BLEGuard: Hybrid Detection Mechanism for Spoofing Attacks in Bluetooth Low Energy Networks (Student Abstract)

Supplement Report for AAAI

Hanlin Cai[1], Yuchen Fang[1], Yuan Meng[2], Zhezhuang Xu[2]
[1] National University of Ireland Maynooth, Maynooth, Co. Kildare, Ireland
[2] School of Electrical Engineering and Automation, Fuzhou University, Fuzhou, China
Corresponding E-mail: hanlin.cai.2021@mumail.ie, zzxu@fzu.edu.cn

Contents

0 README for our Supplements

The document structure tree below outlines the primary components of our submitted supplementary materials. In addition, our complete code and data will be available in the Github repository[1].

```
.
├─dataset                               # sample set of our data
│  ├─profiles                           # sample data of BLE device.
│  └─RSSI                               # partly RSSI feature data recorded.
├─src                                   # source code
│  ├─blemonitor                         # BLE device monitor code.
│  ├─machine-learning                   # relative code includes: SVM, TextCNN etc.
│  │  ├─bayes
│  │  ├─cnn-svm
│  │  ├─cnn-text-classification-pytorch
│  │  ├─cnn-text-classification-tf
│  │  ├─knn
│  │  ├─logistic
│  │  ├─lstm
│  │  ├─random-forest
│  │  ├─svm
│  │  └─tcn
│  └─ubertooth                          # fixed ubertooth code for additional feature.
├─static                                # static resource.
├─README.md                             # README file for our supplements.
└─Supplement-Report.pdf                 # The report you are reading now.
```

This supplement report is organized as follows. In **Section 1**, we provide an overview of the BLE networks and the effects of spoofing attacks. In **Section 2**, we elaborate on our experimental setup and the specific information of the deployment environment and used devices. In **Section 3**, we present the process of attack simulation and the building of advertising datasets. In **Section 4**, we introduce the BLEGuard system, a hybrid detection mechanism that combined cyber-physical features judgment and learning-based methods. Ultimately, **Section 5** discusses the preliminary experimental results and our future works.

1 BLE Network Basics

In this section, we will briefly introduce the background and characteristics of Bluetooth Low Energy (BLE), as well as the fundamentals of spoofing attack and its effects to the BLE devices and networks.

1.1 Background and Specificity

BLE is one of the most widely used protocols for Internet of Things devices (e.g., smart lights, smart sensors and smart thermostats). It is expected that the number of BLE devices will reach 6.5 billion by 2025[2]. However, these devices are vulnerable to spoofing attacks since many of them have limited I/O capabilities and do not support firmware updates. To address the security challenge, an out-of-the-box detection method has been proposed, leveraging BLE's cyber-physical features to defend against advanced spoofing attackers without requiring any interference or updates[3]. Additionally, several works rely on learning-based techniques to identify the malicious packages within BLE networks. A learning framework that integrates supervised and unsupervised learning was suggested to classify packets as benign or malicious inside each suspicious batch with high precision[4]. Nevertheless, most existing methods struggle to strike a balance high accuracy, low false alarm rate and low detection cost, which limits their applicability to a narrow range of scenarios.

Considering the specificity of BLE networks, in this research, four representative cyber-physical features were utilized for detection algorithm design and learning models training[3, 5], as follows:

- Used Channel Numbers (**UCN**): the data channels number used during the exchange of the BLE packets.
- Advertising Interval (**INT**): the time gap between two continuous packets on the same advertising channel.
- Received Signal Strength Indicator (**RSSI**): the signal-to-noise ratio value available in packets exchange.
- Carrier Frequency Offset (**CFO**): the unique offset between the designated and actual carrier frequencies.

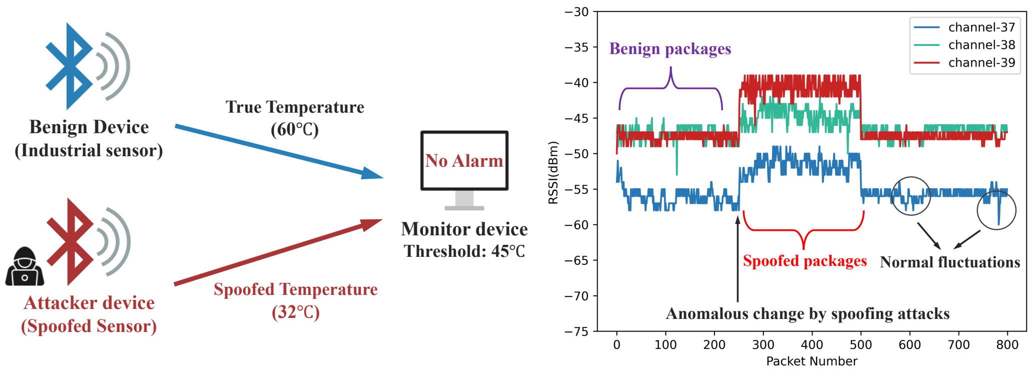

Figure 1: Left: Spoofing attack in BLE sensor network. Right: Observed RSSI values during attack simulation.

