

SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: <https://www.svedbergopen.com/>

Research Paper

Open Access

Mapping Security Vulnerabilities and Malware Threats in Android-Based IoT Ecosystems: A Systematic investigation

S Balakrishna Reddy¹, Dr. Sesaiah Merikapudi²

¹Ph.D. Scholar, Department of Computer Science and Engineering, SJC Institute of Technology, Chickballapur, Karnataka, India, Affiliated to Visvesvaraya Technological University (VTU), Belagavi-590018. E-mail: sama.balakrishnareddy@gmail.com

²Professor, Department of Computer Science and Engineering, SJC Institute of Technology, Chickballapur, Karnataka, India, Affiliated to Visvesvaraya Technological University (VTU), Belagavi-590018. E-mail: merikapudi@gmail.com

ABSTRACT

The convergence of the Android operating system with the Internet of Things (IoT) has produced a vast and heterogeneous ecosystem of smartphones, wearables, smart-home hubs, connected cameras, automotive head units, and industrial edge controllers that all share a common software lineage. This convergence has amplified both the reach and the consequences of cyberattacks, because vulnerabilities inherited from the mobile platform now extend into physical environments, critical infrastructure, and always-on sensor networks. This paper undertakes a systematic investigation of the security challenges and malware threats that specifically affect Android-based IoT devices. Drawing on recent peer-reviewed literature spanning static, dynamic, network-traffic, graph-based, and explainable deep learning approaches, the study first synthesizes the state of the art in Android and IoT malware research, highlighting recurring methodological patterns as well as unresolved limitations such as poor cross-device generalization, concept drift, adversarial fragility, and the scarcity of representative labelled data. Building on this synthesis, the paper develops a structured taxonomy that organizes the threat landscape into five interrelated dimensions: platform and firmware vulnerabilities, network and communication-layer threats, malware families that specifically target constrained IoT endpoints, companion-application and data-privacy risks, and adversarial threats aimed at the AI-based defences themselves. The paper further reviews the public datasets and benchmarks that underpin experimental work in this domain and discusses their coverage gaps. Finally, informed by the identified challenges, a conceptual deep learning-oriented detection and security-enhancement outline is presented, integrating multi-source feature fusion, lightweight on-device inference, federated collaborative learning, adversarial robustness, and explainability as design requirements for future defence systems. The investigation concludes that securing Android-based IoT devices requires detection frameworks that are simultaneously resource-aware, privacy-preserving, generalizable across device families, and resilient to evasion, and it outlines the research directions most likely to close the gap between laboratory performance and real-world deployment.

Keywords: Android Security, Internet of Things, IoT Malware, Deep Learning, Botnets, Network Intrusion Detection, Adversarial Robustness, Explainable AI, Federated Learning, Mobile Threat Landscape

Introduction

Android has grown far beyond the smartphone into the operating layer of the Internet of Things (IoT). Wearable fitness trackers, smart televisions, connected door locks, in-vehicle infotainment units, home security cameras, industrial handheld scanners, and voice-controlled hubs increasingly run on Android or Android-derived firmware, sharing its application framework, permission model, and Linux kernel base. This convergence has been commercially attractive because it lets device manufacturers reuse a mature software stack, a familiar developer ecosystem, and a very large pool of pre-built libraries instead of building bespoke embedded operating systems. At the same time, it means that every architectural weakness, every permission-model loophole, and every malware family that has historically targeted Android smartphones now has a direct pathway into physical devices that sense, actuate, and often operate unattended for years. Industry telemetry illustrates how quickly this threat surface is expanding: Zscaler's ThreatLabz research group reported a 67 percent year-on-year increase in Android malware transactions and found that manufacturing and transportation together accounted for roughly forty percent of all observed IoT malware incidents, with the United States alone receiving more than half of global IoT attack traffic [2]. Google's own defensive telemetry is similarly indicative of scale rather than containment: Google Play Protect now scans in excess of 350 billion application instances every day and, in 2025 alone, blocked 1.75 million policy-violating submissions while identifying more than 27 million malicious applications distributed outside the official store [3].

The stakes of this convergence became globally visible with the Mirai botnet, which in 2016 compromised hundreds of thousands of embedded and IoT devices, most of them running stripped-down Linux firmware with default or hard-coded credentials, and used them to launch some of the largest distributed denial-of-service attacks recorded at the time [4]. Although Mirai itself targeted BusyBox-based firmware rather than Android specifically, the incident reframed IoT devices as a durable attack platform rather than a peripheral curiosity, and it directly motivated the creation of network-traffic benchmarks such as N-BaIoT, which captured genuine botnet infection traffic from nine commercial IoT devices for use in machine learning research [5]. In the years since, security researchers have

documented an expanding and increasingly organized threat landscape spanning cross-architecture IoT malware families, obfuscated Android trojans repackaged for constrained hardware, and adversarially engineered samples designed specifically to evade learning-based detectors [6], [7]. Android-based IoT devices sit at an uncomfortable intersection of these two research traditions: they inherit the enormous, well-studied Android malware corpus, yet they also inherit the resource constraints, heterogeneous connectivity, and weak patch cadence that characterize the broader IoT.

Despite a decade of progress in machine learning and deep learning-based malware detection, most techniques were designed and validated on either smartphone-centric datasets or network-only IoT botnet traces, rarely both, and rarely with the constraints of embedded deployment in mind [1], [8]. Detectors trained on static permission or API-call features from datasets collected years ago degrade rapidly as the underlying application ecosystem drifts, a phenomenon formally documented in space-and-time bias studies of malware classifiers [9], [10]. This creates a pressing need to step back from any single detection technique and instead systematically investigate the full breadth of security challenges and malware threats that specifically affect Android-based IoT devices, so that future defence frameworks can be designed against the actual shape of the problem rather than against a convenient subset of it. That is the objective of this paper. It synthesizes recent literature on Android and IoT malware detection, organizes the threat landscape into a structured taxonomy, reviews the datasets that underpin experimental research in the field, and closes with a conceptual outline of the design requirements that a deep learning-oriented defence framework for this domain should satisfy.

