MalBERT-Temporal: Transformer-Based Zero-Day Malware Detection in Windows Executable Binaries Under Strict Temporal Isolation

Manar Alanazi, Israa Alsiyat
Department of Computer Science, Northern Border University, Arar, Saudi Arabia

***Abstract*—In current malware detection benchmarks, random train/test splits are commonly used, allowing temporal leakage to occur and obscuring performance degradation caused by concept drift. Moreover, traditional classifiers that operate on a one-dimensional PE feature vector do not explicitly model long-range interactions between structurally distant feature groups. This study introduces a Transformer-based malware detection approach named MalBERT-Temporal that reshapes the 2,381-dimensional BODMAS PE feature vector into 16 contiguous feature-group tokens and then processes these tokens with Transformer encoder layers employing multi-head self-attention to model interactions among all tokens. The proposed approach is evaluated using a strict temporal protocol in which the training data comprise only pre-2020 samples, and the test data span the complete 2020 evaluation timeline. Each model is independently calibrated using a fixed validation set with a false-positive-rate budget of 0.1% and is evaluated monthly and on malware families unseen during training. At the calibrated security threshold, MalBERT-Temporal achieves an F1-score of 97.84% with a false-positive rate of 0.138%, outperforming a 1-D CNN baseline, which achieves an F1-score of 92.01% and a false-positive rate of 3.388%, and a Random Forest baseline, which achieves an F1-score of 67.29%. Welch's t-tests confirm statistically significant differences in monthly F1-scores between the proposed approach and both baselines. Moreover, MalBERT-Temporal maintains monthly F1-scores within a 2.2-point band throughout the nine-month evaluation period, indicating improved robustness to temporal distribution shift. A component-wise ablation traces the improvement to the tokenized self-attention mechanism itself: substituting a token-wise feed-forward block for self-attention while keeping every other component costs 2.73 F1 points, and removing the tokenization costs 1.85 points, while positional embeddings, the CLS token, multi-scale pooling, encoder depth, and nonlinearity are each worth 0.47 points or less. A matched random-split control that leaves the model, the preprocessing, the training budget, and the calibration procedure unchanged gives an F1-score of 98.86%, which confirms that chronological evaluation is the stricter protocol.**

Keywords*—Zero-day malware detection; transformer; self-attention; portable executable; temporal isolation; concept drift; BODMAS; structural feature modeling

I. Introduction

Malicious software remains one of the major threats to digital infrastructure as it can facilitate data theft, surveillance, and compromise of systems in both commercial and public-sector environments [1]. Signature-based detection systems remain the dominant approach in operational malware detection and work by matching executables against databases of known patterns; consequently, they cannot detect polymorphic, obfuscated, or previously unseen zero-day malware [1, 2]. Since adversaries frequently develop mutation and obfuscation toolkits that can bypass static fingerprinting, the inadequacy of signature-centric methods for emerging threats is well established in the security literature.

This major weakness has motivated the use of machine-learning and deep-learning methods that are capable of classifying executables based on structural, statistical, and behavioral evidence rather than relying on libraries of curated rules [1]. Windows Portable Executable (PE) static analysis is particularly suitable as a source for large-scale analysis of binary files: information such as file headers, section tables, import directories, exported symbols, string statistics, and byte-entropy profiles can be gathered without executing any computer program, resulting in high-dimensional feature vectors that reflect a binary's design intent, dependency structure, and payload characteristics [3]. These features render structured PE feature analysis a practical foundation for large-scale automated malware classification.

Transformer-based deep-learning models are well suited to operate in such highly structured, high-dimensional environments. Self-attention computes the relationships between all positions in the input sequence simultaneously. This allows the model to capture long-range structural dependencies that are difficult to model explicitly using local methods [2]. In malware analysis, malicious intent is often spread among different attributes that are spatially separated, such as an entropy anomaly in one section and suspicious import patterns in another, rather than being found entirely in a single local region. Traditional machine-learning classifiers treat PE feature vectors as unordered feature sets, while convolutional models mainly focus on local neighboring patterns. As a result, both approaches are limited in their ability to capture interactions between distant feature elements [2].

Another less frequently acknowledged difficulty is the issue of evaluation realism. Malware is non-stationary in nature: new malware families are continually discovered, existing variants undergo change, and even benign software evolves, causing the feature distribution to drift away from any fixed training dataset [4]. The use of random train-test splitting permits samples belonging to a future period to appear within the training data, and thereby gives rise to temporal data leakage [4, 5]. Such temporal data leakage artificially inflates the reported

performance relative to that which would be observed under real-world deployment [4, 5]. This phenomenon, in which the distribution shifts as a consequence of changes in the environment, is termed concept drift. Within only a few months of deployment, concept drift of this kind may degrade detection performance to a considerable degree. Notwithstanding this, the majority of existing studies neither implement chronological partitioning nor evaluate performance over subsequent time windows [5].

Taken together, the literature review reveals two gaps that are closely interlinked. First, PE feature vectors are typically regarded as flat, order-agnostic arrays, with the consequence that models may overlook the long-range relational patterns that attention mechanisms are specifically designed to capture. Second, most evaluations are carried out through random splits and ignore temporal drift monitoring. This makes it difficult to evaluate long-term robustness reliably. Only a limited number of studies have combined both aspects in a single framework focused on Windows PE malware detection.

To bridge these gaps, this study proposes MalBERT-Temporal, a Transformer-based framework for detecting zero-day malware in Windows executables. The framework transforms structured PE feature vectors into sequences of semantically meaningful tokens, captures long-range dependencies using Transformer encoder layers with multi-head self-attention, and relies on an evaluation protocol that strictly separates the training and testing periods chronologically. MalBERT-Temporal is built using the timestamped BODMAS dataset [6], which consists of 134,435 PE samples spanning fourteen years. The model is trained exclusively on pre-2020 data and tested on 2020 data. The framework's performance is then compared with that of CNN and Random Forest baselines under exactly the same temporal conditions to distinguish the impact of architectural design from that of the evaluation protocol. This design addresses the following research questions: 1) to what extent does strict chronological splitting expose performance degradation hidden by conventional random splitting (RQ1); and 2) can a Transformer-based architecture capture long-range structural relationships within PE feature vectors well enough to outperform the CNN and Random Forest baselines under identical temporal conditions (RQ2)?

The primary contributions of this study are the following:

- A rigorous chronological evaluation methodology using the timestamped BODMAS dataset, which enforces temporal isolation and reduces data leakage in zero-day malware detection.
- MalBERT-Temporal, a Transformer-based architecture in which the structured PE feature vector is encoded as a token sequence and multi-head self-attention models long-range structural dependencies in the data.
- A controlled comparison against CNN and Random Forest baselines using identical temporal splits for training and evaluation, isolating the effect of architectural decisions.
- A systematic study of generalization to future and previously unseen malware families under realistic chronological conditions.
- Monthly temporal drift monitoring that tracks detection performance across successive time windows and evaluates temporal stability using statistical testing.
- A set of controlled ablation and control experiments that isolate the source of the reported gains, covering component and group-granularity ablations, a matched random-split control, retraining schedules, bootstrapped zero-day confidence intervals, and feature-level drift analysis.

The rest of the study is structured in the following way. Section II reviews the literature concerning Transformer-based malware detection, alternative representations, and temporal evaluation methodologies. Section III describes the dataset, the preprocessing pipeline, the model designs, and the experimental design. Section IV presents the comparative and temporal-drift results. Section V interprets these results in the light of the research questions, and Section VI concludes the study and outlines directions for future research.

II. Related Work

A. Transformer and Sequence-Based Executable Malware Detection

Numerous researchers have examined the application of attention-based and Transformer models to the analysis of malware. Wang et al., for example, proposed TTDAT, a dual-attention Transformer founded upon API call sequences. The method fuses global multi-head attention with a local attention mechanism to capture dependencies across various scales effectively. This model yielded an average F1-score of 0.90 on a dynamic API dataset [7]. Qian and Cong, along the same lines, enhanced the Transformer encoder by inserting a one-dimensional channel attention module (CAFTrans) to better capture correlations among features within API vectors. They reported state-of-the-art performance on the mal-api-2019 benchmark [8]. The two publications cited above rely on dynamically extracted API sequences, which entail different extraction costs and scalability constraints from those of structured static PE feature vectors.

Lu et al. tested two self-attention Transformer classifiers directly on subsets of BODMAS by means of a two-stage pipeline. SeqConvAttn processed raw bytes, whereas ImgConvAttn used binary image representations [9]. The two-stage pipeline achieved 96.96% accuracy on BODMAS-11, showing that Transformers outperform CNN baselines on PE binary data. However, the input consisted of raw bytes rather than structured PE attributes, and random data partitioning was applied; temporal generalization therefore remains uncharacterized. Labied et al. combined disassembly, assembly-level tokenization, DLL import extraction, and a Transformer encoder to achieve 96.88% accuracy on 11,000 VirusTotal PE samples [10]. The results indicate that attention-based models can be successful for PE analysis, but their approach focuses on assembly-level semantics rather than structured multi-group PE feature vectors. In addition, no chronological evaluation was

presented. Kim et al. adopted a Transformer encoder for pre-training with new masking strategies and finally fine-tuned it for malware detection tasks across different binary formats. Their work indicates that pre-training on raw bytes yields generalizable representations [11]. Tallane et al. used a byte-embedding Transformer encoder to process the first 1,024 bytes of each binary. Their model also used stochastic weight averaging and Mixup. They reached 99% accuracy on one benchmark and 92–94% on two others [12]. Both works show that byte-level Transformers are effective malware detectors. However, they also forgo the semantic structure contained in PE headers, section descriptors, and import tables.

B. Alternative Representations and Hybrid Architectures

Binary visualization techniques convert malware into grayscale images and utilize vision models for classification. Alshomrani et al. combined ConvNeXt-Tiny's local feature extraction with Swin Transformer's global context modeling through adaptive gating, resulting in 94.04% average validation accuracy across three malware image benchmarks and 98% on external validation [13]. Despite the intuitive nature of the image approach, it loses the hierarchical semantic structure of PE files. Alzahrani et al. developed RansomFormer, a cross-modal Transformer that integrates PE byte data with API import features using cross-attention to detect ransomware, demonstrating that merging static bytes with behavioral metadata enhances specificity [14]. Miraoui and Belgacem compared classical machine-learning classifiers and CNN-based models on structured Windows PE feature vectors, finding that both methods achieved high detection accuracy, though neither explicitly captures long-range interactions among distant PE attributes [15]. Dhanya et al. proposed a zero-day IoT malware defense using an anomaly-driven approach that blends attention-enhanced autoencoders, cosine similarity in latent space, Isolation Forest scoring, and a Transformer encoder, achieving 99.38% accuracy on Mirai network traffic [16]. However, the study targeted IoT network traffic rather than Windows PE files, which limited its applicability, and no temporally structured evaluation was performed.

