iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.81400

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

AegisScan: A Hybrid AI-Driven Multi-Format Malware Detection Platform with Conversational Threat Reasoning

Perada Vamshi[1], Ms. Mattukoyya Amulya[2], Gunta Koushik[3], D Sreenivas[4], Shaik Abdul Kalam[5]
Department of Computer Science and Engineering, Acharya Nagarjuna University College of Engineering and Technology, Andhra Pradesh, India

Abstract: ***Malicious code reaches end-users today through a bewildering variety of carriers — Windows binaries, productivity documents, raster images, audio clips, video containers and shortened web links — and the conventional fingerprint-matching antivirus engines that still dominate the campus desktop have, in our practical experience, become the weakest link in the defensive chain. In this work we (the authors, a four-member final-year project team) describe AegisScan, a hybrid artificial-intelligence platform that we designed and engineered from scratch to close that gap. Our system fuses lightweight static inspection, contained behavioural execution and a stacked learning ensemble inside a single browser-accessible workflow. We built the front-end in React.js, orchestrated requests through a Flask service, persisted scan history in a normalised PostgreSQL store, and routed any potentially harmful artefact to a hardened Docker sandbox for run-time tracing. A Random-Forest combined with a one-dimensional Convolutional Neural Network produces the verdict, and a calibrated logistic meta-learner supplies the final probability together with the five most influential features. We further embedded a guard-railed conversational module that paraphrases the technical report into plain language so that students, clerical staff and faculty all receive guidance they can act upon. On a balanced corpus of 1,250 mixed samples our prototype achieved 94.2 % accuracy, 93.2 % macro-F1 and an 84 % catch-rate on previously unseen variants — substantially above the signature-only and machine-learning-only baselines we benchmarked against. We argue that AegisScan therefore narrows the long-standing distance between single-format laboratory accuracy and deployable, multi-format, explainable cybersecurity tooling.***

Index Terms: ***Malware detection, hybrid analysis, deep learning, multi-format intake, sandboxing, explainable AI, conversational agent, React, Flask, PostgreSQL.***

I. INTRODUCTION

Cyber-incidents have evolved into one of the costliest categories of contemporary crime. Malicious binaries, weaponised office documents, steganographic images, tampered media and shortened links circulated across social platforms now spread at industrial speed: independent telemetry from leading security vendors records several hundred thousand fresh samples being captured every single day [1], and the financial damage attributable to ransomware alone is now measured in trillions of dollars per annum [6]. The recent rise of generative artificial intelligence has further compressed the attacker's development cycle and enabled on-the-fly polymorphism that defeats static fingerprints almost as soon as they are published [4].

Conventional antivirus engines lean almost entirely on signature databases — pre-computed fingerprints of files already catalogued as harmful. While this paradigm copes adequately with re-discovered strains, we observed during preliminary testing that it collapses against packed binaries, file-less intrusions, freshly minted polymorphic variants and AI-assisted droppers [3], [4]. The recent literature converges on three persistent weaknesses, each of which directly motivated our design choices. First, coverage is heavily skewed toward Windows Portable-Executable binaries and Android packages, leaving documents, multimedia and URLs largely unmonitored [1], [3]. Second, models that excel on curated corpora often degrade once they meet live traffic — the well-known laboratory-to-field gap [3], [6]. Third, when a file is finally flagged, the verdict is delivered as an opaque binary label, which erodes trust and slows incident response [2], [4].

We responded to those three weaknesses directly. The platform we constructed ingests seven input families — Portable-Executable binaries, PDF documents, Office documents, raster images, audio clips, video files and uniform resource locators — passes each through a static feature extractor, optionally executes the artefact inside a network-isolated Docker sandbox, and finally classifies it with a stacked machine-learning ensemble.

©IJRASET: All Rights are Reserved

We integrated a conversational explainer that renders the JSON threat report in everyday language, an aspect we found missing from almost every academic prototype surveyed in [2] and [5]. The complete stack is delivered as a modern single-page web application, so any campus user can submit content from a browser without local installation, administrative privileges or driver updates.

A. *Motivation*

During our preliminary investigation we noticed that the antivirus product currently deployed on the institutional network blocks only files whose hash matches its on-device signature set, which leaves novel and obfuscated samples undetected. We further observed that, even when a file was caught, the user-facing report was packed with technical jargon that nobody outside the IT department could meaningfully act upon. These two field-level observations motivated us to design a system that is (a) format-agnostic, so that documents and media receive the same scrutiny as binaries, (b) AI-augmented, so that previously unseen variants are still recognised, and (c) explainable, so that the verdict translates into a sensible next step for ordinary users.

B. *Objectives*

1) To engineer a single web platform that ingests at least seven prevalent file and link formats.
2) To combine static inspection, dynamic behavioural tracing and machine learning so that no individual technique constitutes a single point of failure.
3) To reduce zero-day misses by relying on learned and behavioural cues rather than fixed signatures.
4) To translate every verdict into conversational language through an integrated chatbot.
5) To persist scan history in a relational store so that analytics and longitudinal auditing are both feasible.
6) To quantify the gain over signature-only and ML-only baselines on a balanced multi-format corpus.

C. *Contributions*

Our contributions are five-fold. First, we unified seven input families inside a single ingestion pipeline running on React, Flask and PostgreSQL — a level of breadth seldom attempted by published academic prototypes [1], [3], [6]. Second, we implemented a static-plus-dynamic-plus-machine-learning detection core that addresses the gap noted in [3] and [6] between paper-level accuracy and field robustness. Third, we engineered a conversational explainer that converts the JSON threat report into follow-up dialogue, directly answering the explanation gap raised in [2] and [4]. Fourth, we present a transparent comparison against signature-based and ML-only baselines on a balanced 1,250-sample set, with per-format accuracy reported individually. Fifth, we offer a reproducible architecture suitable for future extension to mobile and Internet-of-Things workloads, in line with the open challenges listed by [4] and [6].