1.2 Spoofing Attack

The spoofing attack is a type of cybersecurity attack where an attacker impersonates a legitimate BLE device or entity on the network. In such an attack, the attacker typically pretends to be a trusted BLE device by using forged information, such as a spoofed MAC address or other identifying details, as shown in **Figure 1 (Left)**. In the spoofing attack scenario, the cyber-physical features of BLE network will undergo noticeable affected, resulting in significant deviations from the benign scenario. For instance, the anomalous shift in the RSSI values of the advertising packets indicates the presence of spoofing attacks, as shown in **Figure 1 (Right)**.

2 Experiment Setup

In this section, we will elaborate on the process of building our BLE network testbed, as well as the specific information of the deployment environment and related devices. To ensure the reliability and reproducibility of our experiments, all hardware devices and software platforms utilized are openly available on the Internet.

2.1 Overview

In a word, the testbed environment can be categorized into four parts: (i) BLE devices, (ii) user devices, (iii) attacker platforms, and (iv) network sniffers. **Table 1** illustrates all components used in BLEGuard testbed.

Table 1: Components of BLEGuard testbed.

Component	Description	Devices Example
BLE devices	Used to build the BLE cyberspace environment	nRF51822, DA14580 chips
User devices	Used to connect and simulate usage scenarios	Apple laptop, Android phone
Attacker platforms	Used to launch advanced spoofing attacks	CSR dongle, Lenovo laptop
Network sniffers	Used to capture network advertising package	Raspberry Pi, Ubertooth One

2.2 Deployment Environment

We deployed the testbed in a physical environment: a $15m \times 15m$ office hosting several students in 18 cubicles. We divided the office space into grids of $1m \times 1m$. The office presents a typically complicated and noisy indoor environment for determining the detection performance of **BLEGuard**. While recording RF signals within the sniffers' reception range, we observed substantial channel interference originating from 40 other Bluetooth or BLE-equipped devices (including headsets, sensors, smartphones, and laptops), dozens of Wi-Fi access points, and two microwave devices. Furthermore, we noted that sudden movements of individuals within the office had a significant impact on channel conditions within the monitored environment.

2.3 BLE Devices

For the implementation of our network testbed, we employed **sixteen** widely used BLE devices, covering a variety of Bluetooth chips (such as nRF51822 and DA14580), as shown in **Table 2**. Eventually, we selected **twelve** of these devices, which exhibited relative stability, to gather our advertising datasets. Furthermore, these BLE devices represent a diverse array of mainstream BLE applications (e.g., temperature sensors, locks, and smoke detectors) from reputable manufacturers including Eve, Tahmo, Nest, Ilumi, Apple and Xiaomi.

2.4 User Devices

The user devices are used to connect with BLE devices to simulate network scenarios under normal usage. We also implemented the monitor on the user laptop and PC to interact with sniffers and receive the datasets. **Table 3** illustrates the user devices utilized in our network environment. Besides, it was mentioned above that we have about 40 other network devices in our deployment office, but we only collect the advertising datasets from the BLE devices we deployed, and we do not record any data from other unknown devices.

Table 2: **BLE devices used in BLEGuard testbed.**

Device ID	**Device Name**	**Manufacturers**	**Device Type**	**Link for Details**
1	Smoke detector	Nest	Sensor	Link [1]
2	Indoor Camera	Nest	Camera	Link [2]
3	Nest Mini	Nest	Audio Stream	Link [3]
4	Temperature Sensor	SensorPush	Sensor	Link [4]
5	Temperature Sensor	Tahmo	Sensor	Link [5]
6	Smart Lock	August	Smart Home	Link [6]
7	Door&Window Sensor	Eve	Smart Home	Link [7]
8	Button Remote Control	Eve	Smart Home	Link [8]
9	Energy Socket	Eve	Energy	Link [9]
10	Smart Light Bulb	Ilumi	Smart Home	Link [10]
11	Mi Smart Scale	Xiaomi	Scale	Link [11]
12	Mi Band 8	Xiaomi	Smart Band	Link [12]
13	Mijia Hygrometer 2	Xiaomi	Sensor	Link [13]
14	Key Finder	iHere	Locator	Link [14]
15	Otbeat Sport Band	Otbeat	Heart Rate Monitor	Link [15]
16	HomePod 2	Apple	Audio Stream	Link [16]

