One-Class Anomaly Detection Based on Auto-Encoder for Encrypted Network Attack Analysis

Min-Gyu Kim, and Hwankuk Kim*
SangMyung University, Cheonan-si, South Korea
skystarloid@gamil.com, rinyfeel@smu.ac.kr

Abstract

Using machine learning and deep learning for network anomaly detection is a critical field in mitigating emerging threats. This paper conducts a comparative analysis between two pivotal feature extraction methods, namely, packet-based feature extraction and flow-based feature extraction, which are essential in the development of detection models. Furthermore, to evaluate the performance of these two feature extraction techniques, one-class anomaly detection (OCAD) is carried out using Auto-Encoders. Leveraging the CICIoT2023 dataset, OCAD is performed for web-based Browser Hijacking attacks and SQL Injection attacks. The results of OCAD using both feature extraction methods demonstrate that when employing Flow-based feature extraction, there is an observed increase in accuracy, precision, and F1-score, accompanied by a reduction in recall, as compared to the use of Packet-based feature extraction. In other words, Flow-based feature extraction proves to be more effective than Packet-based feature extraction in permitting fewer missed detections while mitigating a higher number of false alarms.

Keywords: One-Class Anomaly Detection, Feature Extraction, Auto-Encoder, Encrypted Traffic, CICIoT2023

1 Introduction

With the increasing connectivity of numerous IoT devices to the Internet, there is a growing number of IoT devices supporting SSL/TLS protocols to ensure the secure transmission of user data containing sensitive personal information.[1] Furthermore, active research is being conducted on mobile applications and web browsers, driven by the utilization of SSL/TLS.[2][3][4] However, there exists a concern in IoT devices providing web-based services, as attackers can exploit SSL/TLS protocols to carry out attacks with concealed data.[5]

The analysis of encrypted traffic primarily involves methods such as decryption of encrypted traffic, which often comes with limitations such as the risk of sensitive data leakage and significant time overhead, and other methods like network analysis tools such as DPI (Deep Packet Inspection) and the utilization of ML (Machine Learning) and DL (Deep Learning). Therefore, the significance of methods for detecting attacks without decrypting user data has increased.

In the context of anomaly detection through ML and DL in network traffic containing SSL/TLS protocols, researchers have typically employed three categories of feature extraction methods: 1) behavior-based feature extraction, 2) payload-based feature extraction, and 3) supplementary information-based feature extraction. Numerous researchers have been investigating various behavior-based feature extraction methods. Payload-based feature extraction methods involve the use of deep learning or extraction through DPI. These methods often present difficulties in explaining the extracted features and privacy concerns. Supplementary

The 7th International Conference on Mobile Internet Security (MobiSec'23), Dec. 19-21, 2023, Okinawa, Japan, Article No. 7
*Corresponding author: SangMyung University, Cheonan-si, South Korea , rinyfeel@smu.ac.kr

information-based feature extraction involves extracting features from packets other than those containing user data, such as packets from the handshaking process and other packets within the same flow.

This paper aims to compare the anomaly detection performance resulting from two behavior-based feature extraction methods using web-based IoT device attack traffic involving SSL/TLS protocols. Additionally, we utilize autoencoders for one-class anomaly detection. The primary contributions of this study are as follows:

1) We utilized all the normal traffic data from the latest dataset, CICIoT2023, to train a model using an unsupervised deep learning algorithm and conducted network traffic anomaly detection for web-based Browser Hijacking attacks involving SSL/TLS protocols.

2) The results of comparing two feature extraction methods, one based on packet-level metadata and the other on flow-level metadata using a small amount of packet metadata, confirmed that the packet-based feature extraction method outperforms the flow-based feature extraction method in anomaly detection performance for Browser Hijacking attack traffic in IoT environments.

2 Related Work

This section introduces relevant research on methods for extracting features from network traffic data.

Fatani et al. [6] proposed a deep learning-based feature extraction and feature selection algorithm for intrusion detection and malicious identification in IoT environments. They utilized a Convolutional Neural Network (CNN) composed of two 1D convolutional layers and four Fully Connected (FC) layers to perform feature extraction from raw input packets.

