Indian Journal of Computer Science and Technology
https://www.doi.org/10.59256/indjcst.20260503082
Volume 5, Issue3 (September-December 2026), PP: 658-676.
www.indjcst.com
ISSN No: 2583-5300

Genshield AI: Intelligent Ransomware Detection and Prevention Using Generative AI

Vaishnav Ramesh[1], G.Paavai Anand[2]
[1]B.Tech Student, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani, Chennai, Tamilnadu, India.
[2]Assistant professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani, Chennai, Tamilnadu, India.

To Cite this Article: *Vaishnav Ramesh[1], G.Paavai Anand,"Genshield AI: Intelligent Ransomware Detection and Prevention Using Generative AI", Indian Journal of Computer Science and Technology, Volume 05, Issue 03 (September-December 2026), PP: 658-676.*

Copyright: ©2026 This is an open access journal, and articles are distributed under the terms of the Creative Commons Attribution License; Which Permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Abstract: *When information leakage or testing is weak, the accuracy of ransomware detection systems will be very high. This study explores the reimagined GENSHIELD AI, a multi-layered security offering that utilizes autoencoder anomaly detection with supervised machine learning, the Vision Transformer (ViT) family class attribution approach and a moderated connoisseur Generative AI advisory. The datasets used for testing the system were two public datasets: 62,485 records from MLRD and 58,596 records from CIC-MalMem-2022. After cleaning, the MLRD was found to have 32,240 unique feature vectors and out-of-scope testing yielded 39,089 detection records and 19,507 other-malware records from CIC-MalMem-2022. With benign fitted log-standard scaling and 99th percentile threshold, the autoencoder achieved a perfect recall, F1 score of 0.986 and an accuracy of 0.991 in CIC-MalMem-2022 while having a false-positive rate of 0.009. On MLRD, however, only 39.5% was recalled while precision was 0.952. Duplicate data contributed to the strong performance and low observed false-negative rate of the supervised models. With a macro-F1 score of 0.335 for the ViT, family attribution was less certain. Additionally, the memory model detected 100% of the 19,507 malware and trojan entries it came across. Additionally, the memory model flagged 100% of spyware/trojan data for 19,507, indicating a limited detection of ransomware. By eliminating active content and adding a tiny 0.3ms processing time, the guarded Generative AI layer became effective. GENSHIELD AI is more suitably considered as a malware triage system and not as a one-shot ransomware solution for the time being, pending a more extensive deployment and validation.*

Key Word: *Ransomware Detection, GENSHIELD AI, Autoencoder, Vision Transformer, Malware Classification, Generative AI, Information Leakage.*

I. INTRODUCTION

Ransomware is one of the more severely damaging types of cybercrime because it directly impacts availability. Once the system has been encrypted, it is out of use until restored or ransom paid. Data Encrypted for Impact (T1486): This behavior is documented by MITER ATT&CK.ENISA Threat Landscape 2025 provides analysis of a total of 4,875 incidents during the period July 2024 to June 2025. Ransomware is the core element of intrusion activity against European organizations. Of course, these amounts don't take into account downtime, recovery or data-leak costs, which tend to be significantly greater. Chainalysis indicates that, except for downtime, recovery and data-leak costs, more money is paid to destroy or erradicate ransomware than to pay the ransom. But, these amounts do not account for downtime, recovery or data-leak costs, which will typically be significantly higher, as indicated by Chainalysis (Chainalysis, 2025).

The threat is also getting more complicated. Early ransoms were relatively simple to break, with little of the 1359 samples analyzed between 2006 and 2014 exhibiting powerful and complex destructive features, found by (Kharraz et al, 2015). Today, operations involve robust hybrid encryption, man-in-the-middle attacks, affiliations with Ransomware-as-a-Service (RaaS), and the theft of data (Oz et al, 2022; Razaulla et al., 2023; McIntosh et al, 2021). Defenders thus require a control that responds early to a payload usually before it could be run, and is able to be applied to a family that they have not encountered before.

Traditional controls have difficulties achieving both of these requirements. Signature matching is quick and accurate for known samples and easily breakable when samples are repacked, obfuscated or new families appear. (Ucci et al, 2019; Moussaileb et al, 2021) The behaviour monitoring method can reliably detect files that are encrypted; however, once the algorithm has been executed some files will remain unencrypted prior to the alarm. (Scaife et al, 2016; Continella et al., 2016). Sandboxing is an added cost and can be circumvented by environment-aware samples (Kharraz et al., 2016; Urooj et al, 2022). Machine learning (ML) and deep learning (DL) detectors have been suggested to be able to generalise beyond signatures, and it is reported that many of them have very high accuracy (Fernando et al, 2020; Ferdous et al, 2024; Cen et al, 2024a). Security ML is, however, significantly prone to evaluation issues including data leakage, temporal bias,

unrealistic class balance and so on, and often the results achieved in the lab are more optimistic than what can be achieved in deployment (Pendlebury et al.,2019; Arp et al.,2024; Kapoor and Narayanan, 2023). Generative artificial intelligence (GenAI) provides a new dimension, and more specifically, large language models (LLMSM).

Transformer-based LLMs (Vaswani et al.,2017; Brown et al.,2020) can be fed technical evidence, alert explanations, and guidance for respondents or on exploring the detections, thereby assisting in responding to detections when non-specialist users encounter them (Ferrag et al.,2025; Xu et al.,2024). Also, when their text is displayed to users without being validated, the same models hallucinate, can be fooled into responding to prompts they don't have, and can give unsafe responses (Huanget al.2025; Greshakeet al.2023; OWASP, 2025). Benchmarks reveal only 65-72% accuracy for even powerful LLMs to complete CTI tasks (Alamet al.2024). It is thus not important whether GenAI can be embedded in a detector, but what place it can safely be integrated in a detector and how this place should be measured.

II. MATERIALS AND METHODS

2.1. Research Design

The study uses a quantitative, experimental design with repeated, seed-controlled runs. H1: benign-fitted log-standard scaling with validation-calibrated thresholds improves autoencoder detection significantly over the original per-sample scaling and fixed threshold. H2: random record-level splits overestimate performance relative to de-duplicated or sample-grouped splits. H3: a closed-set family model assigns high confidence to many samples from families it has never seen. H4: a schema- and allow-list-based guard removes active content from LLM output without removing legitimate advisory formatting. The original prototype's trained weights and training data were not available, so all models were retrained from scratch with the original architectures adapted to the input dimensions.

2.2. Genshield Ai Framework

2.2.1. Architecture and Evidence Channels

Genshield AI also consists of a chain of layers, with every layer having a single, rigidly testable task (as illustrated in Fig. 1). Questioning the evidence about a suspicious program, to the data acquisition layer, which will transform it into validated numerical features; the detection and attribution layer will determine if the evidence is malicious or not, and if it is and identify which ransomware family it is similar to; the decision layer will convert the output produced by the model into a threat level; and the advisory and response layer, the explanation of the threat itself using the model output, along with suggested actions. The original implementation involved three parts: a Flask web gateway for uploading Windows PE, a PyTorch autoencoder, a ViT classifier and a rule-driven threat level rating, and a Gemini-based advisory. The technical review identified four elements that were not safe or testable in this study and were changed: feature scaling, threshold selection, confidence handling, and the genai skin. This study proposes changes to the components of the original prototype that have been labelled, as well as those that have been implemented.

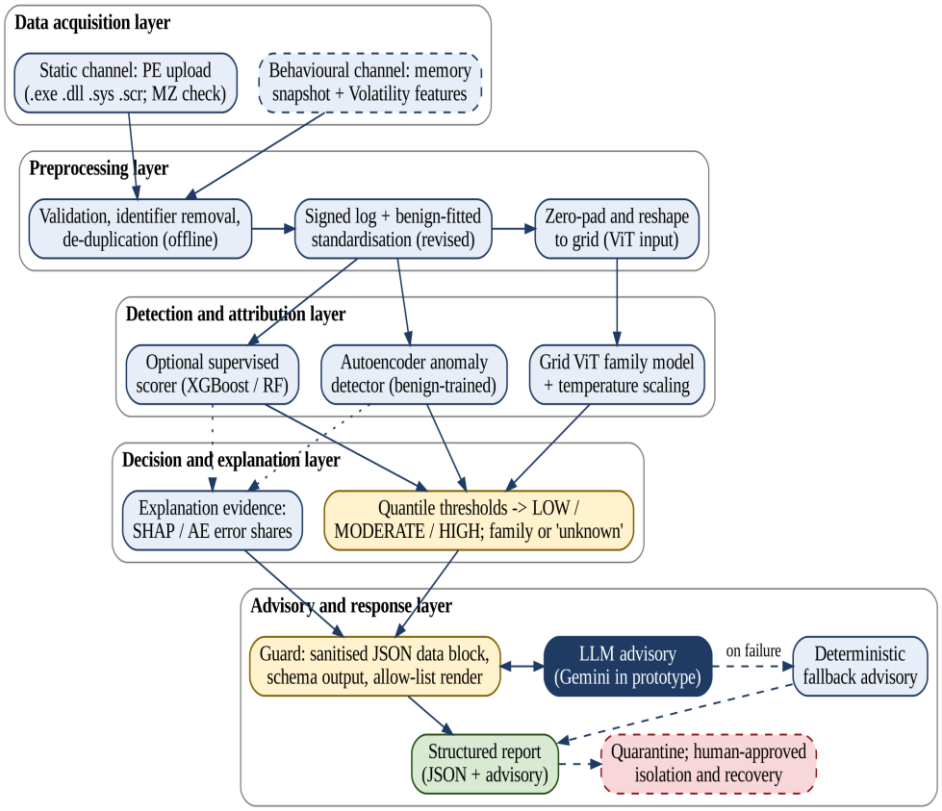


Fig. 1. Layered architecture of GENSHIELD AI. Solid boxes are implemented in the original prototype or revised and evaluated in this study; dashed boxes are proposed and not yet integrated.

Two evidence channels are considered. The static channel analyses a PE file before execution. The original gateway accepts .exe, .dll, .sys and .scr uploads, confirms the PE format by extension or the "MZ" signature, and reads header fields with pefile (Microsoft, 2026). In this study the static channel is evaluated with the MLRD collection of PE-header features from ransomware and benign files (Table 3). The behavioural channel uses memory-forensic evidence: a snapshot of system memory is processed with Volatility plugins that summarise processes, loaded libraries, handles, injected code regions, hidden processes, services and kernel callbacks (Carrier et al, 2022). This channel is evaluated offline with CIC-MalMem-2022; live memory acquisition is not part of the current gateway.

2.2.2. Preprocessing and Behavioural Features

Preprocessing validates the input, removes identifiers, duplicates and scales features. The original code used mean and standard deviation for each feature vector. This per-sample scaling means that one or two of the per-sample scalars can swamp the content of the entire vector, as the image base and timestamps are very large for value x + y, compared to the byte frequencies or the version numbers. GENSHIELD AI therefore applies a signed logarithm to every feature and standardises each feature using statistics estimated on benign training data only (Section 2.4). For the family stage, the scaled vector is zero-padded and reshaped into a square grid, exactly as the original ViT expected, so that a convolutional patch embedding can be applied.

Although the traces are buried in different locations, ransomware behaviour reveals itself in several patterns such as bursts of file writes and renames, high entropy output files, cryptographic API calls and process injection (Scaife et al.,2016; Continella et al.,2016; Kolodenker et al.,2017; Davidian et al.,2024). Process and thread counts, handle counts of object types (files, keys, mutants, events), DLLs not linked in loader lists, injected executable regions reported by malfind, processes not present in the standard listing, service states, and service callbacks are the subset of these indicators used in this study that can be seen in memory snapshots. The outcomes of file-system I/O sequences, API-call traces, and network flows are not claimed since they are not included in the analyzed datasets. They still make sense as the behavioral module's subsequent inputs.

2.2.3. Detection and Attribution Models

There are three different models in the module, which have been selected for particular reasons. (i) An Autoencoder is trained on clean data and identifies inputs that have reconstruction error values that are too large. This maintains the initial objective of GENSHIELD - identify the threats without making use of examples of all ransomware families (Chandola et al, 2009; Pang et al, 2021; Mirsky et al, 2018). In addition (ii), supervised scorers are used as strong tabular baselines and as an optional second opinion, if labelled data is available, such as gradient-boosted trees (Chen and Guestrin, 2016) and random forests. A grid ViT (faithful to the original SimpleViT: convolutional patch embedding, two Transformer encoder layers, flattened tokens and a linear head without class token or position embedding) proposes a ransomware family for flagged inputs (Dosovitskiy et al.,2021). It has the ability to use temperature scaling to calibrate its outputs in a softmax format (Guo et al, 2017).