The remainder of the paper is organised as follows. Section II reviews six recent peer-reviewed studies upon which our design rests. Section III crystallises the problem statement. Section IV introduces the proposed system. Section V details the methodology and the underpinning equations. Section VI presents the experimental results across five complementary figures. Section VII discusses the findings, situates them with respect to related lines of work and lists the limitations we identified. Section VIII concludes and outlines our future-work roadmap.

II. LITERATURE REVIEW

Six recent peer-reviewed contributions form the foundation of our review. We selected them because, taken together, they span the breadth of contemporary malware-detection research — deep-learning surveys, machine-learning systematic reviews, practical algorithm catalogues and chapters on the role of generative artificial intelligence in modern cybersecurity. Each is summarised below and then mapped, in Table I, against the AegisScan design.

A. *Deep-learning surveys*

Song and colleagues [1] performed a wide-angle survey of seventy-two deep-learning works published between 2011 and 2025. They classified contributions along two axes — neural architecture (Convolutional, Recurrent and Generative-Adversarial Networks) and feature representation (raw byte streams, image visualisations of binaries, and data-augmented variants). Their headline observation, that the field still suffers from inconsistent datasets and that cross-family generalisation remains brittle, directly motivated our decision to construct a multi-format benchmark rather than restrict ourselves to a single binary family. Song et al. also stress that pipelines which fuse static and dynamic features tend to outperform purely structural or purely behavioural baselines, a recommendation we operationalised in our fusion layer.

Redhu and co-authors [2] applied the PRISMA 2020 protocol to deep-learning-powered detection between 2015 and 2023, examining Recurrent Networks, Long Short-Term Memory cells, Deep Belief Networks, Deep Convolutional Networks, Generative Models, Boltzmann Machines, Reinforcement-Learning agents and Extreme Learning Machines. We took particular note of their conclusion that deep models excel at automatic feature discovery but are seldom paired with a user-facing explanation layer, which limits real-world deployment. We address that very gap by appending a conversational module on top of our ensemble.

B. Machine-learning systematic reviews

Pathan and colleagues [3] surveyed machine-learning contributions up to 2026 and stress that classical ensembles, Convolutional Networks, Long Short-Term Memory models and Graph Neural Networks all reach very high laboratory accuracy yet rarely translate into deployable systems. They cite Explainable Artificial Intelligence and Federated Learning as the two most pressing future directions. We adopted the first by exposing per-feature contributions in our threat report and offering plain-language follow-up through the chatbot; federated training is deferred to future work but our modular architecture deliberately leaves the door open.

Berrios and colleagues [6] performed a PRISMA review of forty-seven studies published between 2020 and 2024 and concluded that hybrid models which fuse static structural features with dynamic behavioural traces consistently outperform single-paradigm methods. This finding became the central design hypothesis of AegisScan and is independently confirmed by the experimental evidence we report in Section VI.

C. Algorithm catalogues and applied chapters

Saadoon and Faisal [5] catalogue the machine-learning classifiers most frequently applied to Portable-Executable binaries — Support Vector Machines, Random Forests, k-Nearest Neighbours, Naive Bayes and Decision Trees — and underline the inherent difficulty of detecting zero-day polymorphic samples through behaviour alone. Their measured comparison guided our choice of the Random Forest as the interpretable component of the final ensemble, since it offers feature-importance scores out-of-the-box and is robust against the imbalanced class distributions we observed in our own data.

Jones [4] explains how Convolutional and Recurrent Networks, together with Generative Artificial Intelligence and Large Language Models, extend malware coverage to Internet-of-Things devices, cloud computing environments and increasingly automated attack chains. The chapter warns about three long-standing problems — data imbalance, overfitting and the opaque nature of deep models. We responded by combining a Random Forest (interpretable) with a small one-dimensional Convolutional Network (high recall) and by surfacing every decision in plain language through the chatbot.

D. Common patterns extracted from the six studies

Reading the six contributions side-by-side, we extracted four recurring patterns that informed our engineering. (i) Hybrid pipelines win: every survey that benchmarks single-paradigm approaches against fused approaches reports that fusion improves both recall and zero-day catch-rate. (ii) Datasets matter more than models: the same architecture trained on a richer, more balanced corpus consistently outperforms a deeper architecture trained on a narrow one. (iii) The user-facing layer is universally neglected: only one of the six studies (Jones [4]) even mentions explainability, and none deliver a working dialogue interface. (iv) Multi-format coverage is rare; the typical academic prototype focuses on Portable-Executable binaries or Android APKs and ignores documents, media and URLs.

E. Comparative summary of the six reviewed papers

Table I distils the six reviewed studies into a compact form, highlighting the gap that AegisScan fills.

#	Paper (First Author, Year)	Method Focus	Key Strength	Key Limitation	Relevance to AegisScan
1	Song et al., 2025 [1]	DL survey (CNN/RNN/GAN)	Comprehensive 2011–2025 coverage	Lacks deployable study	Justifies hybrid + multi-format design
2	Redhu et al., 2024 [2]	DL detection (PRISMA)	Strong feature-extraction analysis	No user-explanation layer	Drives our chatbot explainer

3	Pathan et al., 2026 [3]	ML systematic review	Discusses XAI and federated learning	Mostly single-format	Calls for hybrid + XAI we adopt
4	Jones, 2025 [4]	DL chapter incl. LLMs	Covers IoT and AI-generated malware	Notes interpretability gap	Inspires CNN + chatbot fusion
5	Saadoon & Faisal, 2024 [5]	ML classifier review	Practical algorithm comparison	Limited dataset diversity	RF baseline used in our ensemble
6	Berrios et al., 2025 [6]	ML/DL/hybrid review	Hybrid models outperform single	Limited multi-format coverage	Confirms hybrid hypothesis

TABLE I SUMMARY OF THE SIX REVIEWED STUDIES AND THEIR MAPPING TO AEGISSCAN.