Vartouni et al. [7] conducted unsupervised anomaly detection by utilizing feature extraction with a Variational Auto-Encoder (VAE). They compared VAE-based feature extraction with AutoEncoder (AE) and Kernel Principal Component Analysis (kPCA) methods, and found that the VAE-based feature extraction method exhibited the best performance.

Jiao Zhang et al. [8] conducted a study focusing on feature extraction from traffic distribution data and traffic clustering using the sliding window algorithm. The research demonstrated the ability to obtain more clustering distribution features compared to the conventional traffic clustering method, the grid method.

Hongping Yan et al. [9] demonstrated the effective classification of Tor network flows by utilizing Time Window-based flow segmentation and bidirectional statistical feature extraction. The classification, achieved through the utilization of packet length, fixed-length intervals, entropy, and similar parameters, exhibited superior speed and recognition rates compared to deep learning methods.

Drawing upon relevant studies, we employ a sliding window algorithm to extract characteristics of traffic. Additionally, for packet-level feature extraction, we expedite the process by utilizing only a few basic metadata of packets. For flow-level feature extraction, we aim to leverage the fundamental metadata of packets to apply overall statistical and bidirectional statistical techniques to the flow.

3 Anomaly detection method in TLS traffic

3.1 Datasets

This study utilized the CIC IoT Dataset 2023, created by the Canadian Institute for Cybersecurity (CIC) at the University of New Brunswick in Canada. This dataset provides data from real-time traffic generated by various attacks attempted in an actual IoT environment. It includes original PCAP files with traffic data and CSV files containing extracted features from the PCAP files. Additionally, it provides example code in an ipynb file for ML-based multiclass classification using this dataset, as well as source code and tool descriptions used for data feature extraction. The dataset encompasses a wide range of IoT attacks divided into seven categories. In this paper, two specific PCAP files were used from the CIC IoT Dataset 2023: one containing normal traffic from IoT devices and the other capturing traffic from Browser Hijacking and SQL Injection attacks involving SSL/TLS protocols.

- **DDoS**: This category is composed of ACK fragmentation, SlowLoris, UDP Flood, SynonymousIP Flood, etc.
- **Dos**: This category is composed of TCP Flood, HTTP Flood, SYN Flood and UDP Flood.
- **Brute Force**: This category is composed of Dictionary brute force.
- **Spoofing**: This category is composed of ARP Sppofing and DNS Spoofing.
- **Recon**: This category is composed of Ping sweep, OS scan and Port scan.
- **Web-based**: This category is composed of Sql injection, Command Injection, Backdoor malware, Uploading attack, XSS and Browser hijacking.
- **Mirai**: This category is composed of GREIP flood, Greeth flood and UDPPlain.

3.2 Feature Extraction Methods

In this section, we explain the processes of two statistical feature extraction methods for network anomaly detection from TCP traffic data containing SSL/TLS protocol packets.

Before feature extraction, the PCAP files in the utilized CICIoT2023 dataset contain a mix of both benign and malicious traffic originating from attacker IoT devices as well as solely benign traffic from normal IoT devices. In this study, to perform network anomaly detection for cases in which previously benign IoT devices generate malicious traffic, we first filtered out the traffic from IoT devices that act as attackers, and thereafter extracted the features. To separate the data traffic originating from IoT devices used in the attacks, we used Tshark to filter by the IP addresses of the devices involved and packets corresponding to the TCP protocol, and generated a new PCAP file containing only the filtered traffic.

In this study, two methods were employed, each with distinct scopes, for feature extraction from network traffic. Both of these feature extraction methods utilize identical packet metadata for feature extraction. For the feature extraction process, we used Scapy package to read information from each protocol layer of the packets.

1) **Packet based feature extraction**: This method involves extracting statistical features on a packet-by-packet basis from the PCAP files. This process was performed in two steps. First, we stored the metadata of individual packets. The metadata used in this step

Feature	Description
packet size	Packet frame size
IAT	The epoch time difference with the previous packet
ack count	ack flag value
syn count	syn flag value
fin count	fin flag value
psh count	psh flag value
rst count	rst flag value
app data size	TLS Application data length

Table 1: packet based features

included information on the packet itself, such as frame size and epoch time, as well as transmission status information (e.g., TCP flags) and encrypted user data information (e.g., TLS application data length). We excluded information that could specifically identify attacking nodes, such as IP addresses, to prevent bias during model training. Second, we extracted statistical features for every set of 10 packets using their metadata and the 10-sliding window algorithm. Table 1 lists statistical features based on packets that were ultimately extracted through these two steps.