Table 3: **User devices used in BLEGuard testbed.**

Device Name	**Operating System**	**MAC Address**	**Link for Details**
Lenovo V15-IIL	Windows 10 Pro	0d:76:9a:3f:e7:0b	Link [1]
MacBook Pro M1	macOS 13.1	0f:2e:4d:1a:8c:5b	Link [2]
Google Pixel 7	Android 13	08:5b:3c:2f:a1:6d	Link [3]
iPhone 13	iOS 16	0a:9f:7e:2d:6b:8f	Link [4]
Surface Laptop 5	Windows 11	06:3d:1f:7e:a8:4c	Link [5]
Dell 7050 PC	Windows 10 Pro	0b:4a:5e:2c:9f:7d	Link [6]

2.5 Attacker Platforms

To generate multiple spoofing attacks, we deployed four distinct types of attacker platforms, each comprising three identical samples situated at different locations. The specifics of the attacker platforms used in our testbed are outlined in **Table 4**. We opted for these platforms due to their accessibility, programmability, and utilization of various transmit power values. In addition, we provided the MAC address for each device to distinguish between identical devices performing different functions, since we have multiple duplicate copies.

Table 4: Attacker platform used in BLEGuard testbed.

Device Name	Operating Form	MAC Address	Link for Details
Lenovo 15IIL laptop	Using Mirage tool	04:6c:59:05:9c:8a	Link [1]
CSR 4.0 BT dongle	Combined with PC	02:42:07:cd:65:a4	Link [2]
HM-10 development board	Combined with PC	02:42:13:02:c7:f0	Link [3]
CYW920735 development board	Using Ostinato tool	00:16:3e:0d:95:65	Link [4]

2.6 Network Sniffers

We positioned three network sniffers at specific grid locations within the office, with each location having an identical copy of the sniffer. These sniffers were built using Raspberry Pi equipped with Ubertooth One, an open platform suitable for secondary development. The Ubertooth One was utilized for capturing network packets and cyber-physical feature information, while the Raspberry Pi managed the transmission of datasets to the user (monitor) devices. The total cost for establishing such a sniffer configuration is approximately $100.

Furthermore, we plan to incorporate other types of sniffers for advertising dataset collection, aiming to reduce the deployment costs of BLEGuard system and minimize the data errors as much as possible. **Table 5** provides several available combinations of devices to implement a BLE network sniffer. We noted that the user laptops can also be used to capture network information using nRF Connect Software without additional overhead.

Table 5: Several combinations of devices to implement the network sniffer.

Communication Platform	Network Capture Tool	Total Cost	Link for Details
Raspberry Pi (Linux 5.4)	Ubertooth One	About $100	Link [1] & [2]
Raspberry Pi (Linux 5.4)	Acxico 1Pcs USB CC2531	About $75	Link [3] & [4]
Raspberry Pi Pico (Linux 4.14)	MDBT42Q-DB-32	About $20	Link [5] & [6]
Google Pixel 7 (Android 13)	nRF Connect Software	(User device)	Link [7] & [8]
Apple MacBook (macOS 13.1)	nRF Connect Software	(User device)	Link [9] & [10]
Dell 7050 PC (Windows 10)	nRF Connect Software	(User device)	Link [11] & [12]

3 Attack Simulation & Datasets Building

In this section, we will present the process of network testbed implementation, spoofing attack simulation, and advertising datasets building. We will provide our datasets and codes in our GitHub Repository[1] to ensure the reliability and reproducibility of our experiments.

3.1 Testbed Implementation

As mentioned above, we implemented BLEGuard system utilizing user monitor devices and network sniffers within the testbed environment. In this process, user monitor devices are used to communicate with sniffers and organize the advertising datasets utilizing Wireshark tool[6]. All network activities including the interaction between monitor and sniffer, advertising packages exchange, and cyber-physical features collection were implemented by around 4000 lines Python code. We have included the main program in our supplement file.

3.2 Attack Simulation

Table 4 above shows the attacker platforms used in BLEGuard testbed. We deployed these four devices, with each having three copies, across twelve distinct locations to enrich the cyber-physical feature datasets for RSSI and CFO. **Figure 2** illustrates the deployment within the testbed environment, where blue circles represent sixteen BLE devices, green squares depict six sniffers, and red triangles signify twelve attacker platforms. In the attack scenario, we conducted the attack by employing cloned identity information to replace the normal network connections between BLE devices and user devices. These clones were created using USB dongles, Mirage tool[7], and Ostinato tool[8]. We note that advanced attackers can easily exploit BLE's vulnerabilities to read, modify, and write settings for these network devices, potentially leading to a range of security issues.