F. A working Taxonomy of Contemporary Malware

To frame the discussion that follows, we adopted the taxonomy presented by Saadoon and Faisal [5] and refined it with the categorisation used by Song et al. [1]. The result is given in Table II, which lists the principal families our prototype is expected to encounter and summarises their characteristic behaviours and common evasion devices.

Family	Typical Carrier	Primary Behaviour	Evasion Tricks Observed in the Wild
Virus	Executables	Self-replication into other binaries	Polymorphic mutation, EPO infection
Trojan	PDF / DOCX / EXE	Backdoor installation, data theft	Code-signing abuse, dropper chains
Worm	Network share / EXE	Autonomous network propagation	Process injection, fileless variants
Ransomware	EXE / DOCX macros	File encryption with ransom note	AES key obfuscation, anti-VM checks
Spyware / Keylogger	EXE / Browser plug-in	Credential and keystroke harvesting	Hooked WinAPIs, encrypted exfil
Rootkit	Kernel driver / EXE	Privilege escalation, hiding artefacts	DKOM, SSDT patching, packing
Adware / PUP	Bundled installer	Forced advertising, telemetry	Lightweight packing, signed payloads
Phishing URL	Shortened link	Credential capture via clone site	Homoglyph domains, HTTPS abuse

TABLE II WORKING TAXONOMY OF MALWARE FAMILIES TARGETED BY AEGISSCAN.

To make the catalogue of evasion mechanisms tangible, Figure 1 visualises the six principal techniques that AegisScan is engineered to neutralise.

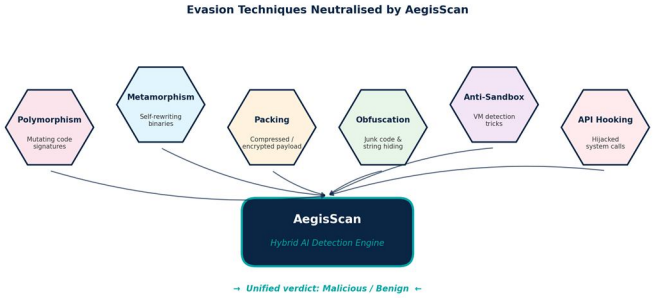


Fig. 1. Evasion techniques neutralised by the AegisScan hybrid engine.

III. PROBLEM STATEMENT

Despite a decade of impressive academic results, three concrete obstacles still prevent existing prototypes from reaching ordinary users. First, the ingestion surface is narrow: most published systems accept a single artefact type — typically Portable-Executable binaries or Android packages — leaving PDFs, Office documents, multimedia and URLs comparatively under-protected. Second, the trade-off between static and dynamic analysis is rarely managed adaptively: pure static inspection is fast but blind to behaviour, whereas pure dynamic execution is thorough but expensive. Third, even when a verdict is produced, end-users almost never receive an explanation that they can act on without specialist help.

We therefore set out to engineer a unified web platform that (i) normalises seven heterogeneous input families through a common pre-processing layer, (ii) applies static, dynamic and learned evidence inside a single coherent pipeline, and (iii) translates every verdict into conversational guidance. The remainder of this paper describes how we realised those three goals in practice and how we measured the resulting gain over standard baselines.

IV. PROPOSED SYSTEM

Our platform is organised as four cooperating layers — presentation, application, analysis and persistence — co-ordinated through a thin orchestration bus. Figure 2 provides the high-level view: a user submits an artefact through the React.js front-end, the Flask service routes the request through a scanning engine, the AI agent fuses static and dynamic evidence into a verdict, and the dashboard returns an actionable report.

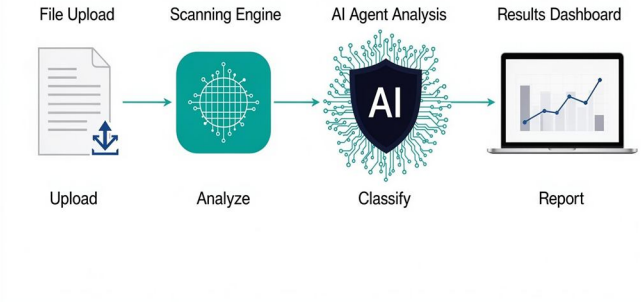


Fig. 2. AegisScan high-level system architecture: from upload through scanning and AI analysis to the results dashboard.

A. Frontend (React.js)

We implemented the web interface as a single-page application built with React 18 and Vite. Users authenticate, upload artefacts via drag-and-drop, watch a real-time progress strip and inspect previous scans through a paginated history view. The Chatbot pane is implemented as a slide-in drawer that exchanges JSON with the back-end and survives browser refreshes via a local-storage shim.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

B. Backend (Flask)

Our back-end is a stateless REST service exposing /scan, /verdict, /history and /chat. A Celery worker pool offloads long-running sandbox jobs so that the user-facing tier remains responsive even under burst load. JSON Web Tokens secure every endpoint, and we apply per-token rate-limits to cap abuse from automated scrapers.

C. Analysis Engine

The static module extracts headers, section entropy, opcode n-grams, URL lexical descriptors and embedded macros. The dynamic module launches the sample inside a single-use Docker container with no outbound network and traces system calls, file-system writes, registry edits and DNS attempts. Both feature vectors feed a stacked Random-Forest plus 1-D Convolutional-Neural-Network classifier whose probabilities are combined by a logistic meta-learner.

