Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 11, No. 01, March 2025.

CURATING DATASETS TO ENHANCE SPYWARE CLASSIFICATION

Mousumi Ahmed Mimi[1], Hu Ng[1] and Timothy Tzen Vun Yap[2]

(Received: 22-Jun.-2024, Revised: 26-Aug.-2024, Accepted: 14-Sep.-2024)

ABSTRACT

Current methods for spyware classification lack effectiveness as well-structured datasets are typically absent, especially those with directionality properties in their set of features. In this particular research work, the efficacy of directionality properties for classification is explored, through engineered features from those on existing datasets. This study curates two datasets, Dataset A which includes features extracted from only single directional packet flows and Dataset B which includes those from bi-directional packet flows. Classification with these features is performed with selected classifiers, where SVM obtained the highest accuracy with 99.88% for Dataset A, while the highest accuracy went to RF, DT and XGBoost for Dataset B with 99.24%. Comparing these results with those from existing research work, the directional properties in these engineered features are able to provide improvements in terms of accuracy in classifying these spywares.

KEYWORDS

Datasets curation, Feature engineering, Packet analysis, Spyware classification.

1. INTRODUCTION

Cybercrimes are increasing due to careless use of online applications and technologies [1]-[2]. Users install various applications on their devices for different purposes, but many are not safe or secure as some disguise themselves as normal applications, such as spyware [3]. Spyware, a malicious software, is installed on the device, gathers sensitive information and transfers it to third parties without user consent [4]. It's very tricky and challenging to distinguish between spyware and legitimate applications, as it disguises itself as a legitimate application [5]. While significant research has been conducted on malware, the exploration of spyware has been overlooked. This creates a research gap for further investigation of the classification methods. Current spyware-classification methods have limitations in feature engineering based on directional properties, which hampers accurate classification. The accuracy of existing spyware classification is often hindered due to inadequate datasets and insufficient analysis for different classifiers, leading to overfit or underfit [6]. Additionally, users sometimes overlook security considerations when installing applications, creating opportunities for hackers. Cybercriminals create clones of popular software on untrustworthy sites with security vulnerabilities. Users carelessly install these cloned applications, allowing attackers to gain access to sensitive information [7]. Thus, this study makes several key contributions to the field of spyware classification.

1.1 Contribution

First, this study aims to enhance spyware classification by curating two new datasets. Dataset A involves annotation based on single-direction packet flow, while Dataset B involves bi-directional packet flow. Information about the packet flow is extracted from major static parameters, such as IP pairs, ports and protocols.

Secondly, feature engineering is applied to form dynamic features from static parameters like Total Forward (Fwd) Packet (Pkt), Total Backward (Bwd) Packet (Pkt), Flow Bytes per Second (Flow Bytes/s), Flow Packets per Second (Flow Pkt/s), … etc., derived from the annotated datasets. The goal is to identify significant features that can provide insights into the effectiveness of using packet-flow information in curating datasets and classifying spyware.

1. M. A. Mimi and H. Ng are with Faculty of Computing and Informatics, Multimedia University, 63100 Cyberjaya, Malaysia. Emails: 1221404218@student.mmu.edu.my and nghu@mmu.edu.my
2. T. T. V. Yap is with School of Mathematical and Computer Sciences, Heriot-Watt University Malaysia, 62200 Putrajaya, Malaysia. Email: timothy.yap@hw.ac.uk

Third, exploring machine-learning (ML) models for spyware classification from the two datasets, with the aim of improving accuracy. This involves applying Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), Naïve Bayes (NB) and Extreme Gradient Boosting (XGBoost) on curated datasets. Comparative analyses of the models trained on these datasets are then conducted to observe clustering patterns for the six different classes (five types of spyware with normal traffic) within each dataset, examining the extent of overlap between them.

The methodologies and the findings offer valuable contributions for enhancing the capabilities of Security Operation Center (SOC) system and Intrusion Detection System (IDS). The curated datasets, based on directional properties, benefit IDS by providing more accurate and contextually rich data for classifiers, that improves classification accuracy and reduces false positive rates for SOC environment. The engineered dynamic features, such as Flow Bytes/s, Flow Pkt/s, enable real-time threat analysis, allowing SOC and IDS to swiftly prioritize alerts for emerging threats. The validation of classification models, including SVM, NB, XGBoost, DT and RF, offer SOCs proven tools for more accurate spyware classification, which improves incident response reliability. Overall, the practical benefits of this research, such as enhanced detection accuracy and improved anti-spyware tools, strengthen the cybersecurity defences of SOCs, reducing the risk of unauthorized data access and privacy violations.

The remaining sections of the study are organized as follows: Section 2 provides a concise summary of prior research on dataset collection, feature extraction and engineering. Section 3 outlines the methodology, including details about dataset acquisition, spyware characteristics and data pre-processing. In Section 4, the proposed method is explained in detail with a focus on packet-flow direction and various approaches for curating Datasets A and B through feature engineering. This section also covers detailed methods for curating Datasets A and B while comparing them to the raw dataset. Section 5 discusses model construction, while Section 6 explores the results and their discussion. Lastly, concluding thoughts are presented in Section 7 to wrap up the document.

2. RESEARCH BACKGROUND

2.1 Spyware Dataset Collection

Cybersecurity is a widely discussed research topic. Researchers have criticized and discussed disciplines within cybersecurity, including spyware. Researchers have collected datasets to detect and characterize spyware.

Qabalin et al. [8] collected a dataset of five different spyware types - Flexispy, Mobilespy, uMobix, TheWispy and mSPY - by capturing packets using PCAPDroid [9]. Then, DT was applied to the dataset, achieving 79% accuracy for binary classification and 77% for multi-class classification.

Conti et al. [10] used the ASAINT (A Spy App Identification System based on Network Traffic) application to identify spyware apps and collected data. Packets were captured using Wireshark [11] and then network traffic was manually analyzed to distinguish between spyware apps and normal ones. The identified classes of spyware applications were cImg, cSms, mSpy and tSpy. Dropbox (DB) [12] and Google Foto (GF) [13] photo uploads were considered as normal applications. Four steps were applied: data collection, pre-processing, training and testing. The data distribution was adjusted using the Synthetic Minority Over-Sampling Technique (SMOTE) during pre-processing. The effectiveness of the datasets was assessed using RF, Linear Regression (LR) and K- Nearest Neighbour (KNN) algorithms, with RF ultimately achieving the best F1-score of 0.85.

M. Naser and Q. A. Al-Haija [14] utilized the Android Spyware-2022 dataset [8] to identify Android spyware, focusing on two out of five spyware classes: MobileSPY and FlexiSPY. After pre-processing the data by removing null and duplicate entries, they analyzed Source IP, Destination IP, Source Port, Destination Port, Duration and Protocol. The testing involved the application of a Fine Decision Tree (FDT), resulting in a 98% accuracy rate.

Noetzold et al. [15] implemented integrated spyware to monitor workplace computers. Integrated spyware represented the utilization of spyware techniques as a fundamental component within the design and functionality of a workplace computer-monitoring solution. The spyware initially sent harmful messages to a Twitter account developed using Python before being applied to the computer. Then, this spyware computer was connected to the workplace computer through an Application

Programming Interface (API) gateway. Subsequently, hate messages were sent from the spyware computer to the workplace one, triggering alerts generated by the API gateway which was connected to a relational database for storing information. Data pre-processing involved the use of normalization and classification techniques to differentiate between hateful and non-hateful speeches. Furthermore, LR, SVM and NB algorithms were utilized to assess prediction validity. NB demonstrated superior accuracy at 80%.