C. Temporal Evaluation, Concept Drift, and Long-Term Robustness

Concept drift and temporal robustness are two issues of increasing concern in malware research. Chen et al. discovered that Android malware classifiers can lose significant detection capability within several months of the training cutoff date and thus developed contrastive learning methods for maintaining classifier reliability under an active-learning budget [4]. Li et al. noted that retraining techniques for Windows PE malware underperform when drift-invariant representations obtained through adversarial domain adaptation are not used [5]. Through the evaluation of timestamping and feature selection methods over a seven-year Android detection period, Guerra-Manzanares and Bahsi demonstrated the significance of labeling budget allocation and temporal placement for achieving long-term performance stability [17]. Maniriho et al. developed MeMalDet, a memory-analysis framework containing explicit temporal attributes and chronological data splits, which achieved an accuracy of 98.82% under concept-drift evaluation conditions, thereby showing that incorporating timestamps into dataset design yields fundamentally more informative benchmarks [18].

D. Synthesis and Identified Research Gap

The reviewed literature consistently validates Transformer architectures for learning contextual relationships in executable data. In addition, concept drift is acknowledged as a practically significant phenomenon. However, a clear yet underexplored combination emerges. Works that depend on raw bytes, assembly tokens, or binary images sacrifice the semantically organized structure of structured PE feature vectors; at the same time, works using structured PE metadata mostly rely on flat-feature classifiers or CNN models that are incapable of modeling long-range attribute interactions; furthermore, the large majority of published evaluations across all representation types employ random train-test splits and omit systematic temporal drift monitoring. Few studies have jointly examined: 1) Transformer-based sequential modeling of structured PE feature vectors, 2) strictly chronological separation of training and test data, 3) evaluation on future and previously unseen malware families, and 4) month-by-month performance monitoring. MalBERT-Temporal is positioned to address this combined gap by integrating attention-based sequence modeling of the full 2,381-dimensional BODMAS feature space, strict temporal isolation, explicit zero-day family generalization testing, and temporal drift analysis with statistical significance testing under controlled baseline comparisons.

III. Methodology

The MalBERT-Temporal framework represents a Transformer-based approach to malware detection. The key idea behind the architecture is to replace the standard flat, order-agnostic representation of Portable Executable (PE) data with a structured token-based representation. Instead of feeding a 2,381-dimensional feature vector to the classifier as a flat array of features, the framework transforms it into a token-based representation, allowing multi-head self-attention to model interactions between semantically related feature groups. To study the effectiveness of this approach independently of other factors, the framework was evaluated against two established structured PE detection approaches: a tree-based ensemble [15], [19] and a one-dimensional convolutional network [9], under the same chronological partitioning. A key methodological principle maintained throughout all experiments is that the chronological order of the data is preserved and all operating thresholds are determined using only a held-out subset of the pre-2020 training data.

A. System Overview

There are four distinct stages in the pipeline, as illustrated in Fig. 1. The first stage involves Data Preparation, in which the timestamp metadata is merged with the feature matrix, malformed instances are removed, the samples are sorted by time, and a strict temporal split is performed. In the second stage, Feature Enhancement, the attributes are preprocessed through signed-logarithmic transformation and robust scaling to make attributes of vastly different orders of magnitude comparable for modeling purposes. The third stage entails the use of a Comparative Modeling Framework, where three models, namely MalBERT-Temporal, a Random Forest model, and a CNN model, are trained on exactly the same data using identical

preprocessing. Therefore, any observed performance difference can be attributed to the modeling strategy rather than differences in data preparation or evaluation. The fourth stage, in turn, involves the Comprehensive Performance and Temporal Stability Analysis, in which the models are assessed based on two dimensions: 1) multi-scenario evaluation of F1-score, accuracy, precision, recall, and zero-day generalization, and 2) temporal drift assessment that tracks monthly performance and evaluates the statistical significance and temporal consistency of the observed differences using Welch's t-test.

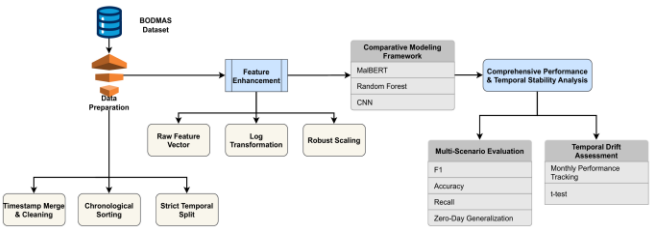

Fig. 1. End-to-end MalBERT-temporal pipeline.

B. Model Architectures

Three model architectures are trained and evaluated under identical temporal partitioning, preprocessing, and training data to provide representative implementations of three commonly used approaches to structured PE malware detection: order-agnostic tree ensembles, local convolutional modeling, and Transformer-based self-attention [2]. This fixes the input pipeline throughout the experiments, making the comparison primarily a test of each model's inductive bias and ensuring that observed performance differences are attributable to the modeling strategy rather than to preprocessing or evaluation.

1) Random forest baseline: The first baseline employs an ensemble of decision trees trained directly on the entire structured PE feature vector, without making any assumptions about sequential information or performing an architecture search [19]. Random Forest models handle heterogeneous tabular features of diverse scales robustly and have long been used as order-agnostic baselines in PE malware detection [15], [19]. The main drawback of the model is architectural in nature, in that feature interactions are encoded only through hierarchical decision paths, making complex relationships between distant attributes less explicit than in attention-based models. Consequently, interactions between feature groups that rarely co-occur along the same decision paths may be more difficult to model effectively.

2) 1-D Convolutional Network (CNN): The second architecture treats the feature vector as a one-dimensional signal and applies a series of convolutional filters aimed at capturing local patterns from neighboring feature indices [9]. This type of architecture has demonstrated strong performance on PE and byte-level representations by exploiting local structural patterns that conventional tabular models may overlook [9]. One of the main disadvantages of such an approach is locality, since the receptive field of a filter contains only neighboring feature indices. Therefore, interactions between widely separated feature groups typically require several convolution and pooling stages before they can be integrated. For example, an entropy anomaly in one section and a suspicious import pattern in another may be combined only after several such stages. As a result, modeling very long-range feature relationships may be less effective than with architectures that explicitly support global interactions.

3) Transformer-based MalBERT-temporal architecture: The architecture proposed herein transforms the flat PE feature vector into a short sequence of tokens and uses multi-head self-attention to model relationships between groups of features. The 2,381 features are divided into 16 contiguous groups, each of which is projected through its own learnable linear transformation, layer-normalized [20], and passed through a GELU activation function [21], leading to one token per feature group. The group boundaries come from an equal-width partition of the ordered index range F1-F2381 and not from any feature-selection or clustering procedure. Since 2,381 is not an integer multiple of 16, three groups hold 148 features and the other thirteen hold 149. Section IV-H examines the choice of sixteen groups empirically. A learnable classification (CLS) token, following the BERT architecture [22], is prepended to aggregate global information, whereas learnable positional embeddings preserve the ordering within the sequence. The resulting 17-token sequence is processed by a stack of pre-normalized Transformer encoder layers [23], based on multi-head self-attention:

$$\text{Attention}(Q,K,V) = softmax\left(\frac{QK^{\top}}{\sqrt{d_k}}\right)V \quad (1)$$

where, Q, K, and V denote the query, key, and value projections of the token sequence, respectively, while d_k denotes the dimensionality of each attention head [24]. As self-attention allows each token to attend directly to every other token, interactions between distant feature groups, such as an entropy anomaly encoded in one group and a suspicious import pattern encoded in another, can be modeled within a single Transformer encoder layer. This global interaction mechanism facilitates learning long-range structural dependencies that are difficult to model using convolutional or tree-based methods. Therefore, the novelty of this research lies not in the introduction of a new Transformer architecture but in the reorganization of structured PE features into a tokenized representation that enables attention-based modeling under a temporally rigorous evaluation framework.

C. Pooling and Threshold Calibration

Each architecture is evaluated independently without being incorporated into an ensemble system. As such, each model generates its own continuous output score, which is then converted into a binary decision using an independently calibrated threshold. This ensures that performance differences between the models are attributable to the individual architectures rather than ensemble effects.

MalBERT-Temporal summarizes the encoder output via three different pooling operations: the CLS token embedding, the mean of the feature-token embeddings, and their element-wise maximum. The concatenated outputs are fed into a classification head regularized with dropout [25]. Combining a

learned global representation with mean and maximum pooling provides complementary global and distributional information before final classification. The CNN and Random Forest models are baseline methods that produce a single output logit or probability directly from their respective classification layers. Therefore, they do not require explicit pooling operations.

Rather than using the standard probability threshold of 0.5, each architecture is calibrated independently using its corresponding validation receiver operating characteristic (ROC) curve. The operating threshold is then determined as the probability level for which the validation set false-positive rate (FPR) is closest to a fixed budget of 0.1%. This approach aligns with the evaluation method of the original BODMAS work [6], which considers maintaining a very low false-positive rate an important operational constraint. Since the three architectures result in different score distributions, the calibrated thresholds also vary, although the target validation FPR budget remains the same. For completeness, all models were additionally evaluated using the standard decision threshold of 0.5, which permits an assessment of the trade-off between false positives and false negatives.

All operating thresholds were determined solely from the pre-2020 validation subset, prior to the observation of any 2020 samples. No recalibration was carried out during the temporal evaluation period, and strict chronological isolation between the development of the model and its testing was thereby maintained. Consequently, changes in the underlying data distribution during 2020 may influence the realized false-positive rate relative to its validation estimate. In order to quantify this effect, the false-positive rates for each month within the evaluation period were reported together with the corresponding performance measures.

D. Dataset and Temporal Integration

This study makes use of the BODMAS dataset introduced by Yang et al. [6]. The dataset provides pre-extracted static features for Windows Portable Executable (PE) files, together with curated malware family labels and first-seen timestamps for each sample. The two principal components of the dataset, namely the main feature table and the timestamp metadata, were merged by means of the SHA-256 file hash. Each record comprises 2,381 numerical features (F1-F2381), which describe PE header information, section properties, import and export tables, string statistics, and byte-entropy histograms. Following the parsing of the timestamps and the removal of the small number of records for which timestamp information could not be recovered, 130,716 samples remained, spanning the period from January 2007 to September 2020. Samples are sorted in chronological order with respect to the year, month, and precise timestamp. Feature columns are kept in their original indexing order (F1–F2381), providing a consistent ordering for the subsequent tokenization process.

The binary class label is directly derived from the Label field of the dataset, where samples belonging to any malware family are labeled as malicious (y = 1), and all others are labeled as benign (y = 0):

$$y = 1 \; if \; Lable > 0, \; otherwise \; y = 0 \tag{2}$$

Consequently, the dataset contains 73,426 benign samples (56.17%) and 57,290 malicious samples (43.83%), resulting in a moderate class imbalance that is intentionally preserved so that the reported operating metrics reflect the natural class distribution. As benign files do not have any malware family or category details, the corresponding metadata fields are assigned the values "non-malicious" and "benign," respectively, which eliminates missing values in the dataset. Fig. 2 depicts the temporal distribution of the samples throughout the fourteen-year collection period. There is an evident clustering of samples in 2019 and 2020, which provides a sufficiently dense and recent time window for creating a temporally separated training set and future test set.