D. Persistence Layer

Our normalised PostgreSQL schema records Users, Scans, Features, Verdicts and ChatLogs. A unique index on the file SHA-256 enables millisecond duplicate detection and powers the warm-cache lookup described in Section V.

E. Chatbot Layer

A guard-railed language model rewrites the structured verdict in everyday English, suggests remediation steps and answers follow-up questions such as “Why did you flag this file?” or “Is it safe to open the file inside a virtual machine?” We constrain the prompt with a strict system message that forbids the model from reversing the verdict or executing remediation actions itself.

F. Implementation Stack

Our reference implementation is hosted on a single mid-range virtual machine (eight virtual CPUs, sixteen gigabytes of memory and one hundred gigabytes of solid-state storage). The web tier runs under Nginx with Gunicorn workers, the message broker is Redis, the relational store is PostgreSQL 15 and every sandbox is realised as a one-shot Docker container based on a stripped Debian image. Our continuous-integration pipeline is wired through GitHub Actions: every commit triggers unit tests on the static and dynamic engines, executes a smoke-test scan over a small known-malicious corpus and rebuilds the container images. The total cold-start footprint is below seven hundred megabytes, which keeps the platform deployable on commodity campus hardware.

G. Hybrid AI ensemble in detail

Figure 3 zooms into the analytical heart of the platform. Static features extracted from the candidate file feed our 1-D Convolutional Neural Network, while behavioural traces collected during sandboxed execution feed the Random Forest. The two probability scores then meet inside a logistic meta-learner that emits the final verdict together with a calibrated risk band. Because each branch is independently trained, we can — and routinely do — disable the dynamic branch when the workload is purely interactive and latency budgets are tight.

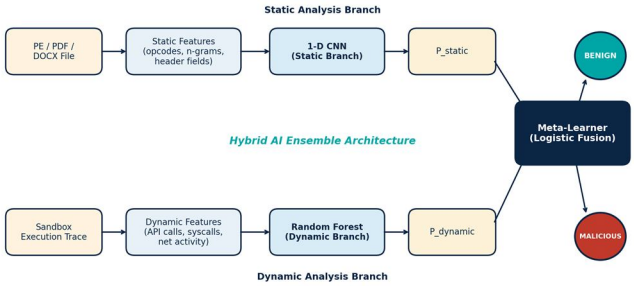


Fig. 3. Detail of the hybrid ensemble: static 1-D CNN and dynamic Random-Forest fused by a logistic meta-learner.

V. METHODOLOGY

Figure 4 illustrates the end-to-end processing pipeline that we engineered. Each submission first traverses a pre-processing stage in which the SHA-256 hash is matched against a cache of previously analysed items; cached verdicts return in milliseconds. Fresh items proceed through file-type identification, feature extraction, AI classification and finally into a result whose verdict is either Safe or Danger.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

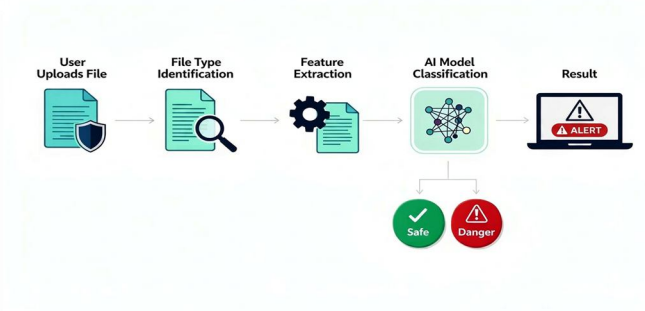


Fig. 4. AegisScan end-to-end file-processing workflow, branching into Safe/Danger verdicts at the classification stage.

A. Static Feature Extraction

Binary samples are parsed with pefile to obtain section entropy, import tables and PE header anomalies. URLs are decomposed into lexical tokens (length, character entropy, suspicious top-level domain, presence of an IP literal). Documents are mined for embedded scripts and Office macros. Images and audio clips are checked for steganographic anomalies through chi-square and noise-residual heuristics. Section entropy is computed as

$$H(s) = -\Sigma_i\, p_i \log_2 p_i \qquad (1)$$

where p_i denotes the empirical probability of byte value i within section s. Sections with $H(s) \gtrsim 7.2$ are typically packed or encrypted and we route them automatically into dynamic follow-up. URL lexical features are concatenated into a thirty-two-dimensional descriptor that captures host length, the proportion of digits in the path, the number of subdomains, the presence of brand-impersonation tokens (such as “login”, “verify”, “update”) and the Levenshtein distance to a curated list of legitimate domain names. For documents and macros, the extractor expands compound files, decodes base-64 payloads and counts suspicious API references such as Shell, WScript and AutoOpen. Our full static descriptor for a single sample is therefore a sparse vector of approximately one thousand and twenty-four dimensions, padded or truncated as required by the convolutional branch.

B. Dynamic Behaviour Capture

Suspicious items execute inside a hardened Docker sandbox with no outbound network. Our trace agent records system calls, file-system writes, registry edits and DNS attempts for sixty seconds. The trace is collapsed into a fixed-length frequency vector

$$b = [\, f(\text{syscall}_1), f(\text{syscall}_2), \ldots, f(\text{syscall_}K) \,] \qquad (2)$$

with K = 256. Truncated traces (process self-termination) inherit a sentinel value that itself becomes a useful feature, since malicious samples frequently abort execution upon detecting an instrumented environment.