2.2.4. Generative AI Module

We use the term Generative AI here, and its usage is meant in the specific context of the LLM. The functions of pipeline are separated in table 1. The autoencoder makes predictions of classification and behaviour pattern analysis, while the supervised scoring models and ViT do so. The Autoencoder is trained with a reconstruction model of a benign behaviour without being used to produce content. The LLM handles structured results that have meaning in the natural language in the following way: it takes the threat level, anomaly score, the most influential features, the calibrated family suggestion and transforms these into an advisory, which reassures, provides the necessary warnings as well as recommending an action. There is no binary available and the decision is not changed to determine who is elected. It takes sanitized and wrapped inputs as input, a strict schema as output, and uses an allow-list to output the results. In addition, a deterministic fallback advisory is provided in case the LLM can't be accessed or returns invalid output.

Function	Component	Output	Status
Predictive classification (detection)	Autoencoder; optional XGBoost/RF scorer	Anomaly score, malicious probability	Original AE revised; scorers added and evaluated
Behavioural pattern analysis	Memory-forensic features + models	Learned decision boundaries	Evaluated offline (CIC-MalMem-2022)
Family attribution	Grid ViT + temperature scaling	Family suggestion, calibrated confidence	Original ViT adapted and evaluated
Natural-language reasoning and explanation	LLM (Gemini in the prototype)	Advisory text in a fixed schema	Integration guarded and evaluated; LLM quality not measured
Threat explanation evidence	SHAP; autoencoder error shares	Top contributing features	Evaluated (Section 3.9)
Automated recommendations	LLM + deterministic fallback	Warnings and actions aligned with CISA guidance	Fallback implemented; automatic actions proposed

Table 1. Functional roles in GENSHIELD AI and the component responsible for each

2.2.5. Decision, Response and Explanation

The combination of the anomaly score and the calibrated family confidence goes into the decision module. If its reconstruction error is above the 99th percentile of these benign validation errors, then it is flagged. A flagged sample satisfies the threshold either when the error also exceeds the 99.9th percentile or when the calibrated family confidence is at least 0.80, otherwise it satisfies the threshold MODERATE (Equation 1).A flagged sample meets the threshold if the error meets the threshold, or if the calibrated family confidence is at least 0.80, otherwise the sample meets the threshold only if the error meets the threshold MODERATE (Equation 1). Confidence level of at least 0.80 are reported whereas a 20% under this level is reported as "family is unknown". Data is not used for these rules, they are based on an existing set of hard rules (loss > 0.35 or confidence > 80%).

$$\text{Level}(x) = \text{HIGH if } L(x) > \tau_{0.99} \text{ and } [L(x) > \tau_{0.999} \text{ or } c(x) \geq 0.80]; \quad \text{MODERATE if } L(x) > \tau_{0.99} \text{ otherwise}; \quad \text{LOW if } L(x) \leq \tau_{0.99} \tag{1}$$

In GENSHIELD AI, the prevention is done pre-execution: means that when a new executable is detected in the system, GENSHIELD AI analyze it beforehand, and indicates with a RESULT = HIGH or MODERATE to the user not to run it. In the proposed prototype this control is based upon the user following the advice. The proposed response layer is adding to the automatic quarantine of the uploaded file, completeness to fast-track responses (even though it is not required for non-destructive feature or file extraction), and the need to be manually approved before deciding to take high-impact actions such as isolating a host, terminating processes or restoring files, and following incident response guidance such as from (Nelson et al, 2025), or from (CISA et a, 2023). The organisable control that is not simulated in the model is recovery from offline backups (Souppaya et al, 2026). In this study these capabilities are not assessed but are proposed.

In each alert there should be a reason that someone can validate. TreeSHAP is GENSHIELD AI's method for ranking features that came in towards predicting ransomware for supervised-scoring. TreeSHAP is the method GENSHIELD AI employs in order to rank features which were used to push a prediction towards ransomware for the supervised-scoring function. In the case of an autoencoder it reports the proportion of reconstruction error due to each feature, meaning the behaviours which the benign model was not able account for. To ensure that the explanation is grounded in the model outputs, these ranked features are passed as evidence to the advisory, not free-form LLM reasoning (Capuano et al, 2022; Saqib et al, 2024).

2.2.6. System Workflow

The whole procedure is demonstrated in Fig. 2. There are two methods of obtaining evidence: a PE file or a memory snapshot. (3) The input is validated and it gives output as features. (3) Features are scaled using ‘benign-fitted' statistics. (4) Compute anomaly score from the autoencoder; optionally add a probability from a supervised scorer. The flagged inputs are fed to the calibrated ViT for the family suggestion. (6) Assigns 'LOW', 'MODERATE' or 'HIGH' and determines if a family name can be displayed. (7) The SHAP values, or error shares are used to compute explanations. (8) Advisory is generated by the guarded LLM, or fallback is in effect. (9) The report is returned, and in the proposed design any files deemed to be "high risk" are placed in quarantine for human review. Analyst feedback to modify labeled data and thresholds (10).

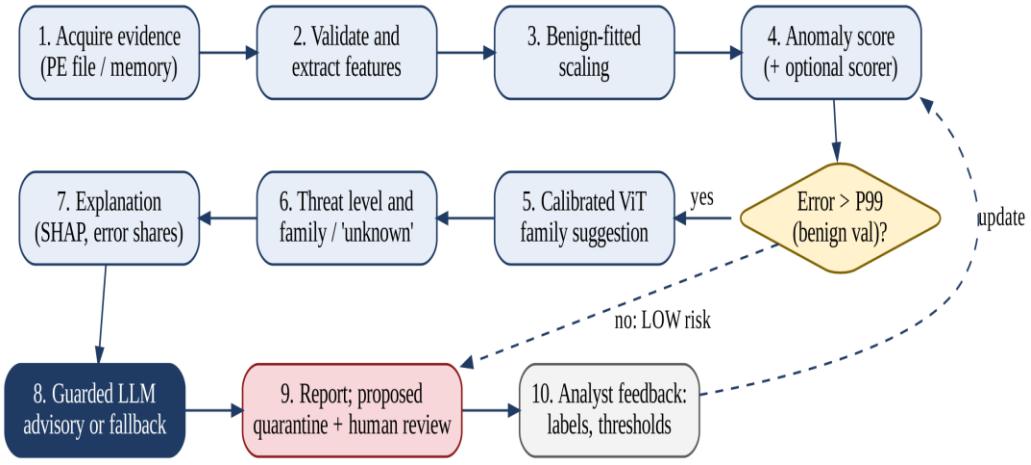


Fig. 2. End-to-end GENSHIELD AI workflow from evidence acquisition to advisory, response and feedback

2.2.7. Prototype Implementation and User Interface

To understand the behavior of the original artefact, the prototype was locally tested using the weights trained in the original artefact (Fig. 3). First the Flask application loads 582 family labels from family_label_map.json into the mesh, then configures the client with gemini-1.5-flash, loads the autoencoder and ViT weights and serves up the interface via the Flask debug server (Fig. 3a). The dashboard (Fig. 3b) contains a threat radar, a four-stage pipeline (file, "AI firewall" autoencoder, ViT classifier and protected server), a risk-score gauge and a prediction report and advisory panel. The files are selected using a file dialog (Fig. 3c) and each success can be opened in a report window or downloaded as a text report (Fig. 3d).

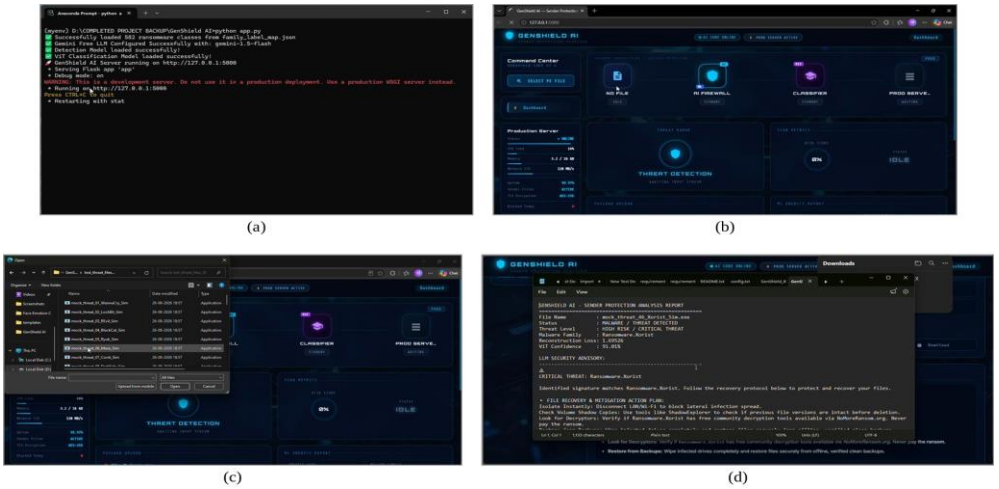


Fig. 3. Operation of the original GENSHIELD AI prototype.

(a) Server start-up log showing the loaded label map, LLM client and models. (b) Dashboard in the idle state. (c) Selection of a file from the set of 50 simulated test files. (d) Downloadable plain-text analysis report, shown for the simulated test file analysed in Section 3.11.

The source code demonstrates how the outputs are created. A 256-byte histogram (256 bins), six header fields, and the entropy calculated from the up to 10 sections are converted to a 2,381 value vector, each sample normalized to this, and then compared to a threshold of 0.15. There are two rules that are important to understand how to interpret the output. If the file is not in the PE binary format it is not analysed, and the server will respond with the SAFE tag and reconstruction loss of 0, and a confidence of 100%. If the file is flagged, the family is extracted from the file name if it contains one of 46 hard-coded families, otherwise the confidence will be randomly drawn from a list from 91% to 97% with the file size as the random seed (ViT is used only when the name matches none). The file is considered HIGH if loss is more than 0.35 or when confidence is above 80%. The server-status numbers on the dashboard (CPU load, memory, uptime, encryption) will not be measures, just display elements. The prototype was equipped with a total of 50 simulated test files whose name involves a number and a family name, such as mock_threat_46_Xorist_Sim.exe; contents of these files were not available to this work and live malware was not handled.

2.3. Datasets

Potential datasets were identified and filtered for public accessibility, traceable origins and for the presence of the ransomware and also for suitable evidence for the two evidence pathways. EMBER (Anderson and Roth, 2018) and BODMAS (Yang et al.,2021) both share a comparable format with the 2381 features suggested by the original code, but they are both general malware collections, with BODMAS further failing to give family names for ransomware, and neither of them was either recoverable in the network restrictions of the experimental framework. The data assembled by the Microsoft Malware Challenge was not applicable, as files of non-ransomware families were disassembled and byte files were provided. Two datasets were chosen (Table 2). The collected features are publicly available in the MLRD distributed ransomware and benign PE-header features from the open source MLRD project, which are available in the copy provided here; the size of this distribution (approx. 62,405 samples, 18 attributes, 57% ransomware) is equivalent to Dataset 1 from (Barnes and Ghafarian, 2026). There is a lack of documentation of its provenance which is a weakness for it. Only a subset of the CIC-MalMem-2022 memory-dump records refers to ransomware (5,930) out of 58,596 records, and all of them are behavioural, not static datasets neither benign nor malicious processes. Both CIC datasets and MLRD repository include a CIC licence file; both are available for use with citation for non-commercial research use only.

Characteristic	MLRD PE-header data (static)	CIC-MalMem-2022 (memory forensics)
Source and integrity	securycore/MLRD repository; SHA-256 6cb5c2…ccb2ac	Two independent public mirrors with identical SHA-256 624ecb…40c4; official description (Carrier et al., 2022)
Records (raw)	62,485 (35,367 ransomware; 27,118 benign)	58,596 (29,298 benign; 9,791 ransomware; 10,020 spyware; 9,487 trojan)
Features	15 numeric PE-header fields (identifiers removed)	55 Volatility features; 3 constant features removed (52 used)
Duplicates	30,229 duplicate feature vectors (48.4%); 8 vectors with conflicting labels	268 duplicate ransomware rows; 67 duplicate benign rows
Grouping	One row per file hash	About 10 dumps per malware sample (997 ransomware sample hashes)

After cleaning	32,240 unique vectors (10,675 ransomware; 21,565 benign)	39,089 rows for detection (benign + ransomware); 19,507 other-malware rows for out-of-scope testing
Ransomware families	Not provided	Ako 2,000 / 200; Conti 1,988 / 200; Maze 1,958 / 176; Pysa 1,717 / 192; Shade 2,128 / 229 (dumps / samples)

Table 2. Characteristics of the datasets used (values measured on the files analysed)

No live malware was collected or executed. Both datasets were downloaded as feature tables from public repositories, and their SHA-256 digests were recorded. For CIC-MalMem-2022, two independent mirrors were compared and found to be byte-identical; record counts and class totals were checked against the official description, which reports 58,596 records with equal benign and malicious shares (Carrieret a, 2022). The official page lists 986 ransomware samples, whereas parsing the sample hash embedded in each record name yields 997 distinct hashes; we used the parsed hashes as grouping keys and report this small discrepancy. For MLRD, the inclusion criterion was a valid label and complete numeric features; all 62,485 rows met it. Exclusion was applied only through de-duplication.

