



ARS-IDS: A Lightweight Hybrid Architecture with Adaptive Rule Synthesis for IoT Intrusion Detection

Lawal Moshood Olatunji^{1*}, Amosa Ramoni Tirimisiyu³, Etudaiye Abdulmumuni Isah³ Lawal Aderonke Tadesse⁴.

¹Department of Cybersecurity, Federal Polytechnic, Ede, Osun State, Nigeria

²Department of Computer Science, Federal Polytechnic, Ede, Osun State, Nigeria

³Department of Cybersecurity, Federal Polytechnic, Ede, Osun State, Nigeria

⁴Department of Computer Science, Federal Polytechnic, Ede, Osun State, Nigeria

*Corresponding Author: Lawal.moshood@federalpolyede.edu.ng

Abstract: Most IoT devices cannot run conventional intrusion detection software, and most compromises trace to weak credentials, not novel exploits. This paper introduces ARS-IDS, a hybrid detection architecture built around Adaptive Rule Synthesis (ARS), turning confirmed machine-learning detections into new deterministic rules and closing a loop that static hybrid designs leave open. ARS is specified algorithmically: detect, confirm against a repeated-pattern criterion, extract a candidate rule, hold it in a non-blocking shadow state using operator acknowledgement as a ground-truth proxy, then admit or discard. Its central constraint: online validation has no labelled oracle, so the shadow stage is a proxy, not a measurement; offline validation is a precondition for trusting it live. We report preliminary validation of the classification layer on BoTNeT-IoT-L01, traffic captured from nine IoT devices under Mirai and Gafgyt attacks. Under a leave-one-device-out protocol, pooled over nine folds, a Random Forest reaches 99.987% accuracy and recall rounding to 100.000% (6 of 1,952,002 malicious flows missed), against 99.916% accuracy and 99.999% recall (14 missed) for a Decision Tree; the Random Forest's worst-case device false-positive rate (1.33%) is about a seventh of the Decision Tree's (9.30%), supporting a hardware-tiering claim: different tree models suit endpoint- versus gateway-class hardware. This validation covers the classification layer alone; the rule layer, ARS, and physical-hardware performance remain untested, with fuller treatment left to future work. We close with the next steps needed to complete that evaluation: an executable rule set, an online ARS trial, and testbed deployment.

Keywords: Adaptive Rule Synthesis, Edge Computing, Hybrid Architecture, Iot Security, Intrusion Detection.

I. Introduction

IoT devices are compromised overwhelmingly through weak or default credentials rather than sophisticated attacks (Zhao et al., 2022; Antonakakis et al., 2017), and the obvious countermeasure, running an IDS at the edge, collides with hardware most of these devices simply do not have. Roy et al. (2022) and Tekin et al. (2022) independently confirm the same practical finding: decision trees are close to the ceiling of what on-device inference can afford, and anything heavier needs to be justified against measured energy cost, not assumed. Daghero et al. (2023) push this further, showing that even tree ensembles need dynamic pruning to stay within budget on ultra-low-power silicon.

A hybrid design, cheap deterministic rules first, with a lightweight classifier only for what the rules cannot resolve, is the natural response to this constraint, and it is not a new idea (Alexander & Kumar, 2023; Beborrtta et al., 2023). What is new in ARS-IDS is what happens after the classifier catches something the rules missed: Adaptive Rule Synthesis (ARS) turns that detection into a candidate rule, so the same attack pattern is caught the next time cheaply. Section II situates this against prior rule-synthesis work; the short version is that automated rule generation from a trained model is an established but unsolved problem (Alcântara et al., 2021, reports only 47.6% top-1 accuracy for exactly this task), and prior hybrid IDS designs do not close this particular loop online.

This paper's contribution is the ARS specification itself, Section III, together with the Architecture it operates inside and a first empirical check that the underlying classification approach is sound. It is

not a claim that the full pipeline has been measured end to end; Section V states that boundary directly, and a longer, more exhaustively validated treatment of this same Architecture, with full reproducibility detail, is left for extended future work.

Section II reviews related work. Section III specifies ARS-IDS and ARS. Section IV reports preliminary classification-layer results. Section V discusses limitations and next steps.