Pierazzi et al. [16] utilized the VirusTotal website [17] to collect spyware. Five types of spyware were identified: HeHe, UaPush, AceCard, Pincer and USBCleaver. Twenty-five features were extracted from each spyware, including file size, permission for sending short-message service, author information, permission for checking phone state, permission to write messages and permission to reboot the system, among others. The Ensemble Late Fusion (ELF) method identified these features as crucial in distinguishing spyware from normal applications. This involved comparing the characteristics of each feature with those of a normal application. Histograms were used to illustrate the variance between spyware and regular applications. Differentiating spyware from normal applications using RF with ELF resulted in an impressive F1-score of 0.96.

Mahesh et al. [18] utilized a Particle Swarm Optimization (PSO) algorithm with Artificial Neural Network (ANN) to improve the prediction of spyware detection. A benchmarked dataset of malware [19] was obtained for the study conducted by Kaggle [20]. Utilizing multi-objective PSO for data pre-processing, the features were then scaled using standard scaling. Additionally, a multi-layer perceptron was utilized along with the Jordan canonical form to remove less significant features and enhance accuracy. The ANN model was finally used to predict the accuracy of the proposed method, achieving an impressive 99% accuracy rate.

Zahan et al. [21] developed a benchmark dataset of malicious and benign software packages from NPM and PyPI to enhance malware-detection tools. The dataset was compiled from existing malicious databases and new malicious and neutral packages. They collected malicious packages from open-source datasets and an internal Socket benchmark and curated a set of neutral packages using manual annotation and automated scanning. The final MalwareBench dataset contained 20,792 samples, of which 6,659 were malicious.

A comprehensive analysis of the search results obtained through the adopted keyword-search approach has been conducted. The reviewed findings are summarized in Table 1.

Table 1. Summary of the reviewed literature in this area of study.

Author(s)	Dataset	Spyware Types	ML Model	Key Findings	Strengths	Weaknesses
Qabalin et al. [8]	Android Spyware-2022	Flexispy, Mobilespy, uMobix, TheWispy, mSPY	DT	Binary classification accuracy: 79.00%, Multi-class: 77.00%	Network-traffic analysis, dataset available	Limited to binary and multi-class classification, lower multi-class accuracy
Conti et al. [10]	ASAINT application, Wireshark	cImg, cSms, mSpy, tSpy	RF, LR, KNN	Best F1-score achieved by RF: 0.85	Effective use of ASAINT	Manual network-traffic analysis
M. Naser and Q. A. Al-Haija [14]	Android Spyware-2022	MobileSPY, FlexiSPY	FDT	Accuracy: 98.00%	High accuracy with FDT	Limited to two spyware classes
Noetzold et al. [15]	Integrated spyware for workplace monitoring	Not specified	LR, SVM, NB	NB demonstrated superior accuracy: 80.00%	Innovative use of integrated spyware	Focused on workplace computers, not mobile spyware
Pierazzi et al. [16]	VirusTotal website	HeHe, UaPush, AceCard, Pincer, USBCleaver	ELF, RF	F1-score: 0.96	High F1-score, Effective feature extraction	Complex ELF method

2.2 Feature Extraction and Engineering

Feature extraction transforms raw data into numerical features that retain the original information, enabling effective processing and improved ML-model performance over direct application of algorithms. Feature engineering, a crucial element in successful ML research, involves data

presentation, refinement and pre-processing tasks. Poorly engineered features can adversely impact model predictions.

Zhang et al. [22] developed a low-cost feature-extraction method for deep learning-based malware detection. The approach involved monitoring API call behaviour, encoding heterogeneous information into homogeneous features using feature hashing and applying gated convolutional neural networks and Bi-Directional Long Short-Term Memory (Bi-LSTM) to capture sequential API call correlations. This yielded a 98.80% area under the ROC curve.

Gibert et al. [23] described a feature-extraction process that combined hand-crafted features from hexadecimal and assembly-language source codes, as well as deep features extracted using deep learning architectures. The hand-crafted features included metadata, byte unigrams, entropy statistics, Haralick features and local binary pattern features. The assembly-language features covered metadata, opcode unigrams, register features, symbol frequency, pixel intensity, API function calls, data define features, section features and miscellaneous features. Deep features were extracted from raw data, including grayscale image-based features, entropy-based features, opcode N-gram features and byte N-gram features. These features were then fused using an early fusion mechanism to create a joint representation, which was used to train a Gradient Boosting (GB) model for malware classification, achieving an accuracy of 99.81%.

Masabo et al. [24] developed a feature-engineering method to classify polymorphic malware (can transform into various forms). The researchers collected a dataset of 5 malware classes (API, Crypto, Locker, Zeus and Shadow brokers.), pre-processed the data and performed feature engineering to identify 11 top features. These included static analysis of portable executable files, packing techniques, file access and registry reading. The developed feature-engineering approach outperformed traditional ML methods (GB), achieving a 94% accuracy.

Nawaz et al. [25] proposed a system to classify Android malware using the Drebin dataset [26]. Static analysis focused on Android intents and permissions, while dynamic analysis utilized network requests and API calls. Apktool [27] was used to decompile and decode the APK files. Feature selection with Info Gain reduced the dimensionality of permissions, intents, API calls and network features. These features were extracted from the APK components and used to train ML classifiers, with RF and GB performing best on the permission features, achieving an F1-score of 0.98.

Jung et al. [28] utilized an APK file from the AndroZoo dataset [29], extracted information on API calls and permissions and generated a feature vector for each application. They applied feature-selection methods to choose the top 20 features from API calls and permissions. The authors then employed RF and grid search to establish optimal hyperparameters and the best accuracy of 96.95% was obtained using Gini importance with the RF model.

Low et al. [30] explored two feature-engineering methods, label encoding and evidence counting, for malware detection. The study involved four main steps: data pre-processing, feature selection, model construction and evaluation. Five malware classes (Advanced Persistent Threats (APT), Crypto, Zeus, Locker and Shadow Brokers) were extracted from the dataset. During pre-processing, data integration, cleaning and transformation were applied. Boruta was used for feature selection and several ML models were constructed, with the optimal parameters identified through grid search. The dataset was balanced using SMOTE. The results showed that RF provided better accuracy for label encoding at 91.34%, while LSTM achieved higher accuracy of 94.64% for evidence counting.

A comprehensive analysis of the search results obtained through the adopted keyword-search approach has been conducted. The reviewed findings are summarized in Table 2.

3. METHODOLOGY

The dataset is initially prepared through pre-processing, organizing the data and converting it into a Comma-Separated Values (CSV) format. Feature engineering is then conducted to choose significant features for classification-model performance. Subsequently, classification models are constructed and their results are recorded for evaluation purposes. Figure 1 illustrates the workflow of the methodology.

Table 2. Summary of the reviewed literature in this area of study.