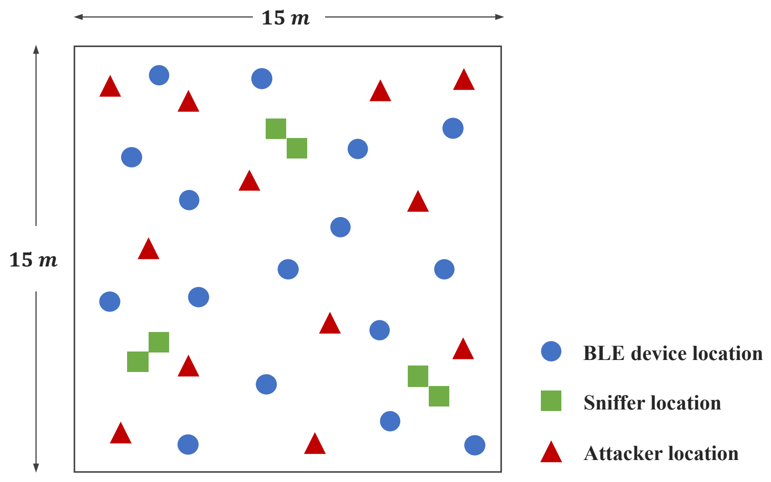

Figure 2: Locations of three types of devices in BLEGuard testbed

3.3 Datasets Building

Regarding the building of our dataset, we collected normal advertising packets from each BLE device for approximately 8 hours (5 hours during the day and 3 hours at night). For each attacker platform located in different positions, we gathered malicious packets for roughly 20 minutes. At present, our dataset comprises 902,980 advertising packets, with benign packets constituting 85.2% and malicious packets comprising 14.8%. **Table 6** shows the sample data of BLE devices recorded during the collecting phase.

Table 6: Sample data of BLE devices recorded during the collecting phase.

Device ID	MAC Address	Advertising Data	Advertising Pattern	Low Bound of *INT* (ms)
1, n075w	02:4f:19:02:c8:f2	06 09 4e 30 39 37 57	Intermittent	1282
2, b3s97	08:7f:2a:1c:5e:b3	0a 18 54 28 6d 81 45	Continuous	1376
3, tfb7a	0e:2b:65:9f:3d:a7	0f 3a 73 59 21 4c 60	Continuous	923
4, r672y	0b:19:0c:7f:8e:2d	0d 0e 76 45 93 82 1a	Continuous	1149
5, o3aqe	01:98:4d:6a:f7:3c	04 13 28 6b 94 3e 77	Intermittent	1423
6, 7wzfy	0c:57:6f:8b:a2:4e	07 23 58 17 9f 02 3d	Continuous	1445
7, 3v9nl	0a:aa:32:ef:1d:80	0b 88 49 2e 76 51 3C	Intermittent	1055
8, 42xpz	03:ef:1a:58:c4:9d	05 47 9d 0a 86 2b 1e	Intermittent	1290
9, qtpv5	09:6d:84:23:5f:ca	0c 6f 3b 0c 98 74 21	Continuous	1178
10, l348y	04:31:2e:7d:0f:6a	0e 67 a1 05 9d 3f 7a	Continuous	1264

Table 7: The hyperparameters of the temporal convolutional network network.

Hyperparameters	Value
Optimizer	RMSprop
Learning rate	5e-4
Kernel size	8
Number of filters	9
Loss function	MSE
Hidden units	10
Dropout rate	0.05
Gradient clipping	1

Table 8: The hyperparameters of the text-convolutional neural network.

Hyperparameters	Value
Optimizer	Adam
Learning rate	1e-4
Batch size	50
Epoch number	50
Loss function	binary cross-entropy
Validation metric	Accuracy
Validation split	0.2
Deep learning framework	Tensorflow 1.13.1, Keras 2.2.4, Gensim (WordVec) 3.7.1

4 Detection Mechanism

In this section, we will discuss our proposed BLEGuard system, a hybrid detection mechanism combined features judgment algorithm with unsupervised and supervised learning technologies. The primary programs are included in the submitted supplement, and the complete models are accessible in our GitHub repository[1].