II. Related Work

Three bodies of work bear on ARS-IDS, identified through a targeted semantic search (Undermind, 2021-2024, queries built around rule synthesis, rule extraction, and neuro-symbolic detection), not a systematic review with a registered protocol. On resource constraints, Tekin et al. (2022) and Daghero et al. (2023) establish, through direct measurement rather than assumption, which model classes are affordable on IoT-class hardware; ARS-IDS’s algorithm choice (Section III-B) follows their findings rather than an independent preference. On hybrid detection, Alexander and Kumar (2023) let a classifier defer to hard-coded rules (93.17% on IoTID20); Bebertta et al. (2023) blend several classifiers (81.77% accuracy); Fatima et al. (2023) pair chi-square feature selection with joint accuracy/resource reporting. Among these three designs, none reports a path back from a confirmed detection into the rule set, which is precisely the gap ARS closes. On automated rule synthesis, Alcântara et al. (2021), Piet et al. (2023), and Krishnamurthy et al. (2021) each show that a model’s judgement can be converted into a rule or a human-readable explanation, offline, against labelled data; Li et al. (2023) extend this to unsupervised anomaly detectors specifically. Among the studies found by this search, none addresses an online setting, where the “labels” available are whatever an operator chooses to confirm, not a static labelled corpus, which is the specific problem Section III-C’s shadow mechanism is built to handle.

Benchmark availability shapes what can currently be validated. CICIDS2017 (Sharafaldin et al., 2018) is general-purpose, not IoT traffic. Meidan et al. (2018) captured genuine IoT device traffic under real botnet infection (N-BaIoT/BoTNeT-IoT-L01), the corpus used in Section IV; Neto et al. (2023) more recently released CICIOT2023 with a wider attack taxonomy across 105 devices, not yet been obtained for this work.

III. ARS-IDS Architecture

A. Overview

ARS-IDS processes each flow through three stages: rule filtering, classification, and fusion, illustrated in Figure 1. A rule R_i is a conjunction of feature-threshold conditions mapping to an attack label; the rule set is checked first because it is cheap, and only flows it cannot resolve reach the classifier. Fusion combines both outputs: $D = 1$ if $R = 1$, or if the classifier flags the flow with confidence above a tuned threshold τ_M ; a missed attack is judged costlier than an extra alert, so the fusion rule is deliberately permissive toward raising an alarm. What makes the loop adaptive rather than static is ARS, described next.