2.4. Data Preprocessing and Feature Engineering

No missing values were present in either dataset. Identifiers (file name and MD5 in MLRD; record names in CIC-MalMem-2022) were removed after being used for grouping. In MLRD, exact duplicate feature vectors with the same label were removed, and vectors that occurred with both labels were discarded. All features are non-negative counts or header values, so no categorical encoding was needed. Two scaling functions were compared. The original GENSHIELD code standardised each sample with its own statistics (Equation 2), where μx and σx are the mean and standard deviation of the d features of sample x. The revised method applies a signed logarithm and standardises each feature j with the mean μj and standard deviation σj estimated on benign training data for the autoencoder, or on all training data for supervised models (Equation 3).

$$z_j = (x_j - \mu_x) / \sigma_x \quad (2)$$

$$z_j = (\text{sign}(x_j) \cdot \log(1 + |x_j|) - \mu_j) / \sigma_j \quad (3)$$

Class imbalance treated without resampling: Anomaly detectors are trained with benign data, thresholds set up on benign validation data and supervised models are trained on a data with natural class distribution and evaluated with an imbalance-aware measure. The feature selection was done only by discarding the features which were constant. It will not compare different models based on feature selection done by feature selection.No other feature selection was applied thus it is not the selection procedure that they are comparing but the various models.

The features defined in Table 3 are considered relevant with respect to ransomware. The static data attributes such as debug-directory and DLL characteristics, linker and image versions and resource sizes simply indicate the way a binary was compiled and bundled; these sorts of fields were found to be extremely discriminative by Barnes and Ghafarian (2026). Persistence, injection and encryption deceive the operating system, as evidenced by traces in the memory data, and reveals how the operating system interacts with the running process (Carrier et al, 2022).

Group	Features	Relevance to ransomware
PE build metadata	Machine, MajorLinkerVersion, MinorLinkerVersion, MajorImageVersion, MajorOSVersion	Toolchain and packing traces; differ between commodity builds and legitimate software
PE directories	DebugSize, DebugRVA, ExportRVA, ExportSize, IatVRA, ResourceSize	Stripped debug data, few exports and unusual resources are common in packed payloads
PE loader settings	NumberOfSections, SizeOfStackReserve, DllCharacteristics	Security flags (e.g., ASLR, DEP) and section layout reflect build practice
Content indicator	BitcoinAddresses	Presence of cryptocurrency wallet strings, a ransom-note artefact
Processes and DLLs	pslist, dlllist counts, average threads and handlers	Process creation and library loading by the malicious process
Handles	handles by type (file, key, event, mutant, section, thread and others)	Many file and registry handles accompany mass encryption; mutexes prevent double infection
Injection and hiding	malfind (injections, commit charge, protection), ldrmodules (not in load/init/memory), psxview	Code injection and stealth techniques
Services and kernel	svcscan, modules, callbacks	Service manipulation and kernel-level persistence

Table 3. *Feature groups used in the experiments and their relevance to ransomware*

2.5. Model Configuration

Table 4 lists the model configurations. The autoencoder keeps the original encoder–bottleneck–decoder design but is resized to the input dimension so that the bottleneck is smaller than the input (d–64–32–b–32–64–d with b = ⌈d/4⌉; b = 4 for MLRD and 13 for CIC-MalMem-2022). Given encoder f_θ and decoder g_φ, the reconstruction and anomaly score are defined in Equations 4 and 5, and the threshold for quantile q is defined on the benign validation set V_b in Equation 6.

$$\hat{x} = g_\varphi(f_\theta(z)) \quad (4)$$

$$L(x) = (1/d) \sum_{j=1}^{d} (z_j - \hat{x}_j)^2 \quad (5)$$

$$\tau_q = Q_q\{ L(x) : x \in V_b \}, \quad q \in \{0.95, 0.99\} \quad (6)$$

The grid ViT reproduces the original SimpleViT: the 52 scaled memory features are zero-padded to 64 values, reshaped to an 8 × 8 grid, embedded by a 2 × 2 convolution with stride 2 into 16 tokens of dimension 64, processed by two Transformer encoder layers with four heads and flattened into a linear classifier. The feed-forward width was reduced from the PyTorch default of 2,048, which the original used for a 2,381-feature input, to 256. Calibrated class probabilities use a temperature T fitted on validation logits o by minimising the negative log-likelihood (Equation 7) (Guo et al., 2017).

$$p_k(x) = \exp(o_k/T) / \sum_{j=1}^{K} \exp(o_j/T) \quad (7)$$

Model	Configuration
Autoencoder (GENSHIELD, revised)	d–64–32–b–32–64–d, ReLU; Adam (lr 0.001), batch 256, ≤150 epochs, early stopping on benign validation MSE (patience 12); thresholds $\tau_{0.95}$, $\tau_{0.99}$ or the original 0.15
Isolation Forest	300 trees, fitted on benign training data, threshold at the 99th percentile of benign validation scores (Liu et al.,2008)
Logistic regression	L2, C = 1, up to 3,000–4,000 iterations, log-standard inputs (Pedregosa et al.,2011)
Random forest	300 trees, default depth, raw inputs
XGBoost	300 trees, depth 6, learning rate 0.1, subsample 0.9, column sample 0.9, histogram method (Chen and Guestrin, 2016)
MLP	Hidden layers 128–64, ReLU, Adam, early stopping on 10% internal validation
Grid ViT (GENSHIELD)	8 × 8 grid, 2 × 2 patches → 16 tokens × 64; 2 encoder layers, 4 heads, feed-forward 256, dropout 0.1; AdamW (lr 0.002, weight decay 0.0001), batch 256, ≤60 epochs, patience 8; temperature scaling

Table 4. Model configurations (identical across seeds; no test-set tuning)

2.6. Generative AI Integration and Security Test

Six fields are called in the prototype to make Gemini available through the google-generativeai library, along with requesting an HTML advisory. As there were two integrations compared, but no call was made to the LLM, the security properties under test are found at the code surrounding the integration. Original integration simply adds the raw file name to an f-string prompt and returns the raw LLM's HTML as is for integration. The file name is normalised (to Unicode NFKC), control and format characters are removed, all characters outside a small allow-list are removed and replaced with a blank space, a JSON reply with a fixed template with HMTL escaping returns a block of reassurance, a block of same action 3 warning, and a block of the same action 3 reply, and otherwise passes the HTML through an allow-list sanitiser (bleach 6.4.0, tags h4, ul, li, b, strong, em, p, div, br, no attributes), with a fixed template providing a deterministic fallback advisory if nothing usable is found. No fine tuning or training was done on LLM. No endpoint of the model was used and the computational cost of the LLM call itself was not measured, just the guard's own cost. The test corpus, written by the authors, contained 38 file names (ordinary names; names containing HTML, quotes, line breaks, null or bidirectional-override characters, homoglyphs, template or injection strings; instruction-like phrases such as "ignore previous instructions"; and over-long names), 40 malicious LLM outputs covering script elements, event-handler attributes, javascript: and data: URLs, frames, forms, meta refresh, SVG and MathML payloads and obfuscated tags, and 20 benign advisories that use the permitted formatting. The output was rendered using an HTML parsing tool that logs potential risky DOM elements, event handler attributes, and URL's containing scripts.

2.7. Training, Validation and Additional Protocols

All partitions used 60% training, 20% validation and 20% test data, repeated with seeds 0–4. MLRD was split by stratified random sampling after de-duplication. CIC-MalMem-2022 was split by sample: all dumps that share a malware hash were assigned to the same partition, groups were shuffled separately within each family, and benign records, which carry no sample identifier, were split at record level. Validation data were used only for early stopping, threshold selection and temperature fitting; no hyperparameters were tuned on test data, and the configurations in Table 4 were fixed in advance from common defaults. Scalers were fitted on training data only. Spyware and trojan records were never used for training and served as an out-of-scope test set. Two leakage experiments repeated the analysis with duplicates retained in MLRD and with random record-level splits for family classification. For open-set testing, each ransomware family was held out in turn from training (seeds 0–2), and the maximum calibrated probability was used as a confidence score (Hendrycks and Gimpel, 2017), with $s(x) = 1 - \max_k p_k(x)$ as the "unknown" score.

Three further protocols complete the design. Robustness was tested with a feature-space header-mimicry attack on MLRD: for every ransomware test record, either the three most important manipulable fields (by XGBoost importance, chosen from MajorImageVersion, MajorLinkerVersion, MinorLinkerVersion, DebugSize, DebugRVA, DllCharacteristics

and SizeOfStackReserve) or all seven were replaced by the values of a randomly drawn benign training record, and recall was re-measured with unchanged models and thresholds. These fields can be edited by an attacker without changing program behaviour, but the attack is a feature-space approximation and was not realised as modified binaries. The end-to-end pipeline was evaluated on CIC-MalMem-2022 by chaining the autoencoder ($\tau_{0.99}$), the temperature-scaled grid ViT and the decision rule of Equation 1 on benign, ransomware and spyware/trojan test records. Explanations were computed with TreeSHAP for XGBoost detection and family models (Lundberg et al., 2020) and, for the autoencoder, as each feature's share of the reconstruction error of flagged ransomware records.

2.8. Baselines and Ablation

Baselines were chosen to answer specific questions. Isolation Forest (Liu et al, 2008) is an unsupervised baseline trained under the same benign-only conditions as the autoencoder. Logistic regression, random forest, XGBoost and an MLP are supervised references that use labels, so they indicate the performance available when labelled ransomware exists; random forest and XGBoost also reproduce the strongest models of (Barnes and Ghafarian, 2026). For family attribution, the grid ViT is compared with the same four supervised models. All baselines use the same splits, features and test sets as the GENSHIELD models.

Tunable thresholds are applied per sample at the ablation stage, along with the following remaining settings: (A1) simply runs the original autoencoder with a fixed threshold of 0.15; (A2) the per-sample scaling is used in conjunction with a threshold, which is calibrated at the 99th percentile; (A3) the log-standard scaling is used, with a fixed threshold of 0.15; (A4) the log-standard scaling is used, with a threshold that is calibrated at the 95th percentile; (A5) the log-standard scaling is used, with a threshold that is calibrated at the 99th percentile as these are used in the decision module. ViT outputs are compared for family attribution, before and after they are calibrated and temperature-scaled. The original and guarded integrations are used on the same inputs for the GenAI layer. A fusion of the AUTOENCODER and supervised scores was not evaluated and is not reported.

2.9. Evaluation Metrics and Statistical Validation

Ransomware is viewed as the positive class. With true and false positives and negatives (TP, FP, TN, FN), we report accuracy, precision, recall, F1-score (Equation 8), false positive rate, false negative rate (Equation 9), specificity (1 − FPR) and the MCC (Equation 10). ROC-AUC and PR-AUC are used to capture the threshold-independent nature of the ranking quality. Why both: if a sample of a ransomware is prevented from being removed, it can encrypt files; if it is allowed, it can result in frequent false alarms, leading to a loss of trust; if classes are not well-balanced, it may be redundant. There are two indices used to evaluate family attribution with accuracy, macro-F1 (Equation 12), and weighted F1 and expected calibration error (ECE) (Equation 13), which compare the confidence and accuracy within M = 15 confidence bins Bm. Parameters that are measured include single-sample latency (average of 500 calls), batch latency and batch model size.

$$\text{Precision} = TP/(TP + FP), \quad \text{Recall} = TP/(TP + FN), \quad F1 = 2\cdot\text{Precision}\cdot\text{Recall}/(\text{Precision} + \text{Recall}) \tag{8}$$

$$FPR = FP/(FP + TN), \quad FNR = FN/(FN + TP) \tag{9}$$

$$MCC = (TP\cdot TN - FP\cdot FN) / \sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)} \tag{10}$$

$$ECE = \Sigma_{m=1}^{M} (|B_m|/n) \cdot |acc(B_m) - conf(B_m)| \tag{11}$$

Values presented are the mean of 5 seeds with 95% confidence interval calculated using the t distribution (and 4 degrees of freedom). Paired t-tests are used to test differences between paired configurations means across seeds. For seed 0, the test statistic for McNemar is the continuity-corrected chi-square statistic if at least 25 pairs are erroneous in at least one of the test sets, else an exact binomial test; the test results in an approximation of the null hypothesis of no difference. The number of seeds tested is limited (five), and effect sizes and CIs are emphasized rather than p-values.