Author(s)	Dataset	Spyware/ Malware Types	Features and Techniques	ML Model	Performance	Strengths	Weaknesses
Zhang et al. [22]	AV-TEST 2017	Various PE malware	Feature extraction using Cuckoo, Feature hashing, Multiple gated CNNs, Bi-LSTM	Multiple gated CNNs, Bi-LSTM	AUC: 98.80%	Effective deep learning for malware detection	Focused on PE files, not mobile apps
Gibert et al. [23]	Not specified	Malware (not specified)	Hand-crafted and deep features, Fusion mechanism, Joint representation of features from multiple modalities	XGBoost	Accuracy: 99.81%	Comprehensive feature extraction from multiple sources	Complex and time-consuming
Masabo et al. [24]	Malware Training Sets	API, Crypto, Locker, Zeus, Shadow brokers	Compute feature importance for feature engineering	KNN, Linear Discriminant Analysis (LDA), GB	Accuracy: 94.00%	Focus on polymorphic malware	Limited dataset, feature-engineering complexity
Nawaz et al. [25]	Drebin dataset	Android malware	Permissions and intents extraction, Network requests, API calls,	RF, NB, GB, Ada Boosting	F1 score: 0.98	High F1-scores with RF and GB	Dynamic-analysis complexity
Jung et al. [28]	AndroZoo dataset, static extraction of API calls	Not specified (general malware)	Gini importance-based method	RF	Accuracy: 96.95%	Effective feature-selection methods	Complexity in feature extraction
Low et al. [30]	Dataset provided by Ramili-2016	Advanced Persistent Threats (APT), Crypto, Zeus, Locker, Shadow Brokers	Label encoding and evidence counting	RF, DT, KNN, SVM, LSTM	Label encoding: 91.34%, Evidence counting: 94.64%	High accuracy with LSTM	High computational requirements

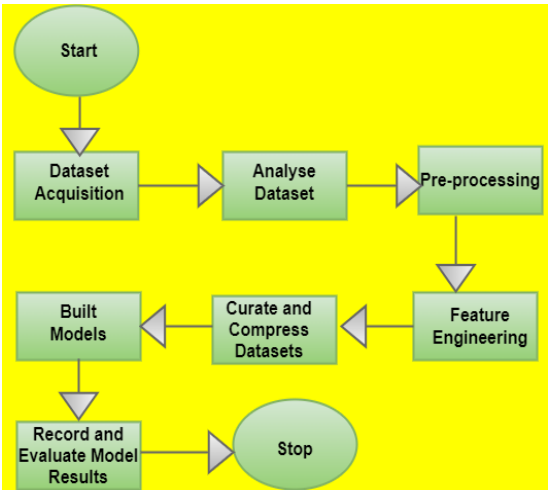

Figure 1. Methodology workflow.

3.1 Dataset Acquisition

The dataset used is called "Android Spyware-2022" [8]. The dataset was generated using PCAPDroid. It is a data-collection tool that can be installed on the Android operating system. The data consisted of five different spywares: FlexiSPY, MobileSPY, mSPY, TheWispy and uMobix; and one normal-traffic class which represents normal-smartphone traffic. Each row in the PCAP file represents a single packet. Analyzing the PCAP data involved extracting static information from each packet, such as Source IP, Destination IP, Source Port, Destination Port, Protocol type, Flags information, Acknowledgment Number and Message content. It also included recording Flow Duration (the time taken for a packet to transfer from source to destination), Packet Header Length and Packet Full Length. The information in the PCAP files is presented in Table 3.

Table 3. Available information in PCAP files.

File Name (.pcap)	Number of Packets	System Name	File Size (MB)	Data Tag
Normal_Traffic	1,04,914	Smart Phone Normal Traffic	78.81	Normal Traffic
FlexiSPY_Installation	19,793	FlexiSPY Inst	16.78	FlexiSPY Inst
FlexSPY_Traffic	35,433	FlexiSPY Traffic	22.32	FlexiSPY Traffic
Mspy Traffic- Part1	35,560	mSPY	25.94	mSPY Traffic
Mspy Traffic- Part2	20,537	mSPY	20.32	mSPY Traffic
mSPY Installation Process	12,976	mSPY	11.34	mSPY Inst
uMobix_Installation	17,312	uMobix	14.37	uMobix Inst
uMobix_Traffic	18,561	uMobix	16.28	uMobix Traffic
MobileSpy_Traffic	28,154	MobileSPY	12.76	MobileSPY Traffic
Mobilespy_Intallation_1	10,139	MobileSPY	8.41	MobileSPY Inst
TheWiSPY_Installation	58,223	TheWiSPY	53.24	TheWiSPY Inst
TheWISPY_Traffic	27,343	TheWiSPY	21.36	TheWiSPY Traffic

In this context, there are two packet types: "Installation" and "Traffic." The "Installation" type represents traffic data captured during the spyware-installation process, while the "Traffic" type represents spyware operation traffic data. Figure 2 illustrates the distribution of the six classes within the dataset. It shows that all five spyware classes overlap with the normal-traffic class. Utilizing the PCAP file information directly for classification may not be ideal, as features are not distinct for each class. Therefore, there is a need to curate new datasets with more distinct features for each class. To achieve this, prominent characteristics of each spyware must be identified and analyzed in the next sub-section.

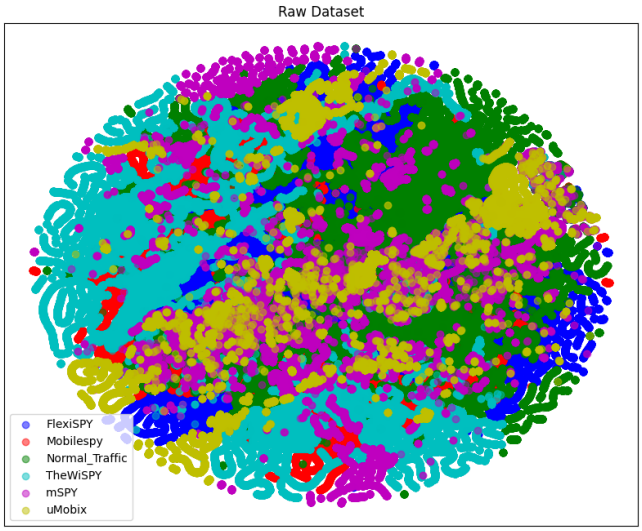

Figure 2. The distribution of the six classes.

3.2 Characteristics of Spyware Acquired from the Dataset

Table 4 presents a concise summary of each type of spyware. The Spying Scope represents different monitoring channels. The Platform highlights the language and framework used to develop spyware. The Upload represents how the data is transmitted to the Command and Control (C2C) server. Sniffing identifies sniffing strategies.

Based on the observations in Table 4, it is noted that each spyware shares similar characteristics in terms of spying scope, platform and sniffing features. This similarity will determine the next course of action for adapting the data pre-processing method.

3.3 Data Pre-processing

The process of converting every individual PCAP file into CSV format is explained in Algorithms 1 and 2. Each PCAP file represents a sample that contains information related to the corresponding spyware. Investigating the utilization of packet-flow direction in feature engineering is explored in the following section to minimize class overlap.

Table 4. Characteristics of spyware acquired from the dataset.

Spyware Classes	Spying Scope	Platform	Upload	Sniffing
FlexiSPY	a) Social-media applications b) Keylogger c) OS activity d) Update history e) Applications manifest f) Phone calls	Java	Periodic-based with fixed time interval	Event-based
MobileSPY	a) SIM Tracker b) Social-media applications c) Keylogger d) OS activity e) Update history. f) Applications manifest g) Phone calls	React Native, Java	Non-adjustable periodic	Event-based
TheWiSPY	a) SIM Tracker b) Social-media applications c)Keylogger d)OS activity e) Phone calls	React Native, Java	Adjustable periodic	Event-based
mSPY	a) Social-media applications b) Keylogger c) OS activity d) Update history. e) Applications manifest e) Phone calls	Java	Periodic-based with fixed time interval	Event-based
uMobix	a) Social-media applications b) Keylogger c) OS activity d) Update history e) Applications manifest f) Phone calls	Java	Adjustable periodic	Adjustable in terms of periodic or event-based

4. FEATURE ENGINEERING

Firstly, when observing the direction of the packet flow, two directional properties are apparent: single-direction and bi-directional. The direction of the packet flow is determined by the Source IP, Destination IP and Protocol. In this case, static features are extracted from the PCAP file including Source IP, Destination IP, Source Port, Destination Port, Protocol, Flow Duration and Packet Length.

