
2	0.98	0.90	0.97	0.99	0.85	0.80
3	1.00	0.91	0.91	0.96	0.87	0.88
4	0.95	0.94	1.00	0.99	0.87	0.89
5	1.00	0.99	0.98	1.00	1.00	0.97
6	1.00	0.94	0.98	0.97	0.92	0.88
7	0.98	0.92	0.97	0.99	0.90	0.80
8	1.00	0.96	0.99	0.98	0.95	0.91
9	1.00	0.91	0.96	0.98	0.86	0.83
10	1.00	0.82	0.89	0.82	0.80	0.78
11	0.99	0.91	0.97	0.99	0.90	0.78
12	0.97	0.93	1.00	0.96	0.78	1.00
13	1.00	0.91	0.99	0.99	0.70	0.96
14	1.00	0.92	0.98	1.00	0.78	0.94
15	1.00	0.91	0.98	1.00	0.72	0.94
16	1.00	0.93	0.96	1.00	0.80	0.96
17	1.00	0.88	0.99	0.99	0.78	0.76
18	1.00	0.79	0.92	0.81	0.72	0.71

7 Conclusion

In this study, we address payload-based detection against payload content and packet flow conversion attacks. To do so, we choose the CNN model and propose basic preprocessing methods such as padding, filter, and histogram to utilise the payload of packets. We also propose to use N-gram to improve preprocessing methods. We then develop two models, single packet detection and sequence detection. Verification is conducted through n-cross validation to increase the reliability of performance verification.

Because single packet detection only considers payloads, it was not possible to detect attacks effectively when using normal packets such as replay deck. To complement this, the sequence detection model considers a packet sequence. To address the problem of learning and testing due to the absence of abnormal traffic, a method of generating abnormal traffic for each situation was devised, which created abnormal traffic for each model situation.

The experimental results showed performance comparisons for each preprocessing method. The padding and filter methods maintain their original form to the full, thereby enhancing the accuracy, compared to the histogram method. The basic preprocessing method was then applied with N-gram, an extended preprocessing method, to improve the performance. We generated a model for N values from 1 through 5, and found that the case, $N = 4$, achieves the highest performance. Therefore, applying N-gram is verified to enhance performance.

The sequence detection model showed 99% of accuracy for normal sequences and 90% of accuracy for abnormal sequences. Based on the results of the experiment, we conclude that the sequence detection model could work effectively. Because both a single packet detection model and a sequence detection model have shown high accuracy, it can be an effective detection method if each situation is judged while operating together.

The methods proposed in this study can easily build detection models without requiring special analysis and design of traffic while achieving high classification accuracy. And we expect to overcome the relatively insufficient abnormalities by strengthening the layer of a model or using more data. As future work, we will study more specific abnormal situations and try to figure out how much performance can be improved by increasing the layer of a model or data.

Acknowledgement

This research was supported by Korea Electric Power Corporation [Grant number: 18A-013].

References

Adepu, S. and Mathur, A. (2016) 'Generalized attacker and attack models for cyber physical systems', *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, June, IEEE, Atlanta, GA, USA, Vol. 1, pp.283–292.

Bhuyan, M.H., Bhattacharyya, D.K. and Kalita, J.K. (2014) 'Network anomaly detection: methods, systems and tools', *IEEE Communications Surveys and Tutorials*, Vol. 16, No. 1, pp.303–336.

Brändle, M. and Naedele, M. (2008) 'Security for process control systems: an overview', *IEEE Security and Privacy Magazine*, Vol. 6, No. 6, pp.24–29.

Cheminod, M., Durante, L. and Valenzano, A. (2013) 'Review of Security Issues in Industrial Networks', *IEEE Transactions on Industrial Informatics*, Vol. 9, No. 1, pp.277–293, 2013.

Denning, D.E. and Neumann, P.G. (1985) *Requirements and Model for IDES - A Real-Time Intrusion Detection System*, Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA.

Garcí-a-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G. and Vázquez, E. (2009) 'Anomaly-based network intrusion detection: techniques, systems and challenges', *Computers and Security*, Vol. 28, Nos. 1–2, pp.18–28.

Hareesh, I., Prasanna, S., Vijayalakshmi, M. and Shalinie, S.M. (2011) 'Anomaly detection system based on analysis of packet header and payload histograms', *2011 International Conference on Recent Trends in Information Technology (ICRTIT)*, June, IEEE, Chennai, India, pp.412–416.

Ingre, B. and Yadav, A. (2015) 'Performance analysis of NSL-KDD dataset using ANN', *2015 International Conference on Signal Processing and Communication Engineering Systems*, January, IEEE, Guntur, India, pp.92–96.

Javaid, A., Niyaz, Q., Sun, W. and Alam, M. (2016) 'A deep learning approach for network intrusion detection system', *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, December, NY, USA, pp.21–26.