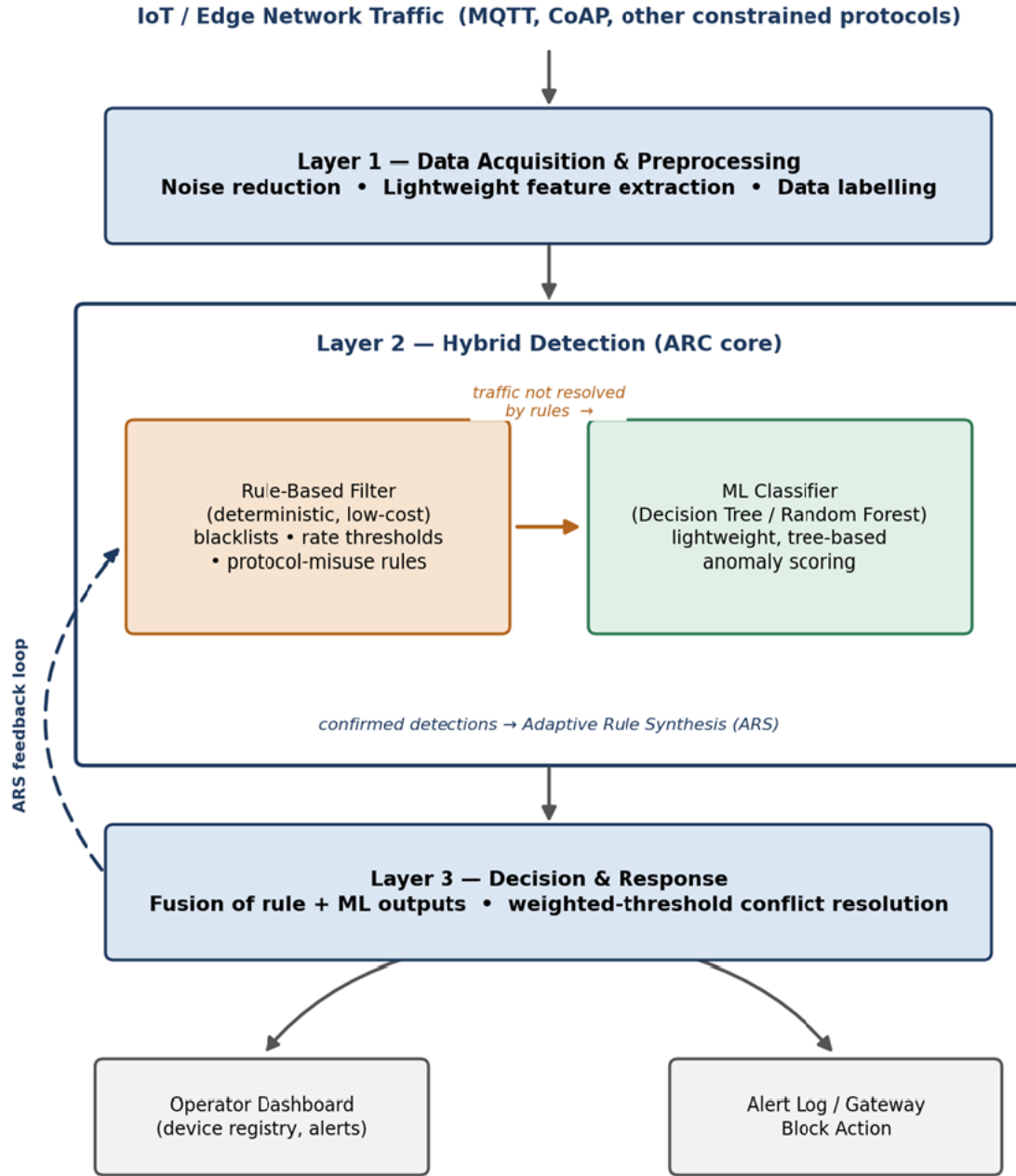


Figure 1. ARS-IDS: rule filtering and classification feed a fusion decision; ARS closes the loop by writing confirmed classifier detections back into the rule set.

B. Classification Layer and Hardware Tiering

ARS-IDS restricts its classifier to Decision Trees and Random Forests rather than deep networks, following the measured energy costs in Section II, and treats the two as targeting different hardware, not as interchangeable options: a single Decision Tree is small enough for a microcontroller-class endpoint (ESP32), while a Random Forest is reserved for gateway-class hardware (Raspberry Pi), where its larger memory footprint is affordable. Section IV tests both under an identical protocol specifically to check whether this tiering claim holds, rather than asserting it.

C. Adaptive Rule Synthesis

ARS is triggered only when the classifier catches what the rule layer missed, $R = 0$ and $M = 1$; a detection the rule layer already caught is not re-synthesised, since a new rule for already-covered traffic adds bookkeeping without adding coverage. Algorithm 1 states the full procedure.

Algorithm 1: Adaptive Rule Synthesis

Input: flow x , rule output R , classifier output $(M, P(M))$

```
1: if  $R = 0$  and  $M = 1$  and  $P(M) \geq \tau_C$  then
2:    $id \leftarrow \text{Hash}(\text{AttackType}, \text{Protocol}, \text{FeaturePattern}(x))$ 
3:   if  $\text{count}(id, \text{window } w) \geq k$  then      # confirmed
4:      $R_{\text{cand}} \leftarrow \text{ExtractPath}(\text{classifier}, x)$  # DT: leaf path;
                                     # RF: support-based
                                     # consensus over
                                     # attack-voting trees
5:      $\text{Shadow}(R_{\text{cand}})$                 # non-blocking;
                                     # log matches only
6:     while awaiting operator acknowledgement:
7:        $\text{Precision}_{\text{hat}} \leftarrow \text{ack\_true} / (\text{ack\_true} + \text{ack\_false})$ 
8:        $\text{Coverage}_{\text{hat}} \leftarrow \text{ack\_true} / (\text{ack\_true} + \text{ack\_false} + \text{pending})$ 
9:       if  $\text{Precision}_{\text{hat}} \geq P_{\text{min}}$  and  $\text{Coverage}_{\text{hat}} \geq C_{\text{min}}$  then
10:         $\text{Admit}(R_{\text{cand}})$                 # -> Active
11:       else:  $\text{Discard}(R_{\text{cand}})$ 
12: # Active rules are Monitored and Pruned on FP drift,
    # disuse, or rule-set overflow (Figure 2)
```