4.1 Features Judgment Algorithm

The specificity features of advertising packages can be used to determine malicious activities within BLE networks. The abrupt changes of UCN and INT can be attributed to the occurrence of potential attacks. Additionally, to detect the advanced spoofing attacks, RSSI and CFO are utilized to implement a continuous judgment mechanism. In BLEGuard, three network sniffers are utilized to collect the value of RSSI and CFO in the lookback window to infer valid ranges, and then inspect relevant values of advertising packets in the observation window. Once the system detects an abnormality in either of these two features, an alarm is raised. Currently, the detection algorithm is still under development and testing, and whole program will be uploaded to the our GitHub repository[1] after final refinement and verification.

4.2 Unsupervised Reconstruction Model

The reconstruction method involves learning the benign behavior of BLE packet exchanges. In the offline training phase, we aim to minimize the error between learned data D_L and original dataset D_T. In the online testing phase, if input data contains any malicious package, the reconstruction error will obviously increase. In this paper, network reconstructions are conducted using a temporal convolutional network (TCN)[9, 10]. The residual is defined as $R(D_T, D_L) = |D_T - D_L|$ with $D_L = f(D)$ and f represents the transformation of TCN auto-encoder. **Table 7** illustrates the hyperparameters of our applied TCN model. Afterwards, we evaluate the residual to determine the anomaly score $\boldsymbol{\alpha}$ for each data batch, as illustrated in the equation below, where R_α represents the corresponding residual, $\boldsymbol{\mu}$ is the mean value of residual, and $\boldsymbol{\sigma}$ is its standard deviation. In a word, reconstruction methods are employed to detect suspicious data batches, in next step, we will utilize the classification model to identify the malicious packets involved in network traffic.

$$\boldsymbol{\alpha} = \begin{cases} 0, \text{when } |R_\alpha - \mu R_\alpha| \leq 3 * \sigma R_\alpha \\ 1, \text{otherwise} \end{cases}$$

4.3 Supervised Classification Models

Upon the identification of suspicious batches, the next stage is to categorize these packages into different classes: benign (0) or malicious (1). In this study, we employed the text-convolutional neural network (text-CNN)[10] for traffic features extraction, and we evaluated the performances of **five** different classifiers (SVM, KNN, Random Forest, Logistic Regression and Naïve Bayes) in package classification[11]. In particular, the payload based features are extracted by converting the payload bytes into low dimensional vectors utilizing the Word2Vec technique. These vectors served as the input for the text-CNN model, and the generating payload features were concatenated with statistical network features and provided for the final classification. Furthermore, the hyperparameters of the Text-CNN model are presented in **Table 8** above. The preliminary experimental results for package classification will be discussed in the **Section 5.3**.

4.4 Overview of Proposed Detection Mechanism

Overall, BLEGuard system aims to balance the detection precision and power overhead within BLE networks. As illustrated in **Figure 3**, when GPU resources are not abundant enough, features judgment algorithm can be deployed online with very low consumption, while reconstruction models can be utilized when detection correctness is highly demanding. In addition, the classification models can identify specific malicious advertising packets very reliably. Preliminary experimental results verified the feasibility and non-interference.