Feature	Description
packet size	Packet frame size mean in flow
IAT	The epoch time difference with the previous packet in the same flow
outbound packet size	outbound packet size mean in a flow
inbound packet size	inbound packet size mean in a flow
transmission rate	Rate of packet transmission in a flow
outbound packet size	Rate of packet outbound transmission in a flow
inbound packet size	Rate of packet inbound transmission in a flow
ack count	Number of packets with ack flag set in the same flow
syn count	Number of packets with syn flag set in the same flow
fin count	Number of packets with fin flag set in the same flow
psh count	Number of packets with psh flag set in the same flow
rst count	Number of packets with rst flag set in the same flow
app data size	TLS Application data length in a flow

Table 2: flow based features

2) **Flow based feature extraction**: This method involved extracting features from the accumulated packet information flowing in one direction between two endpoints from the PCAP file. This process was performed in three steps. First, we stored metadata of individual packets, categorized by flow, based on a 5-tuple. The metadata used in this step were the same as those used in packet-based feature extraction. Second, we extracted statistical features using the accumulated metadata for flows in which new metadata were added. Third, we extracted statistical features for every set of 10 packets using their metadata and the 10-sliding window algorithm. Table 2 lists the final statistical features based on flows that we extracted through these two steps. Additionally, in flow-based feature extraction, we considered the direction of packet flow within a single flow,

extracting features related to inbound and outbound traffic.

4 Experiments and evaluation

In this section, we describe the experimental results of one-class anomaly detection using the two feature extraction methods explained in Section 3.2 from TCP network traffic data containing SSL/TLS for web-based attacks, specifically Browser Hijacking and SQL Injection.

4.1 Experimental setup

Due to the substantial data imbalance, with approximately ten times more data for IoT normal traffic than for attack data, employing a supervised learning-based model would result in an imbalanced dataset. Therefore, in this study, one-class anomaly detection was performed using an AutoEncoder (AE). The layers of the AE model consisted of four Dense layers and two Dropout layers, as shown in Figure 1. The number of units in each layer of the encoder gradually decreased by 25% from the number of input features to a maximum reduction of 50%. In contrast, the number of units in each layer of the decoder increased by 25%, mirroring the AE's input feature count to maintain consistency.

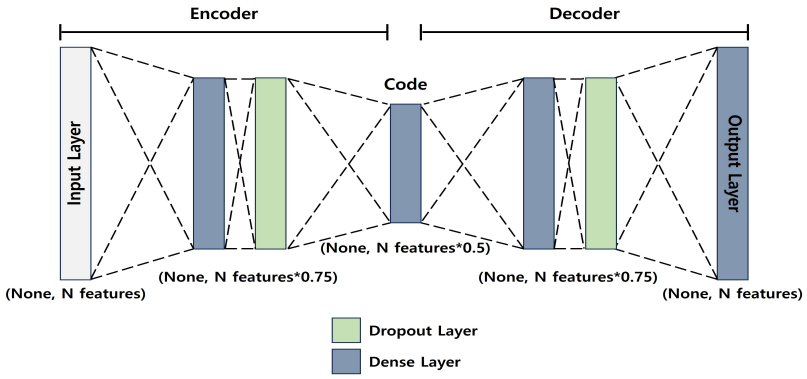


Figure 1: Auto-Encoder Layer Architecture for experimental

The traffic used for feature extraction consists of three types: normal traffic, Browser Hijacking attack traffic, and SQL Injection attack traffic. As shown in Table 3, the packet count for normal traffic is 391,187, and the number of data records in the extracted CSV is 39,119. The packet count for Browser Hijacking attack traffic is 17,385, with 1,739 data records in the extracted CSV. The packet count for SQL Injection attack traffic is 16,981, and the number of data records in the extracted CSV is 1,699.

