Computers & Security 116 (2022) 102672

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Improving SSH detection model using IPA time and WGAN-GP

Junwon Lee*, Heejo Lee

Korea University, Anam-ro 141, Seongbuk-gu, Seoul 02841, Republic of Korea

ARTICLE INFO

Article history:
Received 18 September 2021
Revised 10 January 2022
Accepted 24 February 2022
Available online 26 February 2022

Keywords:
GAN
WGAN-GP
SSH detection
Inter-packet arrival time
Session-based data
Random forest
Generator loss
PCA

ABSTRACT

In the machine learning-based detection model, the detection accuracy tends to be proportional to the quantity and quality of the training dataset. The machine learning-based SSH detection model's performance is affected by the size of the training dataset and the ratio of target classes. However, in an actual network environment within a short period, it is inconvenient to collect a sufficient and diverse training dataset. Even though many training data samples are collected, it takes a lot of effort and time to prepare the training dataset through data classification. To overcome these limitations, we generate sophisticated samples using the WGAN-GP algorithm and present how to select samples by comparing generator loss. The synthetic training dataset with generated samples improves the performance of the SSH detection model. Furthermore, we add the new features to include the distinction of inter-packet arrival time. The enhanced SSH detection model decreases false positives and provides a 0.999 F_1-score by applying the synthetic dataset and the packet inter-arrival time features.

1. Introduction

Advanced persistent threats (APT) targeting major national institutions or companies and insider information leakage have recently become increasingly significant security threats. These security threats continue to transmit data inside the organization to the outside. Moreover, it is challenging to confirm the server IP address information used for hacking and information leakage using Infrastructure as a Service (IaaS), which can temporarily create and terminate servers in the public cloud, has become common.

SSH communication is a useful means of attack attempts and information leakage (SSH, 2021). For example, an attacker can remotely connect to an occupied internal PC by using SSH tunneling (Burande et al., 2014). In other cases, a hacker can infect a PC with malware that connects to an external hacking server via an SSH tunnel or similar protocol using a background process. The SSH tunnel can be abused to leak data or send additional attack commands without user recognition. According to FireEye's APT41 report, hackers use SSH communication to remotely access a PC and infect it with malicious code (Fraser, 2019). SSH access attempts have recently been found using SSH over DNS tunneling (Berg and Forsberg, 2019) to bypass network security controls. SSH communication is generally inferred based on the service port. However, as the TCP port is not dependent on the communication service, it is hard to detect SSH communication with only the TCP port information. Deep packet inspection investigating packet payload in detail (Lin et al., 2008) and machine-learning comprehensively analyzing various features can be adopted to detect SSH communication. Moreover, in the SSH detection model using packet-based data, a high-performance system is required because a number of packets should be simultaneously stored and analyzed (Dharmapurikar et al., 2003). Even though the latest technologies such as machine learning and deep learning are applied, a sufficient training dataset with diverse communication types should be prepared to provide a high detection rate and accuracy.

Our previous research (Lee and Lee, 2021) proposed an SSH detection model using machine learning but shows that the precision is significantly low due to the limited and unbalanced training dataset. The low precision problem occurs when it is difficult to refine genuine outliers (Wang et al., 2013). For the improvement of precision, a diverse and sufficient dataset is required. If outliers are tuned only by focusing on improving recall, the decrease of precision is followed. In an unbalanced dataset, the precision may be low regardless of the low false positive rate (FPR). In Alshammari et al.'s work, even though FPR is 0.017, precision is 0.128 because the class ratio of the test dataset is unbalanced. A detection model dealing with an unbalanced dataset can improve precision by modifying the target cost function, sampling, and generating artificial data (Wang et al., 2019).

In this work, we apply session-based data instead of flow-based data generally used to design the SSH detection model. Session-based data is the information that combines the inbound and outbound flow-based data in the same TCP session. Wheelus et al.

* Corresponding author.
E-mail address: junimirang@korea.ac.kr (J. Lee).

https://doi.org/10.1016/j.cose.2022.102672
0167-4048/

propose a session-based dataset (Session Aggregation for Network Traffic Analysis, SANTA) for detecting internet attacks and security breaches. They explain that session-based data can be used to analyze the context of communication that is obscured by flow-based data. Furthermore, the flow-based and packet-based data do not provide characteristics of the overall session of network communication (Wheelus et al., 2014).

It can be created by aggregating packet-based data or by combining two bound flow-based data. To compensate for limited features of session-based data, we use inter-packet arrival (IPA) time, which is useful for application prediction. We append the average and variance of IPA time obtained from packet-based data as features to session-based data. The size of session-based data is significantly reduced compared to packet-based and flow-based data.

We reviewed the generative adversarial network (GAN) algorithm, which has recently been spotlighted in the field of deep learning that deals with images, as a method to gather samples similar to the training dataset collected in an actual network environment. In 2014, the GAN algorithm, proposed by Ian Goodfellow, generates meaningful output from random noise using a generator and trains the generator using a discriminator (Goodfellow et al., 2014). GAN is actively applied in fields such as speech signal and natural language processing as research on GAN algorithm continues. Recently, the scope of GAN is expanding, studies applying the GAN algorithm to network traffic generation and abnormal traffic detection are continued (Al Olaimat et al., 2020; Lin et al., 2018; Macwan et al., 2021; Ring et al., 2019). Among them, Ring et al. describes a specific preprocessing for applying the GAN algorithm to flow-based data and a data generation method using WGAN-GP (Ring et al., 2019). WGAN-GP improves the clipping weight of GAN and optimizes the learning as a gradient penalty (Gulrajani et al., 2017). In this work, WGAN-GP generates various samples and synthesizes an enhanced training dataset. Since the generated samples with the WGAN-GP have continuous values, we apply the softmax function to adjust the discrete values such as class and evaluate the validity of generated samples. After analyzing the effect of the generator loss (G-loss) and quantity of generated samples on the SSH detection model, we add generated samples to the training dataset, which improves the performance of the SSH detection model.

Compared to our previous work, this work using the same DARPA 99 dataset provides an improvement of 11.74% recall and 63.58% precision. We describe a novel approach to improve the performance of the SSH detection model by proposing a compact session-based dataset and synthetic dataset using WGAN-GP as follows.

- We reduced the size of the training dataset by converting flow-based data into session-based data and propose a method to apply features of inter-packet arrival (IPA) time to features of the session-based data.
- We prove that the WGAN-GP algorithm effectively generates samples compared to the GAN and WGAN algorithms by considering the number of generated samples and the output of softmax function.
- We propose how to select samples by comparing generator loss to improve the detection performance of the SSH detection model.

This paper is organized in the following order. Section 2 describes the insufficient information in session-based dataset, the difficulty in gathering valid training data, and the validation of generated samples. Section 3 discusses the packet-based and flow-based data commonly used in network traffic classification and detection. Then a set of deep generative models used for sample generation is described. Section 4 covers the preprocessing of session-based dataset and how to select and apply samples generated with WGAN-GP. Section 5 measures the performance of the SSH detection model while changing the rate of generated samples, which has a different generator loss to verify Section 4. Finally, we conclude the paper with a discussion and plans for future work.

2. Problem analysis

2.1. Insufficient information in session-based dataset

The network interface divides data into packets, which have the size of a maximum transmission unit (MTU) for efficient transmission in the limited network bandwidth. In transmitting and receiving packets, the information of the packet header also reflects the properties of the application (Mahoney, 2003). However, when combining the header information of packets into a session to minimize the information size and express the communication briefly, the unique properties shown in packet-based data disappear. As a representative example, inter-packet arrival time of bidirectional packets, is an important feature that can estimate application properties (Alshammari and Zincir-Heywood, 2011; Sadasivam et al., 2016). Still, it is not easy to directly express the response time between multiple packets in session-based data.

2.2. Limitations of collecting training dataset

We presented a model for detecting SSH communication using a session-based dataset using a machine learning algorithm in our previous work (Lee and Lee, 2021). The detection model provides high recall, but the precision is relatively low. This problem is caused by false detection of Non-SSH communications such as web and SMTP as SSH communications. Sufficient training dataset are necessary to increase precision (Xu et al., 2020) because they can help to improve the detection rate of non-SSH communications while maintaining the detection rate of SSH. We expected that various training dataset would improve the detection rate of Non-SSH communication, so the internet traffic from 8 to 12, May 2006 provided by the MAWI Working Group is added to the training dataset. As expected, the increased training dataset improves the detection rate of Non-SSH communications. The precision for SSH communication is increased by 29.4%, but the recall for SSH is decreased by 1.8% in the SSH detection model using random forest. We predicted that the training data corresponding to Non-SSH communication affected the detection results of the test dataset. We estimated that the cause of the reduced recall of SSH is the period and network environment in which the MAWI training dataset was collected. In other words, to support the existing high recall and improve precision, it is necessary to gather a sufficient training dataset in the same network environment that reflects the properties of the communication environment. However, since the training dataset should be classified, accurate labeling of the data is essential. Moreover, it takes a lot of effort and time to provide a high-quality training dataset.

