ISSN :2584-2706 DOI: https://doi.org/10.67317/IJMSRT26OCT068

Calendar, Not Traffic: A Timestamp Artefact in the N-BaIoT Dataset and its Effect on IoT Attack Classification

James Nmezi
Engineering Division, Nigerian Television Authority, Umuahia, Nigeria, nmezijames

Chibuzor Henry Amadi
Dept. of Electronic Engineering, Federal University of technology, Owerri, Nigeria, Chibuzor

Reginald Ekene Ogu
Dept. of Computer Engineering, Federal University of technology, Owerri, Nigeria, reginald

Abstract
N-BaIoT is one of the most widely used datasets for detecting botnet attacks on Internet of Things (IoT) devices, and published classifiers routinely report near-perfect accuracy on it. This paper shows that ten of its 115 features, the mean and variance of the host-to-host jitter statistic (HH_jit) at five time windows, frequently hold Unix timestamps instead of traffic statistics. When a channel has seen a single packet, its jitter mean equals the packet's arrival time. Because every attack class was captured in its own short window, on the same dates for all nine devices, these features identify the class by the clock. In a sample of 438,955 records, at least one of the ten features exceeded 10^6 in every attack record but in only 28.9 per cent of benign records. A LightGBM classifier separated the Gafgyt TCP and UDP floods with 99.8 per cent accuracy using the ten features alone and with 50.5 per cent, chance level, without them. Keeping the features raised eleven-class macro-F1 from 0.879 to 0.963 on a random split and from 0.843 to 0.946 when whole devices were held out, while binary detection was unchanged. The artefact therefore inflates exactly the results that are hardest to obtain honestly: multiclass separation and generalisation to unseen devices. We describe how to detect and remove it, give a realistic multiclass ceiling for N-BaIoT, and offer a short checklist for screening other flow-feature datasets for similar leakage.

Keywords: *IoT security; botnet detection; N-BaIoT; data leakage; shortcut learning; intrusion detection datasets; evaluation methodology.*

I. INTRODUCTION

Benchmark datasets shape what the intrusion detection community believes is possible. When a dataset contains a feature that reveals the label for reasons unrelated to the phenomenon being studied, models learn that feature, reported accuracy rises, and the field's sense of the problem's difficulty drifts away from reality. Arp et al. (2022) list spurious correlations of this kind among the most common pitfalls in machine learning for security, and Kapoor and Narayanan (2023) trace a broad reproducibility problem across scientific machine learning to leakage of the same sort. In intrusion detection specifically, Engelen, Rimmer and Joosen (2021) showed that flow-construction and labelling errors in CICIDS2017 were serious enough to change the ranking of models once corrected.

N-BaIoT (Meidan et al., 2018a, 2018b) is a natural place to look for such problems. It records benign traffic and attacks by two botnet families, Mirai and BASHLITE (Gafgyt), on nine commercial IoT devices, and describes each packet by 115 damped-window statistics. Its authors showed that per-device autoencoders could flag every attack without seeing attack labels (Meidan et al., 2018b), and the dataset has since become a standard benchmark for supervised, deep and federated detectors (Alkahtani & Aldhyani, 2021; Rey, Sánchez Sánchez, Huertas Celdrán, & Bovet, 2022; Soe, Feng, Santosa, Hartanto, & Sakurai, 2020).

While building a multiclass detector on N-BaIoT, we found that two of its attack classes, the Gafgyt TCP and UDP floods, could be told apart almost perfectly by a gradient-boosted classifier, yet not at all once ten particular features were removed. Those features turned out to hold capture timestamps. This paper documents that artefact and measures its effect. Its contributions are:

ISSN :2584-2706 DOI: https://doi.org/10.67317/IJMSRT26OCT068

1. It identifies the ten HH_jit mean and variance features as carriers of Unix timestamps, explains the mechanism (a single-packet channel whose jitter is measured from a zero origin) and shows how much of the dataset is affected, class by class.

2. It shows that each attack class was captured in its own short window, shared by all nine devices, so the timestamps identify the class whichever device the record came from.