For model training, we partitioned and merged the three traffic data types into train, validation, and test sets. As depicted in Figure 2, the train set was composed of 50% benign data. The validation set consisted of 25% benign data. The test set was constructed by merging 25% benign data with 100% of each attack type, resulting in the creation of the Browser Hijacking test set and SQL Injection test set.

All experiments were conducted on a PC equipped with an Intel Core i9-10980XE CPU, 128GB RAM, and an NVIDIA GeForce RTX 3090 24GB GPU. Furthermore, for evaluating the performance of the anomaly detection model, we utilized metrics such as Confusion Metrics, precision-recall curve, and f1-score.

Traffic	Packet rows	Csv rows
Benign	391187	39119
Browser Hijacking	17385	1739
SQL Injection	16981	1699

Table 3: Traffic Data Summary

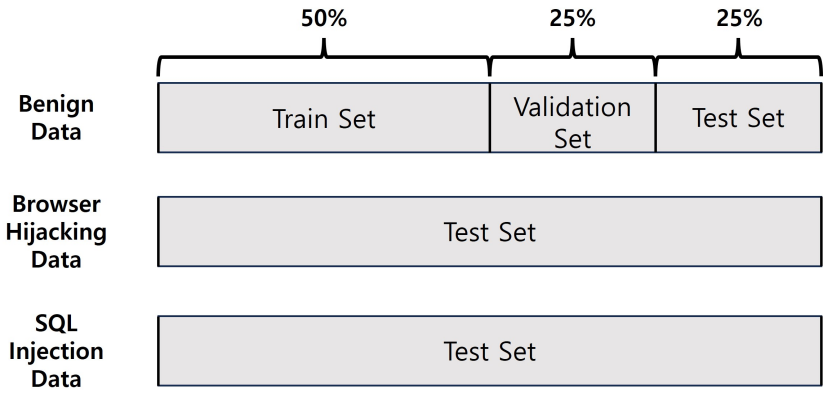


Figure 2: Data composition ratio of train set, validation set, test set

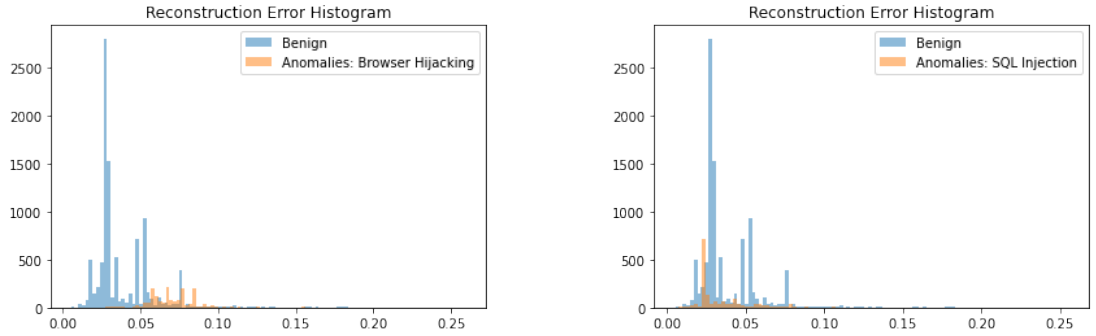
4.2 OC-AD Result with Packet based Feature Extraction

This experiment conducted one-class anomaly detection using packet-based feature extraction. Additionally, the AE trained with packet-based features determined the best threshold value for detecting Browser Hijacking and SQL Injection attacks by identifying the threshold value that maximizes the f1-score. In this experiment, eight packet-based features were used, and the training of the AE was performed with 20 epochs.

Figures 3a and 3b depict the reconstruction error histograms for the AE trained on the Browser Hijacking test set and SQL Injection test set. In the case of the Browser Hijacking test set, Figure 3a shows a noticeable overlap in the reconstruction error value ranges between the benign class and the Browser Hijacking class. Conversely, for the SQL Injection test set, Figure 3b reveals a substantial overlap in the reconstruction error value ranges between the benign class and the SQL Injection class.