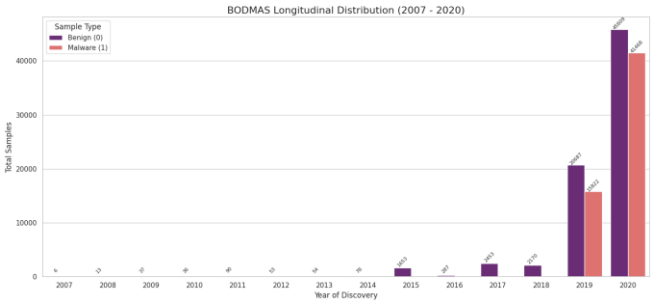

Fig. 2. Longitudinal distribution of benign and malicious BODMAS samples by year of discovery.

E. Signed-Log Transform and Robust Scaling

The raw PE attributes span several orders of magnitude because counts, sizes, and entropy-related measurements are defined on substantially different numerical scales. Consequently, conventional standardization may be disproportionately influenced by a small number of large-valued features. The effect of extreme values that can result from the parsing process is reduced by clipping the magnitudes of the attributes to $\pm 1 \times 10^{10}$. The sign of each attribute is then preserved through a signed logarithmic transformation:

$$\tilde{x} = sign(x) \cdot Ln(1 + |x|) \tag{3}$$

The transformed features are then scaled using the RobustScaler implementation provided by scikit-learn [26], which replaces the mean and standard deviation with the median and interquartile range (IQR), making the transformation more robust to the heavy-tailed distributions commonly observed in PE feature statistics:

$$x' = \frac{\tilde{x} - Q_2}{Q_3 - Q_1} \tag{4}$$

where, Q_1, Q_2, and Q_3 denote the first quartile, median, and third quartile, respectively, estimated exclusively from the training partition. The scaler is fitted only on the training data and is subsequently applied unchanged to the validation and test sets. This approach prevents information from future samples from influencing feature normalization. Any remaining non-finite values are set to zero, while the normalized values are clipped to ±5 to decrease the influence of rare extreme values. Identical preprocessing steps were applied to all three architectures, thereby ensuring that the observed differences in performance arise primarily from the modeling approach rather than from differences in feature preprocessing.

F. Strict Temporal Partitioning and Zero-Day Isolation

All samples collected prior to 2020 were employed for the development of the models, whereas all samples dated within 2020 were reserved exclusively for evaluation. This strict chronological division yielded 43,439 pre-2020 samples for model development and 87,277 temporally subsequent samples for evaluation. Within the pre-2020 group, a stratified 20% subset (8,688 samples) was set aside for validation, leaving 34,751 samples for the training of the models. The validation subset was used solely for threshold calibration and early stopping, and was never used during the test period.

The 2020 test set was further subdivided according to the degree of malware-family exposure encountered during training. If the malware family of a sample was present in the pre-2020 training data, the sample was classified as belonging to a known family; otherwise, it was regarded as a zero-day family. On the basis of this categorization, the test set contained 85,705 known-family samples and 1,572 zero-day samples, the latter representing malware families that had not been observed during training. Table I summarizes the resulting data partition. Since genuinely new malware families occur far less frequently than new variants of existing families, the zero-day subset is comparatively small. This subset nevertheless affords the most direct evaluation of the ability of the model to generalize to previously unseen malware families.

TABLE I. STRICT CHRONOLOGICAL DATA PARTITION OF THE BODMAS DATASET

Partition	Samples	Description
Training (pre-2020)	43,439	All samples dated 2007–2019
• Pure training	34,751	used for model fitting
• Validation	8,688	threshold calibration, early stopping
Test (2020)	87,277	strictly later than training
• Known families	85,705	family seen during training
• Zero-day families	1,572	previously unseen families

G. Training, Calibration, and Evaluation

MalBERT-Temporal was optimized by means of the AdamW optimizer [27]. A cosine-annealing learning-rate schedule with warm restarts was adopted [28], and, in conjunction with this strategy, gradient clipping at 1.0 was applied in order to stabilize the optimization process. The model was trained using binary cross-entropy loss. The checkpoint corresponding to the highest validation F1-score was retained and used for final evaluation on the test set.

Rather than adopting the conventional decision threshold of 0.5, each architecture was calibrated independently on the basis of its validation receiver operating characteristic (ROC) curve. The operating threshold was chosen as the probability cutoff whose validation false-positive rate (FPR) lay closest to a fixed budget of 0.1%. This is consistent with the evaluation protocol used in the original BODMAS study [6], where maintaining a very low false-positive rate is treated as an operational deployment requirement. The resulting calibrated thresholds are 0.4358 for the CNN, 0.9052 for the Random Forest, and 0.9795 for MalBERT-Temporal. For completeness, every model was evaluated using both its calibrated security threshold and the conventional 0.5 decision threshold, allowing the trade-off between false positives and false negatives to be assessed.

Performance is evaluated using accuracy, precision, recall, F1-score, and the false-positive rate,

$$FPR = \frac{FP}{FP+\mathrm{TN}} \tag{5}$$

Computed both over the complete 2020 evaluation period and separately for each month. Generalization to previously unseen malware families is evaluated using the zero-day subset and further analyzed on a per-family basis. Welch's two-sample t-test [29] is used to determine whether the observed performance differences between model pairs are statistically significant based on their monthly F1-score series:

$$t = \frac{\bar{x}_A - \bar{x}_B}{\sqrt{s_A^2/n + s_B^2/n}} \tag{6}$$

where, each calendar month is treated as an independent observation. Welch's test is suitable because it does not assume equal variances between groups and is therefore appropriate for comparing monthly performance series with different levels of variability.

IV. RESULTS AND ANALYSIS

The study's testing horizon comprises nine monthly evaluation periods spanning January to September 2020. Unless otherwise explicitly stated, all comparative results are presented with reference to the calibrated security threshold that corresponds to a validation false-positive rate of 0.1%. The model-specific diagnostic panels, which display the validation ROC curve, the monthly F1-score, the monthly false-positive rate, and the overall confusion matrix, are presented in Fig. 3 for the CNN and in Fig. 4 for the Random Forest. The analysis proceeds from aggregate performance to temporal behavior across successive months, followed by zero-day generalization, statistical significance analysis, and per-family performance.

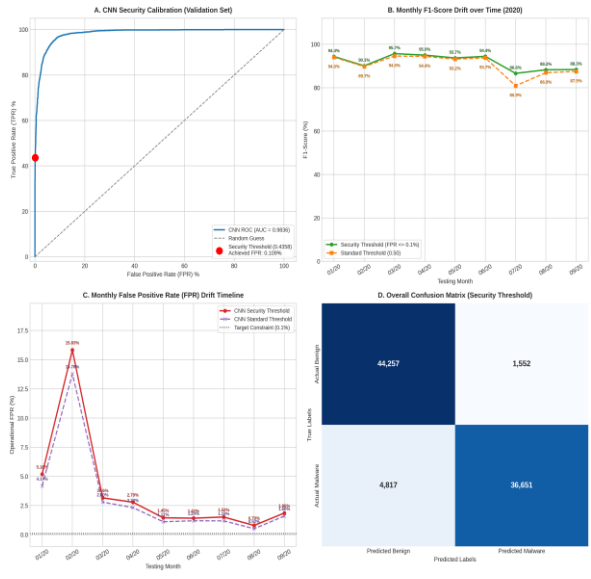

Fig. 3. CNN baseline diagnostics: (a) ROC and security calibration on the validation set, (b) monthly F1 drift, (c) monthly false-positive-rate timeline, and (d) overall confusion matrix at the security threshold.

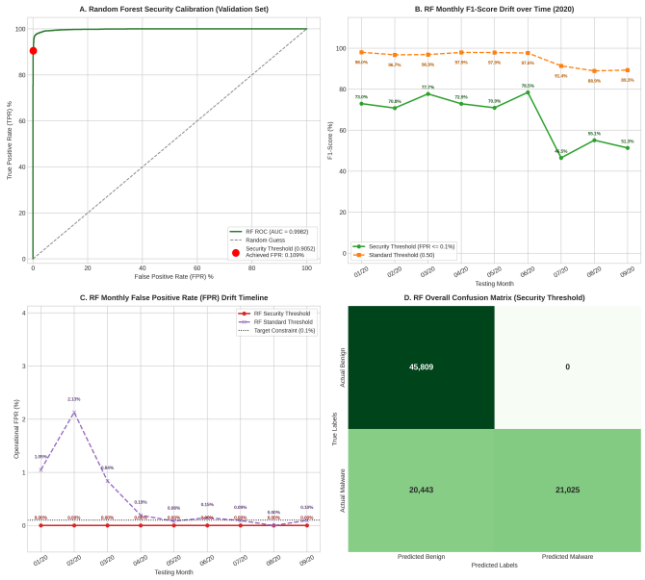

Fig. 4. Random-forest baseline diagnostics: (a) ROC and security calibration, (b) monthly F1 drift, (c) monthly false-positive-rate timeline, and (d) overall confusion matrix at the security threshold.

A. Overall Comparative Performance

Table II summarizes the aggregate performance achieved over the complete 2020 evaluation period and indicates the effect of the proposed attention-based representation under identical temporal evaluation conditions. At its calibrated security threshold, MalBERT-Temporal attains an F1-score of 97.84% while maintaining a false-positive rate of 0.138%. At the conventional 0.5 threshold, the model achieves an F1-score of 98.63%. At the calibrated operating point, the CNN achieves an F1-score of 92.01% but has a substantially higher false-positive rate of 3.39%. The Random Forest has a false-positive rate of 0.000% at its calibrated security threshold; however, this operating point reduces recall to 50.70% and the F1-score to 67.29%, showing that reducing false positives comes at the expense of detecting a considerable proportion of malicious samples. At the conventional 0.5 threshold, the F1-score of the Random Forest increases to 95.21%, although it remains below the performance achieved by MalBERT-Temporal at its stricter calibrated operating point.

TABLE II. OVERALL COMPARATIVE PERFORMANCE ON THE 2020 TEST HORIZON

Model (operating point)	Thr.	FPR%	Acc%	Prec%	Rec%	F1%
MalBERT – Security	0.9795	0.138	97.98	99.84	95.91	97.84
MalBERT – Standard	0.5000	0.362	98.71	99.59	97.69	98.63
CNN – Security	0.4358	3.388	92.70	95.94	88.38	92.01
CNN – Standard	0.5000	2.840	91.73	96.47	85.73	90.78
Random Forest – Security	0.9052	0.000	76.58	100.0	50.70	67.29
Random Forest – Standard	0.5000	0.454	95.64	99.45	91.32	95.21

Fig. 5 represents the model-specific diagnostic panels, which display the validation ROC curve, the monthly F1-score, the monthly false-positive rate, and the overall confusion matrix for MalBERT-Temporal. MalBERT-Temporal achieves an area under the ROC curve (AUC) of 0.9989 (see Fig. 5a). Its confusion matrix at the calibrated security threshold (see Fig. 5d) contains 45,746 true negatives and 63 false positives while correctly identifying 39,772 of the 41,468 malicious samples. These results indicate that the proposed Transformer-based representation maintains a favorable balance between malware detection and false-positive control.