1.1 Why Android-Based IoT Security Is a Distinct Problem

Treating Android-based IoT security as a simple extension of smartphone security, or as a simple extension of embedded-Linux IoT security, understates the problem in three ways. First, Android-based IoT devices are frequently far more resource-constrained than the smartphones on which most detection research is validated, so a classifier that performs well on a flagship handset may be computationally infeasible on a doorbell camera or a wearable, forcing a trade-off between detection depth and on-device inference cost that smartphone-centric literature rarely has to confront [1], [6]. Second, many of these devices operate through a companion smartphone application that configures the device, requests its own broad permissions, and communicates with cloud back-ends on the device's behalf; the attack surface therefore spans the physical device, its firmware, its network traffic, and a separate Android application that is often developed with far less security scrutiny than the device itself [48]. Third, the operational context changes the cost of both false negatives and false positives: an undetected compromise of a smart lock, a medical wearable, or an industrial sensor has physical-world consequences that a compromised note-taking app does not, while an IoT device has none of a smartphone's user interface through which a false alarm could be reviewed or dismissed by an attentive owner. These three characteristics motivate treating Android-based IoT malware detection as its own research problem, with its own taxonomy, its own datasets, and its own design requirements, rather than as a straightforward transfer of results from either parent domain.

II. Literature Review

Research on Android and IoT malware detection has grown along two historically separate tracks that are now converging around Android-based IoT devices: a mobile-security track built on static and dynamic analysis of application packages, and a network-security track built on traffic analysis of embedded and constrained devices. This section reviews recent contributions from both tracks, emphasizing their methodology, reported findings, and the limitations that motivate the taxonomy developed in Section III.

Almobaideen et al. conducted a broad review of machine learning and deep learning techniques applied jointly to Android and IoT malware detection, observing that the two device classes are increasingly discussed together in the literature precisely because Android has become the de facto operating layer for a large share of consumer IoT hardware [1]. The review catalogued feature extraction strategies ranging from permissions and API calls to network flow statistics, and it concluded that although individual detectors report high accuracy on their own benchmark, comparative claims across studies are difficult to sustain because evaluation protocols, feature sets, and dataset vintages differ substantially from one paper to the next.

Liu et al. carried out a systematic literature review specifically focused on deep learning-based defences for Android malware, screening several hundred candidate studies to characterize how the field selects features, architectures, and evaluation datasets [11]. Their synthesis found a strong concentration of studies around a small number of aging datasets and reported that very few works evaluate temporal robustness, meaning that a model's accuracy figures often reflect performance on samples drawn from the same narrow time window used for training. Bensaoud et al. and Gopinath and Sethuraman independently reached a similar conclusion in broader surveys of deep learning-based malware detection that were not limited to Android, noting that convolutional, recurrent, and hybrid architectures have all been applied successfully to malware classification but that reproducibility and cross-dataset generalization remain open problems across the field [12], [13].

Qiu et al. surveyed the use of deep neural models for Android malware detection and organized the literature by input representation, distinguishing approaches that operate on raw bytecode, on structured feature vectors derived from static analysis, and on behavioral sequences captured through dynamic execution [8]. The survey highlighted that deep models trained on structured static features tend to achieve the strongest reported accuracy on held-out test partitions, but that this accuracy is frequently inflated by data leakage between the training and test splits of widely reused

datasets, a concern that echoes the broader temporal-bias findings of Pendlebury et al., whose TESSERACT methodology demonstrated that malware classifiers evaluated without proper temporal separation can overstate real-world performance by a wide margin [10]. Guerra-Manzanares extended this critique in a review that asked whether the field's stated objective, mission-accomplished for Android malware detection, was in fact justified, concluding that open challenges around concept drift, adversarial evasion, and limited operational deployment mean the problem remains substantially unsolved despite two decades of published progress [7].

Razgallah et al. reviewed malware detection specifically within the Android application ecosystem and proposed a set of recommendations for future research, arguing that studies should report performance separately for malware families rather than in aggregate, since aggregate accuracy figures can mask very poor detection of rare but dangerous families such as ransomware and ferocious data-exfiltration trojans [15]. Bilot et al. surveyed the more recent shift toward graph representation learning for malware detection, observing that representing an application as a graph of API calls, control-flow relationships, or function-call topology preserves structural information that flat feature vectors discard, and that this structural information appears to improve robustness to code obfuscation across the studies they reviewed [14].

On the dataset side, Arp et al. introduced Drebin, a static-analysis dataset and companion linear-classifier method that extracts permissions, API calls, hardware component requests, and intent filters directly from the application manifest and Dalvik bytecode, and the resulting feature representation has become one of the most widely reused benchmarks in the field due to its interpretability and its unusually explicit mapping between individual features and human-readable app behavior [39]. The Canadian Institute for Cybersecurity subsequently released CICAndMal2017, which was collected by installing both malicious and benign applications on physical smartphones rather than emulators, specifically to avoid the runtime-evasion behavior that some malware exhibits when it detects an emulated environment, and the dataset organizes its malware samples into adware, ransomware, scareware, and SMS-fraud categories drawn from dozens of distinct families [40]. Karbab et al. proposed MalDozer, a deep learning framework that classifies Android applications directly from sequences of raw API method calls without manual feature engineering, reporting strong detection and family-attribution accuracy while requiring comparatively little preprocessing, which the authors positioned as an advantage for deployment on devices where computational headroom for feature extraction is limited [16].

Turning to work that targets IoT devices directly, Lei et al. proposed EveDroid, an event-aware Android malware detector explicitly designed to resist the model-degrading effects of concept drift on IoT devices, using behavioral events rather than static signatures so that the model remains sensitive to malicious intent even as benign application behavior evolves over time [9]. Jeon, Park, and Jeong combined dynamic analysis with a convolutional neural network to detect IoT malware, converting runtime behavioral traces into an image-like representation that the CNN could classify, and reported that this transformation improved detection of malware variants that shared little static code overlap with previously seen samples [17]. Dhanya et al. pursued a related idea for obfuscated Android applications running on IoT hardware, converting opcode sequences into Markov-transition images before applying a CNN, and found that the image-based representation retained discriminative power even after the applications had been deliberately obfuscated to evade signature-based scanners [18].