Two design choices here are deliberate rather than incidental. First, the pattern identifier at line 2 is behavioural, a hash over attack type, protocol, and feature pattern, rather than source-address-based, because a source-based criterion is defeated trivially by a distributed botnet or spoofed addresses. Second, and more consequentially, lines 6-9 do not assume access to ground truth: a live IDS has no labelled oracle, so $\text{Precision}_{\text{hat}}$ and $\text{Coverage}_{\text{hat}}$ are estimated from what an operator actually acknowledges, with unacknowledged matches (pending) counted as a distinct, unresolved outcome rather than silently treated as false. This is weaker evidence than an offline label by construction, since acknowledgement is selective and delayed; consequently, we propose that any concrete (P_{min} , C_{min}) configuration should first be validated statistically against labelled offline data (Section V) before this shadow mechanism is trusted to admit rules live, an ordering this paper states as a requirement rather than an assumption its design already satisfies.

Illustration. A flood of short flows sharing feature signature f scores $M = 1$ at $P(M) = 0.96$; five occurrences of the same PatternID arrive inside a 60-second window, confirming the detection. The classifier’s path for this leaf is $\text{mean_packet_size} < 64 \wedge \text{inter_arrival_std} < 0.01$, both header/timing-level attributes rather than payload content, written directly as the candidate rule. Over the next 50 shadow-window matches, 40 are acknowledged (36 true, 4 false) and 10 remain pending past the response window; pending matches are excluded from $\text{Precision}_{\text{hat}}$ and re-queued at higher priority rather than defaulting toward admission, so an attacker cannot promote a rule simply by triggering matches an operator never reviews. $\text{Precision}_{\text{hat}} = 36/40 = 0.90$, $\text{Coverage}_{\text{hat}} = 36/50 = 0.72$. With $P_{\text{min}} = C_{\text{min}} = 0.7$, the rule is admitted.

Admitted rules carry creation time, source detection, match count, and false-positive count, and move through the lifecycle in Figure 2: Active while healthy, Monitored continuously, and Pruned on false-positive drift, disuse past an expiry window, or rule-set overflow.