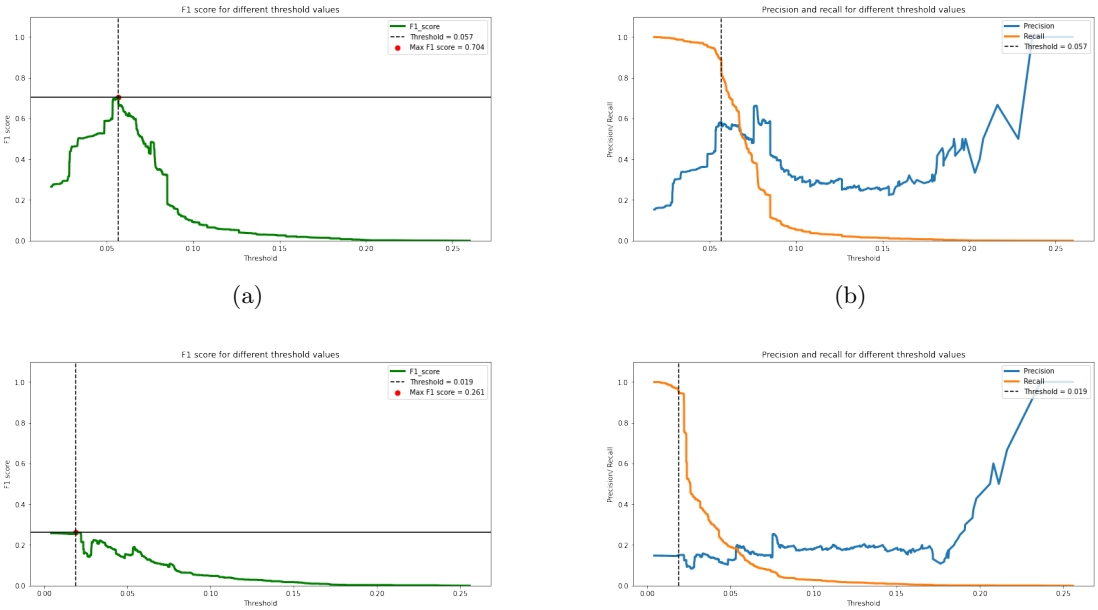
In order to perform anomaly detection for Browser Hijacking and SQL Injection attacks, it is necessary to determine the threshold for the Reconstruction Error. The threshold value was chosen to be the one that yields the highest f1-score for each attack dataset. As shown in Figures 4a and 4c, for the Browser Hijacking test set, the best threshold value is 0.057, resulting in a maximum f1-score of 0.704. In the case of the SQL Injection test set, the best threshold value is 0.019, with a maximum f1-score of 0.261.

Using the Best Threshold, anomaly detection was performed on the Browser Hijacking test set and the SQL Injection test set, and accuracy, precision, recall, and f1-score were measured. Table 4 presents the anomaly detection scores for the two test sets. The results of one-class anomaly detection using packet-based feature extraction indicate that the performance of Browser Hijacking attack detection was approximately 4.5 times better in terms of accuracy, 3.8 times better in precision, and 2.7 times better in f1-score compared to SQL Injection attack



(a) (b)

Figure 3: (a) Reconstruction Error Histogram for Browser Hijacking test set (b) Reconstruction Error Histogram for SQL Injection test set



(a) (b)

(c) (d)

Figure 4: (a) Best Threshold to maximum f1-score with Browser Hijacking Test set, (b) precision and recall value for different threshold values with Browser Hijacking test set, (c) Best Threshold to maximum f1-score with SQL Injection Test set, (d) precision and recall value for different threshold values with SQL Injection test set.

detection. However, the recall score was approximately 1.08 times better for SQL Injection attack detection.