2.3. Synthetic dataset validation

Samples using the GAN algorithm are similar to real data, and they can replace missing data in the original data set (Kwon et al., 2019). Therefore, we tried to obtain a synthetic training dataset by applying GAN. If the output of GAN algorithm offers the class information of samples, the additional labeling work is unnecessary. However, when generating samples using the GAN algorithm, the generated sample needs improvement for mode collapse that is biased towards a specific class that is easy to learn. In addition, it is difficult to determine the validity of the generated sample because network communication data is not recognized by human perception, such as images, texts, and voices. In the detection model using a synthetic dataset, because the validity of the sample can af-

fect the detection performance, we should consider a method for verifying the validity of the sample.

3. Related work

3.1. Packet-based data & flow-based data

Packet-based data is the information obtained through packets establishing a session. Packet-based data contains important information for classifying communication. It can provide additional information by analyzing the correlation of packets and by inspecting the information included in the packet header. Satoh et al. explain that packet size and direction are important features to find the essential singularity of SSH communication (Satoh et al., 2012). Packet-based data describes the shape of transmission and reception and inter-packet arrival time during a session (Garsva et al., 2014). Sadasivam et al.'s work show that inter-packet arrival time of packet-based data is the main feature of the decision tree for classifying successful communication in SSH attacks (Sadasivam et al., 2016). Tcpdump and libpcap are examples of packet-based data. The network traffic is represented in detail by packet-based data captured from the network path. However, because traffic is provided in its entirety, the dataset captured in a real-world network environment is massive. Due to the massive size of packet-based data, analyzing network communication consumes a lot of computing resources and time.

Flow-based data is widely used in network communication analysis due to the collection, storage, and analysis limitations of packet-based data. Flow-based data is data that combines communications with the same source and destination addresses into one. Flow-based data is traffic metadata that combines packet-based data with the same source and destination addresses. The size of flow-based data is reduced because a large number of packet-based data are aggregated into one. NetFlow, a representative flow-based data proposed by Cisco systems, is provided by a switch or router. Flow-based data is used as a network traffic dataset in many studies, including the work of Ring et al.

In Alshammari et al.'s work, when comparing the SSH detection models using packet-based features, the detection model using flow-based features shows a high detection rate and a low false positive rate compared to the SSH detection models using packet-based features (Alshammari and Zincir-Heywood, 2007; 2011).

3.2. Deep generative models

Maximum likelihood based deep generative models are classified by the way how to represent or approximate the likelihood. The production of explicit density divides deep generative models into explicit models and implicit models. An explicit model constructs a model that the explicit density function has a maximum likelihood. In contrast, the implicit model does not explicitly describes a probability distribution. Variational autoencoder (VAE) is a representative explicit model, and GAN is an implicit model suitable for scaling to high dimensional spaces and relatively low computational costs (Goodfellow, 2016).

3.2.1. Variational autoencoder

Variational autoencoder (VAE) is a deep generative model that provides explicit density. VAE consists of input layer, encoder, latent space, decoder, and output layer. The architecture is similar to Autoencoder (AE). A neural network is applied to encoders and decoders. An encoder using Gaussian multi-layered perceptron (MLP) and a decoder using Bernoulli MLP are applied to the VAE model (Kingma and Welling, 2013).

In AE model, an encoder generates latent variable z by dimension reduction, and a decoder constructs the same output layer with the input layer through a decoder. AE can be applied to principal component analysis (Hinton and Salakhutdinov, 2006), information retrieval (Salakhutdinov and Hinton, 2009), anomaly detection (Sakurada and Yairi, 2014). However, the purpose of VAE is to generate new data. In the latent space of VAE, the mean and variance of latent variable z are provided through the encoder, and the distribution follows a standard normal distribution. The decoder reconstructs continuous output data close to the input data by using the sampled data from the latent variable distribution. The decoder reconstructs continuous output data close to the input data by using the sampled data from the latent variable distribution (Kingma and Welling, 2013). VAE has a clear and recognized way to evaluate the quality of the model using the loss function. In contrast, in the image generation, generated samples are much more blurred than images generated from GANs (Genevay et al., 2017)

3.2.2. From GAN to WGAN-GP

In 2014, Ian Goodfellow introduced Generative Adversarial Networks (GANs). GAN is a framework for estimating generative models via an adversarial process. Yann LeCun described that GAN is "the most interesting idea in the last 10 years in Machine Learning" (Beckett, 2017). To replicate a probability distribution, GAN uses loss functions that reflect the distance between the distribution of the data generated by the GAN and the distribution of the real data. The GAN consists of a discriminator D, which represents the probability of judging a given sample as a real data, and a generator G, which generates a synthetic sample through a noise variable. To obtain realistic but non-real data, GAN plays the two-player minimax game of a discriminator network and a generative network. Eq. (1) is the objective function of GAN. In GAN's objective function, the generator is trained to minimize the value of the objective function, and the discriminator is trained to maximize the objective function. In Eq. (1), $G(z)$ means a generated sample by inputting random vector z (Goodfellow et al., 2014).

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \quad (1)$$

If the GAN's generator repeatedly generates the same output, and the next discriminator must reject the corresponding output. However, when the next discriminator stays at the local minimum, it causes a mode collapse problem with limited sample variety. Wasserstein GAN (WGAN) solves the mode collapse problem by using the distance between the probability distribution of real samples and that of generated samples instead of the discriminator-based objective function. Representative distance measures between probability distributions include Total Variation (TV), Kullback–Leibler (KL), and Jensen–Shannon (JS) distance, but they do not converge when the compared probability distributions match each other. Since Earth Mover's (EM's) distance is calculated as a continuous function even when probability distributions match, WGAN trains an optimal model using EM distance as an objective function. Eq. (2) describes the objective function of WGAN. If the optimal generator g_θ satisfies the Lipschitz condition (Eq. (3)), Critic loss (Eq. (2)) is always differentiable to obtain an optimal value. WGAN executes weight clipping to satisfy the Lipschitz condition for each batch section and forcibly adjusts the weight parameters to exist in the compact space, [-c,c] section (Arjovsky et al., 2017; Weng, 2019).

$$Critic\ Loss = \max_{w\in W} \mathbb{E}_{x\sim \mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z\sim p_z(z)}[f_w(g_\theta(z)] \quad (2)$$

$$|\, f(x_1) - f(x_2)\,| \le K\,|\, x_1 - x_2\,| \qquad (Lipschitz\ condition) \quad (3)$$

WGAN supports the stable training of GAN by using EM's distance. However, the poor sample generation and convergence failure of objective function still exist. Gulajani et al. explain that the cause of the problem is the Lipschitz constraint applied to WGAN's critic loss. They proposed WGAN-GP, which can improve clipping weight using gradient penalty. When comparing to WGAN that adjusts the weight parameters, WGAN-GP helps achieve optimal learning by adding a gradient penalty to the WGAN's loss function. Eq. (6) (Critic Loss) is an objective function of WGAN-GP. In Eq. (4), random sample $\hat{x}$ is calculated using real sample x and generated sample $\tilde{x}$. In Eq. (5), if $\hat{x}$'s EM distance slope $|| \nabla_{\hat{x}} D(\hat{x}) ||_2$ is not 1, the penalty is applied. λ is the optimal value experimentally confirmed during the training process. Generator loss (Eq. (7)) is able to describe the reality or similarity of generated samples compared to real samples. Samples with low generator loss generally provide good sample quality (so similar to real samples), but at the same time, they can cause underfitting problems. In the area of anomaly detection, underfitting samples have a negative effect on detecting additional anomaly behaviors.

$$\hat{x} = \epsilon x + (1 - \epsilon)\tilde{x} \tag{4}$$

$$Grdient\ Penalty = \lambda \mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}} \left[(\| \nabla_{\hat{x}} D(\hat{x}) \|_2 - 1)^2 \right] \tag{5}$$

$$Critic\ Loss = \mathbb{E}_{\tilde{x} \sim \mathbb{P}_g}[D(\tilde{x})] - \mathbb{E}_{x \sim \mathbb{P}_r}[D(x)] + Grdient\ Penalty \tag{6}$$

$$Generator\ Loss = -\mathbb{E}[D(G(z))] \tag{7}$$

In the study of Gulajani et al., WGAN-GP showed strong modeling performance and stability compared to DCGAN, LSGAN, and WGAN (Gulrajani et al., 2017).

