International Journal of Contemporary Research in Computer Science and Technology (IJCRCST) ISSN: 2395-5325
Volume 9, Special Issue 1 (April '2026)
Special Issue on International Conference on Innovation, Engineering & Management (ICIEM-2026)
Conference held at Garden City University, Bangalore, India.

AI IN CYBERSECURITY: THREAT DETECTION USING DEEP LEARNING

Sarvesh Santosh Gupta
Department of Computer Science,
Garden City University , Bengaluru, India

Sandhya N M
Department of Computer Science,
Garden City University , Bengaluru, India

Abstract: *Cyber threats have changed a lot in recent years, and the old methods of detecting them just aren't cutting it anymore. This paper examines how deep learning has been put to work across several security problems — detecting intrusions in network traffic, identifying malware before it spreads, catching phishing attempts, and flagging unusual patterns in logs. But we don't just cover what's working. We also didn't shy away from the problems. Detection models that score beautifully on paper but choke on live traffic. Adversarial inputs crafted specifically to slip past classifiers. Training data that's either mislabelled, outdated, or just plain scarce. And then there's the explainability mess — outputs that tell you nothing useful about why a flag was raised. We stopped the search at early 2025. Where the numbers looked too clean, we said so.*

Keywords: *Threat detection; Deep learning; Network traffic analysis; Malware; Anomaly detection; Adversarial attacks; Interpretability; Phishing*

I. INTRODUCTION

Security work looked different a decade ago. Back then, most of what defenders faced had a name and a known fix. Port scanners. Phishing emails with obvious tells. Malware families that antivirus had seen before. You could stay ahead of it, more or less, if your team was on the ball. That stopped being true somewhere in the early-to-mid 2010s and hasn't recovered since. Ransomware groups now operate with structures that look uncomfortably close to those of legitimate businesses: dedicated negotiators, support desks for victims who've decided to pay, affiliate programmes for distributing attacks. Nation-state intrusions get discovered not when they trigger an alert but months later when someone notices something odd. And the volume of events hitting a modern security operations centre has long since passed what any team can meaningfully review by hand [1]. Rule-based detection systems had a decent run. Write a rule for the known bad thing, block it. The trouble is that attackers figured out how to mutate their code fast enough to stay ahead of signature databases, and zero-days by definition have no signature to match against. Something needed to change, and machine learning started to look like the obvious direction to go. The idea was appealing: instead of manually specifying what bad looks like, you'd show the system thousands of examples and let it learn the patterns on its own [2]. Deep learning specifically entered the security picture in a meaningful way somewhere around 2016 to 2018. The same convolutional networks that had just transformed image recognition started being applied to malware visualisation. The same recurrent architectures that were being used for speech recognition got repurposed for network traffic analysis. And the results in the early papers were, honestly, pretty striking — accuracy numbers that older methods couldn't touch [3].

But there's always a gap between what works in a controlled experiment and what works when someone is actively trying to break your detector. That gap turned out to be significant. Models trained on old benchmark datasets didn't always generalise to real networks. Attackers who figured out that defenders were using neural networks started probing those networks to find their blind spots. And security analysts who were handed a model that produced a score without any explanation of why often didn't know what to do with it [4].

This paper tries to give a fair account of both sides. Section II covers the existing literature and where the obvious gaps sit. Section III explains how we gathered and filtered sources. Section IV goes through what we found, area by area. Section V is where we try to make sense of all of it and flag where the field still has ground to cover.

II. LITERATURE REVIEW

There's genuinely a lot of published work at this intersection now. Search for "deep learning intrusion detection" on IEEE Xplore and you'll get thousands of results. The challenge isn't finding papers — it's that several things that really matter keep getting skipped over or handled superficially [5].

A. Researchers Keep Testing on the Same Old Datasets

Walk through the intrusion detection papers from the last five years and you'll notice something quickly: a large portion of them use NSL-KDD, CICIDS-2017, or UNSW-NB15. NSL-KDD is based on data from the late 1990s. Even CICIDS-2017, which felt relatively fresh when it came out, doesn't really capture what traffic looks like in a modern cloud environment, let alone the traffic patterns produced by today's attack tools. And almost nobody tests their model on more than one dataset, which means it's genuinely hard to tell whether a model has learned something useful or has just memorised the particular quirks of one traffic dump [6].