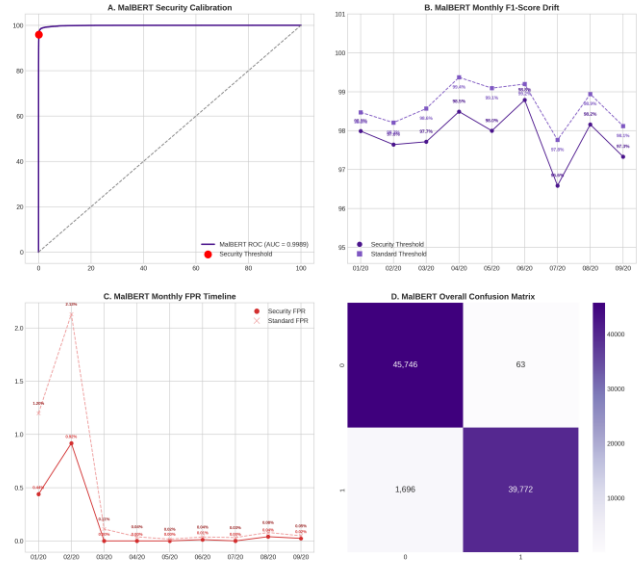

Fig. 5. MalBERT-Temporal diagnostics: (a) ROC curve, (b) monthly F1 drift, (c) monthly false-positive-rate timeline, and (d) overall confusion matrix.

B. Temporal Drift Across the 2020 Horizon

The aggregate results do not reflect the temporal behavior across the evaluation period, which is one of the main elements of the study methodology. Table III provides an analysis of the monthly performance of MalBERT-Temporal at the calibrated security threshold. The F1-score remains within a narrow range, from 96.59% to 98.79%, across all nine months. The false-positive rate is below or equal to 0.1% from March onward, following higher values of 0.44% in January and 0.92% in February. No monotonic degradation is observed as the temporal distance from the training cutoff increases. Performance in August (98.16%) and September (97.33%) remains comparable to that observed in the earlier months, which indicates consistent behavior across the full 2020 horizon. These results point to temporal stability under the proposed evaluation protocol, rather than to performance concentrated within a specific subset of months.

The baseline comparison, presented in Fig. 6, illustrates markedly different behavior. The Random Forest model exhibits a pronounced decline during the second half of the evaluation period, falling from F1-scores in the high 70% range to 46.51% in July, followed by a partial recovery to the low 50% range in August and September. The CNN displays more moderate variation but nevertheless shows a downward trend, reaching the high-80% range in the final months. By contrast, MalBERT-Temporal sustains relatively stable performance throughout the entire period. The monthly F1 and FPR trajectories displayed in

Fig. 5b and Fig. 5c, respectively, confirm that this stability extends to both metrics and is not an artifact of aggregation. Operationally, such temporal stability is relevant in deployment scenarios where retraining is not performed frequently, and models are expected to maintain performance under distribution shift.

TABLE III. MALBERT-TEMPORAL MONTHLY PERFORMANCE IN 2020 (SECURITY THRESHOLD)

Month	#Ben.	#Mal.	FPR%	Acc%	Prec%	Rec%	F1%
01/20	5,921	4,515	0.44	98.28	99.41	96.61	97.99
02/20	3,708	4,264	0.92	97.52	99.18	96.15	97.64
03/20	3,577	4,990	0.00	97.40	100.0	95.53	97.71
04/20	5,201	4,640	0.00	98.60	100.0	97.03	98.49
05/20	6,121	5,449	0.00	98.15	100.0	96.07	98.00
06/20	8,182	4,217	0.01	99.19	99.98	97.63	98.79
07/20	6,387	5,000	0.00	97.10	100.0	93.40	96.59
08/20	2,521	3,816	0.04	97.82	99.97	96.41	98.16
09/20	4,191	4,577	0.02	97.29	99.98	94.82	97.33

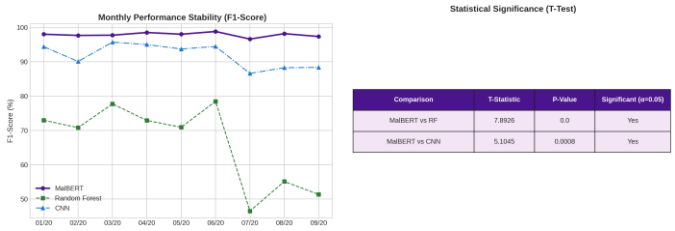

Fig. 6. Monthly F1 stability of the three models across the 2020 horizon (left) and the corresponding Welch t-test results (right).

C. Zero-Day Family Generalization

The zero-day set comprises 1,572 samples belonging to malware families that were not observed during training and therefore provides an evaluation of MalBERT-Temporal's generalization under strict family-level separation. MalBERT-Temporal accurately classifies the majority of these zero-day samples, with only a small number of malware families exhibiting low recall. In several months, at most one zero-day family contributes to misclassifications.

The per-family analysis compares the F1-scores of all malware families present in the test dataset. Substantial improvements are observed for several zero-day-dominated families, such as agentb, aupa, wannacry, waledac, and zenpak, where the F1-score increases from near zero for the Random Forest classifier to perfect or near-perfect classification for MalBERT-Temporal. Improvements over the CNN are also observed for the majority of families. Cases where MalBERT-Temporal underperforms are primarily associated with families containing very few samples, where individual misclassifications have a disproportionate effect on the computed F1-score. Overall, the performance differences are driven by consistent gains on larger and more stable families, with limited and isolated degradations on low-support classes. Several of these families have fewer than ten test samples, so bootstrapped confidence intervals for the per-family zero-day estimates are given in Section IV-K and should be read alongside the point estimates below.

D. Statistical Significance

To assess whether the observed performance differences are statistically significant, Welch's two-sample t-test [29] is applied to the monthly F1-score series for each model pair, as reported in Table IV and Fig. 6. At the security threshold, the improvements in performance for MalBERT-Temporal compared with the Random Forest ($t = 7.89$, $p < 0.001$) and the CNN ($t = 5.10$, $p = 0.0008$) are found to be statistically significant. Similar results are obtained at the conventional 0.5 threshold, where the performance of MalBERT-Temporal is significantly higher than that of the Random Forest ($t = 2.80$, $p = 0.022$) and the CNN ($t = 5.13$, $p = 0.0008$). All comparisons satisfy the significance level of $\alpha = 0.05$, which indicates that the observed differences in performance are unlikely to have arisen from random variation across the monthly evaluation windows.

TABLE IV. WELCH'S T-TEST ON MONTHLY F1 SCORES

Comparison	t-stat	p-value	Sig. (α=0.05)
MalBERT vs RF (Security)	7.8926	< 0.001	Yes
MalBERT vs CNN (Security)	5.1045	0.0008	Yes
MalBERT vs RF (0.50)	2.8011	0.0223	Yes
MalBERT vs CNN (0.50)	5.1272	0.0008	Yes

E. Per-Family Comparative Detection Analysis

Table V presents a representative subset of twenty malware families chosen from the complete per-family evaluation, showing how the overall results presented in Table II are distributed among the individual classes. Across these malware families, the proposed model either matches or exceeds the Random Forest baseline at both operating points and outperforms the CNN in the majority of cases. Specifically, the proposed model achieves a mean security-threshold F1-score of 97.81%, compared with 96.90% for the CNN and 67.22% for the Random Forest.

These differences are largest in comparison with the Random Forest baseline, especially within the agen and agentb family categories, where the Random Forest has an F1-score equal to or near zero, whereas MalBERT-Temporal successfully detects all instances. However, the differences from the CNN model are relatively smaller, with a mean improvement of approximately +0.9 percentage points. This reflects closer parity between the two models for well-represented families. Moreover, decreased performance for some selected families, such as alinaos and aitinject, is associated with limited sample sizes in the test partition, where individual misclassifications have a disproportionate effect on the F1-score. Section IV-K quantifies this effect with confidence intervals, which show that the negative differences reported for these low-support families cannot be statistically distinguished from zero.

TABLE V. COMPARATIVE PER-FAMILY F1-SCORE (%) ACROSS MALBERT-TEMPORAL, CNN, AND RANDOM FOREST AT SECURITY AND STANDARD (0.5) THRESHOLDS FOR TWENTY REPRESENTATIVE MALWARE FAMILIES

Family	MalBERT (Sec)	MalBERT (0.5)	CNN (Sec)	CNN (0.5)	RF (Sec)	RF (0.5)	Imp. vs RF (Sec)	Imp. vs CNN (Sec)
Unknown	100.00	100.00	100.00	100.00	66.67	100.00	33.33	0.00
adload	100.00	100.00	100.00	100.00	100.00	100.00	0.00	0.00
aenjaris	92.86	95.35	80.00	75.00	42.11	88.89	50.75	12.86
agen	100.00	100.00	100.00	66.67	0.00	66.67	100.00	0.00
agensla	100.00	100.00	100.00	100.00	87.50	100.00	12.50	0.00
agent	98.95	99.16	91.93	91.44	74.81	91.93	24.15	7.02
agentb	100.00	100.00	100.00	100.00	0.00	100.00	100.00	0.00
agentp	98.31	100.00	96.55	94.74	84.62	94.74	13.69	1.75
agenttesla	96.00	96.00	86.96	86.96	76.19	91.67	19.81	9.04
agobot	100.00	100.00	100.00	100.00	100.00	100.00	0.00	0.00
agqr	100.00	100.00	100.00	100.00	57.14	100.00	42.86	0.00
ainslot	96.97	96.97	90.32	90.32	52.17	90.32	44.80	6.65
aitinject	91.89	97.44	97.44	97.44	40.00	94.74	51.89	-5.54
alinaos	88.89	88.89	100.00	100.00	57.14	88.89	31.75	-11.11
almanahe	100.00	100.00	100.00	100.00	66.67	100.00	33.33	0.00
alureon	100.00	100.00	100.00	100.00	100.00	100.00	0.00	0.00
androm	100.00	100.00	94.74	88.89	66.67	94.74	33.33	5.26
androme	100.00	100.00	100.00	100.00	100.00	100.00	0.00	0.00
andromeda	100.00	100.00	100.00	100.00	100.00	100.00	0.00	0.00
asacky	92.31	92.31	100.00	92.31	72.73	92.31	19.58	-7.69

The comparison of the security and standard (0.5) thresholds also highlights that MalBERT-Temporal exhibits relatively stable performance across operating points, while the Random Forest model is more sensitive to threshold selection, especially under strict false-positive constraints. Generally, the family-level results show that the observed aggregate improvements are broadly distributed across various malware families and are not concentrated in only a limited number of them.