The dynamic features include Total Forward Packets (Total Fwd Pkt), Total Backward Packets (Total Bwd Pkt), Total Length of Forward Packets, Total Length of Backward Packets, Flow Bytes per Second (bytes/s) and Flow Packets per Second (pkt/s), as well as the statistical values, such as minimum, maximum, average and standard deviation values.

Total forward and backward packets, along with the total length of forward and backward packets, are derived from the direction of packet flow and packet length. Additionally, flow bytes per second (bytes/s) and flow packets per second (pkt/s) are obtained from the direction of packet flow, flow duration and packet length.

After feature-engineering processes, the next step involves curating the two datasets: Dataset A and Dataset B.

4.1 Method for Developing Dataset A

A single-direction packet flow is utilized to curate Dataset A. It examines the IP pairs and Protocol for each row. When the Source IP, Destination IP and Protocol remain constant across two or more consecutive rows, statical measures are calculated to form new features. The features are presented in Table 5. These consecutive rows are combined into a single group, which represents a single-directional packet flow. However, if the Source IP and Destination IP remain unchanged for consecutive rows but the Protocol differs, they cannot be considered part of the same group. They will be considered as part of a different packet flow.

Table 5. Detail description of features after feature engineering.

Feature Name	Feature-engineering Process
Source (Src) IP	From PCAP file
Destination (Dst) IP	From PCAP file
Src Port	From PCAP file
Dst Port	From PCAP file
Protocol	From PCAP file
Flow Duration	Subtract the current flow start time from the last flow end time.
Total Forward (Fwd) Packets	Sum the forward packets.
Total Backward (Bwd) Packets	Sum the backward packets
Total Length of Fwd Packet	Sum the forward packet length
Total Length of Bwd Packet	Sum the backward packet length
Fwd Packet Length Min	Minimum forward packet length
Fwd Packet Length Max	Maximum forward packet length
Fwd Packet Length Mean	Average forward packet length per flow
Fwd Packet Length Std	Standard deviation of forward packet length
Bwd Packet Length Min	Minimum backward packet length
Bwd Packet Length Max	Maximum backward packet length
Bwd Packet Length Mean	Average backward packet length per flow
Bwd Packet Length Std	Standard deviation of backward packet length
Flow Bytes/s	Byte rate in a flow
Flow Pkt/s	Packet rate in a flow

Algorithm 1: Generating Dataset A. Here, p represents the previous row and n represents the next row.

Algorithm 1: Dataset A

```
1.  pcap = read (open ( pcap file))
2.  Require: ip, protocol
3.  for row number in row do
4.      if (p.IP pairs = = n.IP pairs && p.protocol = = n.protocol) then
5.          calculate feature value
6.      else
7.          move to the next row
8.      end if
9.  end for
10. function writeCsv(data, outputFile):
11. processedData = processPcap(pcapFile)
12. writeCsv(processedData, outputFile)
```

4.2 Method for Developing Dataset B

For Dataset B, the process is like Dataset A (Sub-section 4.1) with the exception of utilizing a bi-directional packet flow instead of a single-direction packet flow. Figure 3 illustrates this bi-directional flow. The Source IP in Row-1 and Row-2 subsequently becomes the Destination IP in Row-3 and Row-4. This process subsequently occurs also in Row-5. A similar process occurs for the Destination IPs as well.

Algorithm 2: Dataset B

```
1.  pcap = read (open ( pcap file))
2.  Require: ip, protocol
3.  for row number in row do
4.      if (p.source.ip ||p.destination.ip==n.source.ip||n.destination.ip && p.protocol== n.protocol) then
5.          calculate feature value
6.      else
7.          move to the next row
8.      end if
9.  end for
10. function writeCsv(data, outputFile):
11. processedData = processPcap(pcapFile)
12. writeCsv(processedData, outputFile)
```

4.3 Comparison of the Datasets

After curating Datasets A and B, Dataset A consists of 3,928 rows and Dataset B comprises 2,573 rows, while the raw dataset contains a total of 386,963 rows. Tables 6, 7 and 8 present the features and sample values for the raw dataset, Dataset A and Dataset B. It is important to note that while Table 8

includes features related to bi-directional packets, each feature is listed with respect to one source and destination, which may obscure the indication of data-flow direction. To address this, Figure 3 in sub-section 4.3 illustrates how the data-flow direction is represented in the datasets, enhancing the clarity of the bi-directional packet features.

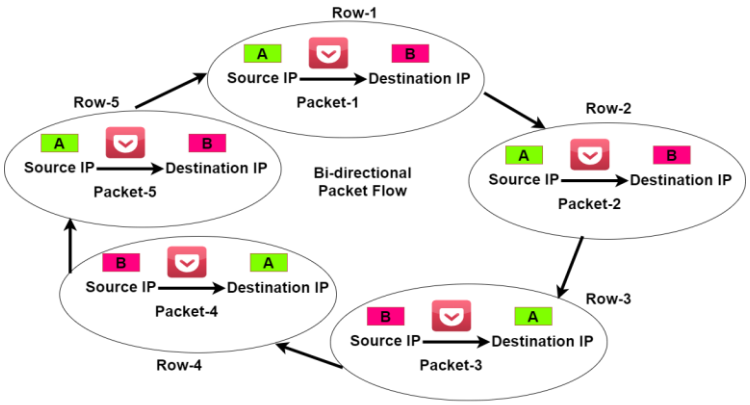

Figure 3. Bi-directional packet flow.

Table 6. Sample rows of the raw dataset.

Src IP	Dst IP	Protocol	Total Fwd. Packets	Total Bwd. Packets	Total Length of Fwd. Packet	Total Length of Bwd. Packet	Fwd. Packet Length Min.	Fwd. Packet Length Max.	Fwd. Packet Length Mean	Fwd. Packet Length Std.	Bwd. Packet Length Min.	Bwd. Packet Length Max.	Bwd. Packet Length Mean	Bwd. Packet Length Std.
10.215.173.1	161.117.185.166	TCP	1	0	60	0	60	60	60	0	0	0	0	0
10.215.173.1	157.240.195.54	TCP	1	0	60	0	60	60	60	0	0	0	0	0
8.8.8.8	10.215.173.1	DNS	0	1	0	48	0	0	0	0	48	48	48	0
8.8.8.8	10.215.173.1	DNS	0	1	0	40	0	0	0	0	40	40	40	0
10.215.173.1	157.240.195.54	TCP	0	1	0	44	0	0	0	0	44	44	44	0
157.240.195.54	10.215.173.1	TCP	1	0	40	0	40	40	40	0	0	0	0	0
10.215.173.1	104.21.81.103	TLSv1.2	1	0	43	0	43	43	43	0	0	0	0	0
157.240.195.54	10.215.173.1	UDP	0	1	0	40	0	0	0	0	40	40	40	0
142.250.200.243	10.215.173.1	TLSv1.3	0	1	0	44	0	0	0	0	44	44	44	0

Table 7. Sample rows of Dataset A.