3. It measures the inflation on a tuned LightGBM classifier under a random split and under leave-one-device-out evaluation, and shows that the gain falls entirely on multiclass separation and cross-device generalisation, not on binary detection.
4. It gives a realistic eleven-class ceiling for N-BaIoT and a short screening checklist that can be applied to other flow-feature datasets.

Section 2 reviews related work. Section 3 describes the artefact, Section 4 the experiments and Section 5 the results. Section 6 discusses what the findings mean for published N-BaIoT results and for dataset practice, and Section 7 concludes.

II. RELATED WORK

2.1 Results on N-BaIoT

N-BaIoT was released with an anomaly-detection study. Meidan et al. (2018b) trained one deep autoencoder per device on benign traffic and reported a true positive rate of 100 per cent on that device's attacks. Their features come from the same family of incremental, damped-window statistics used by the Kitsune intrusion detector (Mirsky, Doitshman, Elovici, & Shabtai, 2018): for each packet, means, variances and related quantities are kept over five exponential decay windows for traffic grouped by source MAC and IP (MI_dir), source IP (H), host-to-host channel (HH), host-to-host jitter (HH_jit) and socket (HpHp).

Later work moved to supervised classification. Soe et al. (2020) built a sequential detector with one lightweight sub-engine per attack type, each using a small selected feature set. Alkahtani and Aldhyani (2021) trained a CNN-LSTM per device on all ten attacks and reported per-device accuracies of roughly 87 to 91 per cent, well below the binary results. Rey et al. (2022) trained federated multilayer perceptrons and autoencoders and evaluated them on devices absent from training. None of these studies reports examining the HH_jit features for leakage, and we did not find the artefact described in the literature we reviewed.

2.2 Leakage and Artefacts in Security Datasets

The broader literature gives good reason to look. Engelen et al. (2021) re-examined CICIDS2017 and found flow-construction and labelling errors that changed model rankings once fixed. Sarhan, Layeghy and Portmann (2022) showed that intrusion datasets use incompatible feature sets and that models trained on one generalise poorly to another unless the features are standardised. Arp et al. (2022) surveyed security papers and found spurious correlations, sampling bias and lab-only evaluation to be widespread. Kapoor and Narayanan (2023) catalogued leakage across many scientific fields and showed how often it produces over-optimistic results. Sasi, Lashkari, Lu, Xiong and Iqbal (2025) give a recent IoT illustration of the consequence: a model that reached 99.96 per cent accuracy on one IoT dataset fell to 33.3 per cent on another processed with the same feature extractor.

The artefact reported here is a specific instance of what these authors warn about. It arises in feature extraction, not in labelling, and it is invisible to the usual precautions of splitting before fitting, scaling on training data and removing duplicates.

III. THE ARTEFACT

3.1 What the HH_jit Features Should Hold

N-BaIoT computes its statistics incrementally over five exponentially damped windows, labelled L5, L3, L1, L0.1 and L0.01, which correspond to roughly 100 ms, 500 ms, 1.5 s, 10 s and 1 min of recent traffic (Meidan et al., 2018b). The HH_jit group describes the timing of packets on a host-to-host channel. For each window it keeps a weight (the decayed count of packets seen), a mean and a variance of the inter-arrival gap. On a busy channel the mean should be a fraction of a second.

3.2 What They Actually Hold

In our sample, HH_jit_L0.1_mean took values between 1.5056×10^9 and 1.5077×10^9 in 42.0 per cent of records. Read as seconds since 1 January 1970, these are times between 17 September and 10 October 2017, a few months before the dataset was released in 2018. Every one of the 184,348 records with such a value had a weight of exactly 1 and a variance of exactly 0: the channel had seen one packet, and the reported mean gap was that packet's arrival time. A further 30,150 records had means between 10^8 and 1.4×10^9 and weights between 1.09 and 15. In these the timestamp had been averaged with a few genuine gaps, and the corresponding variances reached the order of 10^{17}. The same pattern holds at all five windows, so ten features are affected: HH_jit_L5, L3, L1, L0.1 and L0.01, mean and variance.