F. Explicit Definition of the Feature-Token Groups

Table VI gives the full specification of the tokenization stage. The 2,381 ordered features F1-F2381 are split into 16 contiguous groups using a boundary set b(i), which is the integer part of $i \times 2381/16$ for $i = 0, 1, ..., 16$. Group Gi then covers features F(b(i−1) + 1) to F(b(i)). Since 2381/16 = 148.8125, 2,381 is not an integer multiple of 16, and the groups cannot all have the same width. The extra dimensions are spread along the sequence rather than collected into one over-long token: G1, G6, and G11 hold 148 features each, and the other thirteen groups hold 149 each, which gives $3 \times 148 + 13 \times 149 = 2{,}381$. Every feature belongs to one group only, none appears twice, and nothing is padded or truncated.

Groups of different widths are acceptable because each group has its own linear projection layer whose input size is set to the width of that group. The sixteen projections therefore map 148- or 149-dimensional inputs to the common model dimension of 256. No tuning was applied to the boundaries, and they do not come from any feature-selection, clustering, or attribution procedure. They simply follow the native ordering of the published BODMAS feature extractor, in which related descriptors already occupy contiguous index blocks. The tokenization thus inherits whatever grouping that ordering provides and adds no further prior knowledge. Section IV-H measures separately how sensitive the results are to the number of groups.

TABLE VI. EXPLICIT DEFINITION OF THE SIXTEEN FEATURE-TOKEN GROUPS

Group	First	Last	Dim.	Group	First	Last	Dim.
G1	F1	F148	148	G9	F1191	F1339	149
G2	F149	F297	149	G10	F1340	F1488	149
G3	F298	F446	149	G11	F1489	F1636	148
G4	F447	F595	149	G12	F1637	F1785	149
G5	F596	F744	149	G13	F1786	F1934	149
G6	F745	F892	148	G14	F1935	F2083	149
G7	F893	F1041	149	G15	F2084	F2232	149
G8	F1042	F1190	149	G16	F2233	F2381	149

G. Component-Wise Ablation Study

The architecture in Section III-B combines several elements: grouped feature tokens, independent per-group projections, positional embeddings, a CLS token, a four-layer Transformer

encoder, three-way pooling, and nonlinear activations. The aggregate results in Table II do not show which of these elements produces the observed improvement. A component-wise ablation was therefore run, removing or simplifying one element per configuration while keeping everything else fixed: the same pre-2020 training partition, the same validation partition, the same signed-logarithmic and robust-scaling pipeline, the same optimizer and learning-rate schedule, the same 2020 test set, and the same calibration rule, which selects the threshold whose validation false-positive rate is closest to 0.1%. The results are given in Table VII and Fig. 7.

TABLE VII. COMPONENT-WISE ABLATION OF MALBERT-TEMPORAL ON THE 2020 TEST HORIZON

Variant	F1%	ΔF1	FPR%	Acc%	Par. (M)	Min
− Positional embeddings	98.19	+0.02	0.207	98.31	4.00	1.14
Full model	98.17	—	0.279	98.29	4.01	1.23
− Nonlinearity (GELU)	97.95	−0.22	0.266	98.08	4.01	1.13
Mean pooling only	97.93	−0.24	0.124	98.07	3.87	1.13
− CLS token	97.76	−0.41	0.203	97.91	3.87	0.96
Single encoder layer	97.70	−0.47	0.074	97.86	1.64	0.41
− Tokenization (flat)	96.32	−1.85	0.085	96.62	2.03	0.29
− Self-attention	95.44	−2.73	0.090	95.85	1.37	0.37

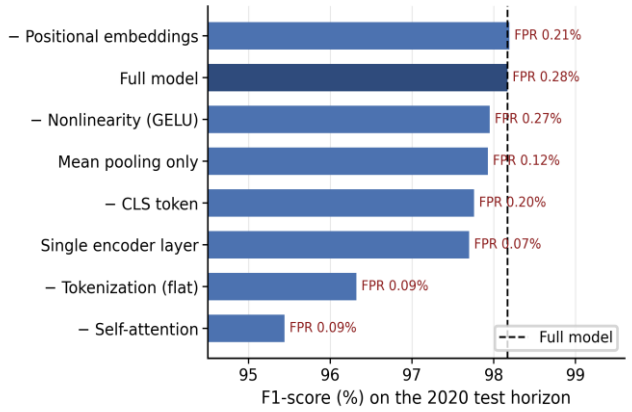

Fig. 7. Component-wise ablation of MalBERT-temporal.

Eight configurations were tested. The full model keeps all components. The "− tokenization" variant drops the grouping completely and feeds the 2,381-dimensional vector to a feed-forward network of similar width; this is the flat-vector control. The "− self-attention" variant keeps the tokenization, the per-group projections, the CLS token, the positional embeddings, and the pooling stage, and replaces only the Transformer encoder with a position-wise feed-forward block applied to each token separately, which separates the effect of self-attention from the effect of the surrounding components. The five remaining variants each make one change: the positional embeddings are removed; the CLS token is removed, and pooling uses the token mean only; mean pooling alone replaces the CLS/mean/max concatenation; identity mappings replace the GELU activations; and the encoder is reduced from four layers to one.

The two variants that change the tokenized-attention mechanism have by far the largest effect. Removing self-attention costs 2.73 F1 points, 95.44% compared with 98.17%, even though all auxiliary components are kept, and removing the tokenization costs 1.85 points, giving 96.32%. Each auxiliary component accounts for at most 0.47 points: a single-layer encoder costs 0.47 points, removing the CLS token costs 0.41 points, mean-only pooling costs 0.24 points, and removing the nonlinearity costs 0.22 points. Removing the positional embeddings changes the F1-score by +0.02 points, which is within run-to-run variation and shows that the order of the feature groups carries little usable positional information. The improvement reported in this study therefore comes from grouped tokenization combined with self-attention over those tokens, and not from the sum of the additional components.

H. Group-Granularity Ablation

The number of feature-token groups G is the design parameter that separates MalBERT-Temporal from a flat-vector Transformer, so the choice G = 16 needs empirical support. Four settings, $G \in \{4, 8, 16, 32\}$, were trained with the protocol used for the component ablation. The group boundaries were produced by the equal-width rule of Section IV-F, and all other hyperparameters were left unchanged. Table VIII repeats the untokenized flat-vector control from Table VII as a reference point, and Fig. 8 plots F1-score against training cost.

TABLE VIII. GROUP-GRANULARITY ABLATION: EFFECT OF THE NUMBER OF FEATURE-TOKEN GROUPS

Tokenization	F1%	FPR%	Acc%	Par. (M)	Min
Flat vector (none)	96.32	0.085	96.62	2.03	0.29
G = 4	97.92	0.124	98.06	3.99	0.48
G = 8	98.24	0.114	98.35	4.00	0.68
G = 16 (adopted)	98.17	0.279	98.29	4.01	1.10
G = 32	98.62	0.262	98.70	4.02	1.99

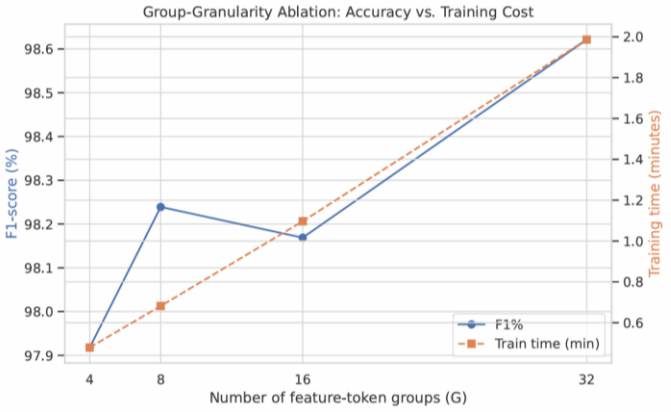

Fig. 8. Group-granularity ablation. F1-score (left axis) and training time (right axis) as a function of the number of feature-token groups G.

All tokenized configurations outperform the flat vector by a large margin. The weakest setting, G = 4, gives 97.92% against

96.32% for the untokenized control, a gap of 1.60 points, and all other settings perform better. The differences between the tokenized settings are small: G = 8 gives 98.24%, G = 16 gives 98.17%, and G = 32 gives 98.62%, a range of 0.45 points. Training time increases monotonically from 0.48 min at G = 4 to 1.99 min at G = 32, because both the number of independent projection layers and the length of the attention sequence increase with G. The gain from finer tokenization therefore saturates. The step from a flat vector to G = 4 already recovers 1.60 of the 2.30 points that separate the flat control from the best configuration, and the remaining 0.70 points require an eightfold increase in G and a fourfold increase in training time. G = 32 gives the highest F1-score in this single-seed comparison, and G = 16 is not claimed to be optimal: the differences between $G \in \{8, 16, 32\}$ are of the same size as the variation between nominally equivalent variants in Table VII. G = 16 is kept for three reasons. It sits in the flat part of the accuracy-cost curve, it keeps the token width close to the size of the contiguous descriptor blocks in the BODMAS feature layout, and it is the setting used for every other result in this study. A deployment that prioritizes accuracy could use G = 32 at about 1.8 times the training cost, and one that prioritizes cost could use G = 8 at about 0.6 times.

I. Random Versus Chronological Splitting

RQ1 asks how much performance loss chronological evaluation reveals that random splitting hides. Answering it requires a control in which only the rule for assigning samples to partitions changes. The control used here keeps the same architecture and hyperparameters, the same signed-logarithmic and robust-scaling pipeline with the scaler refitted on the new training partition, the same training budget, the same partition sizes of 34,751 training, 8,688 validation, and 87,277 test samples, the same calibration rule at a 0.1% validation false-positive-rate budget, and the same metrics. The only change is that the partitions are drawn by stratified random sampling over the full 2007-2020 corpus instead of by date, so 2020 samples can appear in training and pre-2020 samples can appear in the test set.

TABLE IX. RQ1 CONTROL EXPERIMENT: THE SAME MODEL UNDER RANDOM AND CHRONOLOGICAL SPLITTING

Protocol and operating point	Thr.	FPR%	Acc%	Prec%	Rec%	F1%
Random – Security	0.9740	0.116	99.01	99.85	97.89	98.86
Random – Standard	0.5000	0.251	99.41	99.68	98.98	99.33
Chronological – Security	0.9795	0.138	97.98	99.84	95.91	97.84
Chronological – Standard	0.5000	0.362	98.71	99.59	97.69	98.63
Inflation (Security)	—	−0.022	+1.03	+0.01	+1.98	+1.02

The two protocols are compared in Table IX, and Fig. 9 presents the same comparison as a figure. At the security operating point, the random protocol gives an F1-score of 98.86% against 97.84% for the chronological protocol, an inflation of 1.02 points. Recall increases from 95.91% to 97.89%, an inflation of 1.98 points, and precision is essentially unchanged. Measured as error instead of score, the random protocol removes 47% of the residual error at the security threshold, 1.14 points against 2.16 points. At the conventional 0.5 threshold, the inflation is smaller, 0.70 F1 points, but has the same sign.