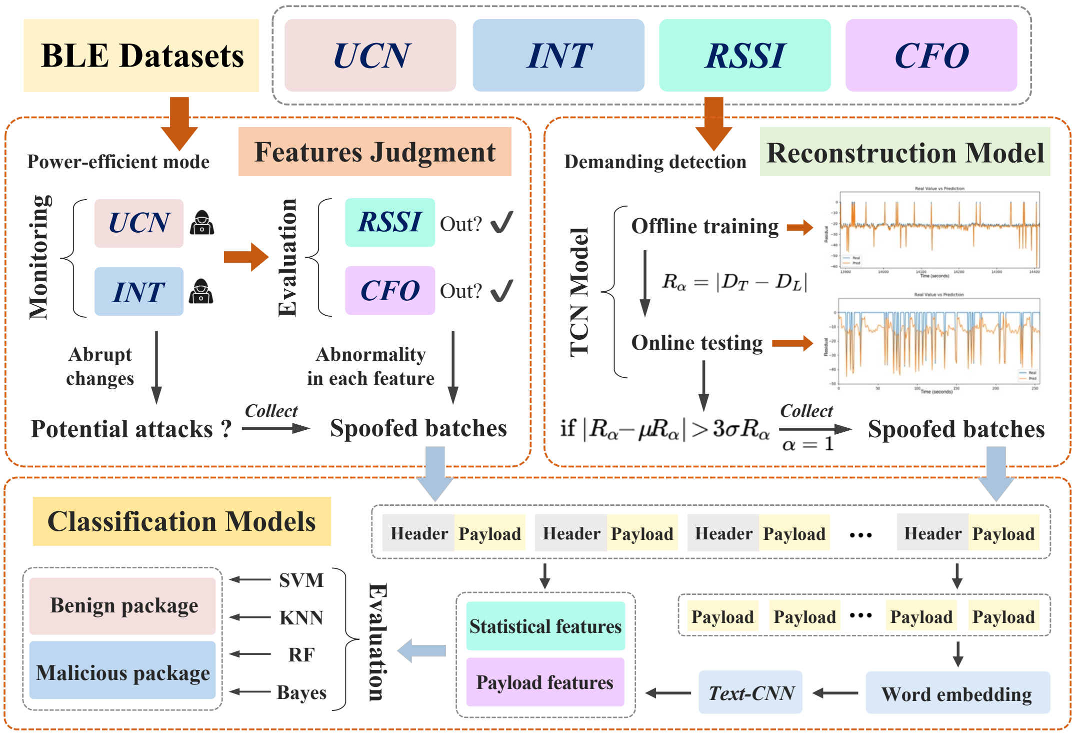

Figure 3: The workflow of BLEGuard detection mechanism.

Table 9: Detection results for features judgment algorithm and classification models (using SVM).

Device ID	Device Name	Accuracy	Overall	
			FAR	UND
1	Smoke detector	97.40	0.09	0.85
2	Indoor camera	99.20	0.07	1.49
3	Nest mini	97.86	0.07	1.96
4	Temperature sensor	97.58	0.09	0.93
5	Tempi temperature sensor	97.26	0.07	1.07
6	Smart lock	98.07	0.14	1.26
7	Door & window sensor	97.49	0.11	0.96
8	Button remote control	98.19	0.05	1.39
9	Energy socket	97.61	0.08	0.74
10	Smart light bulb	98.15	0.12	1.94
11	Mi smart scale	96.96	0.13	1.49
12	Mi band 8	98.62	0.08	1.70
13	Mijia hygrometer 2	99.44	0.08	0.97
14	Key finder	97.05	0.09	1.48
15	Otbeat sport band	98.54	0.04	1.15
16	HomePod 2	98.73	0.12	1.13
	Average	**98.01**	**0.09**	**1.28**

5 Preliminary Results

In this section, we will discuss the preliminary experimental results of our proposed detection mechanism. We will also introduce the evaluation methodology utilized to measure the performance of our methods.

5.1 Performance Evaluation Metrics

As mentioned in **Section 3.3**, to simulate the real-world network environment as similar as possible, the dataset was deliberately designed to be unbalanced, where the amount of benign traffic is far more than malicious traffic. In this case, the benign sample is dominant in number, leading to biased results. So, the general metric of accuracy is not representative and reliable to evaluate the performance of our methods in this scenario. In order to avoid a biased analysis, credible metrics (False Alarm Rate and Un-detection Rate) have been suggested. Therefore, in addition to accuracy, the FAR and UND also be utilized in the performance evaluation. **Table 10** shows these evaluation metrics and corresponding formulas, where TP, TN, FP, FN represent as:

- Ture Position (TP): the number of malicious samples correctly classified as malicious packages.
- Ture Negative (TN): the number of benign samples correctly classified as benign packages.
- False Position (FP): the number of benign samples correctly classified as malicious packages.
- False Negative (TN): the number of malicious samples correctly classified as benign packages.

Table 10: Detection performance metrics.

Evaluation metrics	Corresponding formula
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN} \times 100\%$
False alarm rate (FAR)	$\frac{FP}{FP + TN} \times 100\%$
Un-detection rate (UND)	$\frac{FN}{FN + TP} \times 100\%$

5.2 Performance of Detection Methods

The performance of features judgment algorithm and TCN reconstruction model are highly related to the result of the classification models, since detection methods only identify the potential malicious data batch. In the evaluation phase, three metrics are utilized to access the performance of our methods, as shown in **Table 10**, where accuracy refers to the overall proportion of the correct classifications. False alarm rate represents the error when BLEGuard system triggers a false alarm for a spoofing attack after inspecting benign advertising packets generated by a normal BLE device. Un-detection rate represents the error in which BLEGuard system fails to detect a spoofing attack after analyzing spoofed packets from the attacker devices. **Table 9** illustrates the detection performance for features judgment algorithm and classification models (utilizing SVM classifier). BLEGuard system achieves an average accuracy of 98.01%, with 0.09% FAR and 1.28% UND, which are better than most existing research methods[3-5]. However, we notice that our preliminary training and testing results are based on our accumulated datasets, which are potentially biased since the amount of data is not abundant enough and the testing latitude maybe insufficient. Therefore, more advertising package dataset will be collected while the tests for the TCN reconstruction model will be conducted based on large-scale data.