Src IP	Dst IP	Protocol	Total Fwd. Packets	Total Bwd. Packets	Total Length of Fwd. Packet	Total Length of Bwd. Packet	Fwd. Packet Length Min.	Fwd. Packet Length Max.	Fwd. Packet Length Mean	Fwd. Packet Length Std.	Bwd. Packet Length Min.	Bwd. Packet Length Max.	Bwd. Packet Length Mean	Bwd. Packet Length Std.
10.215.173.1	161.117.185.166	TCP	23	0	6976	0	88	1472	303.30	390.05	0	0	0.00	0.00
10.215.173.1	157.240.195.54	TCP	20	0	1880	0	88	152	94.00	15.10	0	0	0.00	0.00
161.117.185.166	10.215.173.1	TLSv1.2	0	2	0	176	0	0	0.00	0.00	88	88	88.00	0.00
10.215.173.1	10.215.173.2	DNS	38	0	21492	0	88	1548	565.58	586.28	0	0	0.00	0.00
10.215.173.2	10.215.173.1	DNS	0	65	0	11408	0	0	0.00	0.00	116	436	175.51	75.25
10.215.173.1	157.240.196.60	TCP	72	0	82640	0	88	1548	1147.78	564.12	0	0	0.00	0.00
10.215.173.1	157.240.196.60	TLSv1.3	1	0	116	0	116	116	116.00	0.00	0	0	0.00	0.00
10.215.173.1	10.215.173.2	UDP	48	0	7592	0	100	1424	158.17	187.85	0	0	0.00	0.00
10.215.173.1	172.217.171.206	TLSv1.3	2	0	252	0	112	140	126.00	14.00	0	0	0.00	0.00

Figures 4 and 5 provide a comparison of the Datasets A and B and their respective values. As shown in Figure 2, there is an overlap in the values of all six classes within the raw dataset, making it challenging to differentiate between distinct clusters. Conversely, Datasets A and B (depicted in Figures 4 and 5) show minimal or no overlap among the classes, clearly distinguishing between them. These visual representations encompassed all features and depicted the distribution of the six classes.

Table 8. Sample rows of the Dataset B.

Src IP	Dst IP	Protocol	Total Fwd. Packets	Total Bwd. Packets	Total Length of Fwd. Packet	Total Length of Bwd. Packet	Fwd. Packet Length Min.	Fwd. Packet Length Max.	Fwd. Packet Length Mean	Fwd. Packet Length Std.	Bwd. Packet Length Min.	Bwd. Packet Length Max.	Bwd. Packet Length Mean	Bwd. Packet Length Std.
10.215.173.1	161.117.185.166	TCP	10	10	896	4484	88	96	89.60	3.20	88	1004	448.40	373.16
157.240.196.60	10.215.173.1	TCP	64	64	6100	79780	88	500	95.31	51.46	88	1548	1246.56	468.12
10.215.173.1	10.215.173.2	DNS	3	3	184	2800	56	64	61.33	3.77	44	1378	933.33	628.85
37.44.39.12	10.215.173.1	DNS	993	993	61816	1E+06	61	78	62.25	2.91	54	1378	1375.33	59.36
10.215.173.1	157.240.196.60	TCP	63	63	80404	7456	88	1548	1276.25	476.01	88	500	118.35	102.80
172.217.171.206	10.215.173.1	TLSv1.2	10	10	1580	8568	88	648	158.00	168.58	88	1548	856.80	628.06
74.125.206.188	10.215.173.1	TCP	14	14	2464	2748	88	488	176.00	135.51	88	736	196.29	170.65
10.215.173.1	74.125.206.188	TCP	13	13	1152	4080	88	96	88.62	2.13	108	488	313.85	144.58
10.215.173.1	10.215.173.2	UDP	22	22	3356	2520	124	320	152.55	40.96	104	124	114.55	4.76

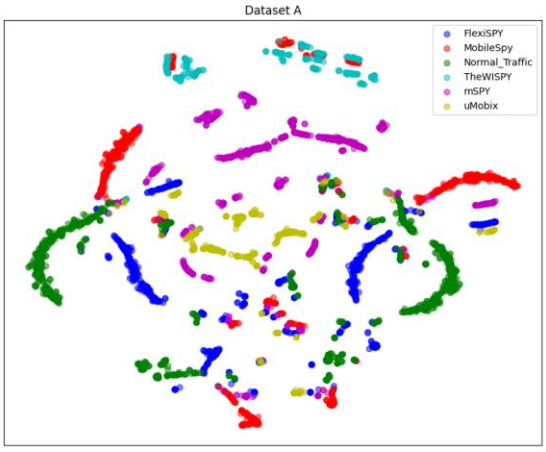

Figure 4. Dataset A.

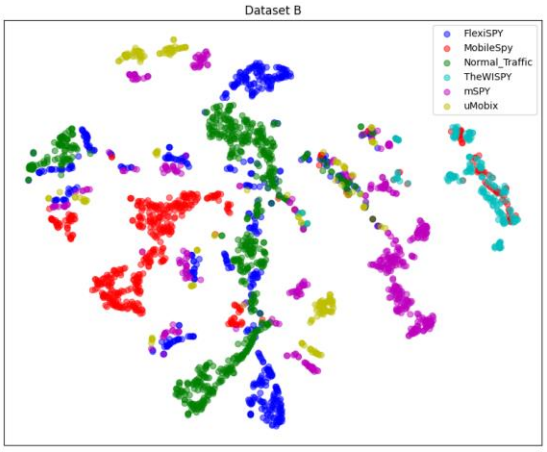

Figure 5. Dataset B.

5. Model Construction

To assess whether a dataset is appropriate for a detection model, thorough testing and analysis are crucial. The aim is to identify the most suitable ML model that aligns with the features of the curated datasets. Through analyzing the confusion-matrix values from different ML models, valuable information about the curated datasets' efficiency and performance will be obtained, enabling well-informed decisions regarding their use in detection tasks.

5.1 Classifiers

To determine the most suitable ML model, this study utilizes NB, XGBoost, RF, DT and SVM with the Radial Basis Function (RBF) kernel. DTs are recognized for their simplicity and interpretability, as they iteratively divide the data based on feature values to create a tree-like structure for classification. They can effectively handle both numerical and categorical data, making them well-suited for datasets with mixed-data types. RF improves upon DT by combining multiple trees, which enhances accuracy and reduces overfitting. SVM with RBF kernel and employing the One- *vs*-Rest (OVR) strategy excels in managing high-dimensional data by identifying optimal decision boundaries for classification. NB is particularly effective for text and categorical data due to its reliance on independence between features. Lastly, XGBoost combines gradient boosting with regularized learning to perform well in diverse datasets as an ensemble model. This research applied these traditional methods because it focused on feature engineering.

6. Results And Discussion

6.1 Results

The study evaluated classification performance using various metrics and different training-testing

dataset splits. The 70-30 split yielded the best results, which were consistent across different approaches and datasets, as illustrated in Tables 9 and 10. This consistency can be attributed to the rigorous curation of the datasets, as described in sub-sections 4.1 and 4.2. The similar results suggested the classification model's robustness across dataset configurations. A thorough review has been conducted to ensure the accuracy and reliability of the dataset-preparation and model-evaluation methods.

Table 9. Results for accuracy, precision, recall and F1-score for Dataset A.