ISSN :2584-2706 DOI: https://doi.org/10.67317/IJMSRT26OCT068

The values are consistent with a first inter-arrival gap measured from a previous timestamp of zero rather than from the channel's first packet. The publicly available Kitsune implementation, which computes this family of statistics, now sets the previous timestamp of a new statistic to its first arrival time, which yields a first gap of zero (Mirsky, 2018). We could not establish which version of the extractor produced N-BaIoT. The mechanism matters less than the result, which is visible in the published files.

3.3 How Much of the Dataset Is Affected

The artefact is not spread evenly. Table 1 shows, for each class, the share of records in which HH_jit_L0.1_mean is a raw timestamp and the share in which any of the ten features exceeds 10^6, a value no genuine jitter statistic on a 1-minute window could reach.

Class	Records	L0.1 mean is a timestamp (%)	Any of the ten features above 10^6 (%)
Benign	45,000	0.4	28.9
Gafgyt combo	45,000	0.1	100.0
Gafgyt junk	45,000	0.1	100.0
Gafgyt scan	45,000	33.5	100.0
Gafgyt TCP flood	41,595	99.9	100.0
Gafgyt UDP flood	42,360	99.9	100.0
Mirai ACK flood	35,000	34.6	100.0
Mirai scan	35,000	99.8	100.0
Mirai SYN flood	35,000	33.9	100.0
Mirai UDP flood	35,000	32.7	100.0
Mirai UDPplain flood	35,000	42.5	100.0
All	438,955	42.0	92.7

Table 1: Share of records carrying the artefact, by class. Sample: the first 5,000 records of each of the 89 N-BaIoT files after removal of 6,045 duplicates.

Two features of the table matter for classification. Every attack record carries at least one contaminated value, against fewer than three in ten benign records, so the features alone go a long way towards separating attacks from normal traffic. More seriously, the timestamps themselves differ by class. Figure 1 shows the capture window of each attack class.