Botnet-oriented research has followed a parallel but distinct trajectory grounded in network-traffic analysis rather than application analysis. Meidan et al.'s N-BaIoT study infected nine commercial IoT devices with the Mirai and BASHLITE botnet families and used deep autoencoders trained exclusively on benign traffic to flag anomalous deviations at inference time, a design choice motivated by the observation that IoT botnet variants evolve faster than any signature database can be updated, so an anomaly-based approach that never needs to see an attack sample during training is inherently more future-proof [5]. Ali et al. proposed a multitask deep learning model that performs IoT malware detection and family identification simultaneously from behavioral network-traffic features, reporting that sharing representations across the two tasks improved identification accuracy relative to training separate single-task models, while Al-Fawa'reh et al. applied deep reinforcement learning to botnet detection in IoT networks, framing detection as a sequential decision problem in which the agent is rewarded for correctly flagging malicious flows with minimal latency [19], [20]. Wei et al. and Nimalasingam et al. both benchmarked classical and deep learning approaches directly on IoT network-traffic captures, with the former reporting that ensemble and deep sequence models consistently outperformed shallow classifiers on the traffic streams they tested, and the latter emphasizing that forensic-quality feature extraction from raw packet captures remains a significant engineering bottleneck that is rarely discussed in papers that instead assume pre-extracted flow features are already available [41], [42]. Feng et al. showed that a two-layer deep learning pipeline applied to Android network traffic, rather than to the application package itself, could detect malicious behavior purely from communication patterns, which is a particularly relevant capability for constrained IoT endpoints where installing an on-device static analysis agent may not be feasible [21].

A growing body of work has explored graph neural networks (GNNs) as a way to capture the relational structure of Android applications more faithfully than flat feature vectors. Gao, Cheng, and Zhang proposed GDroid, which represents an application as a graph connecting API calls and applies graph convolution to learn a classification embedding, reporting that the relational structure improved detection of malware families that reuse code fragments across otherwise dissimilar applications [33]. Li et al. proposed DMalNet, which fuses API-level feature engineering with graph learning over dynamically observed call sequences, and reported that combining engineered features with learned graph representations outperformed either approach in isolation [34]. Zhang et al. and Zaidi et al. both explored variations on this theme, the former using hypergraphs to capture higher-order relationships among API calls that

pairwise graphs cannot represent, and the latter applying standard graph convolutional networks and reporting competitive accuracy while noting that GNN-based detectors are substantially more expensive to train and to run at inference time than comparably accurate CNN-based detectors, a trade-off with direct implications for constrained IoT deployment [31], [32]. Meng and Li extended the graph-based line of work with HerTDroid, a heterogeneous graph transformer that filters influential nodes before classification, reporting improved robustness against the kind of benign-looking padding code that some malware authors insert specifically to dilute the discriminative signal available to graph-based classifiers [43].

A separate line of research has treated multimodal fusion of static, dynamic, and network-level signals as the more promising path forward. Zhang, Su, Liu, and Yang proposed MPDroid, a multimodal pre-training framework that jointly represents static and dynamic features of an application, reporting that pre-training on a large corpus of both types of features before fine-tuning on a labelled detection task improved generalization to malware families absent from the fine-tuning set [22]. Manzil similarly combined multiple analysis modes in DYNAMALDROID, a dynamic-analysis framework built specifically for machine learning-based Android malware detection, while Bashir et al. reported that a hybrid model combining several machine learning algorithms outperformed any single algorithm on their evaluation set, echoing a broader pattern across the literature in which ensemble and multimodal designs consistently, if modestly, outperform single-signal, single-model baselines [38], [37]. Ravi and Chaganti took a different multimodal route, converting Android application binaries into images and applying an EfficientNet-based meta-classifier, reporting that the image-based representation was particularly effective at capturing structural similarity among variants of the same malware family [35]. Mahindru et al. proposed PermDroid, which applies a dedicated feature-selection stage to the Android permission set before classification, reporting that selecting a compact, high-information subset of permissions preserved most of the detection accuracy of the full permission feature set while substantially reducing the computational cost of feature extraction, a property with obvious relevance for resource-constrained IoT deployment [36].

Because collecting centralized, cross-device malware data raises privacy and logistical concerns, particularly for IoT devices deployed inside private homes and healthcare settings, several recent studies have investigated federated learning as an alternative training paradigm. Rey et al. proposed a federated learning framework for IoT malware detection in which participating devices train locally and share only model updates rather than raw traffic, reporting detection performance close to that of a centralized model while additionally studying the framework's vulnerability to poisoning by malicious participants [23]. Alamer extended this line of work with a privacy-preserving federated scheme incorporating secure aggregation specifically for malware detection across IoT resources, and Amjath, Henna, and Rathnayake applied a graph-based federated learning formulation to malware detection in the Internet of Health Things, reporting that combining graph representations with federated training preserved detection accuracy across clinically sensitive, non-uniformly distributed device data while keeping raw patient-adjacent data local to each participating site [24], [25].

Finally, a growing number of studies have foregrounded explainability and adversarial robustness as first-class design requirements rather than afterthoughts. Palma, Ferreira, and Figueiredo applied explainable machine learning techniques to Android malware detection and reported that feature-attribution explanations helped identify which permissions and API calls most consistently distinguished malicious from benign applications across their dataset, information that is valuable both for analyst trust and for guiding future feature engineering [26]. Manthena et al. surveyed the broader use of explainable artificial intelligence in malware analysis, cataloguing SHAP, LIME, and Grad-CAM-style techniques and noting that most explainability work in the malware domain has so far been applied after the fact to already-trained black-box models rather than built into the model architecture from the outset [27]. On the robustness side, Grosse et al. were among the first to demonstrate that small, carefully crafted perturbations to a malware sample's feature representation could cause a trained classifier to misclassify it as benign, establishing adversarial evasion as a concrete threat to learning-based malware defenses rather than a purely theoretical concern [30]. Bhusal and Rastogi surveyed the resulting decade of adversarial-pattern research on Android malware classifiers specifically, distinguishing reactive defenses that are trained after an attack is observed from proactive defenses designed in anticipation of evasion attempts, while Bostani, Zhao, and Moonsamy showed that a large share of an Android classifier's apparent robustness can be explained by spurious correlations in the training data rather than genuine resistance to evasion, and proposed a domain-adaptation technique that measurably narrowed this gap on the Drebin benchmark [28], [29]. Magdy, Abu-Krishna, and Aslan's recent survey of mobile malware detection situates these threads together, concluding that explainability and adversarial robustness are converging from separate subfields into a shared expectation for any detector intended for real deployment rather than for benchmark competition alone [44].

Table 1. Summary of Representative Literature on Android and IoT Malware Detection

Ref.	Focus	Approach	Key Contribution / Finding
[1]	Android + IoT survey	Cross-domain literature synthesis	Documented convergence of Android and IoT malware research but found inconsistent evaluation protocols across studies.

