MalGPT: A Generative Explainable Model for Malware Binaries

Mohd Saqib[1][0000−0003−2125−2162], Benjamin C. M. Fung[1][0000−0001−8423−2906] (✉), Steven H. H. Ding[1], and Philippe Charland[2]

[1] McGill University, 3661 Peel Street, Montreal, Canada
mohd.saqib@mail.mcgill.ca, {ben.fung, steven.h.ding}@mcgill.ca
[2] Defence R&D Canada, Quebec, Canada
philippe.charland@drdc-rddc.gc.ca

Abstract. Explaining malware binaries poses significant challenges, as existing approaches often focus on surface-level features, dynamic behaviors, or assembly code analysis. While these models highlight features contributing to classification, they remain inaccessible to non-experts. In this work, we propose MalGPT, a multi-model and transformer-based approach that generates human-readable explanations of malware binaries in natural language. We manually analyzed malware binaries from different malware families, including benign files, using various tools to create a ground truth dataset with high-level explanations. As per the literature, this is the first contribution of a malware dataset paired with natural language explanations, along with a high-level explanatory model developed for the cybersecurity community. Our approach includes complex feature engineering, followed by a novel architecture, Cross-Hierarchical Attention Network (CHAiN), which learns relationships not only within individual features, but across different feature sets in a multi-model architecture. We developed a Generative Pretrained Transformer (GPT)-style architecture optimized for multi-modal malware binary analysis, designed to seamlessly integrate heterogeneous features, such as numeric data, printable strings, and graph-based representations of assembly code. The architecture aligns syntactic structures with semantic context, to transform encoded multi-modal inputs into coherent and precise explanations. This innovative approach enhances compatibility with diverse data modalities, providing robust and interpretable insights into malware behavior, while enabling detailed and contextually accurate textual explanations. In future work, we aim to scale this approach with larger datasets, enhancing its capacity to explain emerging malware variants and address different cybersecurity landscapes, such as malicious apps or network viruses, ultimately contributing to risk mitigation.

Keywords: GPT Architecture · Explainable AI · Malware Analysis · Large Language Model.

1 Introduction

Deep learning (DL)-based malware detection has received considerable attention in recent years, achieving state-of-the-art detection metrics [4]. However, DL models often face the inherent challenge of being black-box systems, making their decisions difficult to interpret. Reducing the opacity of malware classification models is particularly complex, as traditional explainable AI (XAI) methods, commonly developed for natural language processing and image datasets, fail to capture the intricacies of malicious binaries. Such data require specialized algorithms for effective interpretation.

Saqib et al. [15] identified the limitations of traditional XAI methods and categorized the research into different levels of explanation, concluding that most efforts remain at the basic level, offering limited insight into the decision-making process. While some studies attempted to explain malware behaviors using existing XAI methods [4], others proposed specialized solutions for malware binaries [7][3][14]. For instance, I-MAD [7] introduced an interpretable neural network that reveals the importance of static features, but neglects dynamic aspects or deep code connectivity. Similarly, CFGExplainer [3] highlighted potentially malicious nodes in the control flow graph (CFG) of a malware executable, but lacked robust explanation capabilities, as shown by Saqib et al. [15] in their GAGE model. GAGE [14] proposed a novel representation of portable executable (PE) files, incorporating semantic and structural information through the Canonical Executable Graph (CEG). However, none of these models provide high-level explanations accessible to users without a cybersecurity background. To address these shortcomings, we propose a novel approach that leverages extensive feature extraction and the generative power of Generative Pretrained Transformer (GPT), providing human-understandable explanations for malware binaries.

The proposed algorithm is powered by a multi-model architecture coupled with GPT. The idea behind the multi-model design is to integrate all possible contexts of a malicious file. Previous algorithms have primarily focused on either static or dynamic features, which can miss subtle but crucial information required for explainability. Malware authors often use obfuscation and packing techniques to disguise their intent. Our model overcomes these limitations by considering not only static and dynamic features, but also the semantic and syntactical information extracted from the assembly code. Saqib et al. [14] highlighted that the CEG is more powerful than traditional CFGs. We extended this concept by extracting CEGs from binaries and integrating them into our model. Furthermore, we used a GPT architecture to generate human-readable explanations for these binaries. The encoded vector passed to the GPT model is generated by Cross-Hierarchical Attention Network (CHAiN), which captures all possible contexts and relationships, both inner and intra-feature, for comprehensive explanation generation.