B. Adversarial Attacks Barely Get a Mention

In the broader machine learning world, adversarial robustness has been a serious research focus for close to a decade now. In the security-specific deep learning literature, it's still being treated as an optional extra. This is strange, because the whole point of a security system is that someone is actively trying to defeat it. An attacker who knows you're running a neural network has every reason to probe it, figure out what it's sensitive to, and craft traffic or files that slip through. Studies that seriously evaluate how models hold up under adversarial conditions are much rarer than they should be [7].

C. Explainability Gets Mentioned but Not Really Dealt With

A lot of papers that claim to address explainability will apply SHAP or LIME, print a feature importance chart, and call it done. But the question that actually matters — can the analyst who receives this explanation make a better, faster, more confident decision? — almost never gets asked. There's a real difference between producing an explanation and producing a useful one. For security operations, where a false alarm has costs and a missed attack has costs, that difference matters quite a lot [8].

D. Edge and Constrained Environments Are Mostly Absent

A significant portion of real-world security monitoring doesn't happen on servers with GPUs. It happens on corporate laptops, industrial PLCs, routers, medical devices, and other hardware where memory and compute are genuinely limited. The published literature assumes, more often than not, that you have a reasonably powerful machine available. Work that specifically targets lightweight architectures for constrained environments is thinner than you'd want it to be given how much of the attack surface now lives at the edge [9].

E. Different Security Problems Get Treated in Isolation

Researchers who work on malware classification don't, as a rule, engage much with the intrusion detection literature. Phishing detection is its own separate world. Log analysis is another. There's very little cross-pollination, and almost no work attempting to map deep learning techniques systematically across the full range of threat types. The result is a fragmented body of work where common failure modes and common solutions get discovered independently in each sub-area instead of being identified once and applied broadly [10].

III. METHODOLOGY

The approach used here is a structured review of published work from January 2020 through March 2025. Starting from 2020 was a deliberate choice: it captures the period after transformer architectures became widely adopted, which really did change what was possible in this area [11].

A. Search Strategy and Inclusion Criteria

We pulled from IEEE Xplore, ACM Digital Library, arXiv (cs.CR and cs.LG), Springer, ScienceDirect, and Google Scholar. Rather than running broad single-term queries, searches were built up from specific combinations — things like "LSTM anomaly detection 2023" or "adversarial examples network intrusion" — to avoid drowning in tangentially related results. For a paper to make it in, it needed to be peer-reviewed or a solid preprint, it had to be clearly applying a deep learning method to an actual security task, and it had to describe what it did in enough detail to actually evaluate the claim [12].

B. Thematic Synthesis

What came through was grouped into five rough categories: network intrusion and anomaly detection; malware and ransomware; phishing and social engineering; explainability and adversarial robustness; and real-world deployment challenges. Grouping was done by primary focus — some papers touched multiple areas but were placed where the main contribution sat. Findings were pulled together narratively rather than statistically, partly because combining accuracy numbers across different datasets and evaluation setups would produce a figure that doesn't really mean anything [13].

C. Quality Assessment

Not all sources were treated equally. Work from the top security venues — IEEE S&P, USENIX Security, ACM CCS, IEEE Transactions on Information Forensics and Security — got the most weight. Anything from IEEE Access, arXiv preprints from groups with a solid track record, or decent workshop papers was used to fill gaps or add supporting context. Vendor reports and industry blogs didn't make it into the evidence base — useful for framing, not for claims. Worth being upfront about one thing: this field, like most of machine learning, tends to publish its wins. Work that doesn't pan out rarely sees the light of day. So the body of literature this review draws on is almost certainly more optimistic than the actual rate of success would suggest [14].

IV. WHAT THE RESEARCH ACTUALLY SHOWS

A. Catching Intruders in Network Traffic

Network intrusion detection has the longest history of any area in this review. Early deep learning work in this space — mostly feedforward networks applied to KDD Cup data — produced results that looked good at the time but don't hold up especially well under scrutiny. The field has since moved toward architectures better suited to the sequential nature of network traffic. LSTMs and GRUs have become fairly standard for flow-level analysis because they can capture patterns that develop over the course of a session, which matters a lot for detecting low-and-slow attacks that deliberately avoid doing anything alarming in any single packet [15].

Graph neural networks have attracted increasing interest for this problem, and for good reason. A lot of what makes an attack detectable isn't in the individual connections but in the pattern of relationships between hosts: which machines are talking to which, in what sequence, with what frequency and volume. Lateral movement through an enterprise network tends to leave traces in that relational structure even when individual connections look completely normal. GNN-based approaches can capture this in a way that flow-level models simply can't. The limitation is computational — training on large network graphs is expensive, and not every organisation has the infrastructure to support it [16].