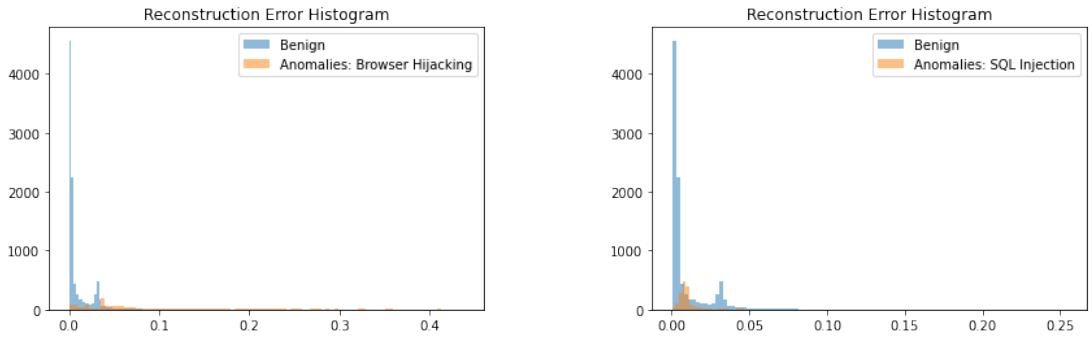
	Browser Hijacking test set	SQL Injection test set
Best Threshold	0.056786	0.018851
True Positive	1544	1633
True Negative	8678	612
False Positive	1102	9168
False Negative	195	66
Accuracy	0.887	0.196
Precision	0.584	0.151
Recall	0.888	0.961
F1-score	0.704	0.261

Table 4: Anomaly Detection score with with Flow based Feature Extraction

4.3 OC-AD Result with Flow based feature Extraction

This experiment involves one-class anomaly detection using flow-based feature extraction. Similarly to Section 4.2, an AE trained with flow-based features was used to determine the best threshold value for detecting Browser Hijacking and SQL Injection attacks by identifying the threshold value that maximizes the f1-score. Thirteen flow-based features were used in this experiment, and the AE's training was conducted with 20 epochs.

Figures 5a and 5b depict the reconstruction error histograms for the AE trained on the Browser Hijacking test set and SQL Injection test set. In the case of the Browser Hijacking test set, Figure 5a shows some overlap in the reconstruction error value ranges between the benign class and the Browser Hijacking class. However, for the SQL Injection test set, Figure 5b demonstrates a considerable overlap in the reconstruction error value ranges between the benign class and the SQL Injection class. This pattern aligns with the results of the Reconstruction Error distribution analyzed using packet-based feature extraction in Section 4.2.



(a) (b)

Figure 5: (a) Reconstruction Error Histogram for Browser Hijacking test set (b) Reconstruction Error Histogram for SQL Injection test set

In order to perform anomaly detection for Browser Hijacking and SQL Injection attacks, it is essential to determine the threshold for the Reconstruction Error, following the same approach as in Section 4.2. The threshold value was chosen to be the one that yields the highest f1-score for each attack dataset. As shown in Figures 6a and 6c, for the Browser Hijacking test set, the

best threshold value is 0.041, resulting in a maximum f1-score of 0.788. In the case of the SQL Injection test set, the best threshold value is 0.004, with a maximum f1-score of 0.49.