Ref.	Focus	Approach	Key Contribution / Finding
[11]	Android, deep learning	Systematic literature review	Found heavy reliance on a small set of aging datasets and limited testing of temporal robustness.
[10]	Temporal bias	TESSERACT evaluation methodology	Showed that ignoring temporal ordering of samples inflates reported classifier accuracy.
[5]	IoT botnets (N-BaIoT)	Deep autoencoders on device traffic	Anomaly detection trained only on benign traffic generalized to unseen Mirai/BASHLITE variants.
[9]	IoT, concept drift	Event-aware detection (EveDroid)	Behavioral events resisted model degradation better than static signatures over time.
[39]	Static analysis dataset	Drebin feature extraction and linear model	Established a widely reused, interpretable permission/API-based benchmark for Android malware.
[33]	Graph learning	GDroid graph convolutional network	Relational API-call structure improved detection of code-reusing malware families.
[23]	Privacy-preserving training	Federated learning across IoT devices	Federated models approached centralized accuracy while keeping raw traffic local to devices.
[30]	Adversarial robustness	Feature-space evasion attack	Demonstrated that small crafted perturbations can flip a malware classifier's decision.
[22]	Multimodal fusion	MPDroid static/dynamic pre-training	Joint pre-training on static and dynamic features improved generalization to unseen families.

III. Taxonomy Of Security Vulnerabilities And Malware Threats In Android-Based IoT Devices

Synthesizing the literature reviewed above, the security challenges and malware threats affecting Android-based IoT devices can be organized into five interrelated categories that differ in where the vulnerability originates, how the corresponding threat manifests, and which forensic or behavioral signal is best suited to detect it. This taxonomy is not intended to imply that the categories are mutually exclusive in practice, since a single real-world incident, such as a compromised smart camera enrolled into a botnet through a companion-app credential leak, can span several categories at once. Rather, the taxonomy is intended to make explicit the range of design requirements that a comprehensive Android-based IoT defense framework must satisfy, since a detector optimized for one category alone will typically leave the others uncovered. Table 2 summarizes the taxonomy before the following subsections describe each category in more detail.

A. Platform, Firmware, and Update-Chain Vulnerabilities

The first category concerns weaknesses that originate in the underlying platform rather than in any specific application. Android-based IoT devices are typically built on customized, OEM-modified forks of the Android Open-Source Project, and this customization frequently lags behind upstream security patches by months or years, leaving known kernel and framework vulnerabilities exposed on shipped hardware long after they have been fixed in mainline Android. Industry-scale telemetry supports the severity of this category: an analysis of tens of millions of monitored consumer IoT devices found that televisions, smart plugs, and digital video recorders accounted for the largest shares of newly discovered device vulnerabilities, and that buffer-overflow and denial-of-service flaws together accounted for more than half of all vulnerability types identified across the studied device population [46]. A separate industry assessment covering billions of connected devices similarly reported hundreds of thousands of attacks observed daily across the global IoT device population, underscoring that platform-level weaknesses are not a theoretical concern but an actively and continuously exploited attack surface [45]. Weak or default authentication credentials compound these platform weaknesses: the Mirai botnet's initial and most effective propagation mechanism was a small dictionary of factory-default username and password pairs that remained unchanged on a very large share of the devices it ultimately compromised, a design flaw that persists on newly manufactured devices in many product categories today [4].

B. Network-Layer and Communication-Protocol Threats

The second category concerns threats that exploit how Android-based IoT devices communicate rather than how they are built. Because these devices are frequently deployed on home or enterprise networks alongside less carefully

secured legacy equipment, and because many communicate using lightweight protocols that were not designed with strong authentication or encryption in mind, they are exposed to distributed denial-of-service recruitment, man-in-the-middle interception, and command-and-control beaconing. The Mirai and BASHLITE botnet families remain the canonical examples of this category, propagating across compromised embedded devices to assemble attack infrastructure capable of overwhelming high-profile internet targets, and the network-traffic fingerprints these botnets leave behind, most notably characteristic spikes in outbound connection attempts and unusually regular beaconing intervals, form the basis of the anomaly-detection datasets and models discussed in Section II [4], [5]. More recent network-layer research has moved beyond signature-style detection toward behavioral and reinforcement-learning-based approaches specifically because botnet command-and-control infrastructure now rotates rapidly enough to evade static blocklists, with multitask and deep reinforcement learning models reporting improved detection of both known and previously unseen botnet behavior directly from traffic features [19], [20].

C. Malware Families Targeting Android-Based IoT Endpoints

The third category concerns the malicious software payloads themselves, once a device or its companion application has been compromised through a platform or network-layer weakness. Current threat telemetry indicates that the composition of this category is shifting quickly: adware-style monetization malware has become the single most frequently detected class of malicious Android application, while spyware families designed for surveillance, extortion, and identity theft grew substantially faster than any other category over the most recently reported measurement period, and banking trojans that steal financial credentials continue to represent a large and growing share of detected transactions [2]. Ransomware, ferocious data-wiping trojans, and SMS-fraud malware round out the categories represented in major benchmark datasets, and because these families are individually rarer than adware or generic trojans, several researchers have specifically cautioned that aggregate detection-accuracy figures can obscure very poor performance on exactly the family types whose consequences are most severe [15], [40]. On resource-constrained IoT hardware, malware authors have also adapted obfuscation techniques originally developed for smartphone malware, including opcode-level transformation and packing, specifically to evade the lighter-weight, lower-latency scanning that constrained devices are able to perform [18].

D. Companion-Application and Data-Privacy Threats

The fourth category concerns the smartphone-side companion application that configures, pairs with, and often continuously communicates with the IoT device on the user's behalf. Because the physical device itself typically has no screen through which a user could review permissions or notice suspicious behavior, the companion application is frequently the only interface through which a user's trust in the overall product is mediated, and it is also frequently developed under weaker security review than either the operating platform or the physical hardware. A systematic security evaluation of companion applications for consumer smart-security devices, including smart locks, cameras, and intruder alarms, found that every tested application was susceptible to at least one commonly occurring vulnerability regardless of whether the associated hardware came from a well-known or a lesser-known manufacturer, and that some companion applications requested permissions and embedded third-party trackers well beyond what the stated device functionality required [48]. This finding is consistent with permission-analysis research on the broader Android ecosystem, which has repeatedly shown that a compact subset of dangerous or unusual permission requests carries most of the discriminative signal separating malicious from benign applications, implying that companion applications requesting broad, loosely justified permission sets warrant particular scrutiny even before any malicious behavior is directly observed [26], [36].