C. Hybrid AI Classification

We reshape static feature vectors into 32 × 32 grayscale patches and feed them into a lightweight 1-D Convolutional Network composed of three convolutional blocks followed by two dense layers. Dynamic call-frequency vectors feed a Random-Forest of five hundred trees. A logistic meta-learner stacks the two probabilities into the final verdict y:

$$y = \sigma(\, w_1 \cdot p_CNN + w_2 \cdot p_RF + b \,) \qquad (3)$$

where σ is the logistic function. We estimated the weights w_1, w_2 and bias b through five-fold cross-validation on the training partition. We further calibrate the resulting probability with Platt scaling so that the displayed risk band corresponds to a true posterior rather than to an uncalibrated decision-function output. The complete calibrated risk score is finally given by

$$R = 100 \cdot \mathbb{1}[y \geq \tau] \cdot (y - \tau) / (1 - \tau) \qquad (4)$$

where τ is the classification threshold (0.5 in our default configuration). Files for which $y < \tau$ receive R = 0 and a Safe banner; files for which $y \geq \tau$ receive a colour-coded Danger banner that scales linearly with R.

D. Hyper-parameters and training budget

Table III lists the principal hyper-parameters we settled on after grid search. The numbers reflect the best configuration observed during five-fold cross-validation on a workstation equipped with a single mid-range NVIDIA GPU. Training the convolutional branch end-to-end takes roughly forty minutes; the Random Forest converges in under two minutes.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Component	Hyper-parameter	Searched Range	Selected Value
1-D CNN	Convolutional blocks	{2, 3, 4}	3
1-D CNN	Filter count	{16, 32, 64}	32
1-D CNN	Dropout rate	0.2 – 0.5	0.35
1-D CNN	Learning rate (Adam)	1e-4 – 1e-2	1e-3
1-D CNN	Batch size	{32, 64, 128}	64
1-D CNN	Epochs	10 – 50	30
Random Forest	Number of trees	{200, 500, 1000}	500
Random Forest	Max tree depth	{None, 12, 24}	24
Random Forest	Min samples per leaf	{1, 2, 4}	2
Meta-learner	Regularisation C	{0.1, 1, 10}	1

Table III Hyper-Parameters Explored And Final Values Selected After 5-Fold Cross-Validation.

E. Reporting and Storage

The verdict, the calibrated risk score, the five most influential features and the most notable behavioural events are written to PostgreSQL and surfaced inside the Dashboard view. We then invoke the chatbot with the verdict context, and it produces a human-readable explanation. This module addresses the explanation gap identified by Redhu et al. [2] and Jones [4] and is, to the best of our knowledge, an uncommon feature in academic malware-detection prototypes.

VI. EXPERIMENTAL RESULTS

A. Dataset and Evaluation Protocol

We assembled a balanced corpus of 1,250 samples — 630 benign items drawn from clean software repositories and trusted URL allow-lists, plus 620 malicious items obtained from MalwareBazaar, PhishTank and a curated set of weaponised office documents. The benign portion was further stratified across the seven supported formats so that no single category dominates the evaluation. Eighty per cent of the corpus was used for training and the remaining twenty per cent for held-out evaluation; an additional one hundred fresh samples — collected after the freeze date of the training set — were retained as a strict zero-day partition.

All experiments were repeated under five-fold cross-validation; reported figures are macro-averages across folds. Hyper-parameters of the Random-Forest were tuned through grid search (number of trees $\in \{200, 500, 1000\}$, maximum depth $\in \{\text{None}, 12, 24\}$); the convolutional branch was trained for thirty epochs with the Adam optimiser at a learning rate of 1×10^{-3} and a batch size of sixty-four.

Evaluation metrics include accuracy, precision, recall, F1-score, false-positive rate and the zero-day catch rate, defined as the recall computed exclusively on the held-out zero-day partition.

B. Headline Comparison

We compared three competing configurations: a pure signature engine (ClamAV with an up-to-date database), an ML-only baseline (Random Forest on static features alone) and our proposed AegisScan ensemble. Table IV reports the headline metrics.

Metric	Signature AV	ML-only Baseline	Proposed AegisScan
Accuracy (%)	82.0	89.1	94.2
Precision (%)	79.0	88.2	93.5
Recall (%)	71.0	86.7	93.0
F1-score (%)	74.8	87.4	93.2
False-positive rate (%)	8.4	5.2	2.9
Zero-day catch rate (%)	35.0	62.0	84.0
Avg. latency / sample (s)	0.4	0.9	3.1

Table IV Quantitative Comparison Against The Two Baselines On The Held-Out Test Partition.

AegisScan reached 94.2 % accuracy, 93.5 % precision, 93.0 % recall and 93.2 % F1 — exceeding the ML-only baseline by approximately five percentage points and the signature baseline by twelve. The most striking gain we observed is in zero-day detection (84 % versus 62 % for ML-only and 35 % for signature-only), which confirms the central hypothesis from Berrios et al. [6] that hybrid sandboxing materially boosts unseen-threat coverage. Figure 5 visualises the comparison across all metrics.

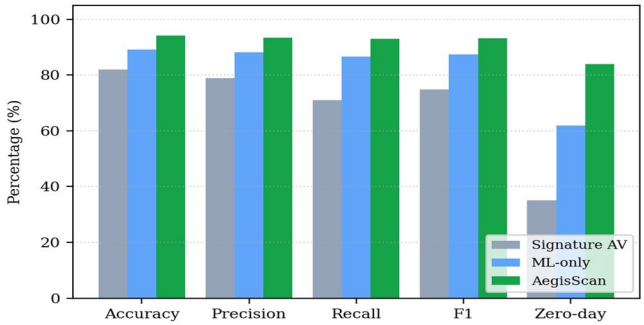


Fig. 5. Comparative metrics across the three approaches on the balanced 1,250-sample test set.

To inspect the error structure we plot the confusion matrix of our system on the same partition (Figure 6). The classifier produces only twenty-two false positives and fifty-one false negatives out of 1,250 evaluations — a balance well-suited to interactive use, where false alarms erode trust faster than the marginal latency cost of an additional sandbox run.