To the best of our knowledge, this is the first work that provides such high-level explanations for malware binaries, as the malware analysis community previously lacked a ground truth explanation dataset for malicious binaries [4]. Our contributions can be summarized as follows:

- MalGPT contributes a unique GPT-style architecture tailored for multi-modal analysis of malware binaries. It integrates diverse feature types, including static features, dynamic behaviors, printable strings, and graph-based representations, into a unified generative framework. This architecture utilizes attention mechanisms to process and align syntactic and semantic relationships across these modalities, enabling both accurate detection and detailed natural language explanations. By bridging multi-modal feature integration with interpretability, MalGPT advances the applicability of GPT models in cybersecurity.
- We introduced CHAiN, an integral component of MalGPT, designed as an advanced multi-model architecture. This architecture leverages hierarchical attention mechanisms to effectively capture both intra-modality and inter-modality dependencies among heterogeneous malware feature modalities. By modeling intricate relationships across features such as static data, dynamic behaviors, and CEGs, CHAiN significantly enhances both the discriminative capability and interpretability of the model, outperforming existing state-of-the-art approaches in malware analysis and explainability tasks.
- We curated a comprehensive dataset utilizing 23.296 GB of binaries, encompassing four malware families (`DownloadAdmin`, `Firseria`, `Emotet`, `Gamarue`) and a benign category. This dataset includes ground truth explanations and diverse features, such as static, dynamic, API call sequences, import/export structures, and CEGs. An attention-based encoder-decoder (AED) was trained on 0.8 million assembly blocks and 28.7 GB of binaries, representing 2,411 CEGs with an average of 546 nodes and 3,567 edges, capturing syntactic and semantic relationships. Integrated with VirusTotal-generated reports, the dataset uniquely delivers natural language explanations for malware behaviors.
- Through comprehensive experiments, we evaluated MalGPT's performance for both detection accuracy and explainability. Quantitative metrics included precision, recall, F1 score, accuracy, ROUGE-L, and BERTScore to assess detection and explanation quality. For qualitative evaluation, we analyzed hallucinations and errors in the generated explanations, focusing on consistency, relevance, and the ability to align with ground truth annotations. This dual evaluation approach ensured a holistic understanding of MalGPT's strengths and areas for improvement in malware detection and explainability.

2 Background

The use of explainable methods in malware analysis has gained significant attention, as traditional "black-box" models offer limited interpretability. Comprehensive surveys, such as [15], highlight various interpretability frameworks, analyzing their benefits and limitations. Saqib et al. [15] provide a categorization of explainability approaches, which can be broadly divided into low-level and high-level explainability techniques. These approaches, however, have yet to fully leverage the potential of large language models (LLMs), which could significantly enhance interpretability for diverse stakeholders with minimal domain knowledge.

2.1 Low-Level Explainability Approaches

Low-level approaches in explainability often rely on local interpretability techniques, such as Local Interpretable Model-Agnostic Explanations (LIME) [13] and Shapley Additive Explanations (SHAP) [9]. These methods approximate the model locally around a specific instance, providing insights into feature importance and the reasoning behind the model's decisions.

Studies leveraging LIME have demonstrated its utility in identifying static malware features. For example, Khan et al. used LIME to explain Conv-LSTM-based autoencoder predictions on network traffic features in industrial IoT environments [4]. Similarly, Kinkead et al. [5] used LIME on Android APK opcode sequences, transforming them into image-based representations for CNN classifiers, while Lu and Thing [8] used LIME alongside BERT-based models for Android application feature extraction. Ambekar et al. [1] utilized a combination of TabNet and LSTM in their model, TabLSTMNet, to fuse LIME-based explanations with DL predictions. Mitchell et al. [10] further extended LIME and introduced hierarchical LIME (H-LIME) to generate sparser explanations at the class and method levels.

SHAP-based studies also illustrate the interpretability of malware features. For instance, Lu and Thing [8] applied SHAP alongside LIME to Android applications, while To et al. [17] used SHAP on PE file analysis, extracting features via VirusTotal and classifying them using logistic regression, decision trees, and k-nearest neighbors. Other studies, such as Smmarwar et al. [16] and Ambekar et al. [1], used hybrid DL approaches such as CNN-BiGRU with SHAP to elucidate malware predictions.

2.2 High-Level Explainability Approaches

In contrast to low-level techniques, high-level approaches focus on model-wide interpretability and behavior explanation. For example, Herath et al. [3] proposed CFGExplainer, a model using control flow graph (CFG) blocks and opcode-based statistics to explain malware behavior comprehensively. Building upon CFGExplainer, Saqib et al. [14] introduced the Genetic Algorithm-based Graph Explainer (GAGE), which extends the control-flow analysis by identifying malicious functions and their caller-callee relationships within a call execution graph (CEG). GAGE demonstrates improved robustness and discriminative power, particularly in explaining complex malware behaviors.

2.3 Exploring Large Language Models (LLMs)

LLMs, especially transformer-based architectures, have seen limited application in malware explainability, though their potential is considerable. Encoder-decoder structures, such as those in [11][19], have been applied primarily for detection rather than for explanation. BERT-based models have also been explored, though primarily for feature extraction rather than interpretability. For instance, Ullah et al. [18] used BERT to generate features for an ensemble detection model.

Demirkiran et al. [2] demonstrated that transformers are effective in classifying highly imbalanced malware families, leveraging the attention mechanisms of transformers to model complex sequence relationships within malware API call data. However, despite these advancements, the application of LLMs for generating interpretable, high-level explanations in malware analysis remains largely unexplored, presenting a promising direction for future research.