4. Methodology

4.1. Overview

In this work, we use a dataset converted from packet-based data to session-based data. A deep generative model generates samples for the synthetic training dataset. First, we convert the inter-packet arrival time of the packet-based features into mean and variance to add the feature on the session-based data. Next, we generate samples using the WGAN-GP algorithm and filter samples using the softmax function and generator loss. The synthetic dataset, mixed with the original and filtered samples, is used as the training dataset. Finally, we present the performance of the random forest-based detection model. Fig. 1 is a diagram illustrating the process from data collection to SSH communication detection. This section describes session-based dataset transform, preprocessing, major feature extraction, GAN algorithm assessment, and sample selection using generator loss.

4.2. Dataset preparation

4.2.1. Session-based dataset transform

The dataset used in this paper is the 1st, 3rd, and 4th-week data of DARPA99 (LINCOLN, 1999). In the tcpdump data of DARPA99, we converted the packet collection time and TCP/IP header information into a session-based dataset. Whereas flow-based data, which is unidirectional, session-based data is bidirectional and expresses data sending and receiving as one. The size of session-based data is further reduced because it is expressed by combining two flow-based data. A session log provided by a firewall or web proxy is an example of representative session-based data. Table 1 shows the number of rows, file size and conversion time (from packet-based data) of the training dataset (Week4 of DARPA99) when it is expressed as packet-based data, flow-based data, or session-based data.

Regarding the studies of Alshammari and Zincir-Heywood (2011), Sadasivam et al. (2016), and Yu et al. (2017), we judged that inter-packet arrival (IPA) time affects the SSH detection rate. The IPA time is calculated using the time records of packets belonging to the same session. The IPA time differs from the duration of flow-based data. Duration refers to the time interval between the first packet's detection and the last packet's detection in a flow with the same bound. On the other hand, IPA time refers to the interval time of continuous packets in a session. Whenever a packet occurs in flow-based data, the difference in duration is limited to displaying only the characteristics of the transmitter or receiver. However, the IPA time can inspect the response time showing the properties of the application between the transmitter and the receiver during communication (Wheelus et al., 2014). We add the mean and variance of the IPA time to the features of the session-based dataset. Fig. 2 depicts the difference between the interval time of flow-based data and the IPA time of session-based time.

Session-based data is combined by analyzing source IP, destination IP, source port, destination port, packet size, and 3-hands shake information included in the TCP header. The process of combining sessions is as follows. First, the row corresponding to the SYN packet of the 3-hands shake is extracted. Next, after the time of the SYN packet, packets with the same IP and port are combined until a FIN packet comes out. If there is no response packet after 300 s have elapsed since the last packet, we conclude that the session is terminated and terminate the session combination. Table 2 shows the session-based data after combining packets.

We can reduce the data size for the training dataset and test dataset from 20,398,592 rows to 595,524 rows by transforming packet-based data into session-based data. It took about 101 min to transform the packet-based data into the session-based data having the feature of IPA time. (We used a workstation with the following specification: 6-core (3.59 GHz) CPU, 32 GB Memory, and Windows 10 21H1.)

4.2.2. Preprocessing

A dataset that has been transformed into a session-based dataset from one session shows only the properties of its session. For comparative analysis between similar communications, association analysis with other sessions is required. In particular, the count of source addresses connected to the same server and the access frequency are valuable indicators in estimating the attribute of communication. This session information is used to detect abnormal communication behavior in IDS/IPS (Moon et al., 2017). We define additional features as the result of comparative analysis between the characteristics of the session and similar communication. Table 3 lists the features of the dataset to be applied to the SSH detection model. In machine learning and deep learning algorithms, datasets with linear characteristics are easy to classify, so we apply a log function to some features to reduce the distribution density of the data and show the linear characteristics as much as possible. Finally, we scale the 9 features (discussed in next section) of the data set except for labels to range from 0 to 100. It should be noted that the preprocessed dataset is further scaled from 0 to 1 before training with the GAN algorithm.

4.2.3. Major feature extraction

In order to improve the detection speed and precision of the detection model, in the SSH detection model using a decision tree, features with low frequency are replaced with principal component analysis (PCA)-applied PC value. To extract the major features, we applied all features in the training dataset to the deci-

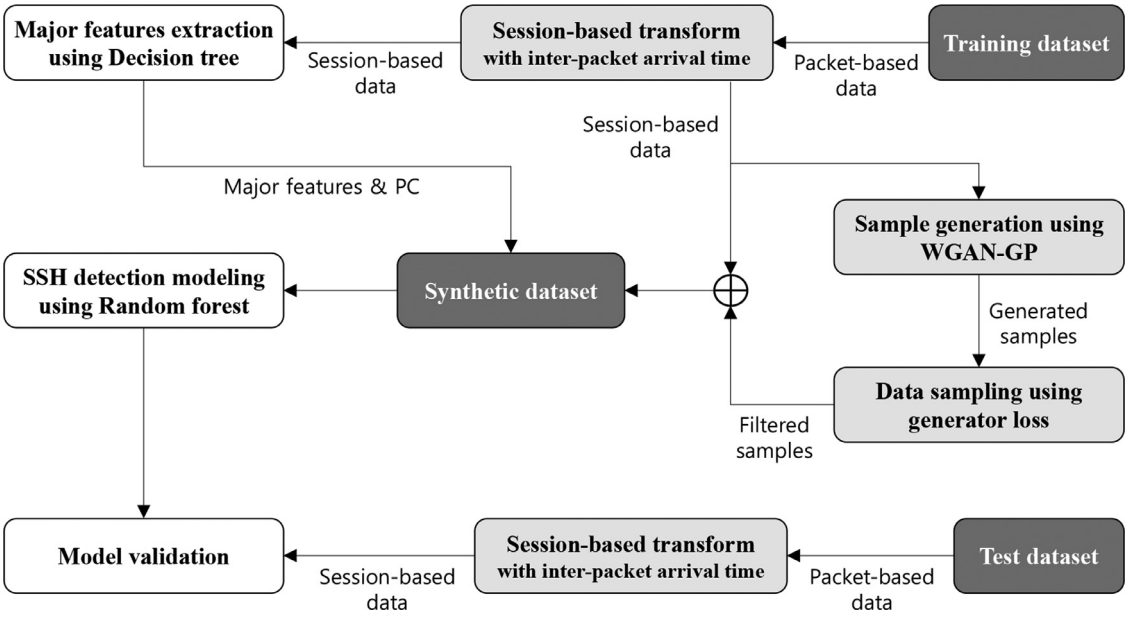

Fig. 1. Overview of the enhanced SSH detection model.

Table 1
Rows, file size, and conversion time comparisons by training dataset type (Week4 of DARPA99).

Comparison item	Session-based data	Flow-based data (NetFlow)	Packet-based data (PCAP)
Rows	175,330	348,583	6,461,795
File size	39.7MB	80.4MB	1.35GB
Conversion type	Packet→ Session	Packet→ Flow	-
Conversion time	288 sec	881 sec	-

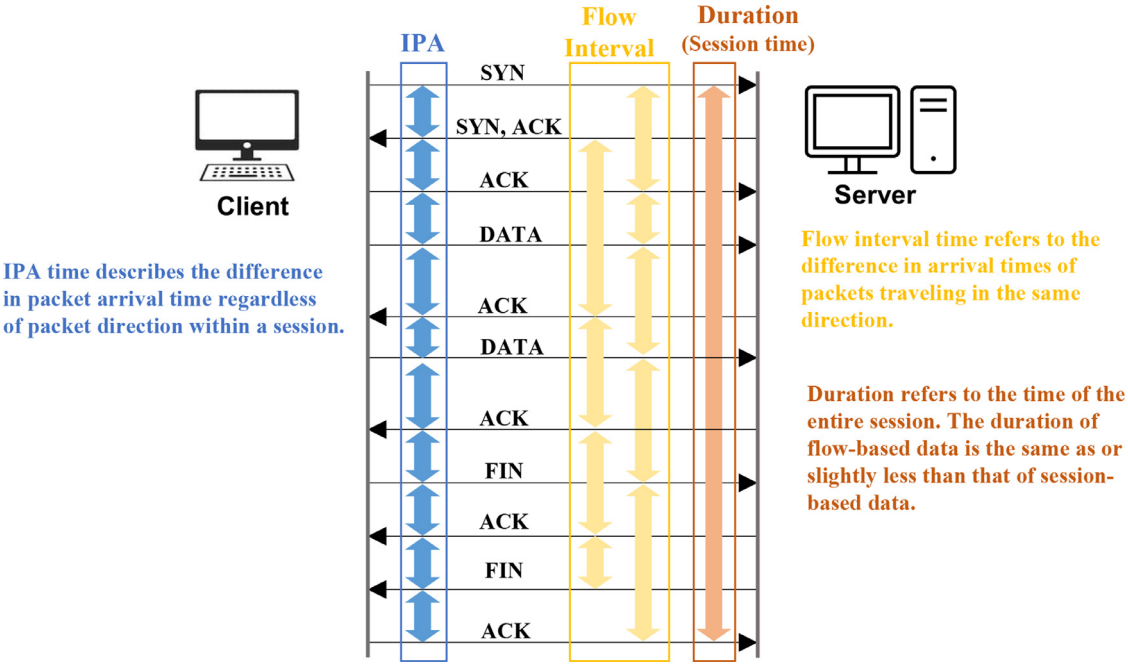

Fig. 2. IPA time, Flow interval, and duration comparison in the TCP communication: IPA time reflects the characteristics of the application by indicating the difference between the packets sent and received.