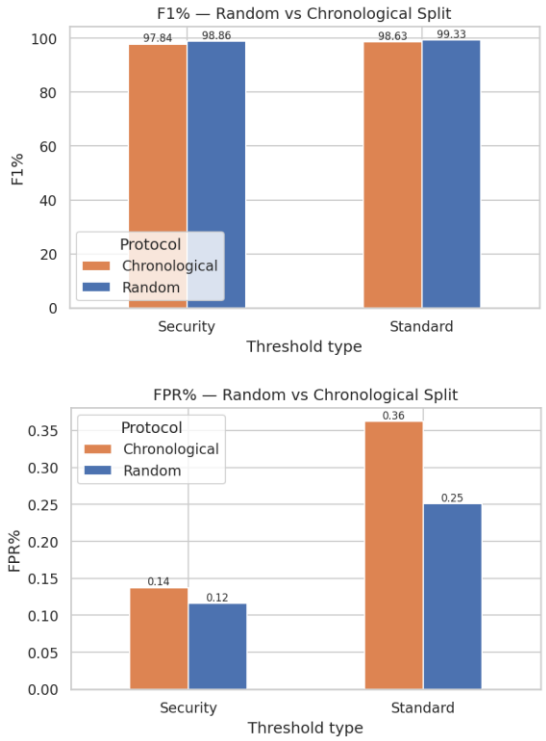

Fig. 9. F1-score and false-positive rate for the same MalBERT-Temporal under random-split vs. chronological-split protocols.

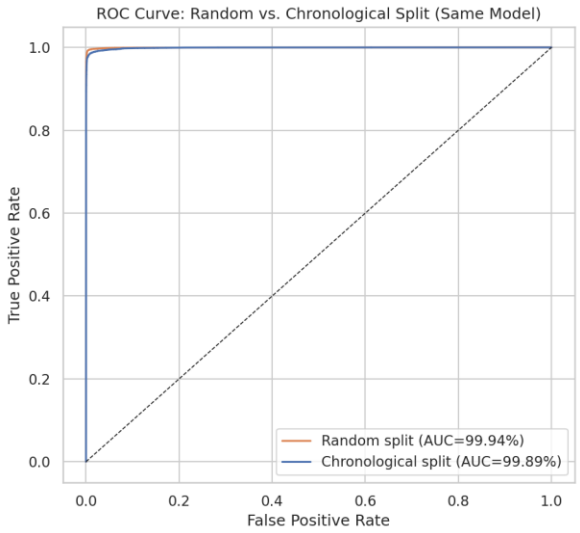

Fig. 10. ROC curves for MalBERT-Temporal under the random and the chronological protocols.

The threshold-free comparison behaves differently. Fig. 10 shows the two ROC curves, and their areas differ by only 0.044 points: 99.939% for random splitting and 99.895% for chronological splitting. Temporal leakage therefore has almost no effect on the ranking quality of the score function, but it does have a measurable effect on the calibrated operating point, which lies far out in the tail of the benign score distribution. This

matters for evaluation practice. Reporting only the area under the ROC curve would have hidden almost all of the effect that RQ1 asks about, since the effect appears only in metrics computed at a fixed low-false-positive operating point.

J. Retraining-Schedule Analysis

The claim that a temporally robust representation reduces the need for frequent retraining cannot be tested on a single fixed model; it requires a comparison of retraining schedules. Three schedules were compared over the nine months of the 2020 horizon. The static schedule is the one used elsewhere in this study: the pre-2020 model is frozen and scores all nine months with no update. The quarterly schedule fine-tunes the model at a learning rate of 1×10^{-4} at the end of March, June, and September, using the labeled samples of the quarter that has just ended. The monthly schedule applies the same fine-tuning step at the end of each month. In every schedule, a month is scored before any update that uses data from that month, so chronological order is preserved, and no prediction uses future information. The schedules do assume that ground-truth labels for a period are available as soon as that period ends. This is optimistic compared with operational labeling latency, so the comparison gives an upper bound on what retraining can achieve.

TABLE X. MONTHLY F1-SCORE (%) UNDER THREE RETRAINING SCHEDULES

Test month	Static	Quarterly	Monthly
01/20	97.99	97.99	97.99
02/20	97.64	97.64	98.60
03/20	97.71	97.71	98.13
04/20	98.49	99.03	99.45
05/20	98.00	98.43	97.75
06/20	98.79	99.09	99.54
07/20	96.59	95.77	94.03
08/20	98.16	95.99	99.50
09/20	97.33	92.91	99.50
Mean	97.86	97.18	98.28

Monthly F1-scores are given in Table X and Fig. 11. The static model averages 97.86% over the nine months, monthly retraining averages 98.28%, and quarterly retraining averages 97.18%. Monthly retraining therefore gains 0.42 F1 points on average over no retraining at all, and quarterly retraining is 0.68 points worse than no retraining.

The month-by-month results explain these averages. Monthly fine-tuning gives the largest gains in the last two months, raising August from 98.16% to 99.50% and September from 97.33% to 99.50%, which is where the drift literature expects a static model to be weakest. The gain is not uniform: in July the F1-score falls from 96.59% to 94.03%, because the update at the end of June adapts the model to a month whose composition differs from that of July. Quarterly fine-tuning is the least stable of the three schedules. After the update at the end of June, its scores fall to 95.77% in July, 95.99% in August, and 92.91% in September, which is consistent with over-adaptation to a narrow and recent labeled window at the expense of the wider pre-2020 evidence.

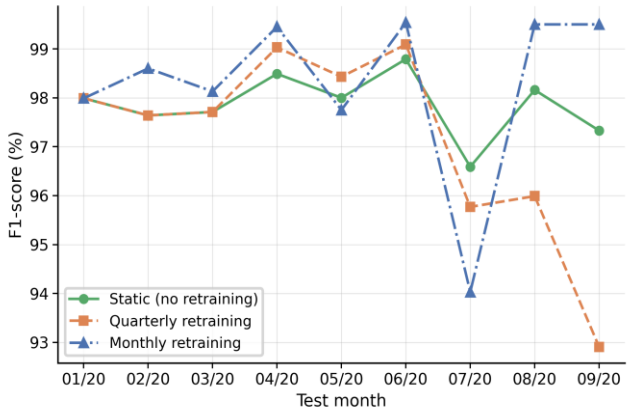

Fig. 11. Monthly F1-score of MalBERT-Temporal under three retraining schedules across the 2020 evaluation horizon.

Two conclusions follow, and the claim in the manuscript is narrowed accordingly. Over a nine-month horizon, the static model stays within 0.42 F1 points of the best schedule tested, so frequent retraining is not needed to maintain performance in this window. Quarterly fine-tuning is worse than no retraining at all, which indicates that periodic updates should be validated on a held-out window rather than adopted without checking.

K. Statistical Reliability of the Zero-Day Per-Family Estimates

The zero-day subset contains 1,572 of the 87,277 test samples, or 1.80% of the test horizon, and it is very unevenly distributed across families. One family, mintluks, provides 670 of these samples, and no other zero-day family provides more than 57. On supports of this size, per-family F1-scores are dominated by single misclassifications, so a point estimate alone overstates how precise the measurement is. A percentile bootstrap was applied to each family to quantify this. The family's test samples were resampled with replacement 1,000 times; the F1-score was recomputed on each resample at the fixed calibrated threshold of 0.9795, and the 2.5th and 97.5th percentiles of the resulting distribution were used as a 95% confidence interval. The threshold is held fixed throughout, so the intervals measure sampling uncertainty in the test set and not uncertainty in the calibration.

Table XI gives the intervals for the largest zero-day families and for the singleton families, and Fig. 12 covers all families with non-trivial uncertainty in their estimates. Interval width follows support closely. For mintluks (n = 670), the interval is 0.55 points wide, so the reported F1-score of 99.76% is reliable. For families with about forty to sixty samples, the intervals are 2.9 to 6.3 points wide, which still supports the qualitative conclusion that these families are detected well. Below about ten samples, the intervals are uninformative: families with two to eight samples routinely span 30 to 60 points, and several of them, including hynamer and foreign with three samples each, run from 0% to 100%. Families with a single sample give degenerate intervals of [100, 100] or [0, 0], which carry no statistical information at all.

TABLE XI. BOOTSTRAPPED 95% CONFIDENCE INTERVALS FOR PER-FAMILY ZERO-DAY F1-SCORES

Zero-day family	n	F1%	95% CI	Width
mintluks	670	99.76	[99.45, 100.00]	0.55
cosmicduke	57	99.10	[97.14, 100.00]	2.86
sillywnse	47	98.88	[96.29, 100.00]	3.71
qakbot	42	97.56	[93.67, 100.00]	6.33
kwbot	41	98.73	[96.00, 100.00]	4.00
Singletons (wannacry, worgtop, zonidel, …)	1	100.00	[100.00, 100.00]	0.00

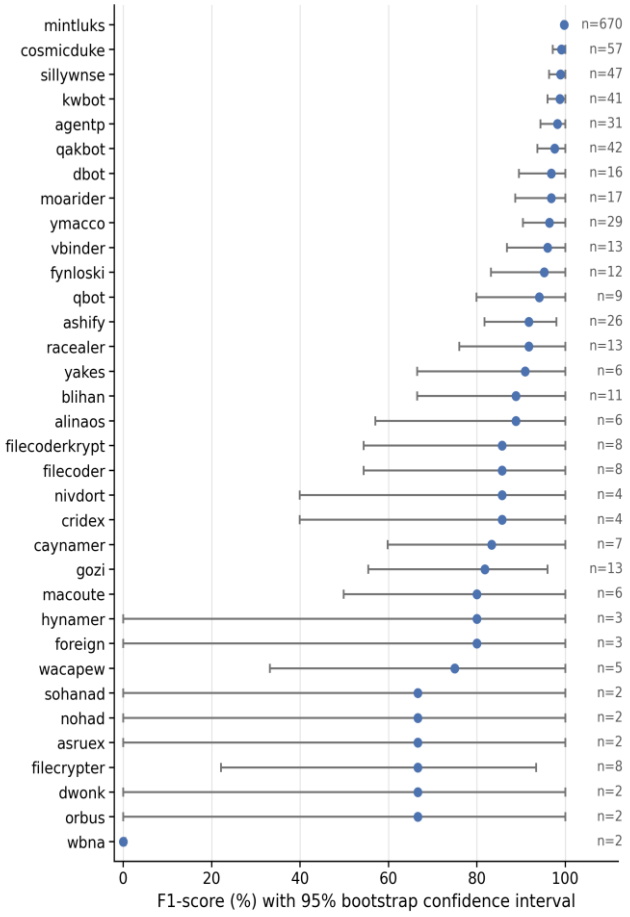

Fig. 12. Bootstrapped 95% CI for zero-day per-family F1-score.

This affects how Table V should be read. The families in that table where MalBERT-Temporal is reported to underperform the CNN all have small test supports.

For example, alinaos is a zero-day family with six test samples and a 95% interval that extends below 60 points (Fig. 12), so the difference of −11.11 points reported for it in Table V cannot be distinguished from zero. The aggregate zero-day conclusion is not affected, because it is dominated by the well-supported families, but per-family entries with small n should be read as indicative only. Future evaluations of zero-day generalization would benefit from a larger zero-day sample, or from reporting intervals together with point estimates as standard practice.