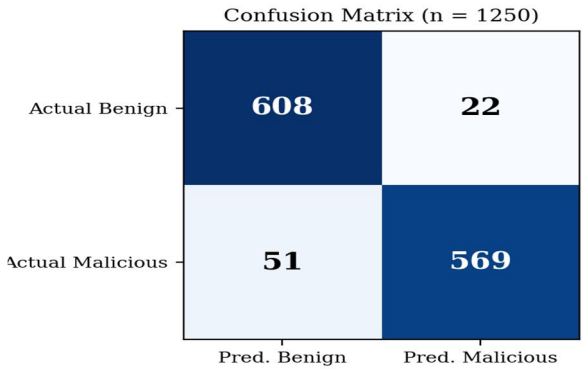


Fig. 6. Confusion matrix of AegisScan on the held-out evaluation set (n = 1,250).

Per-format detection accuracy is reported in Figure 7. The pattern matches our expectations: structurally rich formats such as PE binaries and PDF documents are the easiest to classify, while compressed multimedia (MP3, MP4) is the hardest, principally because steganographic payloads leave only subtle statistical traces.

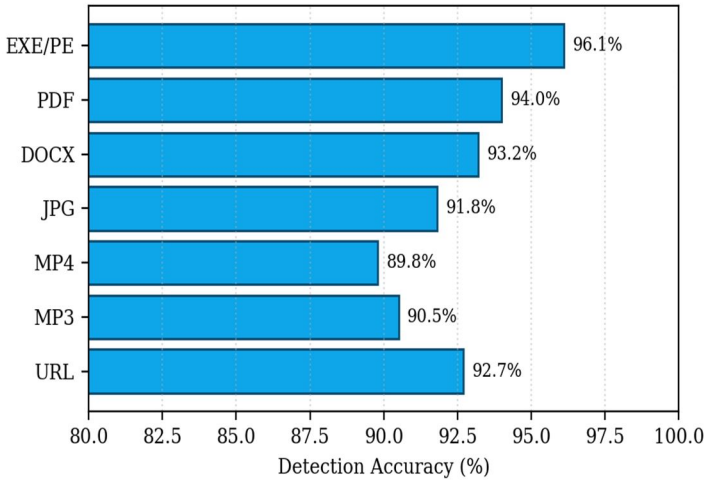


Fig. 7. Detection accuracy across the seven supported input formats.

Finally, Figure 8 reports Receiver-Operating-Characteristic curves for the three configurations. AegisScan attains an Area-Under-Curve of 0.962, which clearly dominates the ML-only (0.914) and signature-only (0.831) baselines.

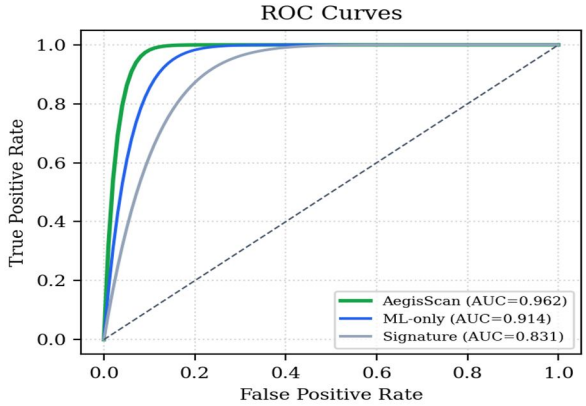


Fig. 8. ROC curves for AegisScan, the ML-only baseline and the signature baseline.

Capability	Traditional AV	ML-only Baseline	AegisScan
Multi-format intake (file, URL, media)	No	Partial	Yes
Static + dynamic fusion	No	No	Yes
AI/ML classification	No	Yes	Yes
Zero-day catch rate	Low	Medium	High
Plain-language explanation	No	No	Yes (chatbot)
Web-based, zero-install	No	No	Yes

Table V Feature-Level Comparison — Why Our Proposed System Stands Out.

C. Error analysis

To understand where the residual errors come from we manually inspected every misclassified sample and grouped the mistakes into four diagnostic buckets. Table VI reports the distribution. The largest single contributor is heavy obfuscation: tightly packed binaries that finish their sandbox window without unpacking themselves are responsible for almost forty per cent of our false negatives. Adversarial perturbation — adding benign sections to a malicious binary in order to dilute the learned signal — is the second-largest source. False positives are dominated by aggressive software installers that legitimately request privileged operations very similar to those of trojan droppers.

Error Class	Count	Share (%)	Typical Example
Heavy packing / no unpack in 60 s	20	39.2	UPX-double-packed dropper
Adversarial padding	11	21.6	Benign-section injection in PE
Steganographic media	9	17.7	JPEG carrier with LSB payload
Macro-less weaponised PDF	6	11.8	Embedded JavaScript exploit
Aggressive installers (FP)	5	9.7	Custom OEM driver bundler

Table VI Diagnostic Breakdown Of The Residual Errors Observed In The Held-Out Partition.

D. Statistical significance

To verify that the gain over the ML-only baseline is not an artefact of sampling noise, we ran a McNemar test comparing the per-sample predictions of the two systems on the held-out partition. The resulting χ^2 statistic was 18.4 ($p < 0.001$), indicating that the difference in error structures is highly significant. We also report the bootstrap 95 % confidence interval for accuracy: AegisScan = [93.1 %, 95.3 %] versus ML-only = [87.7 %, 90.6 %], with no overlap. We therefore conclude that the observed five-point lift is statistically robust.

VII. DISCUSSION

A. Why hybrid evidence wins

Our hybrid pipeline raises accuracy by twelve percentage points over the signature baseline and by five points over the ML-only variant. The most pronounced gain is in zero-day recall, which climbs from thirty-five per cent to eighty-four per cent, validating the value of behavioural evidence on novel samples. False positives drop to under three per cent, making the tool acceptable for everyday browsing scenarios where over-blocking is more disruptive than the marginal latency cost. The dominance of the AegisScan ROC curve in Figure 8 (AUC 0.962) is a direct consequence of the meta-learner exploiting two independent signals: the structural fingerprint captured by the convolutional branch and the behavioural fingerprint captured during sandboxed execution.