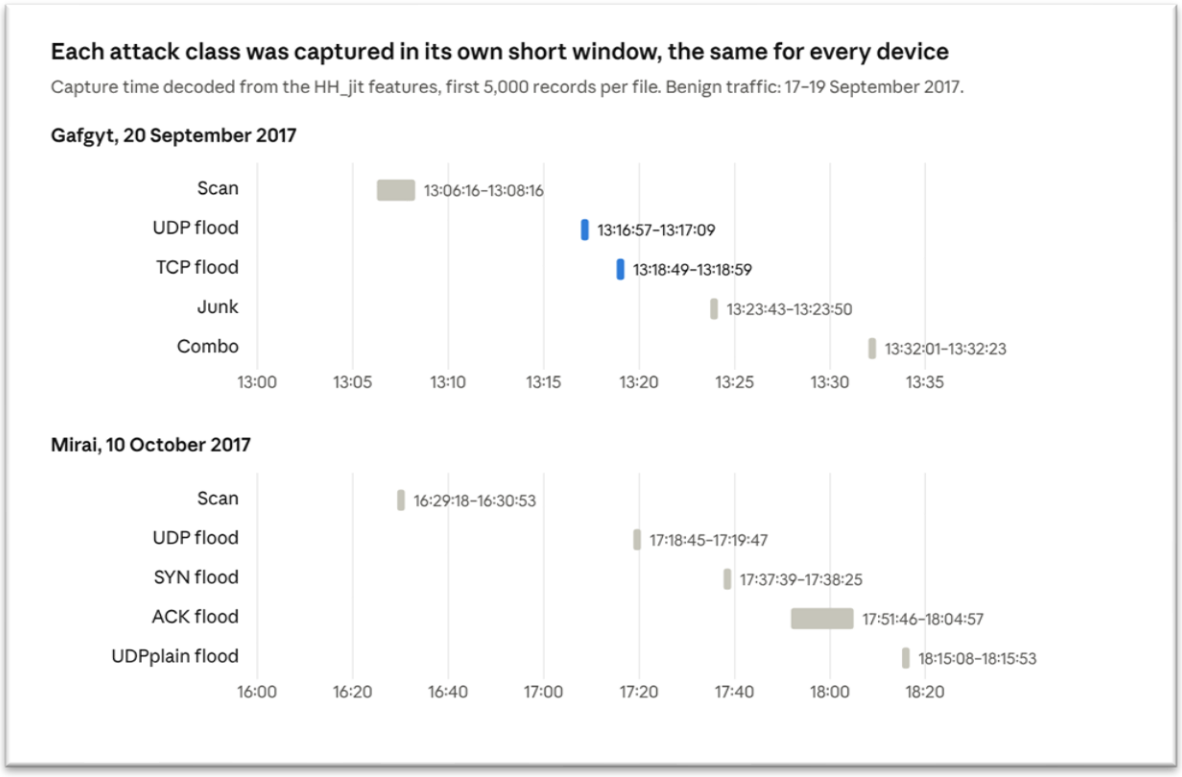


Figure 1 · Times decoded from HH_jit_L0.1_mean in the 438,955-record sample (UTC). Blue: the two floods used as the worked example in Section 5.

ISSN :2584-2706 DOI: https://doi.org/10.67317/IJMSRT26OCT068

Each attack class occupies its own window, and the windows do not overlap. The Gafgyt UDP flood was captured between 13:16:57 and 13:17:09 on 20 September 2017 and the TCP flood between 13:18:49 and 13:18:59, so a single threshold on the timestamp separates them. The windows are also the same for every device: the attacks appear to have been launched on all nine devices at once. A timestamp therefore identifies the class whichever device a record came from, a point that becomes important when models are tested on devices they were not trained on.

IV. EXPERIMENTAL METHOD

4.1 Data

The 89 device-and-class CSV files of N-BaIoT were downloaded from the UCI Machine Learning Repository (Meidan et al., 2018a). To keep the experiments within the reach of a two-core machine, the first 5,000 records of each file were used, 445,000 records in all. Removing 6,045 exact duplicates, all from the Gafgyt TCP and UDP flood files, left 438,955 records in eleven classes, with between 35,000 and 45,000 records per class. Two devices, the Ennio doorbell and the Samsung webcam, have no Mirai traffic. Labels were taken from the file names.

4.2 Experiment A: The TCP and UDP Floods in Isolation

The first experiment asks how much of the separation between two similar classes the artefact accounts for. LightGBM with default settings was trained to tell the 41,595 Gafgyt TCP flood records from the 42,360 Gafgyt UDP flood records, using three-fold stratified cross-validation and three feature sets: all 115 features, the 105 features that remain once the ten HH_jit means and variances are removed, and those ten features alone.

4.3 Experiment B: Eleven-Class Classification

The second experiment measures the effect on the full multiclass task. A LightGBM classifier with balanced class weights was trained with all 115 features and again with the 105 clean ones, with no feature selection in either case. Its settings (400 trees, 94 leaves, learning rate 0.129, at least 79 records per leaf and 55 per cent of features per tree) came from an Optuna search carried out on the clean features in the wider study from which this paper is drawn. Using settings tuned without the artefact makes the comparison conservative.

Two evaluation settings were used:

1. **Random split.** A stratified 70/15/15 split with seed 0; the model was trained on the 70 per cent and tested on the 65,844-record test portion.

2. **Leave one device out.** For each of the nine devices, the model was trained on all records of the other eight and tested on every record of the held-out device. This is the test that most closely matches deployment on hardware the model has not seen.

Reported measures are macro-F1 over the classes present in the test set, recall for the Gafgyt TCP and UDP floods, and binary F1 for attack against benign traffic.

4.4 Software

All experiments used Python 3.13 with pandas, NumPy, scikit-learn 1.9.1, LightGBM 4.7.0 and Optuna 5.0.0 on a cloud virtual machine with two cores of a 2.8 GHz Intel Xeon processor and no GPU. The code is available from the author (see Data and Code Availability).

V. RESULTS

5.1 The TCP and UDP Floods

With all 115 features, LightGBM separated the Gafgyt TCP and UDP floods with a cross-validated accuracy of 99.87 per cent. With the ten artefact features alone it reached 99.85 per cent. With the other 105 features it reached 50.52 per cent, which is chance for two balanced classes. Almost all of the separation between these two floods therefore comes from the timestamps; on the traffic statistics alone, the two floods are the same.