L. Feature-Level Drift and the Early-2020 False-Positive Excess

Table III shows that the realized false-positive rate exceeds its 0.1% validation budget in the first two months of the evaluation horizon, at 0.44% in January and 0.92% in February, and stays at or below 0.04% from March onwards. The absolute counts show the concentration even more clearly than the rates do: of the 63 false positives produced over the whole 2020 horizon, 26 occur in January and 34 in February, so 95% of all false alarms fall in the first two months and the remaining seven months produce three in total. The deviation is confined to a specific period and does not grow with temporal distance, so it is more likely a property of the early-2020 sample population than a progressive failure of the model. This section tests that explanation directly.

All 2,381 features were tested for a change between the pre-2020 training partition and the January and February test samples. Two measures were used: a two-sample Kolmogorov-Smirnov test, for which the training side was subsampled to 2,000 observations to keep the computation tractable, and a standardized mean shift expressed in units of the training standard deviation. The five features with the strongest shift in each month are listed in Table XII, and Fig. 13 shows the fifteen strongest.

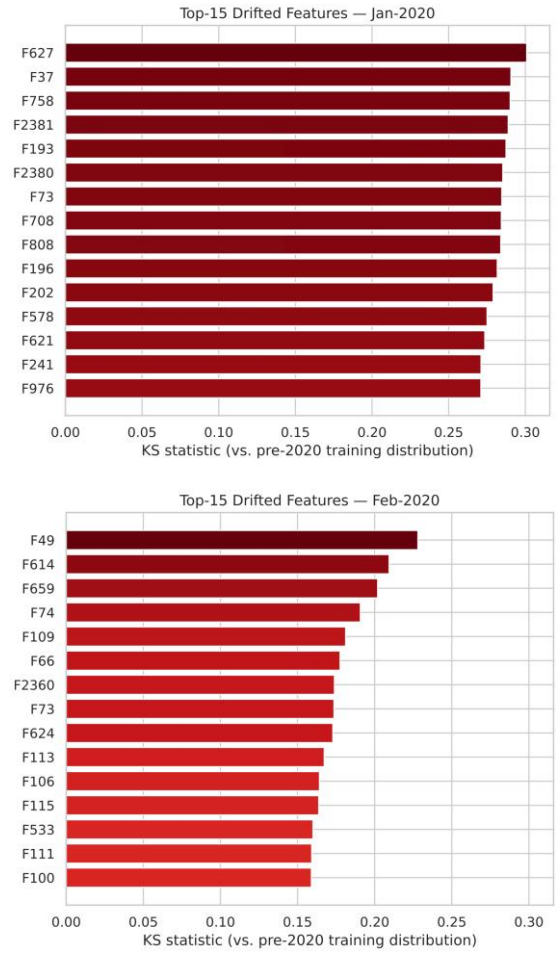

Fig. 13. Feature-level distributional drift between the pre-2020 training partition and the January and February 2020 test months.

TABLE XII. FIVE MOST STRONGLY DRIFTED FEATURES IN JANUARY AND FEBRUARY 2020 RELATIVE TO THE PRE-2020 TRAINING DISTRIBUTION

Period	Feature	KS	p-value	Mean shift
Jan-2020	F627	0.3009	2.9×10^{-135}	−0.124
Jan-2020	F37	0.2906	5.0×10^{-126}	−0.220
Jan-2020	F758	0.2902	1.3×10^{-125}	−0.473
Jan-2020	F2381	0.2889	1.8×10^{-124}	−0.000
Jan-2020	F193	0.2874	3.3×10^{-123}	−0.325
Feb-2020	F49	0.2281	1.9×10^{-73}	−0.234
Feb-2020	F614	0.2094	7.0×10^{-62}	+0.015
Feb-2020	F659	0.2020	1.4×10^{-57}	+0.346
Feb-2020	F74	0.1907	2.3×10^{-51}	−0.245
Feb-2020	F109	0.1812	2.4×10^{-46}	−0.237

Both months differ from the training distribution far beyond sampling variation. In January, the largest KS statistic is 0.301, for F627, and the fifteen most affected features lie between 0.271 and 0.301, with p-values below 10^{-108}. In February, the largest is 0.228 for F49, with the top fifteen between 0.159 and 0.228 and p-values below 10^{-35}. The affected features are not randomly scattered. They cluster in three regions: the low-index header and section-property block (F37, F49, F66, F73, F74, F100-F115, F193-F241), the mid-range import and string block (F533-F808, F976), and the tail of the byte-entropy histogram (F2360, F2380, F2381). The largest standardized shifts in these regions are +1.06 for F976, −0.47 for F758, and −0.46 for F202.

The fixed calibration is what turns this shift into a false-positive excess. The operating threshold of 0.9795 is the 0.1% false-positive quantile of the pre-2020 validation score distribution, so it lies far into the upper tail of the benign score distribution, where the density is low and the cumulative probability changes quickly for small displacements. A shift that moves even a small fraction of benign scores upwards therefore causes a disproportionate change in the realized false-positive rate while leaving accuracy, precision, and recall almost unchanged. This is the pattern seen in Table III: in February the false-positive rate is 9.2 times its budget, but the F1-score for that month is 97.64%, which is inside the band observed across the whole horizon. February also has the third-smallest benign population of the horizon, 3,708 samples, so the same number of tail crossings produces a higher rate than it would in a larger month.

The data-side explanation is also supported by independent evidence. All three systems evaluated in this study have their highest false-positive rate of the whole horizon in February 2020 and their second-highest in January: MalBERT-Temporal at 0.92% and 0.44% (Table III), the CNN baseline at 15.83% and 5.18% (Fig. 3c), the Random Forest baseline at 2.13% and 1.05% at its standard threshold (Fig. 4c). These systems cover the Transformer, convolutional and tree-ensemble families, and they share only the preprocessing pipeline and the calibration rule. A false-positive peak that appears in all of them in the same month, and in no other month, is therefore explained much more consistently by the benign population of January and February 2020 than by any single architecture.

For deployment, this limits the concern rather than removing it. The budget is met in seven of the nine months with no intervention, and the two months that exceed it are the two closest to the training cut-off, so the excess does not grow with temporal distance and does not indicate a progressive drift failure. It does show that a threshold calibrated once on historical data cannot be assumed to keep its nominal false-positive rate on incoming data. A practical deployment could recalibrate the threshold periodically on a small recent sample of confirmed benign files, which requires no retraining of the model, or monitor the KS statistics of the drift-prone features listed in Table XII and trigger recalibration when they exceed a preset bound. Neither option is evaluated in this study; both are given as recommendations, not as results.

V. DISCUSSION

These results are closely related to the concept drift problem that prompted the creation of the BODMAS dataset. Yang et al. [6] introduced BODMAS as a temporal stress-testing benchmark in which gradient-boosted decision trees and deep neural networks were trained on earlier temporal partitions and evaluated on subsequent time periods, revealing a decrease in monthly F1-score as the evaluation progressed. In their analysis, the greatest performance deterioration occurred in August and September 2020, when false-negative rates reached up to 43% for previously unseen malware families. These findings highlighted the sensitivity of static classifiers to temporal distribution shift and the need for periodic retraining.

In the current research, the same evaluation horizon is examined under identical chronological constraints, with a particular focus on the impact of architectural choices on temporal robustness. The results suggest measurable differences among the models under the same temporal evaluation setting.

Table XIII compares the performance of MalBERT-Temporal with the binary classifiers reported in the BODMAS paper over the overlapping evaluation period from January to September 2020 under comparable low-false-positive operating conditions.

TABLE XIII. MONTHLY F1 (%) VS. BODMAS BINARY CLASSIFIERS OF YANG ET AL. [6], 2020 TEST MONTHS (FPR ≈ 0.1%)

Test month	Ember-GBDT	SOREL-DNN	SOREL-GBDT	MalBERT
01/20	93.69	95.41	96.31	97.99
02/20	93.43	93.23	95.73	97.64
03/20	95.75	95.98	98.14	97.71
04/20	96.98	97.30	98.90	98.49
05/20	97.50	96.03	98.64	98.00
06/20	97.76	96.74	98.94	98.79
07/20	96.38	93.86	98.68	96.59
08/20	92.93	85.85	95.99	98.16
09/20	92.14	82.88	95.70	97.33
Mean	95.17	93.03	97.45	97.86

There are two key findings. First, MalBERT-Temporal achieves a mean monthly F1-score of 97.86%, which is slightly higher than that of the best-performing classifier in the BODMAS paper, SOREL-GBDT (97.45%), despite being trained on a substantially smaller dataset (43,439 pre-2020 samples compared with approximately twenty million samples used in SOREL-GBDT). This indicates that improved representation learning may partially offset differences in training data scale in this setting.

Second, differences in temporal behavior are observed in the later evaluation months. The performance of the classifiers in the original BODMAS paper decreases in August and September 2020, consistent with the drift behavior discussed in [6]. Specifically, SOREL-GBDT decreases to 95.99% and 95.70%, SOREL-DNN to 85.85% and 82.88%, and Ember-GBDT to 92.93% and 92.14%, respectively. In contrast, MalBERT-Temporal displays relatively stable performance, achieving 98.16% and 97.33% during the same period.

Three factors may contribute to the observed differences: representational capacity, evaluation protocol, and calibration strategy. From a representational perspective, MalBERT-Temporal models structured PE features as grouped token embeddings and applies self-attention to capture interactions between non-adjacent feature groups. This design allows the model to learn interactions among heterogeneous feature groups, such as section-level statistics and import-table characteristics. In contrast, the Random Forest baseline relies on axis-aligned decision boundaries over individual features, while the CNN baseline primarily captures local feature interactions constrained by its receptive field. The ablation results in Section IV-G provide direct evidence for this representational argument instead of simply asserting it. Removing self-attention while keeping the tokenization, the projections, the CLS token, and the pooling stage lowers the F1-score by 2.73 points, and removing the tokenization lowers it by 1.85 points, while no other component of the architecture is worth more than 0.47 points. The advantage over the baselines therefore comes from the mechanism the paper attributes it to, and not from the additional components that surround that mechanism.

All models were evaluated under a strict chronological partition, in which training was confined to pre-2020 samples, and testing was conducted on temporally subsequent data. This procedure eliminates temporal leakage and provides an evaluation setting that is aligned with deployment conditions.

Furthermore, all models were assessed by means of a unified thresholding strategy based on a fixed validation false-positive-rate budget of 0.1%. This approach ensures consistent operating conditions across the architectures and reflects the practical constraints of malware detection applications, in which false positives must be strictly controlled. The calibration strategy also has a limitation that the monthly results make visible. The threshold is fixed once on pre-2020 data, so the realized false-positive rate can exceed its budget whenever the incoming benign population shifts, and this is what happens in January and February 2020. Section IV-L shows three things about these two months: their feature distributions are the ones that differ most from the training partition, they contain 60 of the 63 false positives recorded across the whole horizon, and three structurally unrelated models calibrated by the same rule all exceed the budget in the same two months. The deployability claim made here should therefore be understood as conditional on periodic recalibration of the operating threshold, an operation that needs only a small sample of recent confirmed benign files and no retraining of the model.