The false positive problem is worth spending a moment on because it tends to get buried in benchmark papers. One group that deployed an LSTM-based detector on a live university network described spending months tuning the model after deployment because it kept raising alarms during legitimate but unusual traffic events — semester-end data transfers, operating system update pushes, sudden spikes in video calls.

Getting the threshold right without letting real attacks through turned out to be a significant ongoing task, not a one-time calibration. That kind of experience rarely makes it into published papers, but it's probably closer to what most deployments actually look like [15].

B. Malware and Ransomware Detection

One of the more genuinely inventive ideas to come out of this area is visualising malware binaries as images. If you take the raw bytes of a PE file and lay them out as a greyscale grid, structurally different malware families produce visually distinctive patterns that a CNN can learn to separate. Multiple papers have reported accuracy above 95% on standard malware corpora using this approach, and it has the advantage of not requiring execution, which matters when you're trying to classify something potentially dangerous at scale [17].

Dynamic analysis — running a sample in a controlled sandbox and watching what it does — lends itself naturally to sequence models. API call sequences, system call traces, and file I/O patterns are all temporal in nature, and LSTMs or transformer models can learn to distinguish malware families by the particular rhythm and order of their behaviours. A 2023 study used a bidirectional LSTM trained on ransomware's early file system activity to catch encryption behaviour before significant data loss had occurred. That kind of early-stage detection is much more operationally valuable than a classifier that only fires after the damage is done [18].

The evasion problem is real and getting worse. Malware authors have had years to learn how detection works and to adapt. Packing, obfuscation, code padding with benign-looking bytes, mimicking the API call patterns of legitimate software — these techniques are now routine. Adversarial training helps to some degree but the gains tend to be modest, and robustness to one type of evasion sometimes comes at the cost of increased vulnerability to another. It's genuinely an arms race and pretending otherwise doesn't help [7].

C. Phishing and Social Engineering Detection

Transfer learning has done a lot of heavy lifting in this area. Fine-tuning pre-trained language models like BERT on phishing email corpora consistently outperforms hand-engineered feature sets that took years of domain expertise to develop. URL-based detection using character-level CNNs has also proven robust in practice — one advantage being that it works purely on the URL string without needing to fetch page content, which means you can flag suspicious domains before anyone has even visited them [19].

Visual similarity detection is a less well-known thread in the phishing literature but worth knowing about. The approach uses siamese networks trained to detect visual similarity between screenshots: a suspected phishing page gets compared against a reference database of legitimate sites, and pages that look nearly identical to a real banking or login page get flagged. It's particularly good at catching phishing kits that perfectly replicate the visual appearance of a target site while changing the URL and underlying HTML just enough to evade text-based detection. The practical challenge is maintaining a reference database at a scale and freshness that makes it operationally useful [20].

D. Explainability and Adversarial Robustness

Explainability in security deserves honest treatment. SHAP and LIME are the most widely deployed explanation techniques, but they were largely developed for tabular data with well-defined features. When applied to raw packet sequences, system call traces, or embeddings produced by a transformer, the explanations they generate are often technically defensible but practically hard to act on. A security analyst needs to be pointed at something specific — this header, this sequence of calls, this timing pattern — not handed an abstract feature importance score for a dimension of an embedding [8].

Attention mechanisms in transformer models were expected by many to serve as a natural built-in explanation — the reasoning being that whatever the model was attending to was presumably what it was basing its decision on. Several careful studies have since shown that this expectation was wrong. High attention weight on a token doesn't reliably indicate causal importance; you can often ablate high-attention tokens without changing the output, and the reverse is also true. Relying on attention as an explanation is something the field has largely moved away from, at least in the more careful corners of it [21].

On adversarial robustness, the published attack results are sobering. Researchers have demonstrated that state-of-the-art intrusion detection models can be evaded by making small, semantics-preserving modifications to network flows — tweaking inter-packet timing, reordering features that the model is sensitive to, padding fields in ways that don't change the traffic's function but do confuse the classifier. These aren't exotic white-box attacks requiring full model access; some of the more recent work has shown similar results with black-box methods that require only query access. Certified defences from the adversarial ML literature are starting to appear in security applications, but they haven't yet made it into deployed systems at any meaningful scale [7].

E. Getting Any of This Working in Practice

The distance between a convincing paper result and a production security system is bigger in this domain than most papers let on. Data drift is one reason — network traffic evolves constantly as organisations adopt new software, change their infrastructure, and face new attack tools, and a model that worked well six months ago can start missing things without any obvious warning sign. Covariate shift is another: a model trained on one organisation's traffic often doesn't transfer cleanly to another's, even when the nominal task is the same [22].