ISSN :2584-2706 DOI: https://doi.org/10.67317/IJMSRT26OCT068

5.2 Eleven Classes on a Random Split

Table 2 compares the two feature sets in Experiment B.

Evaluation	All 115 features	105 clean features	Difference
Random split: macro-F1	0.9626	0.8788	+0.084
Random split: Gafgyt TCP recall	0.6030	0.9998	−0.397
Random split: Gafgyt UDP recall	1.0000	0.0027	+0.997
Random split: binary F1 (attack vs benign)	1.0000	1.0000	0.000
Leave one device out: mean macro-F1	0.9460	0.8426	+0.103

Table 2: LightGBM with and without the ten HH_jit timestamp features. Random split: seed 0, 65,844 test records. Leave one device out: mean over nine devices.

Binary detection did not change: with or without the timestamps, attack and benign traffic were separated almost perfectly. The multiclass result did. Without the artefact, the model put both Gafgyt floods under one label (TCP), and macro-F1 settled at 0.879. With it, the model recovered all of the UDP floods and 60 per cent of the TCP floods, and macro-F1 rose to 0.963.
The clean figure is close to a hard ceiling. If two classes cannot be told apart and a model assigns both to one label, that label's F1 is about 0.67 and the other's is zero. Even with the remaining nine classes perfect, macro-F1 is then (9 + 0.67) / 11 = 0.879, which is what the clean model reached. Splitting the two floods at random would give at best (9 + 2 × 0.5) / 11 = 0.909. On this sample, any eleven-class macro-F1 well above 0.91 needs a feature that tells the floods apart, and the only such features we found are the timestamps.

5.3 Eleven Classes on Unseen Devices

The artefact helped even more when whole devices were held out. Mean macro-F1 across the nine devices rose from 0.843 to 0.946, and every device gained (Figure 2).

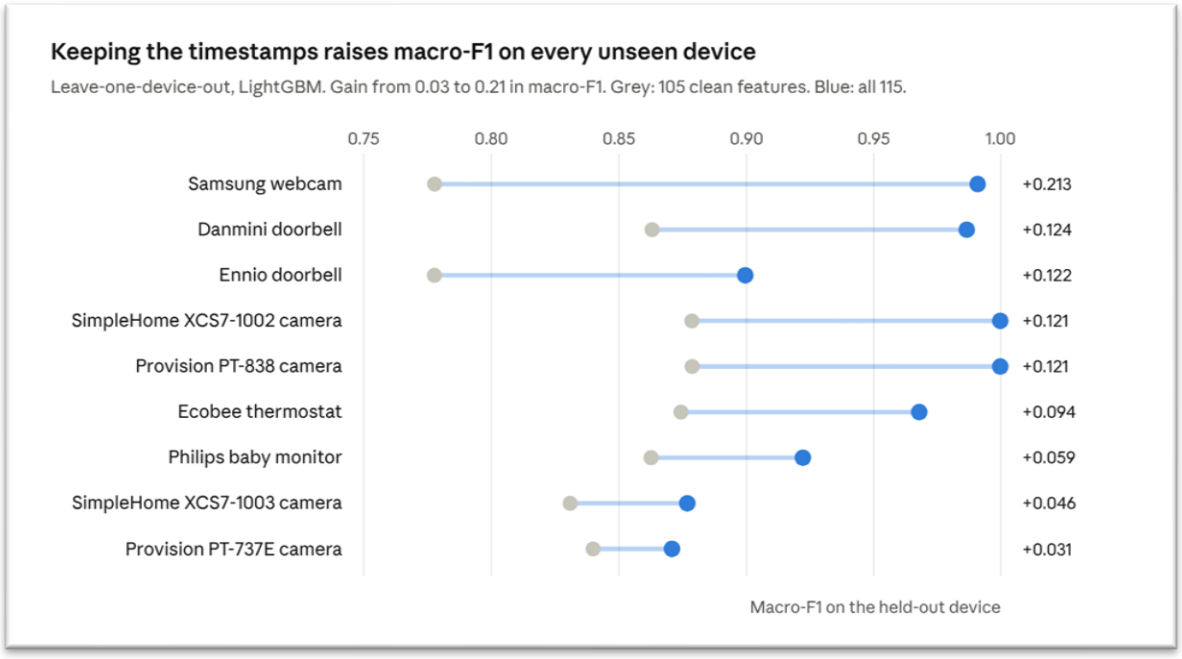


Figure 2 · Experiment B, leave-one-device-out. Macro-F1 over the classes present on each device (eleven, or six for the two devices without Mirai traffic).