Table 2
Session-based dataset including inter-packet arrival (IPA) time information: We applied the mean and variance of IPA time as features to provide the characteristics of IPA time to session-based data.

Src. IP	Dst. IP	Dst. Port	Send Byte	Receive Byte	Session Time	Mean of IPA time	Variance of IPA time
172.16.112.194	196.37.75.158	25	1727	892	0.3092	0.0961	3.01E-03
172.16.113.105	197.182.91.233	79	300	514	0.2357	0.0425	3.46E-03
172.16.114.169	135.13.216.191	25	2158	939	0.1067	0.0221	6.24E-05

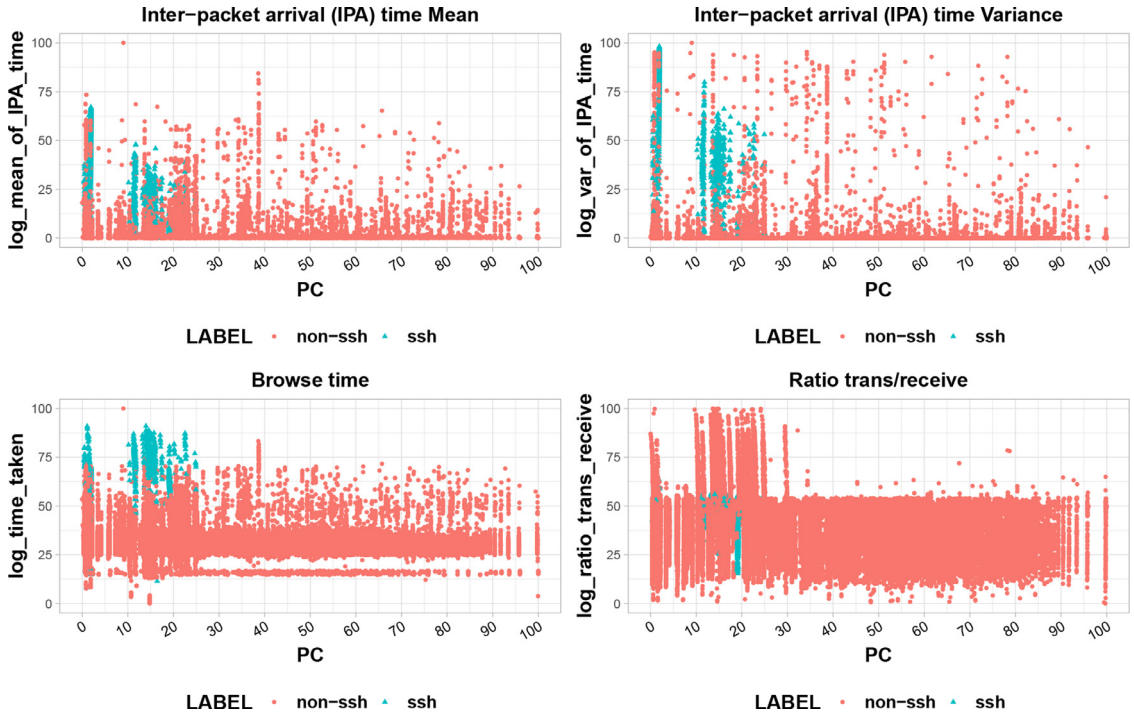

Fig. 3. Major feature analysis: The range of each major feature where 80% of SSH are located is as follows. (IPA time mean: 0∼31, IPA time variance: 0–52, Browse time: 55–82, Ratio trans/receive: 43–64).

Table 3
The feature name and description after preprocessing: After applying the log function to 9 features except for label features, the scale of feature is adjusted.

Dataset feature	Description
count_total_connect	Number of connections to the same Destination IP
count_connect_IP	Number of source IP connected to the same destination IP
count_avg_connect	Average number of connections per IP to the same destination IP
speed_transmit_BPS	Average transfer speed
byte_send	Transmit data size
ratio_trans_receive	Send byte byte/Receive byte
time_taken	Time per session
mean_of_IPA_time	Mean of inter-packet arrival (IPA) time
var_of_IPA_time	Variance of inter-packet arrival (IPA) time
label_ssh	SSH communication
label_non_ssh	Non-SSH communication

sion tree with a maximum of 5 depth and selected the features frequently used as decision nodes. *Ratio_trans_receive, time_taken, mean_of_IPA_time,* and *var_of_IPA_time* were selected as major features, and *count_total_connect, count_connect_IP,*

count_avg_connect, speed_transmit_BPS, and *byte_send* were replaced with PC value and then added to the training dataset. The inter-packet arrival time is mentioned as an essential feature in classifying communications in the related work by Alshammari et al. and Sadasivam et al. We examined whether this feature plays a significant role in classifying SSH communications when converting these features to mean and variance for use with session-based data. Fig. 3 shows the characteristics of major features and principal component (PC) on class classification. All SSH communications in the training dataset are located in the PC range of 0.34–25.0, and most SSH communications are located in a limited range of major features.

We observed the effect of the newly added features *mean_of_IPA_time* and *var_of_IPA_time* on the detection performance. In the result of learning the training dataset to which *mean_of_IPA_time* and *var_of_IPA_time* are added with the random forest (tree number=100) algorithm, recall 11.8% and precision 50.1% were improved compared to the existing detection model (Lee and Lee, 2021).

4.3. Synthetic dataset

4.3.1. Design of a generator and discriminator network

A general GAN algorithm is made up of two networks: a generator network and a discriminator network. In Fig. 4, which depicts a WGAN-GP algorithm, the generator network G and discriminator D in the GAN and WGAN algorithms are the same architecture. This work compares the representative GAN algorithm with WGAN-GP but does not include the optimization of the generator and discriminator networks. Except for the size of the latent vector, the generator and discriminator networks use PyTorch's default settings.

First, we consider the size of the latent vector as an input value. In deep learning, the latent space has a significant impact on the size of the hidden layer. This is due to the small latent space in proportion to the size of the hidden layer, which causes the deep learning model's learning to deteriorate (Pinetz et al., 2019; Tiu, 2020). Given this effect, we chose the latent space (10 dimensions) that was similar to the feature size of dataset.

The generator network generates fake data by inputting a vector of a latent space with a normal distribution and performing four layers of upsampling. The generator network's upsampling layer is made up of a combination of Linear transformation, Batch normalization, and an Activation function, while the discriminator network is made up of a combination of linear transformation and an activation function. The latent variable $\vec{z}$ is expanded to sophisticated and diverse data in the generator network through Linear transformation of each layer. Batch normalization allows for much faster learning rates and easier initialization, and adjustments to avoid distorted distributions (Ioffe and Szegedy, 2015). However, it is suggested to avoid applying batch normalization to the gen-

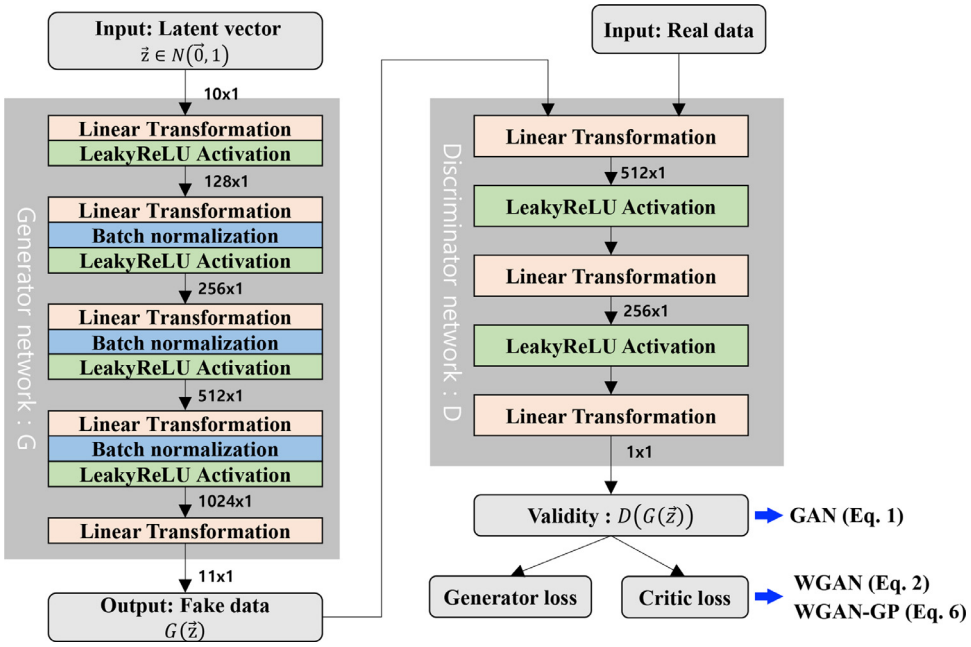

Fig. 4. Network architecture of GAN/WGAN/WGAN-GP: The GAN, WGAN, and WGAN-GP, which compare and test the generation of valid data in this study, all algorithms use the same generator and discriminator networks.

erator output layer and the discriminator input layer because it often to all layers of the GAN may result in sample oscillation and model instability (Radford et al., 2015). The non-linear Activation function deepens the hidden layer and provides the effect of reducing parameters and computations while maintaining accuracy (Sharma et al., 2017). In the discriminator network, it has the same effect. Among non-linear Activation functions, the Leaky ReLU function supports back-propagation in negative space, which adds to ReLU's advantage of fast generator network convergence.