2.10. Experimental Environment, Reproducibility and Ethics

The experiments were performed on Ubuntu 24.04.5 LTS with an Intel Xeon processor at 2.80 GHz (two virtual processors) and 8 GB of RAM memory; there was no GPU. The software stack was Python 3.13.18, PyTorch 2.15.0 (CPU execution; Paszke et al, 2019), scikit-learn 1.9.1 (Pedregosa et al., 2011), XGBoost 3.4.1, SHAP 0.54.1, bleach 6.4.0, NumPy 2.5.3, pandas 3.0.5 and SciPy 1.18.1. There was no use of the original prototype hardware (Intel Core i5, 3 GHz, 4 GB RAM) to run these experiments.

All the experiment code, fixed seeds, configuration values and the digest of the data sets are available as supplementary material; another researcher can be able to retype all the tables shown here using the publicly available data. There are no malware binaries downloaded or executed, only derived features are included in both datasets. No one used the original prototype source code which had a hard-coded LLM API key in it, and which should not be kept or revoked. The adversarial input corpora for the evaluation of GenAI were built by the authors, and intentionally avoided containing harmful or malicious proof-of-concept strings. After deployed, uploaded files may have personal data in them, so they should be deleted after analysis; minimalize names of files sent to an external LLM and any auto-response that impacts a host must have a human sign off before execution.

III. RESULTS AND DISCUSSION

3.1. Detection Performance

All values in this section are means over five seeds with 95% confidence intervals, unless a single seed is stated. Each MLRD test partition contained 6,448 records (2,135 ransomware and 4,313 benign). Each grouped CIC-MalMem-2022 test partition contained 7,775–7,813 records, of which 1,915–1,953 were ransomware dumps; group sizes differ, so

partition sizes vary slightly by seed. A further 19,507 spyware and trojan records served as an out-of-scope test set. In the tables, an asterisk marks supervised models that were trained with ransomware labels; the autoencoder and Isolation Forest were trained on benign data only.

Table 5 and Fig. 4 report detection on both channels. The original autoencoder configuration (per-sample scaling, $\tau = 0.15$) almost never raised an alarm. On MLRD it detected one of 2,135 ransomware files in seed 0 (recall 0.001 ± 0.001), and its ROC-AUC of 0.544 ± 0.058 is close to chance. On CIC-MalMem-2022 its recall was 0.097 ± 0.028 although its ROC-AUC was 0.969 ± 0.003, which shows that the fixed threshold, not only the scaling, was mismatched to the error distribution.

Data	Model	Precision	Recall	F1	FPR	MCC	ROC-AUC
MLRD	AE, original (per-sample, $\tau = 0.15$)	0.733 ± 0.540	0.001 ± 0.001	0.001 ± 0.001	0.000 ± 0.000	0.018 ± 0.014	0.544 ± 0.058
	AE, GENSHIELD (log-standard, $\tau_{0.99}$)	0.952 ± 0.019	0.395 ± 0.026	0.558 ± 0.026	0.010 ± 0.004	0.526 ± 0.021	0.954 ± 0.007
	Isolation Forest ($\tau_{0.99}$)	0.887 ± 0.014	0.126 ± 0.028	0.221 ± 0.043	0.008 ± 0.002	0.262 ± 0.031	0.935 ± 0.010
	Logistic regression*	0.936 ± 0.003	0.934 ± 0.008	0.935 ± 0.003	0.032 ± 0.002	0.903 ± 0.005	0.989 ± 0.001
	Random forest*	0.987 ± 0.002	0.991 ± 0.003	0.989 ± 0.002	0.007 ± 0.001	0.983 ± 0.002	0.999 ± 0.000
	XGBoost*	0.987 ± 0.002	0.990 ± 0.004	0.988 ± 0.002	0.007 ± 0.001	0.983 ± 0.003	0.999 ± 0.000
	MLP*	0.979 ± 0.003	0.984 ± 0.005	0.982 ± 0.003	0.010 ± 0.001	0.973 ± 0.005	0.998 ± 0.001
MalMem	AE, original (per-sample, $\tau = 0.15$)	1.000 ± 0.000	0.097 ± 0.028	0.177 ± 0.046	0.000 ± 0.000	0.272 ± 0.040	0.969 ± 0.003
	AE, GENSHIELD (log-standard, $\tau_{0.99}$)	0.972 ± 0.005	1.000 ± 0.000	0.986 ± 0.003	0.009 ± 0.002	0.981 ± 0.003	1.000 ± 0.000
	Isolation Forest ($\tau_{0.99}$)	0.902 ± 0.068	0.360 ± 0.277	0.489 ± 0.289	0.010 ± 0.001	0.493 ± 0.246	0.988 ± 0.005
	Logistic regression*	0.999 ± 0.001	0.999 ± 0.001	0.999 ± 0.001	0.000 ± 0.000	0.999 ± 0.001	1.000 ± 0.000
	Random forest*	1.000 ± 0.000	1.000 ± 0.001	1.000 ± 0.000	0.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
	XGBoost*	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
	MLP*	0.999 ± 0.001	0.998 ± 0.003	0.999 ± 0.001	0.000 ± 0.000	0.998 ± 0.002	1.000 ± 0.000

*Table 5. Ransomware detection on held-out test data (mean ± 95% CI over five seeds; * = supervised, trained with ransomware labels)*

The precision of the original autoencoder on MLRD has a wide interval because it raised between zero and a few alarms per seed. Values shown as 1.000 or 0.000 are rounded to three decimals

With log-standard scaling and the $\tau 0.99$ threshold, the autoencoder detected every ransomware dump in all five CIC-MalMem-2022 test partitions (recall 1.000) at an FPR of 0.009 ± 0.002, giving an F1-score of 0.986 ± 0.003 and an MCC of 0.981 ± 0.003. The same configuration was accurate on the static MLRD channel (0.952 ± 0.019) but detected only 39.5% of ransomware files at an FPR of 1.0%. ROC-AUC (0.954 ± 0.007) and PR-AUC (0.902) demonstrate that ransomware headers tend to rank higher than benign ones, but most do not stand out enough to pass a 1% false-alarm threshold. Isolation Forest, trained under the same benign-only conditions, was weaker and, on CIC-MalMem-2022, unstable across seeds (F1 0.489 ± 0.289).

The supervised models set the upper reference: random forest and XGBoost reached F1-scores of 0.989 and 0.988 on de-duplicated MLRD, and all four exceeded 0.998 on CIC-MalMem-2022. The out-of-scope test shows what the memory channel actually separates. The GENSHIELD autoencoder flagged 100% of the 19,507 spyware and trojan records, and the supervised ransomware-versus-benign models labelled 98.2–99.3% of them as ransomware. The memory features therefore distinguish malware from benign activity very well, but neither approach is specific to ransomware on this channel.

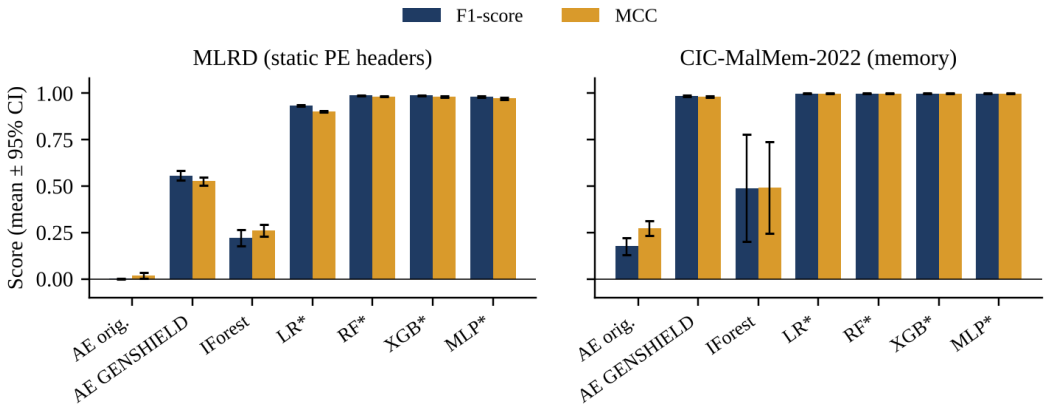


*Fig. 4. Detection F1-score and MCC on the static (MLRD) and memory (CIC-MalMem-2022) channels; error bars are 95% confidence intervals over five seeds. * = supervised models trained with ransomware labels.*

3.2. Comparison with Baselines and Statistical Significance

Relative to the original configuration, the revised autoencoder improved the F1-score by 0.556 on MLRD (paired t = 59.8, p < 0.001) and by 0.809 on CIC-MalMem-2022 (t = 48.6, p < 0.001). McNemar's test on seed 0 agreed: on MLRD, 908 test records were classified correctly only by the revised model and 65 only by the original; on CIC-MalMem-2022 the counts were 1,745 and 67 (p < 0.001 in both cases). Hypothesis H1 is therefore supported. The revised autoencoder also exceeded Isolation Forest in F1-score (by 0.337 on MLRD, p < 0.001, and by 0.497 on CIC-MalMem-2022, p = 0.009), while the difference in ROC-AUC was small (0.019, p = 0.025, and 0.012, p = 0.002). XGBoost exceeded the autoencoder by 0.431 F1 on MLRD (p < 0.001) and by 0.014 on CIC-MalMem-2022 (p < 0.001). On CIC-MalMem-2022, the McNemar counts (67 records correct only for XGBoost and 1 only for the autoencoder) show that the remaining gap consists of the autoencoder's false alarms, not missed ransomware. XGBoost and random forest did not differ significantly (F1 difference −0.0003, p = 0.47; McNemar 6 versus 6, p = 1.0). With five seeds, non-significant differences should not be read as equivalence.

3.3. Ablation Study

Table 6 isolates the two revised components of the detector. Scaling was the dominant factor. Changing from per-sample to log-standard scaling while keeping the original threshold of 0.15 (A1 → A3) raised the F1-score from 0.001 to 0.763 on MLRD and from 0.177 to 0.970 on CIC-MalMem-2022. Calibration of the threshold with per-sample scaling (A2) did not improve much and the ROC-AUC on MLRD stayed at 0.544, because a threshold cannot recover ranking information removed by the scaling. Per-sample standardization scales each record by its own mean and spread, so that a few large header values (e.g. stack and debug sizes) dominate each vector, and differences in the remaining fields become very small. Fig. 5 shows the result. Under per-sample scaling, MLRD benign and ransomware errors overlap almost completely, and most CIC-MalMem-2022 ransomware errors fall just below τ0.99. Under log-standard scaling, the ransomware errors move above the threshold: completely for CIC-MalMem-2022 and partly for MLRD.

Configuration	Threshold	MLRD F1	MLRD FPR	MLRD FNR	MalMem F1	MalMem FPR	MalMem FNR
A1: per-sample scaling (original)	fixed 0.15	0.001 ± 0.001	0.000 ± 0.000	0.999 ± 0.001	0.177 ± 0.046	0.000 ± 0.000	0.903 ± 0.028
A2: per-sample scaling	$\tau_{0.99}$	0.009 ± 0.007	0.011 ± 0.003	0.995 ± 0.003	0.307 ± 0.039	0.008 ± 0.001	0.814 ± 0.028
A3:log-tandard scaling	fixed 0.15	0.763 ± 0.047	0.051 ± 0.007	0.317 ± 0.072	0.970 ± 0.034	0.001 ± 0.000	0.054 ± 0.064
A4:log-tandard scaling	$\tau_{0.95}$	0.762 ± 0.037	0.050 ± 0.006	0.322 ± 0.055	0.929 ± 0.006	0.050 ± 0.005	0.000 ± 0.000
A5:log-tandard scaling (GENSHIELD)	$\tau_{0.99}$	0.558 ± 0.026	0.010 ± 0.004	0.605 ± 0.026	0.986 ± 0.003	0.009 ± 0.002	0.000 ± 0.000

Table 6. Ablation of the GENSHIELD autoencoder detector (mean ± 95% CI over five seeds)

Threshold choice under log-standard scaling is a trade-off that depends on the channel. On MLRD, $\tau_{0.95}$ halves the miss rate of $\tau_{0.99}$ (FNR 0.322 versus 0.605) but raises the FPR from 1.0% to 5.0%. On CIC-MalMem-2022, $\tau_{0.99}$ is better on both counts. The fixed threshold of 0.15 happened to lie close to the MLRD $\tau_{0.95}$ (0.140–0.183 across seeds), which explains why A3 and A4 are almost identical there, but on CIC-MalMem-2022 it missed 5.4% of ransomware on average and up to 12% in individual seeds. The calibrated $\tau_{0.99}$ was 0.348–0.488 on MLRD and 0.029–0.045 on CIC-MalMem-2022, a tenfold

difference. A threshold expressed in error units therefore cannot be transferred between data sources, whereas a quantile of benign validation error can. With per-sample scaling, the CIC-MalMem-2022 autoencoder reached its best validation loss after only 3–14 epochs, because the benign error was already very small (median 1.2×10^{-4} in seed 0).