ML Models	Accuracy (%)	Precision (%)	Recall (%)	F-1 Score (%)
DT	97.97	97.97	97.97	97.97
RF	97.97	97.97	97.97	97.97
XGBoost	96.38	96.43	96.38	96.39
SVM	99.88	99.88	99.88	99.88
NB	97.02	97.02	97.02	97.02

Table 10. Results for accuracy, precision, recall and F1-score for Dataset B.

ML Models	Accuracy (%)	Precision (%)	Recall (%)	F-1 Score (%)
DT	99.24	99.25	99.24	99.24
RF	99.24	99.25	99.24	99.24
XGBoost	99.24	99.25	99.24	99.24
SVM	99.12	99.12	99.12	99.12
NB	99.02	99.02	99.02	99.02

Figures from 6 to 9 show the difference in the evaluation metrics (including accuracy, precision, recall and F1-score) between Datasets A and B. For Dataset A, SVM demonstrated superior performance in accuracy, precision, recall and F1-score. Conversely, DT, RF and XGBoost exhibited better performance on these metrics for Dataset B.

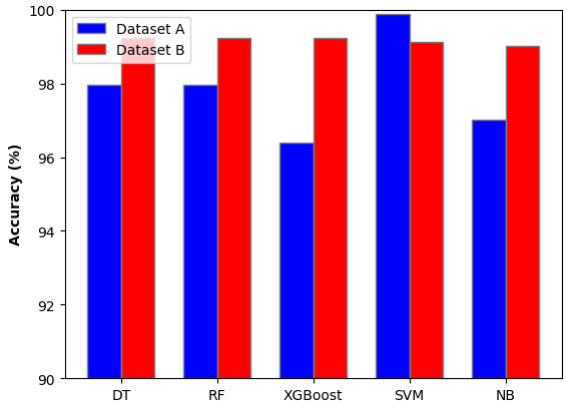

Figure 6. Evaluation of the accuracy of ML models in the context of both Dataset A and Dataset B.

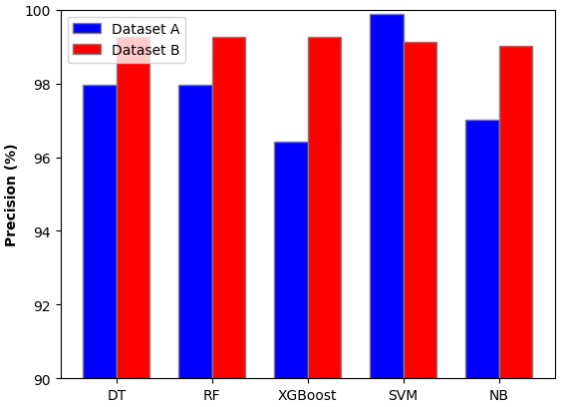

Figure 7. Evaluation of the precision of ML models in the context of both Dataset A and Dataset B.

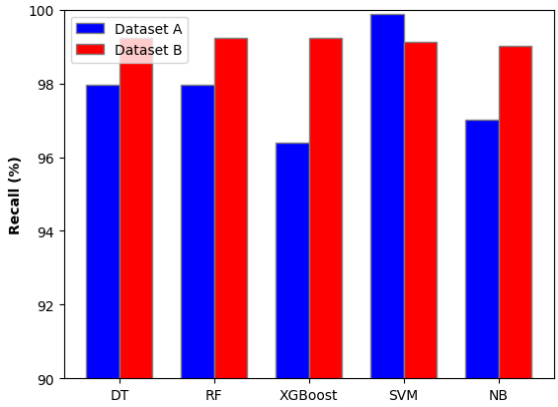

Figure 8. Evaluation of the recall of ML models in the context of both Dataset A and Dataset B.

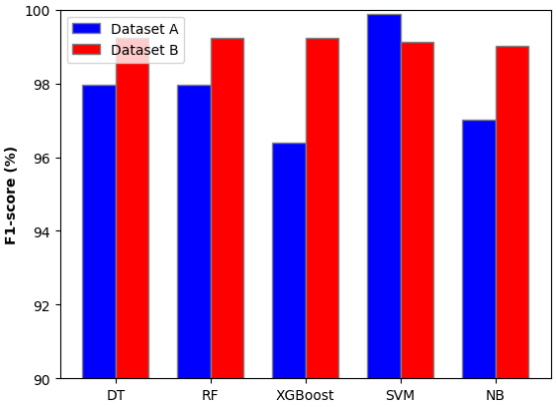

Figure 9. Evaluation of the F1-score of ML models in the context of both Dataset A and Dataset B.

6.2 Discussion

The comparative evaluation aimed to enhance the classification of spyware. Table 11 presents a comparison between the outcomes of earlier studies and those of this research, all utilized on the same dataset. M. Naser and Q. A. Al-Haija [14] categorized two types of spyware: FlexiSPY and MobileSPY, while [8] classified five varieties: FlexiSPY, MobileSPY, TheWiSPY mSPY and uMobix without extracting new features from the datasets used in their research. Feature engineering involved extracting new features based on packet-flow direction, IP pairs and Protocol information to improve spyware classification by identifying five distinct classes.

Table 11. Comparison of classification accuracy.

Research Work	ML Scheme	Dataset	Spyware Classes	Accuracy
Qabalin et al. [8]	DT	Android Spyware-2022 [8]	5	79.00%
M. Naser and Q. A. Al-Haija [14]	FDT	Android Spyware-2022 [8]	2	98.00%
T. N. AlMasri and M. A. N. AlDalaien. [31]	RF	Android Spyware-2022 [8]	5	92.00%
This Work	SVM	Dataset A	5	99.88%
	RF, DT, XGBoost	Dataset B	5	99.24%

After the curation process, both Datasets A and B showed improved performance compared to previous studies in identifying spyware. M. Naser and Q. A. Al-Haija [14] achieved strong results in spyware identification with only two spyware classes, but positive outcomes were achieved by performing well across five different spyware classes. By having more classes, the model had to comprehend more detailed patterns and characteristics associated with each type of spyware. This higher level of detail enabled the model to better discriminate, leading to improved accuracy. Other researchers utilized features directly from the raw dataset without considering directional properties, while this research focused on features utilizing packet flow direction. This approach led to the creation of more pertinent features related to different types of spyware behaviours.

The engineered features of curated Datasets A and B were highly effective, because they incorporated directional properties. These properties facilitated a deeper understanding of network behaviour, aiding in anomaly detection, optimization support and enhancing network analysis for modelling objectives and spyware classification [32]. The data-flow direction enhanced anomaly detection and network-performance optimization by mapping expected sequences, standard frequency, volume of data transmissions and common paths taken by data packets. However, some of these features, although characterized, are relatively trivial. The major contribution of this study is curating two new datasets based on directional properties. Though both datasets have the same features, the values of each feature for both datasets were different because of directional properties. Engineered features included directional properties within the network data for in-depth insight into behavioural baselines, protocol analysis, traffic volume, directionality and time-based patterns. Because of the complex nature of spyware, this method was crucial. By understanding the data flow direction, accurate behaviour modeling could distinguish between normal and suspicious activities with greater precision. The detailed analysis of packet-flow direction and its impact on spyware classification is the novelty of this study. Unlike previous studies, this study insights into data movement were provided by directional properties, which helped identify unusual patterns that signify potential threats or inefficiencies. Unusual data paths represented the potential threats, like exfiltration by spyware. The model also identified network inefficiencies, such as sub-optimal routing or congestion. This novel approach focuses on these directional properties, which provides the model clearer view of network dynamics for early performance optimization and threat detection [33].