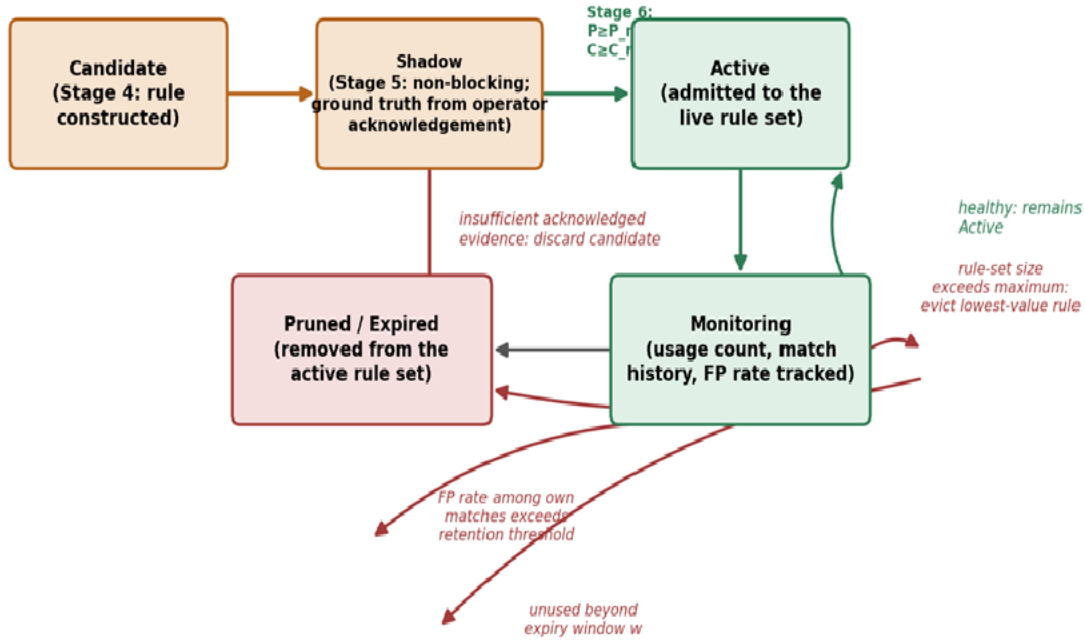


Figure 2. ARS rule lifecycle, from a validated Shadow candidate through Active use to eventual Pruning.

IV. Preliminary Validation

We validate the classification-layer choice in Section III-B, not the Architecture end to end. The dataset is BoTNetIoT-L01 (N-BaIoT; Meidan et al., 2018): 7,062,606 flow records from nine IoT devices under Mirai/Gafgyt attack. Its label field was found, on inspection, to use inverted polarity, malicious flows labelled 0; the target used throughout was derived from the Attack field directly rather than the raw label. A 30% stratified subsample (memory-limited to what a 4-core development workstation could hold) preserved the corpus’s true class balance.

A first pass, a random 15%/85% split with cross-validated model selection, returned near-perfect scores that we did not take at face value: a random split does not respect device boundaries, so near-duplicate flow vectors from the same attack burst can appear on both sides of the split. Leave-one-device-out (LODO), training on eight devices and testing on the ninth, is the test that cannot leak this way, and it is the result we report. Table I gives per-device accuracy, recall, and false-positive rate for a Decision Tree and a Random Forest evaluated on identical folds, generated directly from the saved confusion matrices rather than transcribed by hand.

Table I: Leave-one-device-out results, Decision Tree vs. Random Forest (Acc = Accuracy, Rec = Recall)

Held-out device	DT Acc	DT Rec	DT FPR	RF Acc	RF Rec	RF FPR (false positive rate)
Danmini Doorbell	100.000%	100.000%	0.007%	100.000%	100.000%	0.007%
Ecobee Thermostat	99.997%	99.999%	0.126%	100.000%	100.000%	0.025%
Ennio Doorbell	99.992%	100.000%	0.076%	99.999%	100.000%	0.008%

Philips Baby Monitor	99.910%	100.000%	0.568%	99.985%	100.000%	0.092%
Provision PT 737E	99.997%	99.997%	0.005%	99.997%	99.999%	0.021%
Provision PT 838	99.945%	100.000%	0.464%	99.992%	100.000%	0.064%
Samsung Webcam	99.979%	100.000%	0.154%	99.995%	100.000%	0.038%
SimpleHome XCS7 1002	99.498%	99.999%	9.303%	99.928%	100.000%	1.326%
SimpleHome XCS7 1003	100.000%	100.000%	0.000%	99.999%	100.000%	0.035%
Pooled (9 devices)	99.916%	99.999%	1.063%	99.987%	100.000%	0.160%

Pooled confusion matrices: DT TP=1,951,988 TN=165,007 FP=1,773 FN=14 (n=2,118,782); RF TP=1,951,996 TN=166,513 FP=267 FN=6. Neither model reaches literal 100% recall on every device, stated precisely rather than rounded away: the Decision Tree misses at least one flow on 6 of 9 devices; the Random Forest’s six pooled false negatives are spread across 4 of 9 devices, three on Provision PT 737E, one each on three others whose recall still rounds to 100.000% at the precision reported here, only Provision PT 737E’s recall (99.999%) fails to round to 100.000%. Pooled recall rounds to 100.000% for the Random Forest and 99.999% for the Decision Tree. FPR is the more informative axis: SimpleHome XCS7 1002 is a genuine outlier for both models, and the Random Forest’s tiering advantage is visible exactly there, holding FPR to 1.326% where the single tree reaches 9.303%, a difference that only becomes visible because both models were tested identically. Its near-identical sibling unit, XCS7 1003, shows no such weakness for either model, which suggests a device-instance effect rather than a manufacturer-wide one, though two units is a thin basis for that conclusion. This is evidence against device-identity leakage specifically, not against the untested temporal leakage a device-level split cannot address.