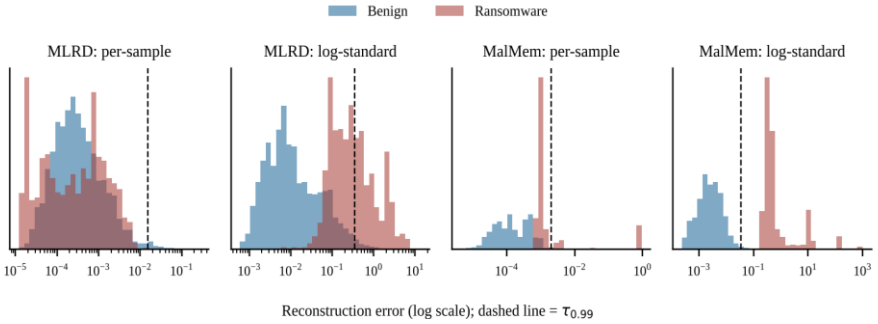


Fig. 5. *Reconstruction-error distributions of benign and ransomware test records (seed 0) under the original per-sample scaling and the revised log-standard scaling; bar heights are the share of each class per bin. The dashed line marks the benign validation $\tau_{0.99}$. Errors above 10^3 (0.6% of CIC-MalMem-2022 ransomware under log-standard scaling) are placed in the last bin.*

3.4. Effect of Data Leakage

When MLRD duplicates were retained, 51.5% ± 0.3% of the records in each test partition had an exact duplicate in the training partition. With this split, random forest showed an accuracy of 0.9965 ± 0.0009 (Table 7), which is in line with the 0.997 reported for this dataset by (Barnes and Ghafarian, 2026). After de-duplication accuracy decreased to 0.9926 ± 0.0011 (paired t = 24.2, p < 0.001) and FNR increased from 0.0026 to 0.0089 (p = 0.005). The change in accuracy is small but the miss rate, which matters the most for ransomware, more than tripled and XGBoost did the same. The autoencoder has never seen ransomware , so it cannot memorize duplicates . Duplicates change the test composition (56.6 % ransomware with duplicates vs 33.1 % without) , which is why the metrics change .

Model	Split	Accuracy	F1	FNR	FPR	MCC
Random forest*	duplicates retained	0.9965 ± 0.0009	0.9969 ± 0.0008	0.0026 ± 0.0006	0.0046 ± 0.0015	0.9929 ± 0.0018
	de-duplicated	0.9926 ± 0.0011	0.9888 ± 0.0016	0.0089 ± 0.0035	0.0067 ± 0.0009	0.9833 ± 0.0024
XGBoost*	duplicates retained	0.9961 ± 0.0009	0.9966 ± 0.0008	0.0030 ± 0.0004	0.0051 ± 0.0020	0.9921 ± 0.0018
	de-duplicated	0.9924 ± 0.0013	0.9885 ± 0.0020	0.0097 ± 0.0039	0.0066 ± 0.0008	0.9828 ± 0.0030
AE, GENSHIELD ($\tau_{0.99}$)	duplicates retained	0.6102 ± 0.0345	0.4787 ± 0.0696	0.6817 ± 0.0612	0.0091 ± 0.0015	0.3950 ± 0.0488
	de-duplicated	0.7930 ± 0.0081	0.5577 ± 0.0256	0.6054 ± 0.0264	0.0098 ± 0.0044	0.5262 ± 0.0207

Table 7. Effect of duplicate records in MLRD (mean ± 95% CI over five seeds)

3.5. Family Attribution and Calibration

Family attribution was evaluated on the 9,791 ransomware dumps of five families (176–229 samples and 1,717–2,128 dumps per family). Under the sample-grouped split, all models were well above the chance level of 0.20 macro-F1 but far from reliable (Table 8). XGBoost and random forest reached macro-F1 scores of 0.447 ± 0.036 and 0.443 ± 0.038, and the grid ViT reached 0.335 ± 0.034, which is 0.113 lower than XGBoost (paired t = 5.12, p = 0.007). The ViT reached its best validation loss after 5–15 epochs, and in seed 0 it assigned 1,113 of the 1,965 test dumps (57%) to one family, Conti. Pysa was the hardest family for every model (per-class F1 0.178 ± 0.070 for the ViT and 0.327 ± 0.065 for XGBoost) and Shade the easiest (0.412 ± 0.083 and 0.546 ± 0.047). Errors were spread over all families rather than concentrated in one pair (Fig. 6b).

Table 8. Ransomware family attribution on CIC-MalMem-2022 (five families; mean ± 95% CI over five seeds)

Model	Accuracy (grouped)	Macro-F1 (grouped)	ECE (grouped)	Macro-F1 (random split)	Inflation (random − grouped)
Grid ViT, uncalibrated	0.358 ± 0.027	0.335 ± 0.034	0.073 ± 0.030	0.462 ± 0.020	+0.128
Grid ViT, temperature-scaled	0.358 ± 0.027	0.335 ± 0.034	0.047 ± 0.019	0.462 ± 0.020	+0.128
Logistic	0.361 ± 0.010	0.347 ± 0.015	0.046 ± 0.015	0.372 ± 0.004	+0.025

regression					
Random forest	0.452 ± 0.038	0.443 ± 0.038	0.109 ± 0.043	0.548 ± 0.007	+0.104
XGBoost	0.457 ± 0.037	0.447 ± 0.036	0.157 ± 0.037	0.567 ± 0.008	+0.120
MLP	0.415 ± 0.019	0.399 ± 0.021	0.087 ± 0.048	0.446 ± 0.003	+0.048

Chance level for five balanced families is 0.20. Temperature scaling changes confidence values but not the predicted class, so accuracy and macro-F1 are identical for both ViT rows. All inflation values are significant (paired t-tests, $p \leq 0.013$).

The Macro-F1 was inflated for all models when using the random record-level splits that allow dumps of the same sample in both the training and test data: 0.128 for the ViT ($p = 0.002$), 0.120 for XGBoost ($p < 0.001$), 0.104 for random forest ($p = 0.002$), 0.048 for the MLP ($p = 0.002$) and 0.025 for logistic regression ($p = 0.013$). Together with the MLRD result, this supports hypothesis H2: both duplicate records and related records from one sample make results look better than they are, and the more flexible the model, the larger the effect. Temperature scaling ($T = 1.08–1.66$) reduced the ViT's ECE from 0.073 to 0.047, so that its mean confidence (0.362) closely matched its accuracy (0.358); with five seeds this reduction was not statistically significant ($p = 0.058$). XGBoost was the most accurate but also the most overconfident model (mean confidence 0.613 against an accuracy of 0.457; ECE 0.157).

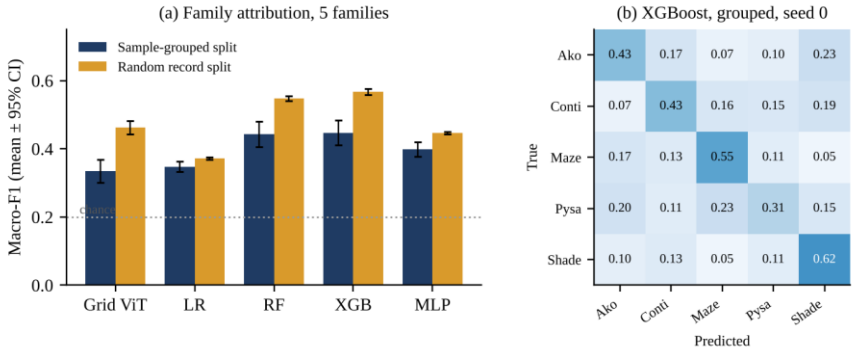


Fig. 6. *(a) Family-attribution macro-F1 under sample-grouped and random record-level splits (mean ± 95% CI over five seeds); the dotted line marks chance. (b) Row-normalised confusion matrix of XGBoost under the grouped split, seed 0.*

3.6. Open-Set Behaviour

In the leave-one-family-out test, each family was removed from training in turn and its dumps were presented to models trained on the other four (Table 9). The maximum calibrated probability of the ViT did not separate unseen from seen families: the AUROC of the "unknown" score was 0.501 ± 0.034, which is chance level. The ViT was rarely confident about anything, so only 2.5% of unseen-family dumps, and 2.2% of seen-family dumps, received a confidence above 0.80. XGBoost behaved differently: its AUROC was also close to chance (0.538 ± 0.027), but 25.9% ± 3.6% of the dumps from families it had never seen received a confidence above 0.80 and would therefore have been reported with a wrong family name. Hypothesis H3 is therefore supported for XGBoost but not for the calibrated ViT, whose low confidence avoids confident errors at the cost of rarely naming a family. For both models, confidence alone could not detect a new family.

Held-out family	**ViT AUROC**	**ViT: unseen with conf. > 0.8**	**ViT: seen with conf. > 0.8**	**XGBoost AUROC**	**XGBoost: unseen with conf. > 0.8**
Ako	0.458 ± 0.123	0.036 ± 0.068	0.025 ± 0.030	0.523 ± 0.023	0.309 ± 0.022
Conti	0.529 ± 0.157	0.023 ± 0.013	0.030 ± 0.035	0.561 ± 0.169	0.240 ± 0.172
Maze	0.536 ± 0.021	0.001 ± 0.004	0.016 ± 0.035	0.542 ± 0.162	0.234 ± 0.207
Pysa	0.518 ± 0.223	0.024 ± 0.059	0.032 ± 0.099	0.549 ± 0.186	0.248 ± 0.275
Shade	0.462 ± 0.144	0.043 ± 0.073	0.007 ± 0.011	0.513 ± 0.034	0.265 ± 0.063
All (15 runs)	0.501 ± 0.034	0.025 ± 0.013	0.022 ± 0.011	0.538 ± 0.027	0.259 ± 0.036

Table 9. *Leave-one-family-out open-set test (seeds 0–2; mean ± 95% CI). AUROC measures how well 1 − max p separates the held-out family from test dumps of the four training families*

3.7. End-to-End Pipeline Outcomes

Chaining the autoencoder, the temperature-scaled ViT ($T = 1.42–1.78$) and the decision rule of Equation 1 gave the threat levels in Fig. 7a. Of the benign test records, 99.1% ± 0.2% were rated LOW, 0.8% MODERATE and 0.1% HIGH. All ransomware dumps were flagged, and 94.7% ± 6.7% were rated HIGH, almost entirely because their reconstruction error exceeded $\tau_{0.999}$. The calibrated family confidence was 0.80 for only 1.0% of flagged ransomware, so a family name would be shown for about one alarm in a hundred. The most probable family was correct 35.3% ± 3.8% of flagged dumps. Further, all spyware and trojan records were flagged and 94.1% ± 7.2% were rated HIGH. The pipeline is thus a trustworthy malware triage stage with a conservative family-naming rule. However, its threat level does not differentiate ransomware from other malware.

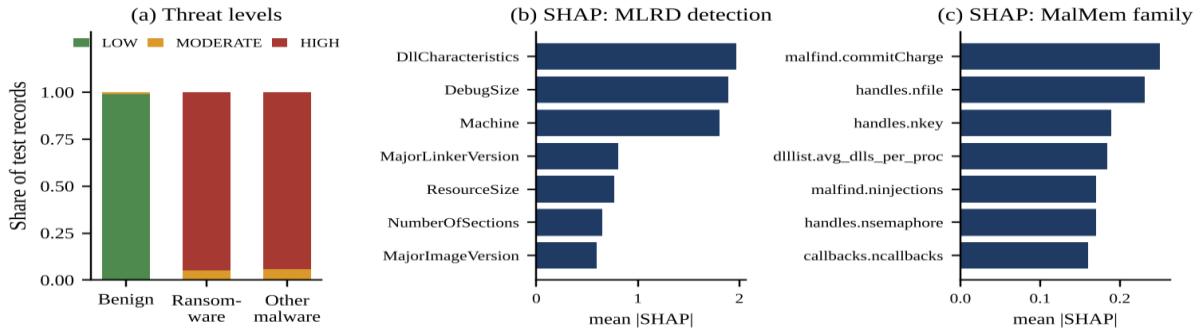


Fig. 7. *(a) Threat levels assigned by the end-to-end pipeline on CIC-MalMem-2022 test records (mean over five seeds). (b) Mean absolute TreeSHAP values of the XGBoost detector on MLRD. (c) Mean absolute TreeSHAP values of the XGBoost family model on CIC-MalMem-2022 (seed 0; seven highest features shown).*

3.8. Robustness to Header Manipulation

Table 10 reports recall on MLRD ransomware test records after the header-mimicry attack. The three fields most important to XGBoost were DebugSize, DllCharacteristics and DebugRVA (MajorImageVersion instead of DebugRVA in one seed). Copying these three fields from benign files reduced the recall of XGBoost from 0.990 to 0.633 and of random forest from 0.991 to 0.619. Copying all seven manipulable fields reduced their recall to 0.155 and 0.117, and logistic regression and the MLP fell to 0.194 and 0.236. The autoencoder never fell below its clean recall: it was 0.577 ± 0.204 under the three-field attack, because mixing benign and ransomware header values creates combinations that are unusual for benign files, and 0.390 ± 0.054 under the seven-field attack, which is higher than that of every supervised model. This advantage must be read together with the autoencoder's low clean recall, and the attack was a feature-space simulation that did not target the autoencoder itself.