The derived features: Total Fwd Pkt and Total Bwd Pkt from IP pairs, Ports and Protocol; provided valuable insights into the source and destination of network traffic with communication protocols (e.g. TCP, UDP). This improved pattern recognition, device-connection analysis and anomaly detection in both TCP and UDP traffics by identifying missing packets or abnormal activities (out-of-order sequences) by recognizing predictable packet sequences during normal activities, such as connection setup (SYN-ACK-ACK) during data transfer. Anomalies in UDP traffic included unexpected increases in packet rate or unusually large packet sizes. Additionally, unusual patterns in destination ports were

observed, which could indicate a DDoS attack. Normal UDP packets consisted of independent, sequenced packets typically used for real-time applications, such as DNS queries or streaming [34].

Metrics like packet-length statistics (minimum, maximum, mean and standard deviation) and flow duration (Flow Bytes/s and Flow Pkt/s) were analyzed to understand the transmission rates and detect anomalies (e.g. unusual packet sizes, irregular transmission rates and unexpected flow durations) in network traffic. An unusually large maximum packet size could indicate data aggregation before transmission, which might be normal for certain applications (e.g. video-streaming services, file-transfer protocols, cloud-storage services and backup solutions), but suspicious for others. The average packet size helped understand the typical packet load on the network. A sudden increase in the mean packet size indicated bulk data transfers. Standard deviation measured the variability in packet sizes. High variability suggested a mix of different types of traffic represented suspicious, while low variability indicated uniform traffic represented normal. A higher Flow Bytes/s rate indicated a high-volume data transfer, which could be legitimate (e.g. video-streaming) or suspicious (e.g. data exfiltration). Packet-length statistics were effective for detecting anomalies by observing baseline establishment, deviation detection and statistical methods [35]. These detail analyses helped identify those unusual patterns which were crucial for maintaining network security and efficiency.

Combining packet-length statistics with flow duration provides detailed analysis encompassing short-lived and long-lasting interactions. Flow metrics like Flow Bytes/s and Flow Pkt/s helped understand data-transmission rates, aiding in identifying abnormal patterns. Integrating packet length statistics with flow duration enables differentiation between short-lived spikes and sustained high-traffic periods, enhancing the understanding of network-flow dynamics.

Traditional ML models incorporated directional properties into their feature sets to more accurately distinguish between normal and anomalous behaviours. For instance, RF and DT benefited from the added granularity in their decision-making processes, while SVM could better separate data points in the feature space. NB, with its probabilistic approach, could more effectively categorize behaviours based on the directional data. This improved recall and precision of anomaly detection, because the algorithms were configured to recognize patterns specific to single-direction and bi-directional flows, yielding more reliable and accurate classification with better detection of anomalies.

Anomaly identification techniques focussed on analyzing unique behaviours within bi-directional and single-direction flows. Understanding the differences between these traffic patterns was important not only for anomaly detection, but also efficient resource allocation. It helped improve tactics by adapting feature sets specific to each type of flow. Targeted flow behaviours contributed to improving data-classification accuracy by reducing false-negative and false-positive results. The model could more precisely identify anomalies by focusing on each flow type's unique characteristics. Also, it reduced false alarms, which improved precision. A higher recall rate for anomaly detection could be achieved by analysing traffic patterns. A higher F1-score was achieved by the balanced improvement in both precision value and recall value, which indicated better overall performance in anomaly detection [36].

By observing dynamic load balancing, fault prediction, forecasting of traffic, protocol routing and the adaptive quality of service, these algorithms addressed bottlenecks and network inefficiencies. Those ensured that the network operated smoothly. As a result, the need for significant processing resources to solve problems after they occurred was reduced. For that reason, the overall processing time for detecting anomalies was shortened, which improved the response time [37].

7. CONCLUSIONS

Engineering features based on the directional properties that captured detailed characteristics of network-traffic behaviour. This enabled the model to identify specific patterns to bi-directional and single-direction traffic, indicating various types of network threats or activities. The high F1-score, recall, accuracy and precision achieved with these features demonstrated their effectiveness in accurately classifying network traffic. These metrics also highlighted their importance in detecting anomalies, which was important for ensuring the security and reliability of network infrastructure.

The investigation analyzed the impact of packet directionality on spyware classification. This was done through the curation of datasets focusing on directional properties. A specific emphasis was placed on IP pairs and Protocol. The analysis found that considering the directional properties

significantly improved spyware classification. RF, NB, SVM, DT and XGBoost were constructed and compared between the two curated datasets. The findings suggested that DT, RF and XGBoost performed better for Dataset B, while SVM showed better performance for Dataset A. These ML approaches demonstrated potential in spyware classification, but further improvements are needed to enhance the model, so that future work should integrate more spyware types with larger number of samples and explore advanced feature-selection and deep-learning techniques. The limited types of spyware and the small number of samples in the dataset represent limitations, so expanding them could improve detection mechanisms. Evaluating the model's performance in diverse real-world scenarios and incorporating realistic benign-traffic data could enhance its ability to distinguish between malicious and benign activities, providing a practical security solution. Integrating real-time data processing and adaptive learning could also be valuable directions for future research.

ACKNOWLEDGEMENTS

This research is supported by TM Research & Development Grant (TM R&D), MMUE/220028.

REFERENCES

[1] T. Munusamy and T. Khodadi, "Building Cyber Resilience: Key Factors for Enhancing Organizational Cyber Security," Journal of Informatics and Web Engineering, vol. 2, no. 2, pp. 59-71, 2023.

[2] M. Al-Hashedi, L.K. Soon, H. N. Goh, A. H. L. Lim and E. G. Siew, "Cyberbullying Detection Based on Emotion," IEEE Access, vol. 11, pp. 53907-53918, 2023.

[3] R. Thangaveloo et al., "Datdroid: Dynamic Analysis Technique in Android Malware Detection," Int. J. on Advanced Science, Engineering and Information Technology, vol. 10, no. 2, pp. 536-541, 2020.

[4] T.A.A. Abdullah, W. Ali, S. Malebary and A. A. Ahmed, "A Review of Cyber Security Challenges: Attacks and Solutions for Internet of Things-based Smart Home," Int. J. of Computer Science and Network Security, vol. 19, no. 9, pp. 139-146, 2019.

[5] A. S. Grillis, "What is Spyware?" [Online], Available: https://www.techtarget.com/ searchsecurity/definition/spyware, Dec. 12, 2023.

[6] S. S. Rawat and A. K. Mishra, "Review of Methods for Handling Class-imbalanced in Classification Problems," arXiv preprint, arXiv: 2211.05456, 2022.

[7] M. Botacin et al., "On the Security of Application Installers and Online Software Repositories," Proc. of the 17th Int. Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA2020), pp. 192-214, Lisbon, Portugal, 2020.

[8] M. K. Qabalin, M. Naser and M. Alkasassbeh, "Android Spyware Detection Using Machine Learning: A Novel Dataset," Sensors, vol. 22, no. 15, pp. 5765-5790, 2022.

[9] Google Play, "PCAPdroid-Network Monitor Apps," [Online], Available: https://play.google.com/apps/, Jan. 08, 2024.

[10] M. Conti, G. Rigoni and F. Toffalini, "ASAINT: A Spy App Identification System Based on Network Traffic," Proc. of the 15th Int. Conference on Availability, Reliability and Security, Article no. 51, pp. 1-8, DOI:10.1145/3407023.3407076, August 2020.