Khandpur, R.P., Ji, T., Jan, S., Wang, G., Lu, C.T. and Ramakrishnan, N. (2017) 'Crowdsourcing cybersecurity: cyber attack detection using social media', *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, November, Singapore, pp.1049–1057.

Kind, A., Stoecklin, M. and Dimitropoulos, X. (2009) 'Histogram-based traffic anomaly detection', *IEEE Transactions on Network and Service Management*, Vol. 6, No. 2, pp.110–121.

Lin, W-C., Ke, S-W. and Tsai, C-F. (2015) 'CANN: An intrusion detection system based on combining cluster centers and nearest neighbors', *Knowledge-Based Systems*, Vol. 78, pp.13–21.

Maglaras, L.A., Kim, K.H., Janicke, H., Ferrag, M.A., Rallis, S., Fragkou, P., Maglaras, A. and Cruz, T.J. (2018) 'Cyber security of critical infrastructures', *ICT Express*, Vol. 4, No. 1, pp.42–45.

Mahoney, M.V. and Chan, P.K. (2001) *PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic*, Dept. of Computer Science, Florida Tech, Tech. Rep. cs-2001-04.

McLaughlin, S., Konstantinou, C., Wang, X., Davi, L., Sadeghi, A.R., Maniatakos, M. and Karri, R. (2016) 'The industrial control systems cyber security landscape', *Proceedings of the IEEE*, Vol. 104, No. 5.

Nguyen, T.T. and Armitage, G. (2008) 'A survey of techniques for internet traffic classification using machine learning', *IEEE Communications Surveys and Tutorials*, Vol. 10, No. 4, pp.56–76.

Nicol, D.M., Sanders, W.H., Singh, S. and Seri, M. (2008) 'Usable global network access policy for process control systems', *IEEE Security and Privacy Magazine*, Vol. 6, No. 6, pp.30–36.

Papadogiannaki, E., Halevidis, C., Akritidis, P. and Koromilas, L. (2018) 'OTTer: A scalable high-resolution encrypted traffic identification engine', *Lecture Notes in Computer Science Research in Attacks, Intrusions, and Defenses*, pp.315–334.

Pervez, M.S. and Farid, D.M. (2014) 'Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs', *The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014)*, IEEE, Dhaka, Bangladesh, pp.1–6.

Ravale, U., Marathe, N. and Padiya, P. (2015) 'Feature selection based hybrid anomaly intrusion detection system using K means and RBF Kernel function', *Procedia Computer Science*, Vol. 45, pp.428–435.

Sekar, R., Gupta, A., Frullo, J., Shanbhag, T., Tiwari, A., Yang, H. and Zhou, S. (2002) 'Specification-based anomaly detection: a new approach for detecting network intrusions', *Proceedings of the 9th ACM Conference on Computer and Communications Security*, November, Washington DC, USA, pp.265–274.

Tavallaee, M., Bagheri, E., Lu, W. and Ghorbani, A.A. (2009) 'A detailed analysis of the KDD CUP 99 data set', *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, July, IEEE, Ottawa, ON, Canada, pp.1–6.

Tsai, C.F., Hsu, Y.F., Lin, C.Y. and Lin, W.Y (2009) 'Intrusion detection by machine learning: A review', *Expert Systems with Applications*, Vol. 36, No. 10, December, pp.11 994–12 000.

Wang, K. and Stolfo, S.J. (2004) *Anomalous Payload-Based Network Intrusion Detection*, Lecture Notes in Computer Science Recent Advances in Intrusion Detection, pp.203–222.

Wang, G., Hao, J., Ma, J. and Huang, L. (2010) 'A new approach to intrusion detection using artificial neural networks and fuzzy clustering', *Expert Systems with Applications*, Vol. 37, No. 9, pp.6225–6232.

Yamashita, R., Nishio, M., Do, R.K.G. and Togashi, K. (2018) 'Convolutional neural networks: an overview and application in radiology', *Insights into Imaging*, Vol. 9, No. 4, pp.611–629.

Yun, J., Hwang, Y., Lee, W., Ahn, H. and Kim, S. (2018) 'Statistical similarity of critical infrastructure network traffic based on nearest neighbor distances', *Lecture Notes in Computer Science Research in Attacks*, *Intrusions*, *and Defenses*, pp.577–599.

Zhang, J., Xiang, Y., Wang, Y., Zhou, W., Xiang, Y. and Guan, Y. (2013) 'Network traffic classification using correlation information', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 24, No. 1, pp.104–117.

Websites

Genomic Data Commons Data Portal, https://portal.gdc.cancer.gov/

National Biomedical Image Archive, https://imaging.nci.nih.gov/ncia/login.jsf

National Cancer Institute, https://biospecimens.cancer.gov/relatedinitiatives/overview/caBig.asp