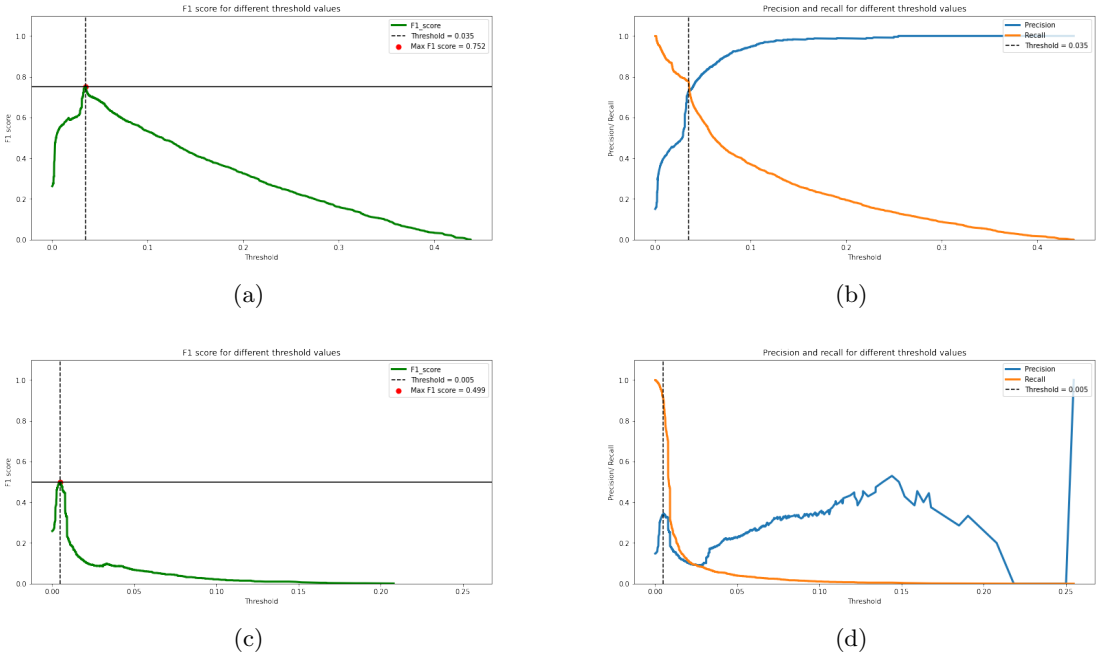


Figure 6: (a) Best Threshold to maximum f1-score with Browser Hijacking Test set, (b) precision and recall value for different threshold values with Browser Hijacking test set, (c) Best Threshold to maximum f1-score with SQL Injection Test set, (d) precision and recall value for different threshold values with SQL Injection test set.

Using the Best Threshold, anomaly detection based on flow features was performed on the Browser Hijacking test set and the SQL Injection test set, and accuracy, precision, recall, and f1-score were measured. Table 4 presents the anomaly detection scores for the two test sets. The results of one-class anomaly detection using packet-based feature extraction indicate that the performance of Browser Hijacking attack detection was approximately 1.3 times better in terms of accuracy, 2.4 times better in precision, and 1.6 times better in f1-score compared to SQL Injection attack detection. However, the recall score was approximately 1.2 times better for SQL Injection attack detection.

4.4 Comparison OCAD Score each Feature Extraction Methods

In this section, we compare the results from Section 4.2 and Section 4.3. For the Browser Hijacking Test set, when using flow-based feature extraction instead of packet-based feature extraction, the accuracy, precision, and F1-score all increased by approximately 1.03 to 1.24 times, while the recall decreased by approximately 1.14 times. Similarly, for the SQL Injection test set, when using flow-based feature extraction as opposed to packet-based feature extraction, accuracy, precision, and F1-score all exhibited an increase of approximately 1.9 to 3.7 times, while the recall decreased by approximately 1.05 times.

Through this, we can discern that using flow-based feature extraction results in an increase

	Browser Hijacking test set	SQL Injection test set
Best Threshold	0.035247	0.005096
True Positive	1352	1555
True Negative	9277	6796
False Positive	503	2984
False Negative	387	144
Accuracy	0.923	0.728
Precision	0.729	0.343
Recall	0.777	0.915
F1-score	0.752	0.499

Table 5: Anomaly Detection score with Flow based Feature Extraction

in precision compared to using packet-based feature extraction, which in turn reduces false positives. However, the decrease in recall, leading to an increase in false negatives, is relatively modest. This discrepancy, as illustrated in Figure 7, suggests that the difference in the extent of improvement is more pronounced in the case of flow-based feature extraction. Therefore, Flow-based feature extraction proves to be more effective in permitting fewer missed detections while mitigating a higher number of false alarms compared to packet-based feature extraction.