[11] WireShark, Go Deep, [Online], Available: https://www.wireshark.org/, Dec. 12, 2023.

[12] Google Play, "DroidBox Mikrotik Config Tool-Apps," [Online], Available: https://play.Google.com/ store/ apps, Dec. 12, 2023.

[13] Google, "Google Photos," [Online], Available: https://www.google.com/photos/about/, Jan. 08, 2024.

[14] M. Naser and Q. A. Al-Haija, "Spyware Identification for Android Systems Using Fine Trees," Information, vol. 14, no. 2, pp. 1-10, 2023.

[15] D. Noetzold et al., "Spyware Integrated with Prediction Models for Monitoring Corporate Computers," Preprints.org, vol. 1, DOI: 10.20944/preprints .202301.0580.v1, 2023.

[16] F. Pierazzi, R. Emilia, R and I. V. S. Subrahmanian, "A Data-driven Characterization of Modern Android Spyware," ACM Transactions on Management Information Systems, vol. 11, pp. 1-38, 2020.

[17] VirusTotal-Home, [Online], Available: https://www.virustotal.com/gui/home/, Dec. 07, 2023.

[18] V. Mahesh and S. D. KA, "Detection and Prediction of Spyware for User Applications by Interdisciplinary Approach," Proc. of 2020 Int. Conf. on Computational Intelligence for Smart Power System and Sustainable Energy (CISPSSE), DOI: 10.1109/CISPSSE49931.2020.9212222, Keonjhar, India, July 1-6, 2020.

[19] O. F. Catak, "API Call Based Malware Dataset," [Online], Available: https://www.kaggle.com./datasets/ focatak/ malapi2019, Dec. 08, 2019.

[20] Kaggle, "Your Machine Learning and Data Science Community," [Online], Available: https://www.kaggle.com/, Nov. 01, 2024.

[21] N. Zahan, P. Burckhardt, M. Lysenko, F. Aboukhadijeh and L. Williams, "MalwareBench: Malware

Samples Are Not Enough," Proc. of 2024 IEEE/ACM 21st Int. Conf. on Mining Software Repositories (MSR), pp. 728-732, DOI: 10.1145/3643991.3644883, April 2024.
[22] Z. Zhang, P. Qi and W. Wang, "Dynamic Malware Analysis with Feature Engineering and Feature Learning," Proc. of 34th AAAI Conf. on Artificial Intelligence (AAAI-20), pp. 1210-1217, April 2020.
[23] D. Gibert et al., "Fusing Feature Engineering and Deep Learning: A Case Study for Malware Classification," Expert Systems with Applications, vol. 207, pp. 117957-117974, 2022.
[24] E. Masabo, K. S. Kaawaase, J. S. Otim, J. Ngubiri and D. Hanyurwimfura, "Improvement of Malware Classification Using Hybrid Feature Engineering," SN Computer Science, vol. 1, pp. 1-14, 2020.
[25] A. Nawaz, "Feature Engineering Based on Hybrid Features for Malware Detection over Android Framework," Turkish J. of Computer and Mathematics Education, vol. 12, no. 10, pp. 2856-2864, 2021.
[26] M. Humayun, N. Z. Jhanjhi and M. Z. Alamri, "Smart Secure and Energy Efficient Scheme for E-Health Applications Using IoT: A Review," Int. J. of Computer Science and Network Security, vol. 20, no. 4, pp. 55-74, 2020.
[27] Apktool, "Apktool," [Online], Available: https://apktool.org/, Dec. 01, 2024.
[28] J. Jung, J. Park, S. J. Cho, S. Han, M. Park and H. H. Cho, "Feature Engineering and Evaluation for Android Malware Detection Scheme," J. of Internet Technology, vol. 22, no. 2, pp. 423-440, 2021.
[29] K. Allix et al., "AndroZoo: Collecting Millions of Android Apps for the Research Community," Proc. of the 13th Int. Conf. on Mining Software Repositories (MSR), pp. 468-471, Austin, USA, May 2016.
[30] M. X. Low et al., "Comparison of Label Encoding and Evidence Counting for Malware Classification," Journal of System and Management Sciences, vol. 12, no. 6, pp. 17-30, 2022.
[31] T. N. AlMasri and M. A. N. AlDalaien, "Detecting Spyware in Android Devices Using Random Forest," Proc. of the 2023 Int. Conf. on Advances in Comput. Research (ACR'23), pp. 294-315, 2023.
[32] N. Ben-Asher, S. Hutchinson and A. Oltramari, "Characterizing Network Behavior Features Using a Cyber-security Ontology," Proc. of MILCOM 2016-2016 IEEE Military Communications Conf., pp. 758-763, Baltimore, USA, November 2016.
[33] S. Misra, M. Tan, M. Rezazad, M. R. Brust and N. M. Cheung, "Early Detection of Crossfire Attacks Using Deep Learning," arXiv preprint, arXiv: 1801.00235, 2017.
[34] L. Zhou et al., "DDOS Attack Detection Using Packet Size Interval," Proc. of the 11th Int. Conf. on Wireless Comm., Networking and Mobile Computing (WiCOM), pp. 1-7, Shanghai, China, 2015.
[35] A. Iorliam et al., "Flow Size Difference Can Make a Difference: Detecting Malicious TCP Network Flows Based on Benford's Law," arXiv preprint, arXiv: 1609.04214, 2016.
[36] N. Davis, G. Raina and K. Jagannathan, "A Framework for End-to-End Deep Learning-based Anomaly Detection in Transportation Networks," Transportation Research Interdisciplinary Perspectives, vol. 5, pp. 100-112, 2020.
[37] M. Kuchnik et al., "Plumber: Diagnosing and Removing Performance Bottlenecks in Machine Learning Data Pipelines," Proc. of Machine Learning and Systems, vol. 4, pp.33-51, 2022.

ملخص البحث:

تفتقر الطُّرق الراهنة لتصْنيف برامج التّجسُّس إلى الفعالية لغياب مجموعات البيانات جيدة التّنظيم، وبخاصّةٍ مجموعات البيانات ذات الخصائص الاتّجاهية. في هذه الورقة، يتمّ استكشاف فعالية الخصائص المرتبطة بالاتّجاهية في مجموعات البيانات من أجل التّصنيف، من خلال خصائص تجري هندستها من الخصائص الموجودة في مجموعات البيانات. وتعمل هذه الدّراسة على تنظيم اثنتين من مجموعات البيانات. المجموعة A تحتوي على خصائص يتمّ استخلاصُها من تدفّقات حُزَم أحادية الاتّجاه، بينما تحتوي المجموعة B على خصائص يتمّ استخلاصها من تدفّقات حُزَم ثنائية الاتّجاه. وتجدر الإشارة إلى أنّ عملية التّصنيف من خلال تلك الخصائص تتمّ باستخدام مجموعةٍ مختارة من خوارزميات التّصنيف. وقد حقّق مصنِّف (SVM) الدّقّة الأعلى بالنّسبة لمجموعة البيانات A 99.88%، في حين ذهبت الدّقّة الأعلى بالنّسبة لمجموعة البيانات B إلى مُصنِّفات (RF و DT و XGBoost) 99.24%.

وبمقارنة الطّريقة المقترحة في هذا البحث مع غيرها من الطّرق الواردة في أدبيات الموضوع، تبيَّن أنّ الخصائص الاتجاهية في مجموعات البيانات من شأنها أنْ تحقّق تحسيناتٍ من حيث الدّقّة في تصنيف برامج التّجسُّس.