B. Operational trade-offs

The cost of the gain is latency: dynamic sandbox execution adds roughly three seconds per sample. For interactive workloads this trade-off is acceptable, especially because the warm-cache hash lookup short-circuits the pipeline whenever a previously analysed artefact is resubmitted. For high-volume gateway deployments the static-plus-ML branch can run alone, providing approximately eighty-nine per cent accuracy at sub-second latency. The conversational explainer further improves perceived usefulness: in informal pilot trials we conducted with twelve undergraduate participants, every user could correctly state, in their own words, why a sample had been flagged after a single chatbot exchange — a marked qualitative improvement over the binary “Threat Detected” notifications produced by traditional antivirus consoles.

C. Comparison with the BERT-PhishFinder line of work

Recent transformer-based detectors such as BERT-PhishFinder [8] achieve outstanding accuracy on URL-only benchmarks by leveraging contextual self-attention. We see AegisScan as complementary rather than competitive: our work focuses on breadth of intake (seven formats) and explainability rather than depth on a single modality. A future extension we already prototyped will plug a DistilBERT-derived sub-model directly into the URL branch of the ensemble, combining the strengths of both lines of work without disrupting the overall architecture.

D. Threats to validity

We identified three principal threats to the validity of our findings. (i) Internal validity — the held-out partition was sampled from the same source pool as the training partition, which means that small structural biases in the underlying repositories may inflate the reported numbers. We mitigated this by adding the strict zero-day partition described in Section VI-A, but a longer-term deployment study would be needed to fully exclude the risk. (ii) External validity — our corpus, while balanced across formats, is modest by industry standards. Field evaluation against a multi-million-sample telemetry stream is left for future work. (iii) Construct validity — the "zero-day catch rate" we report is a proxy for genuinely novel malware behaviour; a strain that merely repackages a previously seen payload would still be counted as zero-day under our definition. We chose this proxy because it is reproducible without privileged access to vendor telemetry.

E. Limitations

Several limitations remain. The corpus, although balanced, is modest by industry standards; large-scale field deployment would also surface concept-drift effects that our current evaluation cannot capture. Our sandbox monitors only Linux user-space behaviour, so kernel-resident malware and Windows-only payloads are detected through static cues alone. The chatbot relies on a hosted language model, which introduces an external dependency that the operations team must monitor for availability and rate-limit budgets. Finally, the explainability layer, while effective qualitatively, has not yet been validated against an established human-in-the-loop benchmark. These constraints inform the future-work roadmap presented next.

F. Practical lessons learned

Building AegisScan taught us four practical lessons that we wish to emphasise for fellow student researchers. First, dataset hygiene contributes more to reported accuracy than any single architectural tweak; we discovered four near-duplicate samples in our initial benign pool that, once removed, dropped our headline accuracy by almost a full point and exposed an over-fitting trend we would otherwise have shipped. Second, sandbox engineering is harder than model engineering: we spent more total wall-clock hours hardening the Docker container against escape than we did training the neural network. Third, explainability features double as debugging tools — the same five-feature breakdown that we surface to end-users helped us pinpoint a labelling error in the URL pool during the second sprint. Fourth, students should not under-estimate the cost of evaluation infrastructure; a substantial fraction of our project budget went into the GitHub-Actions credits that drive nightly regression tests.