On two cameras, the Provision PT-838 and the SimpleHome XCS7-1002, keeping the timestamps lifted macro-F1 from about 0.88 to 0.9998, an almost perfect score on a device the model had never seen. The largest gain, 0.21, came on the Samsung webcam. This is what Figure 1 predicts. A model that has learnt that records stamped around 13:17 on 20 September 2017 are UDP floods will apply the rule correctly to any device, because every device was attacked at the same time. A study that kept these features would report better cross-device generalisation than the traffic itself supports.

ISSN :2584-2706 DOI: https://doi.org/10.67317/IJMSRT26OCT068

VI. DISCUSSION

6.1 What This Means for Published N-BaIoT Results

The artefact inflates precisely the results that are hardest to earn. Binary detection on N-BaIoT is easy with or without it. Multiclass separation and generalisation to new devices are where the field's claims are most interesting, and those are the results the timestamps improve. A model trained on all 115 features can therefore look like a stronger detector, and a better generaliser, than the traffic supports.

We have not re-run other authors' pipelines, and nothing here shows that any particular published result depends on the artefact. Many studies select features, aggregate records or use only part of the dataset, and some of these steps may remove the HH_jit features by accident. What the results do show is a simple test a reader can apply. On this sample, an eleven-class macro-F1 well above 0.91, or near-perfect separation of the Gafgyt TCP and UDP floods, is not achievable from the traffic statistics alone. A paper that reports such a figure using all 115 features should be checked against the artefact before its result is accepted as a property of the traffic.

The finding also sharpens how earlier multiclass results read. Alkahtani and Aldhyani (2021) reported per-device accuracies of roughly 87 to 91 per cent for a CNN-LSTM, well below binary results. In the wider study, clean eleven-class tree models reached about 90 per cent accuracy on pooled devices, and almost all of the remaining error came from the indistinguishable TCP and UDP floods. Results in that range may be close to the honest limit of the dataset rather than evidence of weak models.

6.2 Recommendations

For work on N-BaIoT, we recommend removing the ten HH_jit mean and variance features (HH_jit_L5, L3, L1, L0.1 and L0.01, mean and variance) before any modelling, and reporting results with and without them where the comparison matters. Setting the contaminated values to missing is a possible alternative, but it would leave the fact of contamination itself, which differs sharply between benign and attack traffic, available to the model.

For any flow-feature dataset, five quick checks would have caught this artefact before it reached a model:

1. **Range check.** Compare each feature's range with what the statistic can physically take. A jitter mean of 1.5×10^9 seconds is not a jitter.

2. **Single-feature test.** Train a simple classifier on each feature alone. A single feature that separates two classes almost perfectly needs an explanation grounded in the traffic.

3. **Decode suspicious values.** Values near 10^9 to 2×10^9 may be Unix timestamps. Convert them to dates and plot them by class.

4. **Check capture windows.** If classes were captured at different times, any feature that leaks time will leak the label. Device-wise or session-wise splits do not protect against this when every device was attacked at once.

5. **Ablate and compare.** Report the effect of removing any suspect feature group. A large multiclass gain with no binary gain is a warning sign.

The same family of incremental statistics is used by Kitsune (Mirsky et al., 2018) and by other datasets built with similar extractors. We have not examined those datasets, but they deserve the same checks.