Table 10. Robustness and efficiency of the detectors on MLRD (recall: mean ± 95% CI over five seeds; latency and size: seed 0, CPU only)

Model	Recall, clean	Recall, top-3 fields	Recall, all 7 fields	Latency, single (ms)	Batch (µs/sample)	Model size
AE, GENSHIELD ($\tau_{0.99}$)	0.395 ± 0.026	0.577 ± 0.204	0.390 ± 0.054	0.115	0.53	29.8 KB (6,483 params)
Logistic regression*	0.934 ± 0.008	0.266 ± 0.020	0.194 ± 0.006	0.123	0.28	0.8 KB
Random forest*	0.991 ± 0.003	0.619 ± 0.100	0.117 ± 0.006	77.2	18.1	16.4 MB
XGBoost*	0.990 ± 0.004	0.633 ± 0.061	0.155 ± 0.022	0.306	4.17	0.67 MB
MLP*	0.984 ± 0.005	0.598 ± 0.052	0.236 ± 0.051	0.122	0.87	0.13 MB

Recall of supervised models uses a probability threshold of 0.5; the autoencoder uses its benign $\tau_{0.99}$. Isolation Forest required 23.7 ms per single sample and 17.8 µs per sample in batches.

3.9. Computational Efficiency and Explainability

All models ran on two CPU cores without a GPU (Table 10). The autoencoder needed 0.115 ms per single MLRD sample and 0.178 ms per CIC-MalMem-2022 sample, with 6,483 and 11,841 parameters (29.8 KB and 51.2 KB). XGBoost was similarly fast (0.31–0.37 ms) and small (0.25–0.67 MB). Random forest needed 74–77 ms per single sample, probably because of per-call parallelisation overhead across its 300 trees, and occupied 16.4 MB on MLRD. The grid ViT (105,413 parameters) needed 0.71 ms per single sample and trained in 35–60 s per seed. At these costs, detector choice on a two-core machine need not be driven by CPU time.

TreeSHAP explanations agreed with prior findings in some respects and explained our robustness results in others (Fig. 7b–c). For MLRD detection, DllCharacteristics, DebugSize and Machine had the largest mean absolute SHAP values (1.97, 1.89 and 1.81), in line with (Barnes and Ghafarian, 2026). Bitcoin Addresses, the only explicitly ransomware-related field, contributed least (0.013). Two of the three most influential fields are editable fields used in the mimicry attack, which explains why the supervised models were fragile. For CIC-MalMem-2022 detection, service counts (svcscan.nservices and svcscan.shared_process_services) and the average number of handles per process dominated. For family attribution, malfind.commitCharge, handles.nfile and handles.nkey ranked highest, but contributions were small and similar (0.12–0.25), which is consistent with weak family separability. The autoencoder's own explanation, the share of reconstruction error per feature, pointed to code injection for flagged ransomware: malfind.uniqueInjections accounted for 17.7% of the error, followed by callbacks.ncallbacks (9.6%), malfind.commitCharge (9.3%) and svcscan.kernel_drivers (7.2%). These shares are the "most influential features" that the decision module can pass to the advisory.

3.10. Security Evaluation of the Generative AI Integration

Table 11 compares the original and guarded GenAI integrations on identical inputs. In the original integration, every file name reached the prompt verbatim, 14 of the 38 names (36.8%) contained characters that can break out of the prompt or HTML context, and 38 of the 40 malicious LLM outputs (95%) produced active content when rendered. In the guarded integration, no sanitised name contained break-out characters or exceeded 64 characters, all prompts carried a valid JSON data block, and no malicious output produced active content; malicious text inside the JSON schema fields was rendered as escaped text. Legitimate advisories were not damaged: no benign output was flagged, and all permitted formatting tags and visible text were retained, although inline style attributes were removed. The guard cost 8.3 µs per prompt and 273 µs per output. Hypothesis H4 is therefore supported.

Table 11. Security test of the original and guarded GenAI integrations (38 file names, 40 malicious and 20 benign LLM outputs)

Measure	Original integration	Guarded integration
File name inserted into the prompt verbatim	38/38 (100%)	0/38 (sanitised)
Names containing prompt or HTML break-out characters	14/38 (36.8%)	0/38 (0%)
Maximum file-name length in the prompt	unbounded (304 characters in the corpus)	64 characters
Names containing instruction-like keywords†	10/38 (26.3%)	10/38 (26.3%)
Prompt fields passed as a valid JSON data block	not applicable	38/38
Malicious outputs producing active content when rendered	38/40 (95%); 53 hits	0/40 (0%); 0 hits
Malicious JSON fields producing active content	active	escaped, inactive
Benign outputs flagged as active	0/20	0/20
Permitted formatting tags and visible text retained	100%	100%
Added latency	none	8.3 µs per prompt; 273 µs per output

† Keyword match on ignore, system, instruction, safe or override; the match is deliberately broad (it also counts a path containing system32)

One weakness remains. Instruction-like phrases such as "ignore previous instructions" consist of ordinary characters, and all 10 names that contained instruction-like keywords (26.3%) still contained them after sanitisation. The guard limits what such text can do, because it is presented as data, the reply must follow a schema, and the reply cannot execute in the browser, but it does not remove the text. Whether a particular LLM would follow such instructions was not tested, because no LLM was called. For the same reason, the quality, factual accuracy and usefulness of LLM advisories were not measured in this study.

3.11. Demonstration of the Original Prototype

The original prototype was run with two inputs: the simulated test file mock_threat_46_Xorist_Sim.exe and the Python source file app.py (Figs. 8 and 9). The end-to-end workflow operated as designed: the file moved through the pipeline display, the result appeared on the dashboard and in a report window, an advisory was shown, and a text report could be downloaded (Fig. 3d). For the simulated Xorist file, the dashboard reported THREAT DETECTED, and the report window listed MALWARE / THREAT DETECTED, HIGH RISK / CRITICAL THREAT, the family Ransomware.Xorist, a reconstruction loss of 1.69526 and a ViT confidence of 91.01% (Fig. 8a–b). For app.py, the dashboard showed SYSTEM SAFE, and the report listed SAFE FILE, LOW RISK / SAFE, the family None (Clean Document/Text), a loss of 0 and a confidence of 100% (Fig. 8c–d).



Fig. 8. Results shown by the original prototype.

(a) Dashboard and (b) analysis report after uploading the simulated test file mock_threat_46_Xorist_Sim.exe. (c) Dashboard and (d) analysis report after uploading the non-PE file app.py. The family and confidence in (a)–(b) come from

a file-name rule, and the values in (c)–(d) are fixed responses for non-PE files.

These outputs do not indicate the accuracy with which detection and attribution is carried out. However, in the case of the Xorist file, the loss is 1.69526 but, due to the unit variance normalisation of each feature vector, it's actually a measure of the reconstruction, not ransomware content. The file-name rule gave the family label, no the ViT, and therefore, also the HIGH grade rating follows from the (high) level of file-name confidence (91.01%) – and not directly from the ViT. Two display only states: QUARANTINED and BLOCKED. Note: The QUARANTINED and BLOCKED labels on the dash board are only for display – no quarantine function. The result in the case of app.py is fixed - the SAFE status and zero loss are returned for all non-PE files, a script or a document containing mal-content will get the same result.

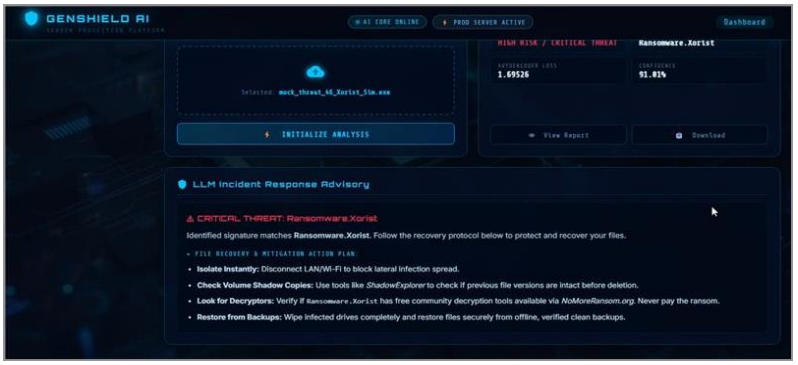


Fig. 9. Incident-response advisory displayed by the original prototype for the simulated Xorist test file (screenshot). Whether the text was generated by the LLM or by an interface template could not be established.

The Xorist file advisory (Fig. 9) suggests that the victim should disconnect the machine from the network, check Volume Shadow Copies before removal, try using NoMoreRansom.org to find an updated free decryptor and restore from safe offline backups. The safeguards below align with previously released ransomware response guidance from CISA et al. (2023). While the text used matches a string in the fallback text for its backend, that string wasn't available as a back-end code snippet and the front end code wasn't available, making it uncertain whether the text was generated by the LLM or by a template in the interface. Consequently, the demonstration proves a successful workflow but reveals three limitations of the original artefact,that is the name-based attribution, the lack of an analysis of non-PE inputs, and the verification of origin of literature/advisory material. These are included in the revisions being evaluated here (the calibrated thresholds, the file-name rule being only applied at the level of confidence reached after calibration and the guarded advisory) and extend to two deployment requirements: remove the file-name rule; analyse or explicitly deny file types not supported; and report protected file types as safe.

3.12. Synthesis of Findings

In the case of RQ1, the detection notion backgrounded by GENSHIELD AI - behaviour training and flagging of behaviour deviations - is effective on the memory-forensic features, though it comes after two engineering decisions. Most information that the autoencoder required was removed in per-sample standardisation, and the fixed threshold of 0.15 did not have a consistent significance across the different data sources. The architecture scaled correctly and at less than 1% FPR for each ransomware dump in CIC-MalMem-2022. It performed well on static PE headers (ROC-AUC 0.954) but it only found about 40% at the same false alarm cost. There are 15 header fields describing the nature of the construction of a file, and many ransomware binaries will be built in the same way as normal software, so an anomaly in one or more of the header fields isn't that meaningful as a behavioural indicator, but evidence from memory is.

The second is that the memory channel won't necessarily block ransomware from other malware; it will block malware from benign operations. All spyware and all trojans were identified by the autoencoder, while more than 99% of them were identified by the supervised binary models as ransomware. This reinforces the static result found in (Barnes and Ghafarian, 2026) who found a ransomware versus other malwares MCC of only 0.112. A 'probably malicious' alarm from either channel, and further evidence for the term ‘ransomware'.

Family attribution from the 52 memory features was weak once dumps of the same sample were kept together: the best grouped macro-F1 was 0.447 against a chance level of 0.20, and no model could recognise a family absent from training. A family name from these features is therefore a weak hint, and the rule of naming a family only above a calibrated confidence of 0.80 is essential; the calibrated ViT almost never reached that level, whereas the uncalibrated XGBoost would have named a wrong family for about a quarter of unseen-family samples. The ViT did not outperform tree ensembles, which is consistent with the observation that vision transformers lack the inductive biases of convolutional networks and need large training sets (Dosovitskiy et al, 2021); here the training set was small and the 8 × 8 grid of unrelated counters has no spatial meaning.

For the supervised models were accurate but fragile: copying seven editable header fields from benign files reduced their recall from about 0.99 to 0.12–0.24, in line with evidence that static detectors can be evaded by functionality-preserving changes (Demetrio et al, 2021). The autoencoder's recall did not fall, which suggests that benign-only and supervised models fail in different ways and could complement each other. The guard removed the two concrete risks found in the original code, prompt-context break-out and execution of LLM-generated HTML, at a cost below 0.3 ms per advisory.

3.13. Comparison with Previous Studies

Direct comparison only can be made with published numbers if they are on fact the same data and protocol. Lastly, on MLRD with duplicates being retained, our random forest gave an accuracy of 0.997, which matches the accuracy reported by (Barnes and Ghafarian, 2026). Our result adds that about half of such test records have exact duplicates in training, and that removing them more than triples the miss rate. Without any reporting of dumps being grouped by sample, (Hussain et al, 2024) achieved a detection accuracy of 99.99% and a family accuracy of 74.65% detecting CIC MalMem-2022. Our supervised detection results do agree with the first one, however the detection results of our grouped family is much lower. Since their family definition and models and split vary from ours, the difference can't be solely blamed on leakage; our random-split experiment finds that the difference is significant enough that it's caused by the split protocol. The same observations as done by (Arp et al, 2024), (Pendlebury et al, 2019), and (Kapoor and Narayanan, 2023) seem to hold true for these scenarios: duplicate or related records in partitioned tables can make security models appear more robust than they are. Here, calibration and open-set behavior (which we studied for ransomware both recently (Gazzan et al, 2024) and again recently (Cen et al, 2024b) were also important.