E. Adversarial and Evasion Threats Against AI-Based Defenses

The fifth category is distinct from the first four in that it targets the defensive system itself rather than the device, network, or application under protection. As machine learning and deep learning-based detectors have become the default approach for Android and IoT malware classification, they have simultaneously become a target for adversarial manipulation. Early work demonstrated that small, carefully crafted perturbations to a malware sample's feature representation, additions or modifications that preserve the sample's underlying malicious functionality while changing its observed features, could reliably flip a trained classifier's decision from malicious to benign [30]. Subsequent research has shown that a meaningful share of a classifier's apparent robustness against such evasion attempts is actually attributable to spurious correlations learned from a narrow training distribution rather than genuine resistance to manipulation, meaning that robustness figures reported on a single static benchmark can be unreliable predictors of real-world resilience [28], [29]. Because this category specifically targets the defense mechanism, it has direct implications for the design of any detection framework proposed for this domain: a detector's own robustness to evasion must be treated as an explicit design requirement and evaluation criterion, not as a property that follows automatically from otherwise strong detection accuracy [7].

Table 2. Taxonomy of Security Vulnerabilities and Malware Threats in Android-Based IoT Devices

Category	Representative Threats	Primary Detection Cue / Representative Work
A. Platform, firmware, and update-chain vulnerabilities	Default/weak credentials, unpatched kernel and framework flaws, fragmented OEM firmware, insecure bootloaders	Firmware and configuration auditing; device vulnerability telemetry [45], [46]; default-credential propagation as in Mirai [4]
B. Network-layer and communication-protocol threats	Botnet recruitment, DDoS participation, command-and-control beaconing, man-in-the-middle interception	Traffic-flow anomaly detection and behavioral/reinforcement learning on network features [4], [5], [19], [20]
C. Malware families targeting IoT endpoints	Adware, spyware, banking trojans, ransomware, SMS-fraud malware, obfuscated payloads	Per-family detection accuracy on static/dynamic feature datasets [2], [15], [18], [40]
D. Companion-application and data-privacy threats	Over-broad permission requests, embedded trackers, insecure device-app pairing, weak companion-app review	Permission and static feature attribution; companion-app security evaluation [26], [36], [48]
E. Adversarial and evasion threats against AI defenses	Feature-space perturbation attacks, spurious-correlation exploitation, model poisoning in federated settings	Adversarial robustness evaluation and security curves [7], [28], [29], [30]

IV. Datasets and Benchmarks

Experimental progress in Android-based IoT security research is bounded by the datasets available to train and evaluate detection models, and the review in Section II shows that the field currently relies on a small number of benchmarks that were each built for one side of the Android/IoT divide rather than for their intersection. Table 3 summarizes the datasets most frequently used in the literature reviewed above.

Table 3. Datasets and Benchmarks Relevant to Android-Based IoT Security Research

Dataset / Benchmark	Source	Description
Drebin	Arp et al. [39]	Static-analysis dataset of 5,560 malware samples across 179 families and over 120,000 benign applications, with features drawn from permissions, API calls, hardware requests, and intents extracted from the application manifest and bytecode.
CICAndMal2017	Canadian Institute for Cybersecurity [40]	10,854 samples (4,354 malware, 6,500 benign) installed on physical smartphones rather than emulators, with malware organized into adware, ransomware, scareware, and SMS-fraud categories drawn from 42 families.
N-BaIoT	Meidan et al. [5]	Network-traffic captures from nine commercial IoT devices infected with the Mirai and BASHLITE botnet families, widely used to train and evaluate anomaly-based botnet detectors.
IoT behavioral / network-traffic corpora	Multiple studies [19], [20], [41], [42]	Device- and flow-level traffic captures assembled by individual research groups for multitask, reinforcement-learning, and forensic botnet-detection studies; coverage and labelling conventions vary considerably across studies.
Companion-application evaluation corpora	Allen et al. [48]	APK collections of smartphone companion applications paired with consumer smart-security devices, used to assess permission scope, embedded trackers, and static/dynamic vulnerabilities.

Two structural limitations emerge from this landscape. First, no dataset in current wide use jointly captures the application package, the device's network behavior, and the companion application's own code and permissions for

the same underlying product, which means that studies aiming to detect threats spanning more than one of the taxonomy categories in Section III must either combine incompatible datasets or restrict their scope to a single category. Second, several of the most heavily reused benchmarks, including Drebin and the original N-BaIoT captures, were collected years before this paper was written, and because both Android malware and IoT botnet families evolve quickly, models trained and validated exclusively on these older samples risk the same temporal-bias problem documented by Pendlebury et al. for smartphone-only malware classifiers [10]. Researchers have generally addressed dataset aging either by supplementing older benchmarks with more recently collected samples of their own, as several of the network-traffic and behavioral studies reviewed in Section II did, or by explicitly testing cross-family and cross-time generalization rather than relying on a single random train-test split [9], [17], [20]. Both practices should be treated as a minimum methodological standard for future work in this domain rather than as an optional robustness check, given how directly dataset vintage has been shown to inflate reported accuracy.

V. Synthesis Of Current Defense Paradigms And Open Challenges

Read together, the studies reviewed in Section II and the threat categories developed in Section III point to a consistent pattern: the research community has produced strong, narrowly scoped solutions to individual pieces of the Android-based IoT security problem, but comparatively little work has addressed the pieces jointly. Five open challenges recur across the literature and directly shape the design requirements for any framework intended to defend this class of device in practice.

Feature-Source Fragmentation

Most detectors reviewed in Section II commit to a single feature source: static application features in the Drebin-descended literature [39], [16], [36]; dynamic behavioral traces in the CNN and multitask IoT studies [17], [18], [20]; or raw network-traffic statistics in the botnet-detection literature [5], [19], [21]. Multimodal studies that fuse two or more of these sources are still comparatively rare, yet the few that exist report measurable accuracy gains from fusion, suggesting that single-source commitment is a limitation of convenience rather than of necessity [22], [37], [38]. Because the taxonomy in Section III shows that real threats routinely span platform, network, and application-level categories at once, a detector committed to a single feature source is structurally unable to observe threats that only manifest in a different source.

The Resource-Constraint Blind Spot

Graph neural network and transformer-based detectors report some of the strongest accuracy figures in the literature reviewed here [31], [32], [33], [43], but these architectures are also the most computationally expensive, and very few of the studies that propose them evaluate inference latency, memory footprint, or energy consumption on hardware representative of an actual constrained IoT endpoint rather than a research workstation or cloud GPU. This gap matters specifically for this domain: a wearable, doorbell camera, or industrial sensor cannot offload every classification decision to the cloud without reintroducing the latency, connectivity-dependence, and privacy costs that on-device detection is meant to avoid. Lightweight alternatives exist, including permission-selection approaches that discard low-information features before classification and image-based representations that reuse efficient convolutional backbones, but these efficiency-oriented techniques and the accuracy-oriented graph and transformer techniques have so far been evaluated largely in isolation from one another rather than on a shared accuracy-versus-efficiency frontier [35], [36].