The discriminator network, which was trained with real training data, receives fake data $G(\vec{z})$ and outputs validity $D(G(\vec{z}))$. The GAN algorithm calculates the objective function (Eq. (1)) using $D(G(\vec{z}))$, whereas the WGAN and WGAN-GP use the probability distribution of real samples and the probability of generated samples to determine the validity of fake data. In contrast to the generator network, the discriminator's linear transformation reduces dimensionality while providing sophisticated validity through multiple layers.

4.3.2. GAN algorithm assessment using softmax

We use the WGAN-GP algorithm as a method to generate samples. In this work, our experiment confirms that the WGAN-GP algorithm could generate stable and valid samples compared to the GAN and WGAN algorithm. Samples are generated from the training dataset (DARPA99 Week 4) with 11 features of Table 3 using GAN, WGAN, and WGAN-GP algorithms. The GAN, WGAN, and WGAN-GP algorithms used Pytorch-GAN (Linder-Norén, 2018), an open-source implemented with PyTorch. We repeated the data generation six times using GAN, WGAN, and WGAN-GP algorithms to prepare sufficient samples. To avoid underfitting data, data from 0 to 10 epochs were excluded from the synthetic dataset. The generated samples should be similar to the data that can occur in the real network environment. We classified the output of the softmax function based on a specific threshold to select samples applicable to the training dataset. Table 4 shows the sample counts for each output range of the softmax function. In the range of high softmax output, the WGAN-GP algorithm generates more samples compared to GAN and WGAN. The reason GAN and WGAN generate insufficient samples compared to WGAN-GP is the limited sample variety caused by the mode collapse phenomenon of GAN (Gulrajani et al., 2017) and the critic's limited discrimination performance (Arjovsky et al., 2017) due to the vanishing gradient caused by the small weight clipping of the WGAN.

Table 4
Sample generation result by GAN, WGAN, and WGAN-GP (epoch 100, batch 64, latent space 10): When comparing the number of samples generated for each range of the softmax function, only the WGAN-GP algorithm generates samples with more than 0.75 softmax output.

Algorithm	Class	Softmax(Class)				
		> 0.90	> 0.85	> 0.8	> 0.75	> 0.70
GAN	SSH	0	0	0	0	565,148
	Non-SSH	0	0	0	0	724,770
WGAN	SSH	0	0	0	0	14
	Non-SSH	0	0	0	0	11,829
WGAN-GP	SSH	0	667	2,968	13,815	42,468
	Non-SSH	0	1	23	1,488	8,582,565

4.3.3. Sample generation using WGAN-GP

We measured the accuracy of the detection model according to the range of the critic output value of the generated dataset in order to select a suitable virtual sample as the training dataset of the SSH detection model. Fig. 5 shows the changes in the critic loss (Eq. (6)) and the generator loss (Eq. (7)) according to the number of epochs of the first dataset among six generated dataset.

We use the value of softmax output as a criterion to distinguish samples similar to actual communication. Samples over the softmax output value 0.7 are selected (Table 4). Fig. 5 shows how critic loss and generator loss change as the epoch of the decoding increases. The critic loss converges to 0 and shows stable learning. As the epoch increases, the generated sample more and more similar to the real data, and the generator loss gradually decreases.

However, if samples are more sophisticated and more trained to generate a limited data diversity, the synthetic dataset has a limited variation. We analyzed the generator loss to select samples to be applied to the synthetic dataset to improve the performance of the SSH detection model. In the Fig. 6, as the epoch increases, the

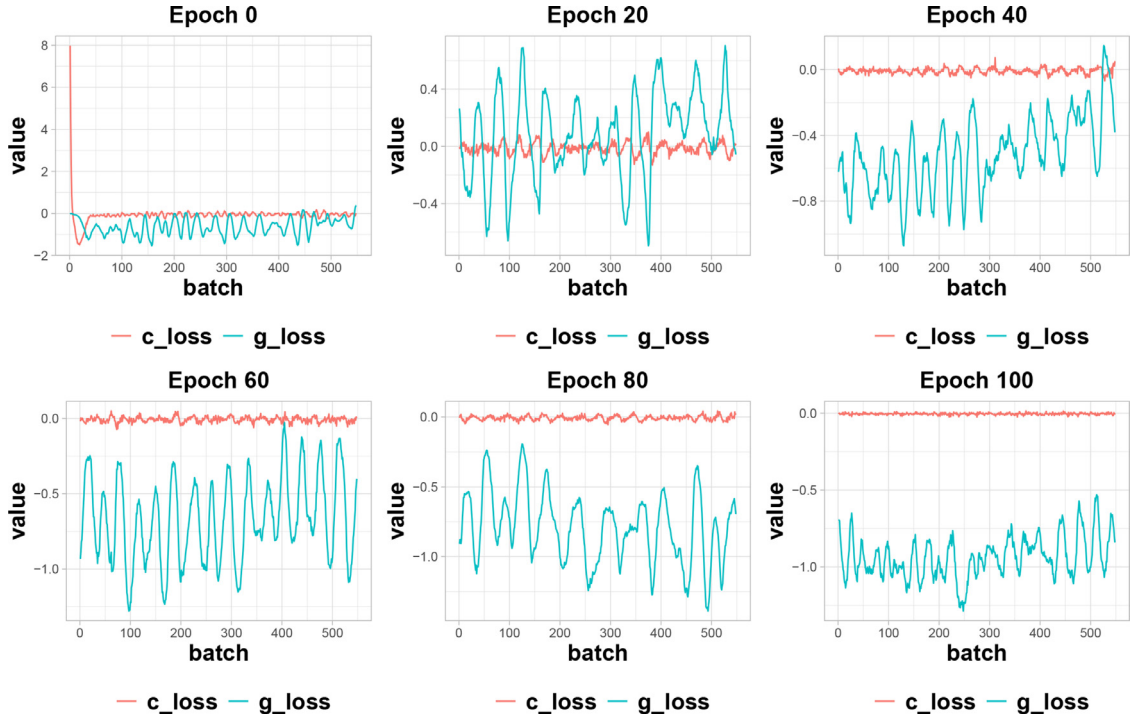

Fig. 5. Changes in critic loss(c_loss) and generator loss(g_loss) by the number of epochs: As the epoch increases, the training of the WGAN-GP becomes more stable. Training in high epochs causes the critic loss to converge to zero and the generator loss to decrease gradually.

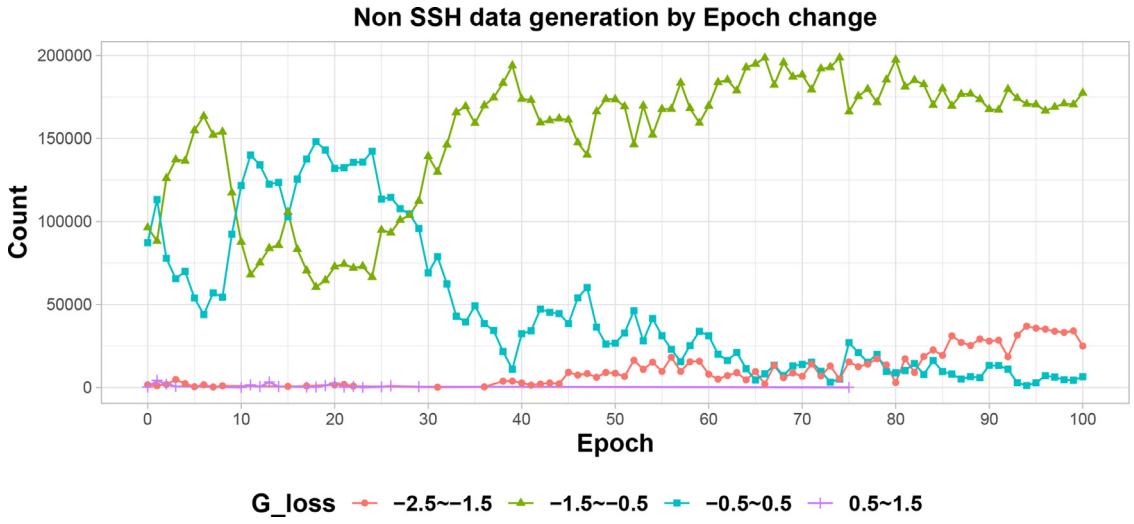

Fig. 6. Changes in critic loss and generator loss by number of epochs.

sample corresponding to the low generator loss section (data similar to the real data) is generated at a high rate. In addition, after the 20–30 epoch range of Fig. 6, the generation rate of data with low generator loss increases. In order to sufficiently select samples similar to the original data, it is possible to select sufficient samples by training until high epoch or by multiple sample generation.