Further limitations should also be noted. The evaluation spans a nine-month horizon, and Section IV-J shows that quarterly fine-tuning can perform worse than no retraining, so update schedules should be validated on a held-out window before adoption rather than applied on a fixed calendar. The method also relies exclusively on static PE features, so packed or obfuscated binaries attenuate the structural signal on which the tokenized representation depends; combining it with lightweight dynamic or byte-level features is the natural mitigation. Per-family zero-day estimates below roughly ten samples likewise carry confidence intervals too wide to support conclusions, as quantified in Section IV-K, which a larger zero-day sample or the routine reporting of intervals would remedy.

Tables II and IV present the results obtained under identical preprocessing, temporal partitioning, and evaluation procedures. Under these controlled conditions, MalBERT-Temporal achieves higher overall performance and statistically significant improvements in monthly F1-score over both baselines.

VI. Conclusion

This study has investigated whether the representation of structured PE features as a token sequence within a Transformer network is capable of improving zero-day Windows malware detection, relative to conventional baseline methods, under a temporally consistent evaluation protocol. Using a strict chronological split of the BODMAS dataset, where the training data include samples prior to 2020 and evaluation is conducted exclusively on 2020 data, MalBERT-Temporal achieved an F1-score of 97.84% with a false-positive rate of 0.138%. The model maintained relatively stable performance over the nine-month testing period, with its monthly F1-scores varying within a narrow range, including periods when the baseline models exhibited noticeable degradation. Furthermore, MalBERT-Temporal outperformed a 1-D CNN and a Random Forest model trained and tested under the same conditions, with statistical significance confirmed using Welch's t-test, and achieved performance comparable to or exceeding that of strong baselines reported in the BODMAS paper, despite being trained under a more constrained data regime.

The key findings of this study are as follows. First, representing structured PE features as a token sequence and applying self-attention enables the modeling of long-range feature interactions, which contributes to improved generalization to previously unseen malware families. Second, Transformer-based representations demonstrate greater robustness under temporal distribution shift. Across the nine-month horizon examined, a model that is never retrained stays within 0.42 F1 points of the best of the three retraining schedules tested, so performance in this window can be maintained without frequent retraining. The quarterly schedule nevertheless shows that periodic fine-tuning can also reduce performance, and no claim is made about horizons longer than nine months. These findings address Research Question 1 (RQ1) by showing

that strict chronological evaluation reveals performance degradation that is not observable under random splitting, while also indicating that MalBERT-Temporal maintains comparatively stable performance under such conditions. A matched control experiment in which only the partition rule changes quantifies degradation of 1.02 F1 points and 1.98 recall points at the calibrated security threshold, whereas the area under the ROC curve differs by only 0.044 points. The effect is thus visible at a fixed low-false-positive operating point and almost invisible in threshold-free metrics. They also address Research Question 2 (RQ2) by demonstrating consistent improvements over the CNN and Random Forest baselines under identical temporal constraints. Therefore, the novelty of this study does not lie in the development of a new learning paradigm but in a representation-level redesign of structured PE analysis that improves robustness under temporal drift.

Potential directions for future research include extending the study by evaluating longer temporal horizons beyond 2020, performing multi-seed experiments to quantify variance, integrating static PE features with lightweight dynamic or byte-level representations, and analyzing the attention patterns in greater detail to improve interpretability. Further evaluation of the method's robustness under adversarial evasion scenarios would also strengthen the practical applicability of the approach. Overall, the findings provide evidence that attention-based modeling of structured PE features is a promising direction for temporally robust malware detection.

REFERENCES

[1] S. Berrios, D. Leiva, B. Olivares, H. Allende-Cid, and P. Hermosilla, "Systematic review: malware detection and classification in cybersecurity," Appl. Sci., vol. 15, no. 14, Art. no. 7747, 2025, doi: 10.3390/app15147747.

[2] M. Alshomrani, A. Albeshri, B. Alturki, F. S. Alallah, and A. A. Alsulami, "Survey of Transformer-based malicious software detection systems," Electronics, vol. 13, no. 23, Art. no. 4677, 2024, doi: 10.3390/electronics13234677.

[3] D. Z. Syeda and M. N. Asghar, "Dynamic malware classification and API categorisation of Windows Portable Executable files using machine learning," Appl. Sci., vol. 14, no. 3, Art. no. 1015, 2024, doi: 10.3390/app14031015.

[4] Y. Chen, Z. Ding, and D. Wagner, "Continuous learning for Android malware detection," in Proc. 32nd USENIX Security Symp. (USENIX Security 23), Anaheim, CA, USA, 2023, pp. 1127–1144, doi: 10.48550/arXiv.2302.04332.

[5] A. S. Li, A. Iyengar, A. Kundu, and E. Bertino, "Revisiting concept drift in Windows malware detection: adaptation to real drifted malware with minimal samples," in Proc. Network and Distributed System Security Symp. (NDSS), San Diego, CA, USA, 2025, doi: 10.48550/arXiv.2407.13918.

[6] L. Yang, A. Ciptadi, I. Laziuk, A. Ahmadzadeh, and G. Wang, "BODMAS: an open dataset for learning based temporal analysis of PE malware," in Proc. 2021 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 2021, pp. 78–84, doi: 10.1109/SPW53761.2021.00020.

[7] P. Wang, T. Lin, D. Wu, J. Zhu, and J. Wang, "TTDAT: two-step training dual attention Transformer for malware classification based on API call sequences," Appl. Sci., vol. 14, no. 1, Art. no. 92, 2024, doi: 10.3390/app14010092.

[8] L. Qian and L. Cong, "Channel features and API frequency-based Transformer model for malware identification," Sensors, vol. 24, no. 2, Art. no. 580, 2024, doi: 10.3390/s24020580.

[9] Q. Lu, H. Zhang, H. Kinawi, and D. Niu, "Self-attentive models for real-time malware classification," IEEE Access, vol. 10, pp. 95970–95985, 2022, doi: 10.1109/ACCESS.2022.3202952.

[10] N. Labied, H. Benbrahim, and A. Amine, "A deep learning-based model for malware detection using Transformer architecture," in Proc. 10th Int. Conf. Optimization and Applications (ICOA), Almeria, Spain, 2024, pp. 1–6, doi: 10.1109/ICOA62581.2024.10753854.

[11] E.-J. Kim, Y.-K. Lee, S.-M. Lee, J.-N. Kim, A. R. Kang, M.-S. Kim, and Y.-S. Jeong, "Malware detection using pre-trained Transformer encoder with byte sequences," PLoS ONE, vol. 20, no. 10, Art. no. e0332307, 2025, doi: 10.1371/journal.pone.0332307.

[12] R. B. Tallane, A. U. Priantoro, and R. Muhida, "Development of a model for malware detection and classification at the byte level based on Transformer," IIUM Eng. J., vol. 27, no. 1, pp. 142–159, 2026, doi: 10.31436/iiumej.v27i1.4009.

[13] M. Alshomrani, A. Albeshri, A. A. Alsulami, and B. Alturki, "An explainable hybrid CNN–Transformer architecture for visual malware classification," Sensors, vol. 25, no. 15, Art. no. 4581, 2025, doi: 10.3390/s25154581.

[14] S. Alzahrani, Y. Xiao, S. Asiri, N. Alasmari, and T. Li, "RansomFormer: a cross-modal Transformer architecture for ransomware detection via the fusion of byte and API features," Electronics, vol. 14, no. 7, Art. no. 1245, 2025, doi: 10.3390/electronics14071245.

[15] M. Miraoui and M. B. Belgacem, "Binary and multiclass malware classification of Windows Portable Executable using classic machine learning and deep learning," Front. Comput. Sci., vol. 7, Art. no. 1539519, 2025, doi: 10.3389/fcomp.2025.1539519.

[16] L. Dhanya, R. Chitra, and A. M. Anusha Bamini, "Hybrid deep-learning approach with Transformer integration for unknown malware detection in IoT environments," J. Sens. Sci. Technol., vol. 34, no. 4, pp. 289–296, 2025, doi: 10.46670/JSST.2025.34.4.289.

[17] A. Guerra-Manzanares and H. Bahsi, "Experts still needed: boosting long-term Android malware detection with active learning," J. Comput. Virol. Hacking Tech., vol. 20, pp. 901–918, 2024, doi: 10.1007/s11416-024-00536-y.

[18] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "MeMalDet: a memory analysis-based malware detection framework using deep autoencoders and stacked ensemble under temporal evaluations," Comput. Secur., vol. 142, Art. no. 103864, 2024, doi: 10.1016/j.cose.2024.103864.

[19] T. Becker, A.-J. Rousseau, M. Geubbelmans, T. Burzykowski, and D. Valkenborg, "Decision trees and random forests," Am. J. Orthod. Dentofacial Orthop., vol. 164, no. 6, pp. 894–897, 2023, doi: 10.1016/j.ajodo.2023.09.011.

[20] B. Casella, R. Esposito, A. Sciarappa, C. Cavazzoni, and M. Aldinucci, "Experimenting with normalization layers in federated learning on non-IID scenarios," IEEE Access, vol. 12, pp. 47961–47971, 2024, doi: 10.1109/ACCESS.2024.3383783.

[21] S. Phunsa and T. Chomsiri, "LSGELU: improved Gaussian error linear units for enhanced performance in simple and complex neural networks," J. Adv. Inf. Technol., vol. 17, no. 3, pp. 477–487, 2026, doi: 10.12720/jait.17.3.477-487.

[22] M. S. Islam and L. Zhang, "A review on BERT: language understanding for different types of NLP task," Preprints.org, 2024, doi: 10.20944/preprints202401.1857.v1.

[23] J. Kim, B. Lee, C. Park, Y. Oh, B. Kim, T. Yoo, S. Shin, D. Han, J. Shin, and K. M. Yoo, "Peri-LN: revisiting normalization layer in the Transformer architecture," in Proc. 42nd Int. Conf. Machine Learning (ICML), vol. 267, 2025, pp. 30400–30436, doi: 10.48550/arXiv.2502.02732.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in Neural Information Processing Systems, vol. 30, 2017, pp. 5998–6008, doi: 10.48550/arXiv.1706.03762.

[25] C. F. G. dos Santos and J. P. Papa, "Avoiding overfitting: a survey on regularization methods for convolutional neural networks," ACM Comput. Surv., vol. 54, no. 10s, pp. 1–25, 2022, doi: 10.1145/3510413.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas,

A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html

[27] N. J. Outmezguine and N. Levi, "Decoupled weight decay for any p norm," arXiv:2404.10824, 2024, doi: 10.48550/arXiv.2404.10824.

[28] G. Vrbančič and V. Podgorelec, "Efficient ensemble for image-based identification of pneumonia utilizing deep CNN and SGD with warm restarts," Expert Syst. Appl., vol. 187, Art. no. 115834, 2022, doi: 10.1016/j.eswa.2021.115834.

[29] J. de Winter, "The risks of defaulting to Welch's t-test with unequal sample sizes and skewed distributions," unpublished.