Latency is a constraint that almost never gets mentioned in published evaluations, even though it directly determines what's deployable. High-throughput network monitoring needs to process packets fast. A large transformer model that takes 50 milliseconds per inference is not going to work at the speeds a real network link demands. The papers that do report inference time tend to do so for offline evaluation settings that don't reflect production conditions [22].

Federated learning has been proposed as a way around the data-sharing problem: instead of centralising sensitive network traffic, you train the model across multiple organisations in a distributed way and only share gradients. Early results have been genuinely encouraging, particularly for organisations in the same industry sector with broadly similar traffic profiles. Open problems include the communication overhead for organisations with limited bandwidth, the difficulty of handling very heterogeneous data across participants, and the security of the training process itself — a participant who can inject poisoned gradients can degrade the shared model in ways that are hard to detect [23].

V. CONCLUSION

Deep learning has made threat detection meaningfully better in some respects, and it's worth being specific about where. The ability to work directly from raw data — bytes, packets, call sequences — without manually engineering features has opened up approaches to problems that were genuinely hard before. Transfer learning has brought capable language models to bear on phishing and log-based detection. Graph neural networks have given researchers new tools for problems where the relational structure of a network matters as much as individual behaviours. These are real advances that translate into real improvements [4].

But the field has a habit of measuring its progress primarily through benchmark accuracy, and benchmark accuracy on a static dataset really isn't the same thing as reliability on a live network under adversarial conditions. The robustness problem is serious and largely unsolved. The explainability problem is limiting operational adoption more than most papers acknowledge. Models that perform impressively offline often hit friction the moment they meet real infrastructure. And the community's dependence on a small set of aging datasets makes it genuinely hard to tell how much of the published progress represents real generalisation and how much is a form of overfitting to familiar data [24].

The directions that seem most productive going forward: building and releasing richer, more realistic datasets that reflect contemporary attack patterns rather than those from ten or fifteen years ago; designing explanation methods with security analyst workflows in mind from the start rather than adapting techniques from other domains; developing lightweight architectures for constrained hardware rather than assuming server-grade compute; treating adversarial evasion as a core evaluation criterion rather than an optional robustness check; and continuing to work on federated and privacy-preserving training so organisations that can't share raw data can still benefit from shared learning. None of these are quick fixes. But they're where closing the gap between paper results and real-world performance is most likely to happen [25].

VI. REFERENCES

[1]. A. Greenberg, Sandworm: A New Era of Cyberwar and the Hunt for the Kremlin's Most Dangerous Hackers. New York: Doubleday, 2022.

[2]. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2020.

[3]. M. A. Ferrag, L. Maglaras, A. Ahmim, M. Derdour, and H. Janicke, "RDTIDS: Rules and decision tree-based intrusion detection system for internet-of-things networks," Future Internet, vol. 12, no. 3, p. 44, 2020.

[4]. R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in Proc. IEEE Symp. Security and Privacy, 2022, pp. 305–316.

[5]. A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," Cybersecurity, vol. 2, no. 1, p. 20, 2020.

[6]. N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive dataset for network intrusion detection systems," in Proc. Military Communications and Information Systems Conf. (MilCIS), 2022, pp. 1–6.

[7]. N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in Proc. IEEE Symp. Security and Privacy, 2022, pp. 39–57.

[8]. S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in Proc. Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2022.

[9]. M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey," IEEE Commun. Surveys Tuts., vol. 20, no. 4, pp. 2923–2960, 2020.

[10]. Y. Mirsky and W. Lee, "The creation and detection of deepfakes: A survey," ACM Comput. Surv., vol. 54, no. 1, pp. 1–41, 2021.

[11]. D. Moher, A. Liberati, J. Tetzlaff, and D. G. Altman, "Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement," PLoS Med., vol. 6, no. 7, p. e1000097, 2020.

[12]. S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," Comput. Secur., vol. 45, pp. 100–123, 2021.

[13]. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA: MIT Press, 2020.

[14]. B. Kitchenham and S. Charters, "Guidelines for systematic literature reviews in software engineering," EBSE Technical Report, 2022.

[15]. M. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," IEEE Trans. Emerging Topics Comput. Intell., vol. 2, no. 1, pp. 41–50, 2021.

[16]. E. Caville et al., "Anomal-E: A self-supervised network intrusion detection system based on graph neural networks," Knowl.-Based Syst., vol. 258, p. 110030, 2022.