5.3 Package Classification Results

As mentioned in **Section 4.3**, we utilized the text-CNN for features extraction, and evaluated the performances of five different classifiers (SVM, KNN, Random Forest, Logistic Regression and Naïve Bayes) in package classification. **Figure 4** depicts the accuracy performance for five machine learning classifiers. Within the training phase, the Random Forest (RF) model attains the highest accuracy at 99.41%, whereas the SVM model exhibits the superior performance of 98.01% in the testing results. In terms of the accuracy metric, the discrepancies between training and testing results are marginal, with one notable exception being the Logistic Regression (LR). This observation suggests that the LR classifier underperforms in realistic testing scenario.

Figure 5 provides a visual representation of the FAR comparison among five classifiers. In this context, FAR signifies the percentage of benign traffic incorrectly classified as malicious. Among these models, the Random Forest model exhibits the most favorable FAR of 0.01% during the training experiments, while the SVM model outperforms others with a FAR of 0.09% in the testing experiments. Nevertheless, when scrutinizing the variation between the training and testing scenarios, it becomes apparent that the Random Forest model experiences a relatively substantial increase, rising from its initial 0.01% to 0.24%. This upward shift suggests a potential instability in the model's performance when subjected to realistic testing conditions. Furthermore, the Logistic Regression model continues to exhibit suboptimal performance, registering the highest FAR of 2.47% among all models, thereby establishing itself as the least performing model in this phase.

Figure 6 presents the performance of UND for the five models. In this stage, UND denotes the percentage of malicious traffic inaccurately classified as benign traffic. Notably, the SVM model excels in this regard, achieving the lowest UND rate at 1.28%. Following SVM, the Naïve Bayes (Bayes) model demonstrates a UND rate of 2.46%, while the K-Nearest Neighbors (KNN) model and Random Forest (RF) model exhibit UND rates of 2.94% and 3.71%, respectively. Conversely, the LR model displays a notably poor performance, recording a high UND rate of 56.73%, so we eventually removed it in from our submitted 2-page abstract.

Overall, the preliminary results unequivocally highlight the outstanding performance of the Support Vector Machine (SVM) model, with the 98.01% accuracy, 0.10% FAR, and 0.81% UND, as illustrated in **Table 9**.

5.4 Future Works

To further enhance the effectiveness of the proposed BLEGuard system, our future plans are given as follows:

- **More Dataset Collection:** We will actively work on amassing more comprehensive advertising package datasets. This expansion will provide a richer and more diverse data for the performance evaluations.
- **Learning Model Improvement:** We aim to improve the learning models' training and testing. This involves refining existing models and exploring new approaches to ensure robust and accurate detection.
- **Precision and Efficiency Optimization**: Our focus will also be on fine-tuning the system to achieve the optimal configurations between high precision in detecting malicious activity and minimizing power consumption. This optimization process will involve rigorous testing with real-world datasets.