Figure 3. Pooled leave-one-device-out results on BoTNeTIoT-L01: Decision Tree vs. Random Forest

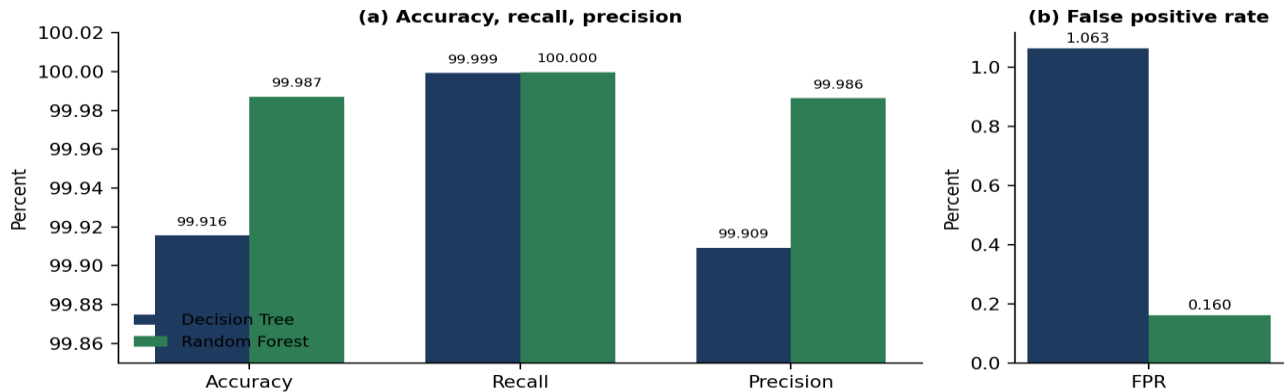


Figure 3. Pooled leave-one-device-out results on BoTNeTIoT-L01: Decision Tree vs. Random Forest, accuracy/recall/precision (left) and false positive rate (right).

This supports Section III-B’s tiering claim, a Random Forest earns its larger footprint with materially better worst-case behaviour, but it establishes nothing about the rule layer, ARS, or fusion, none of

which were exercised, and nothing about performance on target IoT hardware, since every measurement here ran on the workstation described above, not a Raspberry Pi or ESP32. It also covers one attack family; BoTNeTIoT-L01 contains no infiltration, web, or protocol-specific traffic, which a broader-taxonomy dataset such as CICIOT2023 would be needed to test. Full reproducibility detail (software versions, exact seeds, hyperparameter grid, and confidence intervals around these point estimates) is reported in an extended treatment of this work rather than repeated here.

V. Discussion and Next Steps

ARS-IDS's central open question is not whether ARS is well-specified, Section III-C states it precisely, but whether the shadow-evaluation proxy it depends on actually behaves well once real, slow, selective operator acknowledgement replaces the abstraction in Algorithm 1. That question cannot be answered offline. Three further gaps follow directly from Section IV's scope: the rule layer has no executable seed-rule set yet, only the specification in Section III-A; ARS itself has not been exercised, online or offline; and no CPU, memory, or latency figures exist for the Raspberry Pi or ESP32 targets this Architecture is designed for.

The concrete next steps, in order, are: instantiate an executable seed-rule set and test rule-layer performance against packet-level data CICIOT2023 would provide; run ARS against a replayed, time-ordered detection stream to test the shadow mechanism under realistic acknowledgement delay, before trusting it live, as Section III-C requires; and deploy the resulting pipeline to physical hardware for the resource measurements this paper has deliberately not estimated. Each is a precondition for calling ARS-IDS validated rather than specified.