4.3.4. Sample selection for synthetic dataset

Our previous work and the SSH detection model proposed in Section 4.2 both show better recall and precision. However, the precision is relatively low compared to the improvement of recall. It is necessary to improve the detection model to decrease false positives and false negatives by increasing the detection accuracy for non-SSH communication (Eqs. (8) and (9)). We examined the effect on detection rate and detection accuracy by changing the generator loss (similarity) and quantity of the sample added to the training dataset (Table 5). DARPA99 Week 4 dataset is used for the training dataset to make a synthetic dataset, and Week 1 and Week 3 datasets are used for the test dataset. First, we have measured the effect of SSH samples on the performance of the SSH detection model and confirmed that the added SSH samples increase false positives and significantly reduce precision. In the next experiment using non-SSH samples, they have effected to increase the precision by improving the false positive of the SSH detection model.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{8}$$

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{9}$$

Table 5
SSH detection performance using synthetic dataset: a synthetic dataset that mixes 150 K samples with high generator loss and 50K samples with low generator provides the highest F_1-score. (Week 1, 3 test dataset)

Generator Loss		Count	Recall (RF)	Precision (RF)	F_1-score (RF)
No samples added		0	0.9996	0.8637	0.9267
High range (G_loss>-0.5)		100,000	0.9992	0.9877	0.9934
		150,000	**0.9972**	**0.9985**	**0.9978**
		200,000	0.9892	0.9996	0.9944
		250,000	0.9848	1.0000	0.9923
		300,000	0.9692	1.0000	0.9844
		400,000	0.9416	1.0000	0.9699
Low range (G_loss<-1.5)		100,000	1.0000	0.9119	0.9539
		150,000	1.0000	0.9157	0.9560
		200,000	1.0000	0.9358	0.9668
		250,000	1.0000	0.9446	0.9715
		300,000	1.0000	0.9491	0.9739
		400,000	1.0000	0.9530	0.9759
Mixed range	**High**	**150,000**	**0.9984**	**0.9988**	**0.9986**
	low	**50,000**			
	High	50,000	1.0000	0.9646	0.9820
	low	150,000			
	High	100,000	1.0000	0.9838	0.9919
	low	100,000			

$$F_1 - score = \frac{2 \times Recall \times Precision}{Recall + Precision} \quad (10)$$

Table 5 shows the effect on the SSH detection model under the different conditions of non-SSH samples. The precision shows different aspects according to the value of generator loss (G-loss). Non-SSH samples in the high G-loss range (> -0.5) help to significantly improve the precision when adequate amounts are added to the training dataset. However, the recall tends to decrease slightly compared to using the original training dataset. On the other hand, non-SSH samples in the low G-loss range improve both recall and precision as the quantity of samples increases. Although the precision is improved less than non-SSH samples with high generator loss, it is expected that both recall and precision of the SSH detection model can be slightly improved. However, it takes a lot of time to generate enough samples and to train the SSH detection model with huge samples.

Considering that the ratio of non-SSH communication is very high compared to SSH communication in the training dataset, the precision of the SSH detection model is very important along with recall. Non-SSH data with high generator loss provides high precision with small samples, and non-SSH data with low generator loss has the characteristic of maintaining high recall. Since we observed that high and low generator loss have different effects on the recall and precision, we created a synthetic dataset by mixing samples in the different G-loss ranges.

The process of mixing non-SSH samples in the different G-loss ranges is as follows. First, we measure the performance of the SSH detection model with samples in the single side G-loss range. Next, we select the case of high generator loss that shows the highest F_1-score. Finally, we find the maximum recall and precision by gradually adding samples with low generator loss. Table 5 shows the performance of the SSH detection model trained by changing the count of samples with different ranges. We selected the case of 150,000 samples indicating the highest F_1-score in the high range of generator loss. Measuring the performance by increasing the count of samples with low generator loss, the highest 0.9986 F_1-score was measured when 50,000 samples with low generator loss was added. Using samples with mixed range generator loss, the F_1-score is increased by up to 3.18% compared to using 200,000 non-SSH samples with single side (low range) generator loss. Moreover, this mixed case shows the highest F_1-score among 100,000–400,000 non-SSH samples.

4.4. Detection performance comparison

While using non-SSH samples to reduce false positives of the SSH detection model, the decrease of true positives is minimal. The reason lies in the algorithm of the SSH detection model. In this work, the SSH detection model defines various sessions with the same destination IP and port as the same service. If one of the various sessions is detected as SSH communication, all sessions with the same destination IP and port are classified as SSH communication. This SSH detection model algorithm prevents the decrease of true positives from the generated samples.

Table 6 compares the performance of SSH detection models. F_1-score and precision are improved compared to other SSH detection models. Considering that DARPA99, MAWI, and AMP are unbalanced datasets with a small percentage of SSH communication, the improved precision by reducing false positives is remarkable in terms of the accurate precision of SSH communication. Compared with our previous study (Lee and Lee, 2021), the precision is improved by 63.6%, and the number of false positives is reduced from 39 to 0∼1. Considering that there are 25 SSH servers in the test dataset, the decrease in precision saves time and effort in classifying the final target data.

4.5. Sample validation

The similarity between non-SSH samples and real non-SSH data is verified through the distribution of non-SSH based on major features. Sample validation uses 50,000 real non-SSH data randomly selected from the original training dataset and 50,000 non-SSH samples randomly selected from the synthetic dataset. The case showing the highest F_1-score (high G-loss: 150,000, low G-loss: 50,000) was selected for the synthetic dataset. In the Fig. 7, the shape of non-SSH samples is similar to real non-SSH, but the distribution of non-SSH samples is wider than real non-SSH. The nature of the sample distribution supports the SSH detection model to improve precision by reducing false positives for SSH.

Table 6
Comparison of SSH detection performance by dataset feature type: The detection model applying the average and variance of inter-packet arrival time to the training dataset shows improved precision and F_1-score.

Detection Model (Algorithm)	Recall	Precision	F_1-score	Test dataset
Vinayakumar et al. (2017) (CNN-LSTM)	0.988	0.990	0.989	Flow-based features (MAWI/AMP)
Alshammari and Zincir-Heywood (2011) (SBB-GP)[a]	0.983 (+0.00%)	0.128 (+0.00%)	0.226 (+0.00%)	Flow-based features (DARPA99 Week 1,3)
Lee and Lee (2021) (Random Forest)	0.881 (-10.2%)	0.363 (+23.5%)	0.514 (+28.8%)	Session-based features (DARPA99 Week 1,3)
Enhanced model with IPA time (Random Forest)	1.00 (+1.7%)	0.864 (+73.6%)	0.927 (+70.1%)	Session-based features (DARPA99 Week 1,3)
Enhanced model with IPA time & WGAN-GP (Random Forest)	**0.998 (+1.5%)**	**0.999 (+87.1%)**	**0.999 (+77.3%)**	**Session-based features (DARPA99 Week 1,3)**

[a] Symbiotic Bid-based (SBB) GP source code.