Temporal Robustness and Concept Drift

Pendlebury et al.'s demonstration that ignoring the chronological order of malware samples can substantially inflate reported accuracy is one of the most consequential methodological findings in the field, yet the practice it warns against, evaluating on a random split of a static, aging dataset, remains common outside the subset of studies that explicitly target concept drift [10]. EveDroid and the multitask IoT detectors reviewed in Section II show that behavior-oriented and periodically-retrained detectors degrade more gracefully than purely static-signature detectors as the underlying malware and application ecosystems evolve, but drift-aware evaluation is still the exception in the wider literature rather than the norm [9], [20].

Adversarial Fragility

The taxonomy category described in Section III-E is not a hypothetical concern: feature-space evasion attacks against malware classifiers were demonstrated years ago, and more recent work has shown that a meaningful share of a classifier's apparent robustness against such attacks is attributable to spurious correlations rather than genuine resilience [30], [28], [29]. Because federated learning is increasingly proposed as a privacy-preserving training paradigm for exactly this domain, and because federated aggregation has itself been shown to be vulnerable to poisoning by malicious participants, robustness against adversarial manipulation must be treated as a property of the entire training and deployment pipeline rather than of the trained classifier in isolation [23].

Explainability as an Afterthought

Explainability techniques applied to malware detection have so far been used predominantly to interpret already-trained black-box models after the fact, rather than to shape model architecture or training objectives from the outset

[26], [27]. For Android-based IoT devices specifically, where an incorrect detection decision can trigger an automated response, such as isolating a device from the network, with no on-device user interface through which an affected owner could review or contest that decision, post-hoc explainability is arguably insufficient; the case for building interpretable decision pathways into the detector's architecture is correspondingly stronger in this domain than in general-purpose malware research.

Taken together, these five challenges argue against pursuing further incremental accuracy improvements on existing single-source, single-dataset benchmarks, and in favor of detection frameworks that are explicitly designed around multi-source fusion, resource awareness, temporal and adversarial robustness, and built-in explainability as joint requirements. Section VI outlines such a framework at the conceptual level, informed directly by the gaps identified here.

VI. Toward A Unified Deep Learning-Oriented Detection And Security-Enhancement Outline

The investigation in Sections II through V points toward a set of design requirements rather than a single algorithm, since no individual technique reviewed above addresses feature fragmentation, resource constraints, temporal and adversarial robustness, and explainability simultaneously. This section outlines, at a conceptual level, how a deep learning-oriented framework for Android-based IoT devices could be structured to satisfy these requirements jointly, drawing directly on the strongest ideas identified in the preceding review. The outline is deliberately conceptual: it is intended to define the architecture and design commitments that the next stage of this research program should implement and evaluate empirically, rather than to report new experimental results.

1. Multi-Source Feature Fusion

Rather than committing to a single feature source, the framework should ingest static application features (permissions, API calls, and manifest attributes, following the Drebin tradition [39]), dynamic behavioral traces of the kind used in CNN- and image-based IoT malware detectors [17], [18], network-flow statistics of the kind used in botnet detection [5], [21], and, where a companion application is present, the permission and code-level features of that application evaluated on its own terms [48]. A shared preprocessing stage would normalize these heterogeneous inputs into aligned representations before a fusion layer combines them, following the multimodal designs that reported measurable accuracy gains over single-source baselines in Section V [22], [37].

2. Resource-Aware, Tiered Backbone

To close the resource-constraint blind spot identified in Section V, the feature-extraction backbone should be tiered rather than monolithic: a lightweight, permission- and image-based classifier of the kind proposed by Ravi and Chaganti and by Mahindru et al. would run directly on the constrained device for continuous, low-latency screening, while a more expressive graph- or transformer-based model of the kind proposed by Gao, Cheng, and Zhang and by Meng and Li would run on a nearby gateway or edge server, invoked selectively when the lightweight tier flags a sample as suspicious rather than on every observation [35], [36], [33], [43]. This tiering directly targets the accuracy-versus-efficiency trade-off that Section V identified as evaluated in isolation rather than jointly in the current literature.

3. Federated, Privacy-Preserving Collaborative Learning

Because Android-based IoT devices are deployed across privacy-sensitive settings, including private homes and healthcare environments, the framework should adopt federated training in the manner demonstrated by Rey et al., Alamer, and Amjath, Henna, and Rathnayake, allowing a fleet of devices to collaboratively improve a shared detection model without centralizing raw traffic or application data [23], [24], [25]. Given the demonstrated vulnerability of federated aggregation to poisoning by compromised participants, the aggregation mechanism itself should incorporate the kind of robust-aggregation safeguards that federated malware-detection research has already begun to evaluate, rather than assuming participant honesty by default [23].

4. Adversarial Robustness Module

Given that feature-space evasion attacks and spurious-correlation exploitation have both been demonstrated against Android malware classifiers, the framework should incorporate adversarial training and the kind of domain-adaptation techniques proposed by Bostani, Zhao, and Moonsamy as a standing component of the training pipeline rather than as a one-time hardening step, and it should be evaluated using security curves that report accuracy under increasing attacker perturbation budgets rather than accuracy alone [30], [28], [29].

5. Built-In Explainability Layer

Rather than applying explanation techniques after the fact, the framework should surface feature-attribution and structural explanations, in the style of the permission- and API-level attributions used by Palma, Ferreira, and Figueiredo, as a direct output of each detection decision, alongside a manipulation or malicious-behavior confidence score [26], [27]. For a device class with no user-facing screen through which an owner could review a decision, these explanations are intended primarily for the fleet-management console or security operations interface responsible for the device, supporting analyst review before any automated remediation action such as network isolation is taken.

6. Fleet-Level Risk Scoring and Deployment Integration

Finally, the outputs of the preceding components, static and dynamic detection confidence, network-behavior anomaly scores, and adversarial-robustness confidence, should be combined into a single per-device risk score that a fleet-management platform can use to prioritize which devices require human review, an automated firmware update, or temporary network isolation. This mirrors the reliability-scoring approach used in comparable proposed frameworks for other high-stakes AI-security domains, adapted here to the specific taxonomy of platform, network, malware, companion-application, and adversarial threats developed in Section III, and it allows the modular architecture to be extended as new device categories, malware families, or evasion techniques emerge without requiring the entire framework to be redesigned.