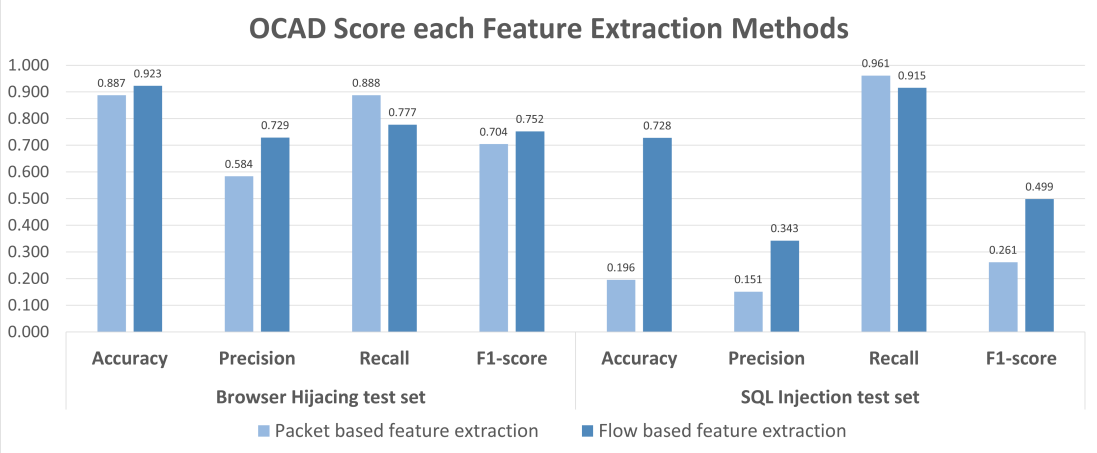


Figure 7: OCAD Score each Feature Extraction Methods

5 Conclusion and Future Works

This paper compared two feature extraction methods for one-class anomaly detection using Auto-Encoder, using the same type of packet metadata. One-class anomaly detection was performed on CICIoT2023's SSL/TLS traffic data for Browser Hijacking and SQL Injection attacks. The results demonstrated that using Flow-based feature extraction method yielded better one-class anomaly detection performance compared to the Packet-based feature extraction method.

However, this paper has the following limitations: 1) Performance was evaluated for only one type of attack, and 2) the number of SSL/TLS traffic data in the attack samples was

limited. Therefore, future research will compare the anomaly detection performance using the same feature extraction methods on datasets with a larger amount of TLS traffic. Additionally, further research will explore session-based feature extraction methods beyond the packet and flow-based feature extraction methods proposed in this paper

Acknowledgement

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.RS-2023-00235509, Development of security monitoring technology based network behavior against encypted cyber threats in ICT convergence environment)

References

[1] Muhammad Talha Paracha, Daniel J Dubois, Narseo Vallina-Rodriguez, and David Choffnes. Iotls: understanding tls usage in consumer iot devices. In *Proceedings of the 21st ACM Internet Measurement Conference*, pages 165–178, 2021.

[2] Marten Oltrogge, Nicolas Huaman, Sabrina Amft, Yasemin Acar, Michael Backes, and Sascha Fahl. Why eve and mallory still love android: Revisiting {TLS}({In) Security} in android applications. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 4347–4364, 2021.

[3] Damilola Orikogbo, Matthias Büchler, and Manuel Egele. Crios: Toward large-scale ios application analysis. In *Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices*, pages 33–42, 2016.

[4] Adrienne Porter Felt, Richard Barnes, April King, Chris Palmer, Chris Bentzel, and Parisa Tabriz. Measuring {HTTPS} adoption on the web. In *26th USENIX security symposium (USENIX security 17)*, pages 1323–1338, 2017.

[5] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. Sok: Security evaluation of home-based iot deployments. In *2019 IEEE symposium on security and privacy (sp)*, pages 1362–1380. IEEE, 2019.

[6] Abdulaziz Fatani, Abdelghani Dahou, Mohammed A. A. Al-qaness, Songfeng Lu, and Mohamed Abd Elaziz. Advanced feature extraction and selection approach using deep learning and aquila optimizer for iot intrusion detection system. *Sensors*, 22(1), 2022.

[7] Rong Yao, Chongdang Liu, Linxuan Zhang, and Peng Peng. Unsupervised anomaly detection using variational auto-encoder based feature extraction. In *2019 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 1–7. IEEE, 2019.

[8] Jiao Zhang, Li Zhou, Angran Xiao, Sai Zeng, Haitao Zhao, Jibo Wei, et al. Sliding window based feature extraction and traffic clustering for green mobile cyberphysical systems. *Mobile Information Systems*, 2017, 2017.

[9] Hongping Yan, Liukun He, Xiangmei Song, Wang Yao, Chang Li, and Qiang Zhou. Bidirectional statistical feature extraction based on time window for tor flow classification. *Symmetry*, 14(10):2002, 2022.