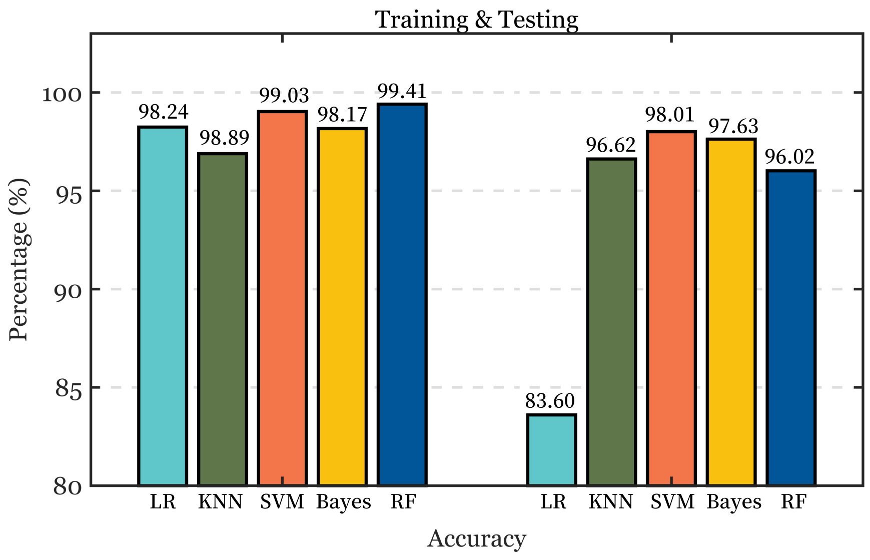

Figure 4: Accurary of package classification for five models

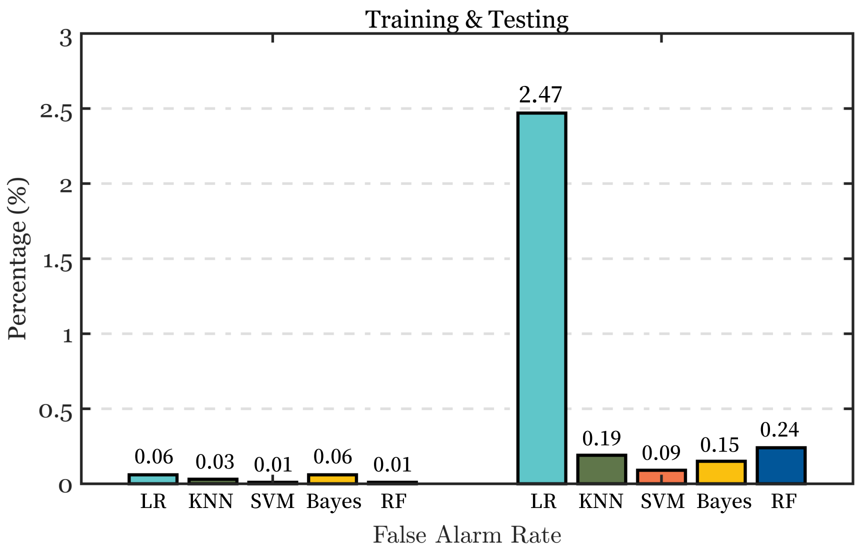

Figure 5: False alarm rate of package classification for five models

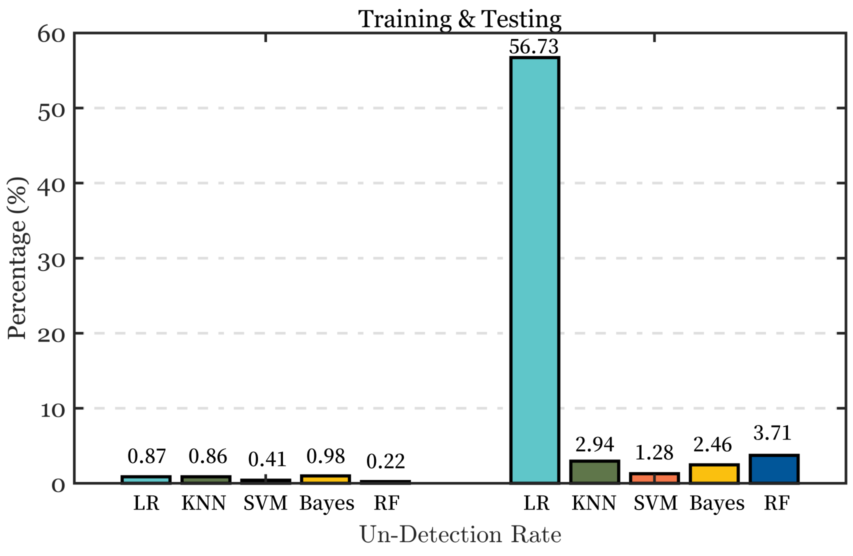

Figure 6: Un-detection rate of package classification for five models

References

[1] Github Repository for BLEGuard. Online Access: https://github.com/BLEGuard/supplement . 2023.

[2] Bluetooth SIG. Bluetooth Market Update. Online Access: https://www.bluetooth.com/bluetooth-resources/2023-bluetooth-market-update/ . 2023.

[3] Wu J, Nan Y, et al. {BlueShield}: Detecting spoofing attacks in bluetooth low energy networks; proceedings of the 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020), F, 2020.

[4] Lahmadi A, Duque A, et al. MitM Attack Detection in BLE Networks Using Reconstruction and Classification Machine Learning Techniques. ECML PKDD 2020 Workshops. 2020: 149-164.

[5] Yaseen M, Iqbal W, et al. MARC: A Novel Framework for Detecting MITM Attacks in eHealthcare BLE Systems. J Med Syst, 2019, 43(11): 324.

[6] Wireshark Tool (Network Protocol Analyzer). Online Access: https://www.wireshark.org/ . 2023.

[7] Mirage Networking Tool. Online Access: https://miragenet.github.io/Mirage/ . 2023.

[8] Ostinato Traffic Generator Tool. Online Access: https://ostinato.org/ . 2023.

[9] Bai S, Kolter J Z, et al. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:180301271, 2018.

[10] Chen X, Hao Z, et al. Cruparamer: Learning on parameter-augmented api sequences for malware detection. IEEE Transactions on Information Forensics and Security, 2022, 17: 788-803.

[11] Teixeira M A, Salman T, et al. SCADA system testbed for cybersecurity research using machine learning approach. Future Internet, 2018, 10(8): 76.