Evaluating this outline empirically, across the resource, temporal, adversarial, and explainability dimensions identified in Section V, and across a dataset that jointly captures application, network, and companion-app signals for the same underlying devices, is the logical next stage of this research program and falls outside the systematic-investigation scope of the present paper.

VII. Conclusion

Android's expansion from the smartphone into the Internet of Things has created a device class that inherits the mature, well-documented malware ecosystem of Android alongside the resource constraints, heterogeneous connectivity, and physical-world consequences of embedded IoT hardware. This paper undertook a systematic investigation of the security challenges and malware threats affecting this device class, synthesizing recent literature spanning static, dynamic, network-traffic, graph-based, federated, and explainable deep learning approaches to Android and IoT malware detection. The investigation found that individual studies have made genuine progress on narrowly scoped problems, from event-aware detectors that resist concept drift to graph neural networks that capture relational code structure to federated schemes that preserve device-level privacy, but that comparatively little work addresses these problems jointly, and that current industry telemetry shows the underlying threat continuing to grow quickly across both the mobile and IoT dimensions of the problem.

Building on this synthesis, the paper organized the threat landscape into a five-part taxonomy spanning platform and firmware vulnerabilities, network-layer and communication threats, malware families that specifically target constrained endpoints, companion-application and data-privacy risks, and adversarial threats directed at the AI-based defenses themselves. It further reviewed the datasets that underpin experimental work in this domain and identified two structural gaps: the absence of a benchmark that jointly captures application, network, and companion-app signals for the same devices, and the risk that reliance on aging datasets systematically inflates reported detection accuracy. Informed by these findings, the paper outlined the design requirements that a deep learning-oriented detection and security-enhancement framework for Android-based IoT devices should satisfy, namely multi-source feature fusion, a resource-aware tiered inference architecture, privacy-preserving federated collaborative learning, adversarial robustness treated as a standing pipeline requirement, explainability built into the model rather than applied after the fact, and fleet-level risk scoring to support real-world deployment decisions.

Several directions follow directly from this investigation. Future empirical work should construct and share benchmark corpora that jointly label application, network, and companion-app behavior for the same underlying devices; evaluate proposed detectors explicitly along accuracy-versus-resource-cost and accuracy-versus-time frontiers rather than on accuracy alone; and report adversarial security curves as a standard component of model evaluation rather than as an optional addendum. More broadly, as Android-based IoT devices continue to expand into healthcare, industrial, and critical-infrastructure settings where the cost of an undetected compromise is measured in physical rather than purely informational harm, closing the gap between narrowly scoped laboratory detectors and the joint, resource-aware, privacy-preserving, and explainable systems that real deployments require should be treated as a central objective for the field rather than a secondary refinement of existing techniques.

References

1. W. Almobaideen, O. Abu Alghanam, and M. Abdullah, et al., "Comprehensive review on machine learning and deep learning techniques for malware detection in android and IoT devices," *International Journal of Information Security*, vol. 24, art. 110, 2025.
2. Zscaler ThreatLabz, "2025 Mobile, IoT, and OT Threat Report," Zscaler, Inc., San Jose, CA, Nov. 2025.
3. Google Security Blog, "Keeping Google Play & Android App Ecosystems Safe in 2025," Google LLC, Feb. 2026.
4. M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the Mirai Botnet," in *Proc. 26th USENIX Security Symposium*, 2017, pp. 1093–1110.
5. Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-BaIoT: Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.
6. R. Darwish, M. Abdelsalam, and S. Khorsandroo, "Deep Learning Based XIoT Malware Analysis: A

- Comprehensive Survey, Taxonomy, and Research Challenges,” *Journal of Network and Computer Applications*, 2025.
7. A. Guerra-Manzanares, “Machine Learning for Android Malware Detection: Mission Accomplished? A Comprehensive Review of Open Challenges and Future Perspectives,” *Computers & Security*, vol. 138, art. 103654, 2024.
 8. J. Qiu, J. Zhang, W. Luo, L. Pan, S. Nepal, and Y. Xiang, “A Survey of Android Malware Detection with Deep Neural Models,” *ACM Computing Surveys*, vol. 53, no. 6, pp. 1–36, 2021.
 9. T. Lei, Z. Qin, Z. Wang, Q. Li, and D. Ye, “EveDroid: Event-Aware Android Malware Detection Against Model Degrading for IoT Devices,” *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6668–6680, 2019.
 10. F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, “TESSERACT: Eliminating Experimental Bias in Malware Classification across Space and Time,” in *Proc. 28th USENIX Security Symposium*, 2019, pp. 729–746.
 11. Y. Liu, C. Tantithamthavorn, L. Li, and Y. Liu, “Deep Learning for Android Malware Defenses: A Systematic Literature Review,” *ACM Computing Surveys*, vol. 55, no. 8, pp. 1–36, 2022.
 12. A. Bensaoud, J. Kalita, and M. Bensaoud, “A Survey of Malware Detection Using Deep Learning,” *Machine Learning with Applications*, vol. 16, art. 100546, 2024.
 13. M. Gopinath and S. C. Sethuraman, “A Comprehensive Survey on Deep Learning Based Malware Detection Techniques,” *Computer Science Review*, vol. 47, art. 100529, 2023.
 14. T. Bilot, N. El Madhoun, K. Al Agha, and A. Zouaoui, “A Survey on Malware Detection with Graph Representation Learning,” *ACM Computing Surveys*, vol. 56, no. 11, pp. 1–36, 2024.
 15. A. Razgallah, R. Khoury, S. Hallé, and K. Khanmohammadi, “A Survey of Malware Detection in Android Apps: Recommendations and Perspectives for Future Research,” *Computer Science Review*, vol. 39, art. 100358, 2021.
 16. E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, “MalDozer: Automatic Framework for Android Malware Detection Using Deep Learning,” *Digital Investigation*, vol. 24, pp. S48–S59, 2018.
 17. J. Jeon, J. H. Park, and Y.-S. Jeong, “Dynamic Analysis for IoT Malware Detection with Convolution Neural Network Model,” *IEEE Access*, vol. 8, pp. 96899–96911, 2020.
 18. K. Dhanya, P. Vinod, Y. Y. Suleiman, T. Abhiram, K. S. Ashil, and T. Gireesh Kumar, “Obfuscated Malware Detection in IoT Android Applications Using Markov Images and CNN,” *IEEE Systems Journal*, vol. 17, no. 2, pp. 2756–2766, 2023.
 19. M. Al-Fawa’reh, J. Abu-Khalaf, P. Szewczyk, and J. J. Kang, “MalBot-DRL: Malware Botnet Detection Using Deep Reinforcement Learning in IoT Networks,” *IEEE Internet of Things Journal*, vol. 11, no. 6, pp. 9610–9629, 2024.
 20. S. Ali, O. Abusabha, F. Ali, M. Imran, and T. Abuhmed, “Effective Multitask Deep Learning for IoT Malware Detection and Identification Using Behavioral Traffic Analysis,” *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 1199–1209, 2023.
 21. J. Feng et al., “A Two-Layer Deep Learning Method for Android Malware Detection Using Network Traffic,” *IEEE Access*, vol. 8, pp. 125786–125796, 2020.
 22. S. Zhang, H. Su, H. Liu, and W. Yang, “MPDroid: A Multimodal Pre-Training Android Malware Detection Method with Static and Dynamic Features,” *Computers & Security*, vol. 150, art. 104262, 2025.
 23. V. Rey, P. M. S. Sánchez, A. H. Celdrán, and G. Bovet, “Federated Learning for Malware Detection in IoT Devices,” *Computer Networks*, vol. 204, art. 108693, 2022.
 24. A. Alamer, “A Privacy-Preserving Federated Learning with a Secure Collaborative for Malware Detection Models Using Internet of Things Resources,” *Internet of Things*, vol. 25, art. 101015, 2024.
 25. M. Amjath, S. Henna, and U. Rathnayake, “Graph Representation Federated Learning for Malware Detection in Internet of Health Things,” *Results in Engineering*, vol. 25, art. 103651, 2025.
 26. C. Palma, A. Ferreira, and M. Figueiredo, “Explainable Machine Learning for Malware Detection on Android Applications,” *Information*, vol. 15, no. 1, art. 25, 2024.
 27. H. Manthana et al., “Explainable Artificial Intelligence (XAI) for Malware Analysis: A Survey of Techniques, Applications, and Open Challenges,” *IEEE Access*, 2025.
 28. D. Bhusal and N. Rastogi, “Adversarial Patterns: Building Robust Android Malware Classifiers,” *ACM Computing Surveys*, vol. 57, no. 8, pp. 1–34, 2025.
 29. H. Bostani, Z. Zhao, and V. Moonsamy, “Improving Adversarial Robustness in Android Malware Detection by Reducing the Impact of Spurious Correlations,” in *Computer Security. ESORICS 2024 International Workshops, Lecture Notes in Computer Science*, Springer, 2025.
 30. K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, “Adversarial Examples for Malware Detection,” in *Proc. European Symposium on Research in Computer Security (ESORICS)*, 2017, pp. 62–79.
 31. D. Zhang et al., “Android Malware Detection Based on Hypergraph Neural Networks,” *Applied Sciences*, vol. 13, art. 12629, 2023.
 32. A. R. Zaidi et al., “Deep Learning-Based Detection of Android Malware Using Graph Convolutional Networks (GCN),” *Statistics, Computing and Interdisciplinary Research*, vol. 6, pp. 57–73, 2024.