3.14. Role and Limits of Generative AI

GENAI does not provide a detection capability for ransomware, nor is it the purpose of this study to respond to that. It has the role of interpreting and providing an opinion on the structured output of the detector in an advisory manner that can be followed by a non-expert. Nevertheless, we showed that this role does not create new attack surface; this is significant, as LLM-integrated applications can be the victim of indirect prompt injection and unsafe output handling (Greshake et al.,2023; OWASP, 2025; Yao et al.,2024). The guard first sees the LLM as an untrusted text generator, requiring minimization of input, always considering it as data, requiring outputs to follow a schema, and forbidding anything generated from being executed in the browser. None of the LLMs were invoked to measure whether the advisories were helpful, correctly high or low (Huang et al.,2025) was measured. The true value of the GenAI layer to the users is, however, a hypothesis that would have to be confirmed through a user study coupled with expert rating of advisories. Deterministic fallback is an important element in the design, as the provider of the model (Google AI for Developers in 2026) indicated that the model in question will be unsupported soon.

3.15. Practical Implications

There are four implications for practitioners.There are four implications for practitioners. First, thresholds for anomaly detectors should be defined as quantiles of the benign validation error and should change over time whenever either the source data set or feature extraction algorithm varies. Second, memory-dump datasets ought to be clustered by sample, static datasets should be deduplicated, prior to deciding on using it to select a product or model. Third, a family label cannot be placed above a calibrated confidence and should explicitly be accompanied by an ‘unknown family' outcome - the response actions recommended by (CISA et al.,2023) depend on knowledge of the family. Fourth, a system should include a benign-only anomaly detector, a supervised scorer, and additional, run-time behavioural controls, in order to compensate for the common vulnerability of the strong detectors, otherwise alerts from the former will often be falsepositives. Combined detector was not tested in this study.

3.16. Limitations and Future Directions

Internal validity is limited because the original training data were undocumented and the original weights could not be applied to the public feature datasets; all evaluated models were retrained, so the quantitative results describe the GENSHIELD design, not the original artefact, which was examined only qualitatively on two inputs (Section 3.11). The ViT feed-forward width was reduced for CPU training, and hyperparameters were fixed defaults, so tuned models might perform better. Construct validity is limited because both datasets contain derived features, not binaries or live behaviour; the header attack was a feature-space simulation, and the memory channel was evaluated offline. Benign CIC-MalMem-2022 records have no sample identifiers, so some benign near-duplicates may remain across partitions. External validity is limited to two public datasets. CIC-MalMem-2022 contains only five ransomware families from one collection period, and the family composition and collection dates of MLRD are documented only at repository level; temporal drift was not tested (Pendlebury et al.,2019). There are only five seeds on which statistical conclusions are based. Last, the GenAI evaluation was not about the LLM but about the code that surrounds the LLM.

Further studies could further test the advisory layer with a fixed version of the LLM, expert correctness and usefulness ratings, elicited hallucination occurrences, and a controlled user study. It should also test a fused detector consisting of the benign-only autoencoder together with a supervised scorer against adaptive attacks realised as modified binaries, apply time-stamped data training with test-new, train-old splits (Pendlebury et al.,2019), examine family attribution with more informative behavioral traces and open-set methods, and employ the proposed quarantine and recovery functions with human approval in a remote setting (Nelson et al.,2025; CISA et al.,2023). The overall environment suggests that GENSHIELD AI can be considered as a tuned malware triage pipeline that features an explainable, protected advisory on top. It's memory-forensic detector is correct, static detector is one of the weaker first filters it has, family names are hints to be taken with a grain of salt, and its Generative AI piece can be incorporated without creating new attack vectors, but each ransomware-specific claim needs more behavioural evidence.

IV. CONCLUSION

This study shows that GENSHIELD AI can be useful for ransomware and malware triage when its results are tested with proper data handling. The revised autoencoder performed much better than the original design. On CIC-MalMem-2022, it detected 100% of ransomware dumps, with an F1-score of 0.986 and a false-positive rate of 0.9%. On the static MLRD data, however, recall was only 39.5%, although precision was 95.2%. This shows that PE-header information alone is not strong enough for reliable ransomware detection. The study also found that data leakage can make models look better than they really are. Removing duplicate records increased the random forest false-negative rate from 0.0026 to 0.0089. Family identification was also limited, with the ViT achieving a macro-F1 of only 0.335 under sample-grouped testing. Therefore, family predictions should be seen as supporting information, not final decisions. The guarded Generative AI layer effectively blocked unsafe active content with a processing cost of less than 0.3 ms. Overall GENSHIELD AI is best used as a calibrated malware triage system. Future work should test on real binaries, richer behavioral data, time-based testing, stronger open-set methods, fused detection models and expert evaluation of the LLM advisory to validate its practical value in the practice.

REFERENCES

1. Al-rimy, B.A.S., Maarof, M.A. and Shaid, S.Z.M. (2018) 'Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions', Computers & Security, 74, pp. 144–166. Available at: https://doi.org/10.1016/j.cose.2018.01.001.
2. Alam, M.T., Bhusal, D., Nguyen, L. and Rastogi, N. (2024) 'CTIBench: A benchmark for evaluating LLMs in cyber threat intelligence', in Advances in Neural Information Processing Systems 37 (NeurIPS 2024), Datasets and Benchmarks Track, pp. 50805–50825. Available at: https://doi.org/10.52202/079017-1607.
3. Anderson, H.S. and Roth, P. (2018) 'EMBER: An open dataset for training static PE malware machine learning models', arXiv preprint, arXiv:1804.04637 [not peer reviewed]. Available at: https://arxiv.org/abs/1804.04637.
4. Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L. and Rieck, K. (2024) 'Pitfalls in machine learning for computer security', Communications of the ACM, 67(11), pp. 104–112. Available at: https://doi.org/10.1145/3643456.
5. Ashwini, K. and Nagasundara, K.B. (2024) 'An intelligent ransomware attack detection and classification using dual vision transformer with Mantis Search Split Attention Network', Computers and Electrical Engineering, 119, 109509. Available at: https://doi.org/10.1016/j.compeleceng.2024.109509.
6. Barnes, G. and Ghafarian, A. (2026) 'Machine learning-based static ransomware detection using PE header features and SHAP interpretation', Journal of Cybersecurity and Privacy, 6(2), 58. Available at: https://doi.org/10.3390/jcp6020058.
7. Beaman, C., Barkworth, A., Akande, T.D., Hakak, S. and Khan, M.K. (2021) 'Ransomware: Recent advances, analysis, challenges and future research directions', Computers & Security, 111, 102490. Available at: https://doi.org/10.1016/j.cose.2021.102490.
8. Berrios Vasquez, S.I., Hermosilla Monckton, P.A., Leiva Muñoz, D.I. and Allende, H. (2025) 'Zero-day threat mitigation via deep learning in cloud environments', Applied Sciences, 15(14), 7885. Available at: https://doi.org/10.3390/app15147885.
9. Brown, T. et al. (2020) 'Language models are few-shot learners', in Advances in Neural Information Processing Systems 33 (NeurIPS 2020), pp. 1877–1901. Available at: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.
10. Capuano, N., Fenza, G., Loia, V. and Stanzione, C. (2022) 'Explainable artificial intelligence in cybersecurity: A survey', IEEE Access, 10, pp. 93575–93600. Available at: https://doi.org/10.1109/ACCESS.2022.3204171.
11. Carrier, T., Victor, P., Tekeoglu, A. and Lashkari, A. (2022) 'Detecting obfuscated malware using memory feature engineering', in Proceedings of the 8th International Conference on Information Systems Security and Privacy (ICISSP 2022). Setúbal: SciTePress, pp. 177–188. Available at: https://doi.org/10.5220/0010908200003120.
12. Cen, M., Deng, X., Jiang, F. and Doss, R. (2024b) 'Zero-Ran Sniff: A zero-day ransomware early detection method based on zero-shot learning', Computers & Security, 142, 103849. Available at: https://doi.org/10.1016/j.cose.2024.103849.
13. Cen, M., Jiang, F., Qin, X., Jiang, Q. and Doss, R. (2024a) 'Ransomware early detection: A survey', Computer Networks, 239, 110138. Available at: https://doi.org/10.1016/j.comnet.2023.110138.
14. Cevallos-Salas, D., Grijalva, F., Estrada-Jiménez, J., Benítez, D. and Andrade, R. (2024) 'Obfuscated privacy malware classifiers based on memory dumping analysis', IEEE Access, 12, pp. 17481–17498. Available at: https://doi.org/10.1109/ACCESS.2024.3358840.
15. Chainalysis (2025) 35% year-over-year decrease in ransomware payments, less than half of recorded incidents resulted in victim payments. Chainalysis Blog, 5 February. Available at: https://www.chainalysis.com/blog/crypto-crime-ransomware-victim-extortion-2025/ (Accessed: 1 November 2026).
16. Chandola, V., Banerjee, A. and Kumar, V. (2009) 'Anomaly detection: A survey', ACM Computing Surveys, 41(3), pp. 1–58. Available at: https://doi.org/10.1145/1541880.1541882.
17. Chen, T. and Guestrin, C. (2016) 'XGBoost: A scalable tree boosting system', in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, pp. 785–794. Available at: https://doi.org/10.1145/2939672.2939785.
18. CISA, MS-ISAC, NSA and FBI (2023) #StopRansomware guide. Washington, DC: Cybersecurity and Infrastructure Security Agency. Available at: https://www.cisa.gov/stopransomware/ransomware-guide (Accessed: 1 November 2026).
19. Continella, A., Guagnelli, A., Zingaro, G., De Pasquale, G., Barenghi, A., Zanero, S. and Maggi, F. (2016) 'ShieldFS: A self-healing, ransomware-aware filesystem', in Proceedings of the 32nd Annual Conference on Computer Security Applications (ACSAC '16). New York: ACM, pp. 336–347. Available at: https://doi.org/10.1145/2991079.2991110.
20. Davidian, M., Kiperberg, M. and Vanetik, N. (2024) 'Early ransomware detection with deep learning models', Future Internet, 16(8), 291. Available at: https://doi.org/10.3390/fi16080291.
21. Deldar, F. and Abadi, M. (2023) 'Deep learning for zero-day malware detection and classification: A survey', ACM Computing Surveys, 56(2), pp. 1–37. Available at: https://doi.org/10.1145/3605775.
22. Demetrio, L., Coull, S.E., Biggio, B., Lagorio, G., Armando, A. and Roli, F. (2021) 'Adversarial EXEmples: A survey and

experimental evaluation of practical attacks on machine learning for Windows malware detection', ACM Transactions on Privacy and Security, 24(4), pp. 1–31. Available at: https://doi.org/10.1145/3473039.