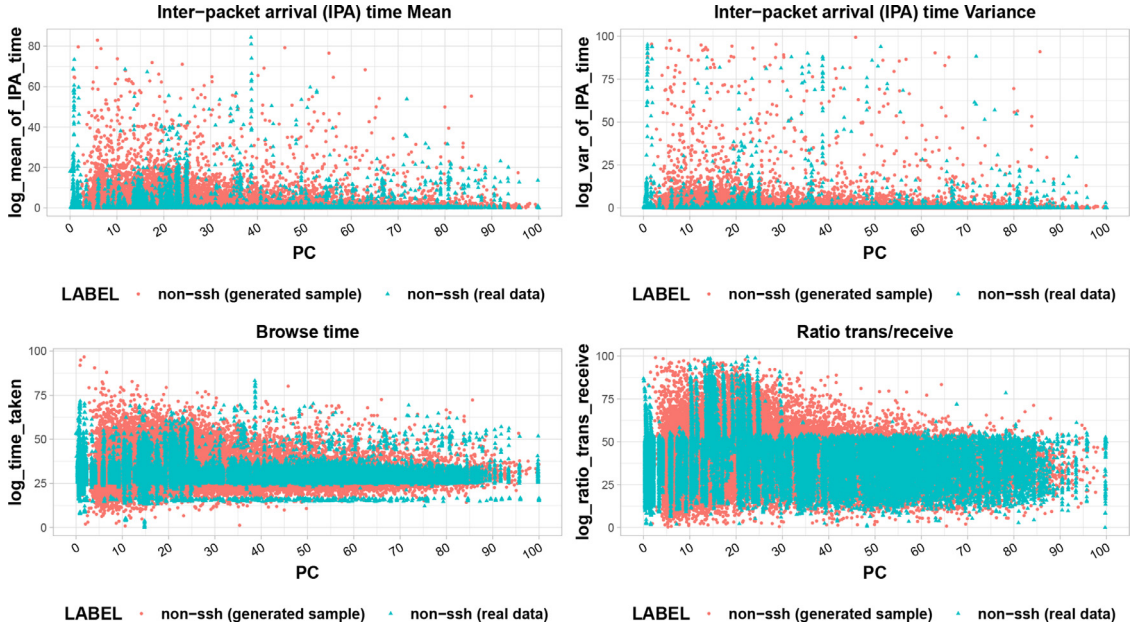

Fig. 7. Comparison of the distribution of real non-SSH and non-SSH samples: It shows the distribution of data randomly selected by 50,000 each from real non-SSH and non-SSH samples.

While using non-SSH samples to reduce false positives of the SSH detection model, the decrease of true positives is minimal. The reason lies in the algorithm of the SSH detection model. In this work, the SSH detection model defines various sessions with the same destination IP and port as the same service. If one of the various sessions is detected as SSH communication, all sessions with the same destination IP and port are classified as SSH communication. This SSH detection model algorithm prevents the decrease of true positives from the generated samples.

5. Evaluation

We evaluated the detection performance using the DARPA 99 week 5 dataset as another test dataset. After adopting the mean and variance of inter-packet arrival time, the detection model presents 81.82% recall and 74.73% precision with the week 5 test dataset, and the recall is relatively low compared to the week 1, 3 test datasets. The detection performance for week 5 is limited because the detection model cannot detect abnormal SSH communication like a denied access. Although we added SSH samples to improve recall, the recall did not improve further and the precision decreased.

To evaluate the effect of samples using the WGAN-GP algorithm, we applied non-SSH samples (high generator loss 150,000, low generator loss 50,000) that showed high detection performance in the test datasets at week 1, 3 without adding SSH samples. The results shows that the false positives are reduced and the precision of Eq. (9) is increased to 100%. However, because the false negative increases, the recall of Eq. (8) decreases and the F_1-score is very low. When applying SSH samples and non-SSH samples at the same time, we find that the false negatives do not increase and the recall of Eq. (8) does not decrease.

The SSH communication ratio (3.36%) of the week 5 dataset is large in comparison to the SSH communication ratio of weeks 1 and 3. (0.36%). Therefore, sophisticated SSH samples (low generator loss) should be added to the synthetic dataset used for the SSH detection model of week 5. In Table 7, the highest detection performance based on F_1-score appears on the synthetic training dataset having 500 SSH samples (low generator loss), 150,000 non-SSH samples (high generator loss), and 50,000 non-SSH samples (low generator loss). SSH samples improve the recall but negatively affects the precision. Moreover, the SSH samples with high generator loss make the false positive increase more than the SSH samples with low generator loss. Table 8 shows the effect of the synthetic data on the recall and precision of the SSH detection model.

Table 7
The effect of sample type and quantity on the performance of SSH detection model (Week5 test dataset).

Detection Model	Count of samples				Model performance		
	G-loss (SSH)		G-loss (Non-SSH)		Recall	Precision	F_1-score
	High	Low	High	Low			
Without IPA time mean and variance	**0**	**0**	**0**	**0**	**0.6991**	**0.3929**	**0.5031**
With IPA time mean and variance	**0**	**0**	**0**	**0**	**0.8182**	**0.7473**	**0.7811**
With SSH samples	0	100	0	0	0.8182	0.5398	0.6505
	0	300	0	0	0.8182	0.4039	0.5408
	0	500	0	0	0.8182	0.2954	0.4341
	100	0	0	0	0.8182	0.4509	0.5814
	300	0	0	0	0.8182	0.3054	0.4448
	500	0	0	0	0.8182	0.1787	0.2933
With SSH & non-SSH samples	0	0	150,000	50,000	0.4590	0.1000	0.1642
	0	100	150,000	50,000	0.5782	0.1000	0.1705
	0	300	150,000	50,000	0.7963	0.9796	0.8785
	0	**500**	**150,000**	50,000	**0.8182**	**0.9930**	**0.8972**
	0	750	150,000	50,000	0.8182	0.9545	0.8811
	100	0	150,000	50,000	0.7173	0.9281	0.8092
	300	0	150,000	50,000	0.8109	0.8708	0.8398
	500	0	150,000	50,000	0.8182	0.8533	0.8354
	750	0	150,000	50,000	0.8182	0.7260	0.7540

Table 8
Effect of diverse samples on the SSH detection performance: Properly synthesized samples can improve SSH detection performance. However, if the SSH detection does not improve furthermore, the training dataset needs to be further updated by real samples.

Sample type	Analysis of effect
Generated samples	Generated sample improves the detection performance by generating missing data from the original dataset and refining the outliers.
Generated samples with low generator loss	Samples with low generator loss slightly increase true positives and false positives because of the high similarity with the original dataset.
Generated samples with high generator loss	Samples with a high generator loss have a relatively large increase in true positives and false positives due to low similarity.
Synthetic samples	In Table 5 and 7, samples with high generator loss and low generator loss should be synthesized and added to improve the recall and precision.
Real sample update	Generated samples are in the limited range similar to the training dataset (real samples). Real samples are continuously updated to generate informative samples.

6. Conclusion

Our work improves the performance of the SSH detection model using features of inter-packet arrival time and samples generated from WGAN-GP algorithm. The generated sample is meaningful in that it reinforces missing data to the training dataset without label classification (Goodfellow, 2016). In this work, we convert the inter-packet arrival time of packet-based data into mean and variance for the session-based data. The converted features are added to the session-based dataset. The performance of the SSH detection model improved by 11.9% in recall and 50.1% in precision compared to our previous work (Lee and Lee, 2021). In the finally enhanced model using generated samples, the precision is further improved by 13.5%. Additionally, our work analyzes the characteristics of generated samples to explain the optimal quantity and generator loss range. In the evaluation, we demonstrate that the distinction of the test dataset is related to the selection of the sample condition. When applying the SSH detection model using WGAN-GP to the actual network environment, the synthetic training dataset should be adjusted the generated sample condition for the optimal detection by considering the target dataset.

Through repeated experiments, we determined the number of samples suitable for the synthetic dataset. However, we have still to conduct a model study to predict the number of generated samples required for a synthetic dataset in our work. Furthermore, there are constraints that must be considered before applying our study's WGAN-GP algorithm to the detection model in the real world. The SSH communication ratio of the target dataset to be predicted, as seen in evaluation, influences the properties and quantity of samples to be added to the synthetic dataset. Because the SSH communication ratio in the actual network environment is constantly changing due to various factors such as hacking and the implementation of new services, it is necessary to consider that it is not possible to determine the properties and quantity of samples to be added solely based on the proportion of the training dataset. To overcome these constraints, we will continue to investigate how to adjust the class range and quantity of generated samples to be appropriate for the target data without requiring an experiment. We will investigate further a model that classifies SSH traffic details (command script transfer, file upload, file download, reverse connection) in order to reduce false positive detection and detect anomaly traffic more accurately. We open the python codes (SSH detection model using random forest, and WGAN-GP based sample generation) and dataset used in this work on the author's Github (https://github.com/junimirang/) to help related studies.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Junwon Lee: Conceptualization, Methodology, Software, Validation, Formal analysis, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Heejo Lee:** Conceptualization, Writing – review & editing, Supervision, Project administration.

Acknowledgment

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-01697 Development of Automated Vulnerability Discovery Technologies for Blockchain Platform Security, No. 2019-0 -01343 Regional Strategic Industry Convergence Security Core Talent Training Business, and No. IITP-2021-2020-0- 01819 ICT Creative Consilience program).

References

Al Olaimat, M., Lee, D., Kim, Y., Kim, J., Kim, J., 2020. A learning-based data augmentation for network anomaly detection. In: Proceedings of the 29th International Conference on Computer Communications and Networks (ICCCN). IEEE, pp. 1–10.

Alshammari, R., Zincir-Heywood, A.N., 2007. A flow based approach for SSH traffic detection. In: Proceedings of the IEEE international conference on systems, man and cybernetics. IEEE, pp. 296–301.