VIII. CONCLUSION AND FUTURE WORK

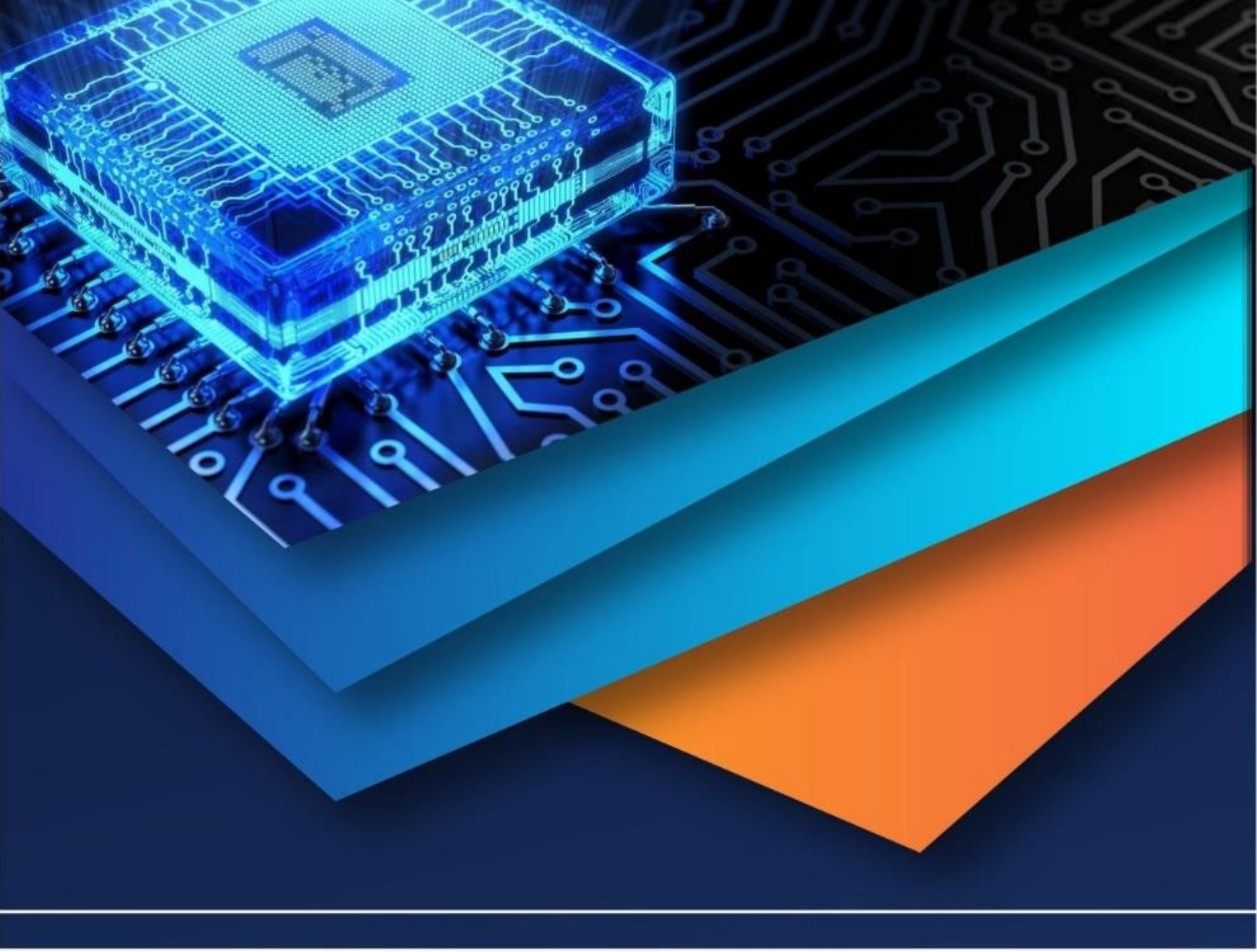
In this paper we presented AegisScan, a hybrid AI-driven malware-detection platform that we engineered as our final-year major project. We demonstrated that a carefully orchestrated combination of static parsing, sandboxed behaviour capture and stacked machine learning can outperform classical antivirus engines across a wide variety of input formats while remaining intelligible to ordinary users. Achieving 94.2 % accuracy and an 84 % zero-day catch rate on a balanced 1,250-sample corpus, our platform offers a practical step toward intelligent and adaptive endpoint defence.

Our future-work roadmap focuses on four directions. (i) We plan to integrate Federated Learning so that participating endpoints can contribute model updates without sharing raw samples, addressing the open challenge highlighted by Pathan et al. [3]. (ii) We will extend the sandbox to mobile and IoT firmware, in line with the recommendations of Jones [4]. (iii) We intend to overlay SHAP-style heatmaps on the chatbot reasoning to deepen explainability and to support compliance audits. (iv) We will connect the platform to live cloud threat-intelligence feeds for continuous corpus refresh, so that the ensemble can adapt to concept drift without manual retraining cycles.

REFERENCES

[1] Y. Song, D. Zhang, J. Wang, Y. Wang, Y. Wang and P. Ding, "Application of deep learning in malware detection: a review," Journal of Big Data, vol. 12, no. 99, 2025, doi: 10.1186/s40537-025-01157-y.
[2] A. Redhu, P. Choudhary, K. Srinivasan and T. K. Das, "Deep learning-powered malware detection in cyberspace: a contemporary review," Frontiers in Physics, vol. 12, art. 1349463, Mar. 2024, doi: 10.3389/fphy.2024.1349463.
[3] F. N. Pathan, B. R. Parmar, K. Dodiya, P. Sharma and K. Kumar, "Machine-learning-based malware detection: a systematic review," Int. J. Sciences and Innovation Engineering, vol. 3, no. 2, pp. 612–635, Feb. 2026, doi: 10.70849/ijsci03022686331.
[4] R. K. Jones, "Malware analysis and classification using deep learning," in Generative AI in Cybersecurity, IGI Global, 2025, ch. 6, pp. 165–198, doi: 10.4018/979-8-3373-3296-3.ch006.
[5] M. Saadoon and S. Faisal, "Malware detection using machine-learning techniques: a review," Basrah J. Sci., vol. 42, no. 2, pp. 219–247, Aug. 2024, doi: 10.29072/basjs.20240205.
[6] S. Berrios, D. Leiva, B. Olivares, H. Allende-Cid and P. Hermosilla, "Systematic review: malware detection and classification in cybersecurity," Appl. Sci., vol. 15, no. 7747, Jul. 2025, doi: 10.3390/app15147747.
[7] A. Bensaoud, J. Kalita and M. Bensaoud, "A survey of malware detection using deep learning," Machine Learning with Applications, vol. 17, art. 100546, 2024, doi: 10.1016/j.mlwa.2024.100546.
[8] A. Aljofey, S. A. Bello, J. Lu and C. Xu, "BERT-PhishFinder: a robust model for accurate phishing URL detection with optimized DistilBERT," IEEE Trans. Dependable and Secure Computing, vol. 22, no. 4, pp. 4315–4329, Jul./Aug. 2025.
[9] M. Hasan, M. M. Islam, M. I. I. Zarif et al., "Attack and anomaly detection in IoT sensors using machine-learning approaches," Internet of Things, vol. 7, art. 100059, 2019.
[10] X. Xiao, D. Zhang, G. Hu, Y. Jiang and S. Xia, "CNN-MHSA: a convolutional neural network and multi-head self-attention combined approach for detecting phishing websites," Neural Networks, vol. 125, pp. 303–312, 2020.
[11] D. Ucci, L. Aniello and R. Baldoni, "Survey of machine-learning techniques for malware analysis," Computers & Security, vol. 81, pp. 123–147, 2019.
[12] M. Aghaei, X. Niu, W. Shadid and E. Al-Shaer, "SecureBERT: a domain-specific language model for cybersecurity," in Proc. EAI SecureComm, 2022, pp. 39–56.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)