23. Dosovitskiy, A. et al. (2021) 'An image is worth 16x16 words: Transformers for image recognition at scale', in International Conference on Learning Representations (ICLR 2021). Available at: https://openreview.net/forum?id=YicbFdNTTy.
24. ENISA (2025) ENISA threat landscape 2025. Athens: European Union Agency for Cybersecurity. Available at: https://www.enisa.europa.eu/Publications/Enisa-Threat-Landscape-2025 (Accessed: 1 November 2026).
25. Ferdous, J., Islam, R., Mahboubi, A. and Islam, M.Z. (2024) 'AI-based ransomware detection: A comprehensive review', IEEE Access, 12, pp. 136666–136695. Available at: https://doi.org/10.1109/ACCESS.2024.3461965.
26. Fernando, D.W., Komninos, N. and Chen, T. (2020) 'A study on the evolution of ransomware detection using machine learning and deep learning techniques', IoT, 1(2), pp. 551–604. Available at: https://doi.org/10.3390/iot1020030.
27. Ferrag, M.A., Alwahedi, F., Battah, A., Cherif, B., Mechri, A., Tihanyi, N., Bisztray, T. and Debbah, M. (2025) 'Generative AI in cybersecurity: A comprehensive review of LLM applications and vulnerabilities', Internet of Things and Cyber-Physical Systems, 5, pp. 1–46. Available at: https://doi.org/10.1016/j.iotcps.2025.01.001.
28. Gazzan, M. and Sheldon, F.T. (2024) 'Novel ransomware detection exploiting uncertainty and calibration quality measures using deep learning', Information, 15(5), 262. Available at: https://doi.org/10.3390/info15050262.
29. Gemini Team, Google (2024) 'Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context', arXiv preprint, arXiv:2403.05530 [not peer reviewed]. Available at: https://arxiv.org/abs/2403.05530.
30. Ghourabi, A. and Chouaib, H. (2026) 'A ransomware detection approach based on LLM embedding and ensemble learning', Computers, Materials & Continua, 87(1), 98. Available at: https://doi.org/10.32604/cmc.2026.074505.
31. Gómez-Hernández, J.A., Álvarez-González, L. and García-Teodoro, P. (2018) 'R-Locker: Thwarting ransomware action through a honeyfile-based approach', Computers & Security, 73, pp. 389–398. Available at: https://doi.org/10.1016/j.cose.2017.11.019.
32. Google AI for Developers (2026) Gemini API: Deprecations. Last updated 22 October 2026. Available at: https://ai.google.dev/gemini-api/docs/deprecations (Accessed: 1 November 2026).
33. Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T. and Fritz, M. (2023) 'Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection', in Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec '23). New York: ACM, pp. 79–90. Available at: https://doi.org/10.1145/3605764.3623985.
34. Gulmez, S., Gorgulu Kakisim, A. and Sogukpinar, I. (2024) 'XRan: Explainable deep learning-based ransomware detection using dynamic analysis', Computers & Security, 139, 103703. Available at: https://doi.org/10.1016/j.cose.2024.103703.
35. Guo, C., Pleiss, G., Sun, Y. and Weinberger, K.Q. (2017) 'On calibration of modern neural networks', in Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, 70, pp. 1321–1330. Available at: https://proceedings.mlr.press/v70/guo17a.html.
36. Gupta, M., Akiri, C., Aryal, K., Parker, E. and Praharaj, L. (2023) 'From ChatGPT to ThreatGPT: Impact of generative AI in cybersecurity and privacy', IEEE Access, 11, pp. 80218–80245. Available at: https://doi.org/10.1109/ACCESS.2023.3300381.
37. Hendrycks, D. and Gimpel, K. (2017) 'A baseline for detecting misclassified and out-of-distribution examples in neural networks', in International Conference on Learning Representations (ICLR 2017). Available at: https://openreview.net/forum?id=Hkg4TI9xl.
38. Homayoun, S., Dehghantanha, A., Ahmadzadeh, M., Hashemi, S. and Khayami, R. (2020) 'Know abnormal, find evil: Frequent pattern mining for ransomware threat hunting and intelligence', IEEE Transactions on Emerging Topics in Computing, 8(2), pp. 341–351. Available at: https://doi.org/10.1109/TETC.2017.2756908.
39. Huang, L. et al. (2025) 'A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions', ACM Transactions on Information Systems, 43(2), pp. 1–55. Available at: https://doi.org/10.1145/3703155.
40. Hussain, A., Saadia, A., Alhussein, M., Gul, A. and Aurangzeb, K. (2024) 'Enhancing ransomware defense: Deep learning-based detection and family-wise classification of evolving threats', PeerJ Computer Science, 10, e2546. Available at: https://doi.org/10.7717/peerj-cs.2546.
41. Kapoor, S. and Narayanan, A. (2023) 'Leakage and the reproducibility crisis in machine-learning-based science', Patterns, 4(9), 100804. Available at: https://doi.org/10.1016/j.patter.2023.100804.
42. Kaur, R., Gabrijelčič, D. and Klobučar, T. (2023) 'Artificial intelligence for cybersecurity: Literature review and future research directions', Information Fusion, 97, 101804. Available at: https://doi.org/10.1016/j.inffus.2023.101804.
43. Kharraz, A., Arshad, S., Mulliner, C., Robertson, W. and Kirda, E. (2016) 'UNVEIL: A large-scale, automated approach to detecting ransomware', in Proceedings of the 25th USENIX Security Symposium. Austin, TX: USENIX Association. Available at: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/kharaz.
44. Kharraz, A., Robertson, W., Balzarotti, D., Bilge, L. and Kirda, E. (2015) 'Cutting the Gordian knot: A look under the hood of ransomware attacks', in Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA 2015). Lecture Notes in Computer Science, vol. 9148. Cham: Springer, pp. 3–24. Available at: https://doi.org/10.1007/978-3-319-20550-2_1.
45. Kolodenker, E., Koch, W., Stringhini, G. and Egele, M. (2017) 'PayBreak: Defense against cryptographic ransomware', in Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security (ASIA CCS '17). New York: ACM, pp. 599–611. Available at: https://doi.org/10.1145/3052973.3053035.
46. Kritika (2025) 'A comprehensive literature review on ransomware detection using deep learning', Cyber Security and Applications, 3, 100078. Available at: https://doi.org/10.1016/j.csa.2024.100078.
47. Liu, F.T., Ting, K.M. and Zhou, Z.-H. (2008) 'Isolation forest', in 2008 Eighth IEEE International Conference on Data Mining (ICDM). Pisa: IEEE, pp. 413–422. Available at: https://doi.org/10.1109/ICDM.2008.17.
48. Lundberg, S.M. and Lee, S.-I. (2017) 'A unified approach to interpreting model predictions', in Advances in Neural Information Processing Systems 30 (NIPS 2017). Available at: https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html.
49. Lundberg, S.M. et al. (2020) 'From local explanations to global understanding with explainable AI for trees', Nature Machine Intelligence, 2(1), pp. 56–67. Available at: https://doi.org/10.1038/s42256-019-0138-9.
50. McIntosh, T., Kayes, A.S.M., Chen, Y.-P.P., Ng, A. and Watters, P. (2021) 'Ransomware mitigation in the modern era: A comprehensive review, research challenges, and future directions', ACM Computing Surveys, 54(9), pp. 1–36. Available at: https://doi.org/10.1145/3479393.
51. Microsoft (2026) PE format. Microsoft Learn, updated 10 September 2026. Available at: https://learn.microsoft.com/en-us/windows/win32/debug/pe-format (Accessed: 1 November 2026).

52. Mirsky, Y., Doitshman, T., Elovici, Y. and Shabtai, A. (2018) 'Kitsune: An ensemble of autoencoders for online network intrusion detection', in Proceedings of the Network and Distributed System Security Symposium (NDSS 2018). San Diego, CA: Internet Society. Available at: https://doi.org/10.14722/ndss.2018.23204.
53. MITRE (2026) Data encrypted for impact, technique T1486 – Enterprise (version 1.5). MITRE ATT&CK. Available at: https://attack.mitre.org/techniques/T1486/ (Accessed: 1 November 2026).
54. Moussaileb, R., Cuppens, N., Lanet, J.-L. and Le Bouder, H. (2021) 'A survey on Windows-based ransomware taxonomy and detection mechanisms', ACM Computing Surveys, 54(6), pp. 1–36. Available at: https://doi.org/10.1145/3453153.
55. Nelson, A., Rekhi, S., Souppaya, M. and Scarfone, K. (2025) Incident response recommendations and considerations for cybersecurity risk management: A CSF 2.0 community profile. NIST Special Publication 800-61r3. Gaithersburg, MD: National Institute of Standards and Technology. Available at: https://doi.org/10.6028/NIST.SP.800-61r3.
56. NIST (2024a) Artificial intelligence risk management framework: Generative artificial intelligence profile. NIST AI 600-1. Gaithersburg, MD: National Institute of Standards and Technology. Available at: https://doi.org/10.6028/NIST.AI.600-1.
57. NIST (2024b) The NIST cybersecurity framework (CSF) 2.0. NIST Cybersecurity White Paper 29. Gaithersburg, MD: National Institute of Standards and Technology. Available at: https://doi.org/10.6028/NIST.CSWP.29.
58. OWASP (2025) 2025 top 10 risk and mitigations for LLMs and Gen AI apps. OWASP GenAI Security Project. Available at: https://genai.owasp.org/llm-top-10/ (Accessed: 1 November 2026).
59. Oz, H., Aris, A., Levi, A. and Uluagac, A.S. (2022) 'A survey on ransomware: Evolution, taxonomy, and defense solutions', ACM Computing Surveys, 54(11s), pp. 1–37. Available at: https://doi.org/10.1145/3514229.
60. Pang, G., Shen, C., Cao, L. and van den Hengel, A. (2021) 'Deep learning for anomaly detection: A review', ACM Computing Surveys, 54(2), pp. 1–38. Available at: https://doi.org/10.1145/3439950.
61. Paszke, A. et al. (2019) 'PyTorch: An imperative style, high-performance deep learning library', in Advances in Neural Information Processing Systems 32 (NeurIPS 2019). Available at: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.
62. Patsakis, C., Casino, F. and Lykousas, N. (2024) 'Assessing LLMs in malicious code deobfuscation of real-world malware campaigns', Expert Systems with Applications, 256, 124912. Available at: https://doi.org/10.1016/j.eswa.2024.124912.
63. Pedregosa, F. et al. (2011) 'Scikit-learn: Machine learning in Python', Journal of Machine Learning Research, 12(85), pp. 2825–2830. Available at: https://jmlr.org/papers/v12/pedregosa11a.html.
64. Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J. and Cavallaro, L. (2019) 'TESSERACT: Eliminating experimental bias in malware classification across space and time', in Proceedings of the 28th USENIX Security Symposium. Santa Clara, CA: USENIX Association, pp. 729–746. Available at: https://www.usenix.org/conference/usenixsecurity19/presentation/pendlebury.
65. Razaulla, S., Fachkha, C., Markarian, C., Gawanmeh, A., Mansoor, W., Fung, B.C.M. and Assi, C. (2023) 'The age of ransomware: A survey on the evolution, taxonomy, and research directions', IEEE Access, 11, pp. 40698–40723. Available at: https://doi.org/10.1109/ACCESS.2023.3268535.
66. Ribeiro, M.T., Singh, S. and Guestrin, C. (2016) '"Why should I trust you?": Explaining the predictions of any classifier', in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, pp. 1135–1144. Available at: https://doi.org/10.1145/2939672.2939778.
67. Rjoub, G., Bentahar, J., Abdel Wahab, O., Mizouni, R., Song, A., Cohen, R., Otrok, H. and Mourad, A. (2023) 'A survey on explainable artificial intelligence for cybersecurity', IEEE Transactions on Network and Service Management, 20(4), pp. 5115–5140. Available at: https://doi.org/10.1109/TNSM.2023.3282740.
68. Saqib, M., Mahdavifar, S., Fung, B.C.M. and Charland, P. (2024) 'A comprehensive analysis of explainable AI for malware hunting', ACM Computing Surveys, 56(12), 314. Available at: https://doi.org/10.1145/3677374.
69. Scaife, N., Carter, H., Traynor, P. and Butler, K.R.B. (2016) 'CryptoLock (and drop it): Stopping ransomware attacks on user data', in 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS). Nara: IEEE, pp. 303–312. Available at: https://doi.org/10.1109/ICDCS.2016.46.
70. Seneviratne, S., Shariffdeen, R., Rasnayaka, S. and Kasthuriarachchi, N. (2022) 'Self-supervised vision transformers for malware detection', IEEE Access, 10, pp. 103121–103135. Available at: https://doi.org/10.1109/ACCESS.2022.3206445.
71. Souppaya, M., Barker, W., Fisher, W. and Kent, K. (2026) Ransomware risk management: A cybersecurity framework 2.0 community profile. NIST Internal Report 8374r1. Gaithersburg, MD: National Institute of Standards and Technology. Available at: https://doi.org/10.6028/NIST.IR.8374r1.
72. Ucci, D., Aniello, L. and Baldoni, R. (2019) 'Survey of machine learning techniques for malware analysis', Computers & Security, 81, pp. 123–147. Available at: https://doi.org/10.1016/j.cose.2018.11.001.
73. Urooj, U., Al-rimy, B.A.S., Zainal, A., Ghaleb, F.A. and Rassam, M.A. (2022) 'Ransomware detection using the dynamic analysis and machine learning: A survey and research directions', Applied Sciences, 12(1), 172. Available at: https://doi.org/10.3390/app12010172.
74. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I. (2017) 'Attention is all you need', in Advances in Neural Information Processing Systems 30 (NIPS 2017). Available at: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.
75. Xu, H. et al. (2024) 'Large language models for cyber security: A systematic literature review', arXiv preprint, arXiv:2405.04760 [not peer reviewed; version 5, 2025]. Available at: https://arxiv.org/abs/2405.04760.
76. Yang, L., Ciptadi, A., Laziuk, I., Ahmadzadeh, A. and Wang, G. (2021) 'BODMAS: An open dataset for learning based temporal analysis of PE malware', in 2021 IEEE Security and Privacy Workshops (SPW). San Francisco, CA: IEEE, pp. 78–84. Available at: https://doi.org/10.1109/SPW53761.2021.00020.
77. Yao, Y., Duan, J., Xu, K., Cai, Y., Sun, Z. and Zhang, Y. (2024) 'A survey on large language model (LLM) security and privacy: The good, the bad, and the ugly', High-Confidence Computing, 4(2), 100211. Available at: https://doi.org/10.1016/j.hcc.2024.100211.
78. Zhu, J., Jang-Jaccard, J., Singh, A., Welch, I., Al-Sahaf, H. and Camtepe, S. (2022) 'A few-shot meta-learning based siamese neural network using entropy features for ransomware classification', Computers & Security, 117, 102691. Available at: https://doi.org/10.1016/j.cose.2022.102691.