Alshammari, R., Zincir-Heywood, A.N., 2011. Can encrypted traffic be identified without port numbers, ip addresses and payload inspection? Comput. Netw. 55 (6), 1326–1350.

Arjovsky, M., Chintala, S., Bottou, L., 2017. Wasserstein generative adversarial networks. In: Proceedings of the International Conference on Machine Learning. PMLR, pp. 214–223.

Berg, A., Forsberg, D., 2019. Identifying DNS-tunneled traffic with predictive models. arXiv preprint arXiv:1906.11246.

Beckett, J., 2017. What's a generative adversarial network? Inventor explains. https://blogs.nvidia.com/blog/2017/05/17/generative-adversarial-networks/.

Burande, A., Pise, A., Desai, S., Martin, Y., D'mello, S., 2014. Wireless network security by SSH tunneling. Int. J. Sci. Res. Publ. 4 (1), 2250–3153.

Dharmapurikar, S., Krishnamurthy, P., Sproull, T., Lockwood, J., 2003. Deep packet inspection using parallel bloom filters. In: Proceedings of the 11th Symposium on High Performance Interconnects, 2003. IEEE, pp. 44–51.

Fraser, N., Plan, F., Oeary, J., Cannon, V., Leong, R., Perez, D., Shen, C.E., 2019. Apt41: a dual espionage and cyber crime operation. https://www.mandiant.com/resources/apt41-dual-espionage-and-cyber-crime-operation/.

Garsva, E., Paulauskas, N., Grazulevicius, G., Gulbinovic, L., 2014. Packet inter-arrival time distribution in academic computer network. Elektron. Elektrotech. 20 (3), 87–90.

Genevay, A., Peyré, G., Cuturi, M., 2017. GAN and VAE from an optimal transport point of view. arXiv preprint arXiv:1706.01807.

Goodfellow, I., 2016. NIPS 2016 tutorial: generative adversarial networks. arXiv preprint arXiv:1701.00160.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets. Adv. Neural Inf. Process. Syst. 27.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A., 2017. Improved training of Wasserstein GANs. arXiv preprint arXiv:1704.00028.

Hinton, G.E., Salakhutdinov, R.R., 2006. Reducing the dimensionality of data with neural networks. Science 313 (5786), 504–507.

Ioffe, S., Szegedy, C., 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Proceedings of the International Conference on Machine Learning. PMLR, pp. 448–456.

Kingma, D. P., Welling, M., 2013. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

Kwon, D., Kim, H., Kim, J., Suh, S.C., Kim, I., Kim, K.J., 2019. A survey of deep learning-based network anomaly detection. Cluster Comput. 22 (1), 949–961.

Lee, J., Lee, H., 2021. An ssh predictive model using machine learning with web proxy session logs. Int. J. Inf. Secur. 1–12.

Lin, P.-C., Lin, Y.-D., Lai, Y.-C., Lee, T.-H., 2008. Using string matching for deep packet inspection. Computer 41 (4), 23–28.

Lin, Z., Shi, Y., Xue, Z., 2018. IDSGAN: generative adversarial networks for attack generation against intrusion detection. arXiv preprint arXiv:1809.02077.

Macwan, R., Das, S., Das, M.L., 2021. Gids: anomaly detection using generative adversarial networks. In: Proceedings of the Security in Computing and Communications: 8th International Symposium, SSCC 2020, Chennai, India, October 14–17, 2020, Revised Selected Papers, 1364. Springer Nature, p. 244.

Mahoney, M.V., 2003. Network traffic anomaly detection based on packet bytes. In: Proceedings of the ACM Symposium on Applied Computing, pp. 346–350.

Moon, D., Im, H., Kim, I., Park, J.H., 2017. DTB-IDS: an intrusion detection system based on decision tree using behavior analysis for preventing apt attacks. J. Supercomput. 73 (7), 2881–2895.

Radford, A., Metz, L., Chintala, S., 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.

Ring, M., Schlör, D., Landes, D., Hotho, A., 2019. Flow-based network traffic generation using generative adversarial networks. Comput. Secur. 82, 156–172.

Sadasivam, G.K., Hota, C., Anand, B., 2016. Classification of SSH attacks using machine learning algorithms. In: Proceedings of the 6th International Conference on IT Convergence and Security (ICITCS). IEEE, pp. 1–6.

Sakurada, M., Yairi, T., 2014. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In: Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis, pp. 4–11.

Salakhutdinov, R., Hinton, G., 2009. Semantic hashing. Int. J. Approx. Reason. 50 (7), 969–978.

Satoh, A., Nakamura, Y., Ikenaga, T., 2012. SSH dictionary attack detection based on flow analysis. In: Proceedings of the IEEE/IPSJ 12th International Symposium on Applications and the Internet. IEEE, pp. 51–59.

Sharma, S., Sharma, S., Athaiya, A., 2017. Activation functions in neural networks. Towards Data Sci.ss 6 (12), 310–316.

Tiu, E., 2020. Understanding latent space in machine learning. https://towardsdatascience.com/understanding-latent-space-in-machine-learning-de5a7c687d8d/.

SSH.COM, 2021. How an SSH tunnel can bypass firewalls, add encryption to application protocols, and help access services remotely. https://www.ssh.com/academy/ssh/tunneling

Vinayakumar, R., Soman, K., Poornachandran, P., 2017. Secure shell (SSH) traffic analysis with flow based features using shallow and deep networks. In: Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE, pp. 2026–2032.

Wang, F., Chawla, S., Surian, D., 2013. Latent outlier detection and the low precision problem. In: Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description, pp. 46–52.

Wang, Z., Wang, P., Zhou, X., Li, S., Zhang, M., 2019. FlowGAN: unbalanced network encrypted traffic identification method based on GAN. In: Proceedings of the IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom). IEEE, pp. 975–983.

Linder-Norén, E., 2018. eriklindernoren/PyTorch-GAN, https://github.com/eriklindernoren/Pytorch-gan/

Pinetz, T., Soukup, D., & Pock, T., 2019. Impact of the latent space on the ability of GANs to fit the distribution.

Weng, L., 2019. From GAN to WGAN. arXiv preprint arXiv:1904.08994.

Wheelus, C., Khoshgoftaar, T.M., Zuech, R., Najafabadi, M.M., 2014. A session based approach for aggregating network traffic data–the santa dataset. In: Proceedings of the IEEE International Conference on Bioinformatics and Bioengineering. IEEE, pp. 369–378.

Xu, L., et al., 2020. Synthesizing Tabular Data Using Conditional GAN. Massachusetts Institute of Technology Ph.D. thesis.

Yu, Y., Long, J., Cai, Z., 2017. Session-based network intrusion detection using a deep learning architecture. In: Proceedings of the International Conference on Modeling Decisions for Artificial Intelligence. Springer, pp. 144–155.

LINCOLN LAB., 1999. Darpa intrusion detection evaluation dataset, https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset/

Junwon Lee He is a principal engineer in Samsung SDS and has a Ph.D. in the Computer Science and Engineering. He worked as a network engineer for 8 years (2002–2009). He has consulted, proposed, operated, and built the infrastructure of enterprise companies (Tesco, Hynix, Samsung). Since 2010, He has been working as a security engineer in the Integrated Security Center of Samsung. He is doing a network security review and audit, a security review of system architecture, and a security incident investigation and response. In a Ph.D. course, he is researching security anomaly detection using machine learning and deep learning and designing a public cloud-based IoT platform. And he is actively working on projects that apply these studies to the security of the company.

Heejo Lee is a Professor at the Department of Computer Science and Engineering, Korea University, Seoul, Korea, and the director of Center for Software Security and Assurance (CSSA). Before joining Korea University, he was at AhnLab, Inc. as a CTO from 2001 to 2003. From 2000 to 2001, he was a Postdoctorate Researcher at the Department of Computer Science and CERIAS at Purdue University. In 2010, he was a visiting professor at CyLab/CMU. Dr. Lee received his B.S., M.S., Ph.D. degree in Computer Science and Engineering from POSTECH, Pohang, Korea. Dr. Lee serves as an editor of IEEE Trans. on Vehicular Technology, Journal of Communications and Networks, and International Journal of Network Management. He has been working on the consultation of the cyber security in the Philippines (2006), Uzbekistan (2007), Vietnam (2009), Myanmar (2011), Costa Rica (2013) and Cambodia (2015). He is a recipient of the ISC2 ISLA award and got the most prestigious recognition of the asiapacific community service star in 2016. He is a founding member and co-CEO of IOTCUBE Inc., which is a spin off of CSSA, Korea University. The IoTcube open platform has been developed from 2015 to enable non-security professionals to analyze security vulnerabilities professionally, which is available freely for individual users at https://iotcube.net. As a professional service of IoTcube, Labrador provides software bills of materials (SBoM), and examines security vulnerabilities and license violations.