33. H. Gao, S. Cheng, and W. Zhang, "GDroid: Android Malware Detection and Classification with Graph Convolutional Network," *Computers & Security*, vol. 106, art. 102264, 2021.
34. C. Li, Z. Cheng, H. Zhu, L. Wang, Q. Lv, Y. Wang, N. Li, and D. Sun, "DMalNet: Dynamic Malware Analysis Based on API Feature Engineering and Graph Learning," *Computers & Security*, vol. 122, art. 102872, 2022.
35. V. Ravi and R. Chaganti, "EfficientNet Deep Learning Meta-Classifer Approach for Image-Based Android Malware Detection," *Multimedia Tools and Applications*, vol. 82, no. 16, pp. 24891–24917, 2023.
36. A. Mahindru et al., "PermDroid: A Framework Developed Using Proposed Feature Selection Approach and Machine Learning Techniques for Android Malware Detection," *Scientific Reports*, vol. 14, no. 1, art. 10724, 2024.
37. S. Bashir et al., "Hybrid Machine Learning Model for Malware Analysis in Android Apps," *Pervasive and Mobile Computing*, vol. 97, art. 101859, 2024.
38. H. H. R. Manzil, "DYNAMALDROID: Dynamic Analysis-Based Detection Framework for Android Malware Using Machine Learning Techniques," in *Proc. Int. Conf. on Knowledge Engineering and Communication Systems (ICKES)*, IEEE, 2022.
39. D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket," in *Proc. Network and Distributed System Security Symposium (NDSS)*, vol. 14, 2014, pp. 23–26.
40. Canadian Institute for Cybersecurity, University of New Brunswick, "CICAndMal2017: Android Malware Dataset," [Online]. Available: <https://www.unb.ca/cic/datasets/andmal2017.html>, 2018.
41. Y. Z. Wei, M. Md-Arshad, A. A. Samad, and N. Ithnin, "Comparing Malware Attack Detection Using Machine Learning Techniques in IoT Network Traffic," *International Journal of Innovative Computing*, vol. 13, no. 1, pp. 21–27, 2023.
42. N. Nimalasingam, J. Senanayake, and C. Rajapakse, "Detection of IoT Malware Based on Forensic Analysis of Network Traffic Features," in *Proc. Int. Research Conf. on Smart Computing and Systems Engineering (SCSE)*, vol. 5, IEEE, 2022, pp. 122–130.
43. X. Meng and D. Li, "HerTDroid: Android Malware Detection Method with Influential Node Filter and Heterogeneous Graph Transformer," *Applied Sciences*, vol. 14, art. 3150, 2024.
44. M. Magdy, I. Abu-Krishna, and H. Aslan, "Machine Learning and Deep Learning Approaches for Mobile Malware Detection: A Survey," in *Model and Data Engineering (MEDI 2025)*, *Lecture Notes in Computer Science*, vol. 16427, Springer, 2026.
45. ORDR, Inc., "IoT Security Statistics 2026: 21B Devices, 820K Daily Attacks," 2026.
46. NETGEAR, Inc. and Bitdefender, "The 2024 IoT Security Landscape Report," 2025.
47. R. Prakash, J. Neeli, and S. Manjunatha, "A Survey of Security Challenges, Attacks in IoT," *E3S Web of Conferences*, vol. 491, art. 04018, 2024.
48. A. Allen, A. Mylonas, S. Vidalis, and D. Gritzalis, "Security Evaluation of Companion Android Applications in IoT: The Case of Smart Security Devices," *Sensors*, vol. 24, no. 17, art. 5465, 2024.