6.3 Limitations

The experiments used the first 5,000 records of each file rather than all seven million records, so the capture windows in Figure 1 describe the start of each capture session, and the share of affected records may differ in the full files. The mechanism, a single-packet channel reporting its arrival time, does not depend on the sample. Experiment B used one model family, LightGBM; other models may exploit the artefact more or less. Finally, we inferred the cause from the published values and the public Kitsune code, and could not confirm which version of the feature extractor produced the dataset.

VII. CONCLUSION

Ten features of N-BaIoT, the HH_jit means and variances, record when traffic was captured rather than how it behaved. Because each attack class was captured in its own short window, shared by every device, those features identify the class directly. They account for almost all of the apparent separability of the Gafgyt TCP and UDP floods, add 0.08 to eleven-class macro-F1 on a random split and 0.10 on unseen devices, and leave binary detection untouched. Removing them gives an honest multiclass ceiling of about 0.88 to 0.91 on this sample. We recommend that future work on N-BaIoT drop these features, that reviewers ask for the check when multiclass results exceed that ceiling, and that the five screening steps in Section 6.2 become routine for flow-feature datasets.

ISSN :2584-2706 DOI: https://doi.org/10.67317/IJMSRT26OCT068

Data and Code Availability

N-BaIoT is publicly available from the UCI Machine Learning Repository (Meidan et al., 2018a). The scripts used to sample the data, detect the artefact and run both experiments are available from the corresponding author on request. [Author to add a repository link before submission.]

REFERENCES

[1]. Alkahtani, H., & Aldhyani, T. H. H. (2021). Botnet attack detection by using CNN-LSTM model for Internet of Things applications. Security and Communication Networks, 2021, Article 3806459. doi:10.1155/2021/3806459

[2]. Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., . . . Rieck, K. (2022). Dos and don'ts of machine learning in computer security. In Proceedings of the 31st USENIX Security Symposium (pp. 3971–3988). Boston, MA: USENIX Association.

[3]. Engelen, G., Rimmer, V., & Joosen, W. (2021). Troubleshooting an intrusion detection dataset: The CICIDS2017 case study. In 2021 IEEE Security and Privacy Workshops (SPW) (pp. 7–12). San Francisco, CA: IEEE.

[4]. Kapoor, S., & Narayanan, A. (2023). Leakage and the reproducibility crisis in machine-learning-based science. Patterns, 4(9), 100804. doi:10.1016/j.patter.2023.100804

[5]. Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Breitenbacher, D., & Shabtai, A. (2018a). Detection of IoT botnet attacks N_BaIoT [Data set]. UCI Machine Learning Repository. doi:10.24432/C5RC8J

[6]. Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., & Elovici, Y. (2018b). N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders. IEEE Pervasive Computing, 17(3), 12–22. doi:10.1109/MPRV.2018.03367731

[7]. Mirsky, Y. (2018). Kitsune-py [Computer software]. Retrieved from https://github.com/ymirsky/Kitsune-py

[8]. Mirsky, Y., Doitshman, T., Elovici, Y., & Shabtai, A. (2018). Kitsune: An ensemble of autoencoders for online network intrusion detection. In Proceedings of the Network and Distributed System Security Symposium (NDSS 2018). San Diego, CA: Internet Society. doi:10.14722/ndss.2018.23204

[9]. Rey, V., Sánchez Sánchez, P. M., Huertas Celdrán, A., & Bovet, G. (2022). Federated learning for malware detection in IoT devices. Computer Networks, 204, 108693. doi:10.1016/j.comnet.2021.108693

[10]. Sarhan, M., Layeghy, S., & Portmann, M. (2022). Towards a standard feature set for network intrusion detection system datasets. Mobile Networks and Applications, 27, 357–370.

[11]. Sasi, T., Lashkari, A. H., Lu, R., Xiong, P., & Iqbal, S. (2025). An efficient self attention-based 1D-CNN-LSTM network for IoT attack detection and identification using network traffic. Journal of Information and Intelligence, 3(5), 375–400. doi:10.1016/j.jiixd.2024.09.001

[12]. Soe, Y. N., Feng, Y., Santosa, P. I., Hartanto, R., & Sakurai, K. (2020). Machine learning-based IoT-botnet attack detection with sequential architecture. Sensors, 20(16), 4372. doi:10.3390/s20164372