References

- Alcântara, L., Padilha, G., Abreu, R., & d'Amorim, M. (2021). Syrius: Synthesis of rules for intrusion detectors. *IEEE Transactions on Reliability*, 70(1), 370–381. <https://doi.org/10.1109/TR.2021.3061297>
- Alexander, M., & Kumar, D. (2023). Hybrid rule based classification of attacks in Internet of Things (IoT) intrusion detection system. In *2023 7th International Conference on Computing Methodologies and Communication (ICCMC)* (pp. 1249–1254). IEEE. <https://doi.org/10.1109/ICCMC56507.2023.10083504>
- Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J. A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K., & Zhou, Y. (2017). Understanding the Mirai botnet. In *Proceedings of the 26th USENIX Security Symposium* (pp. 1093–1110). USENIX Association.
- Bebortta, S., Panda, T., & Singh, S. K. (2023). An intelligent hybrid intrusion detection system for Internet of Things-based applications. In *2023 International Conference on Network, Multimedia and Information Technology (NMITCON)* (pp. 1–6). IEEE. <https://doi.org/10.1109/NMITCON58196.2023.10275968>
- Daghero, F., Burrello, A., Macii, E., Montuschi, P., Poncino, M., & Pagliari, D. J. (2023). Dynamic decision tree ensembles for energy-efficient inference on IoT edge nodes. *IEEE Internet of Things Journal*, 10(1), 742–757. <https://doi.org/10.1109/JIOT.2023.3286276>
- Fatima, M., Rehman, O., & Rehman, I. M. H. (2023). Li-IDS: An approach towards a lightweight IDS for resource-constrained IoT. In *2023 International Conference on Smart Applications, Communications and Networking (SmartNets)* (pp. 1–6). IEEE. <https://doi.org/10.1109/SmartNets58706.2023.10216096>

- Krishnamurthy, P., Khorrami, F., Schmidt, S., & Wright, K. (2021). Machine learning for NetFlow anomaly detection with human-readable annotations. *IEEE Transactions on Network and Service Management*, 18(2), 1885–1898. <https://doi.org/10.1109/TNSM.2021.3075656>
- Li, R., Li, Q., Zhang, Y., Zhao, D., Jiang, Y., & Yang, Y. (2023). Interpreting unsupervised anomaly detection in security via rule extraction. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*.
- Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Breitenbacher, D., Shabtai, A., & Elovici, Y. (2018). N-BaIoT: Network-based detection of IoT botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 17(3), 12–22. <https://doi.org/10.1109/MPRV.2018.03367731>
- Neto, E. C. P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., & Ghorbani, A. A. (2023). CIIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. *Sensors*, 23(13), 5941. <https://doi.org/10.3390/s23135941>
- Piet, J., Nwoji, D., & Paxson, V. (2023). GGFASST: Automating generation of flexible network traffic classifiers. In *Proceedings of the ACM SIGCOMM 2023 Conference*. ACM. <https://doi.org/10.1145/3603269.3604840>
- Roy, S., Li, J., Choi, B.-J., & Bai, Y. (2022). A lightweight supervised intrusion detection mechanism for IoT networks. *Future Generation Computer Systems*, 127, 276–285. <https://doi.org/10.1016/j.future.2021.09.027>
- Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)* (pp. 108–116). SciTePress. <https://doi.org/10.5220/0006639801080116>
- Tekin, N., Acar, A., Aris, A., Uluagac, A. S., & Gungor, V. C. (2022). Energy consumption of on-device machine learning models for IoT intrusion detection. *Internet of Things*, 21, 100670. <https://doi.org/10.1016/j.iot.2022.100670>
- Zhao, B., Ji, S., Lee, W.-H., Lin, C., Weng, H., Wu, J., Zhou, P., Fang, L., & Beyah, R. (2022). A large-scale empirical study on the vulnerability of deployed IoT devices. *IEEE Transactions on Dependable and Secure Computing*, 19(3), 1826–1840. <https://doi.org/10.1109/TDSC.2020.3037908>