3 Proposed Method

In this section, we introduce MalGPT, a novel architecture combining CHAiN and GPT for malware analysis and explanation generation (Figure 1). MalGPT leverages comprehensive features engineering mechanism, followed by CHAiN to effectively process and integrate heterogeneous data types, such as PE header features, API call sequences, and graph-based CEG representations, ensuring both intra- and inter-modality relationships are captured. The fused representations are then passed to a GPT module, which generates comprehensive, human-readable explanations, enhancing transparency and interpretability in malware detection.

3.1 Features Engineering Mechanism

To enable comprehensive analysis of binary files, we extracted four distinct feature types, each capturing a unique aspect of the binaries.

PE Header Features: This category includes PE header features such as `ExportsNbDLL`, `SectionsMeanEntropy`, and `ResourcesMeanSize`. Categorical data within this group were encoded using a label encoder to ensure compatibility with the model.

Import/Export Features: We categorized import/export functionalities into four subcategories: `open`, `create`, `delete`, `close`, `resume`, `kill`, `call`, and `other`. These were encoded using the SBERT model [12], which generates embeddings for similar lists with high semantic similarity. After obtaining individual embeddings, we performed aggregation (average) to produce a uniform-sized vector representation for individual subgroups.

Printable Strings: This feature type contains extensive lists ranging from 500 to 2,000 entries per binary. Categories include `DIR`, `email`, `sentences`, `keywords`, `files`, `IP addresses`, and `URLs`. Using the same SBERT-based encoding [12] strategy as for import/export features, we created subcategory-specific embeddings and aggregated them into a fixed-length vector.

CEG Construction: This feature involves extracting assembly code from binary files and transforming it into a graph representation. Each node in the

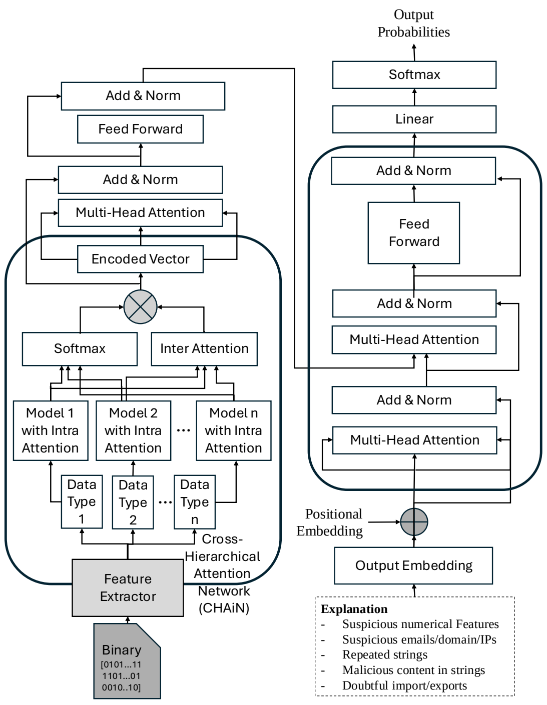


Fig. 1. MalGPT architecture

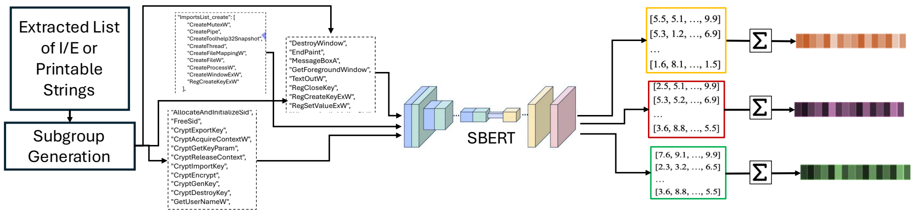
CEG encapsulates a set of instructions, which were encoded into numerical vectors using the *PalmTree*[3] model. These instruction-level vectors were aggregated at the node level using an AED architecture, effectively capturing both local and sequential information. The AED model, which integrates the strengths of autoencoders and sequence-to-sequence architectures, was trained on a dataset of 0.8 million assembly code blocks, each capped at 512 instructions. Using these embeddings, CEGs were constructed based on the definitions outlined by Saqib et al. [14]. Subsequently, we trained a graph encoder to generate dense vector representations for the entire graph. This encoder was trained on a dataset of 28.7 GB, comprising 2,411 CEGs from benign and malicious binaries. On average, each CEG contained 546 nodes and 3,567 edges, capturing intricate structural and behavioral information from the binaries.

3.2 CHAiN Network

Binary files are composed of diverse feature types, such as numerical values (e.g., floats or integers), labeled features, sequences (e.g., API call lists), and assembly code extracted into CEGs [14]. The CEG representation uniquely integrates

[3] https://github.com/palmtreemodel

both syntactical and semantic information into node embeddings, while edge features encapsulate the structural attributes of PE files. To effectively learn from this heterogeneous dataset and capture both intra-modality and inter-modality dependencies, we introduce the CHAiN.



```
"ImportsList_create": [
    "CreateMutexW",
    "CreatePipe",
    "CreateToolhelp32Snapshot",
    "CreateThread",
    "CreateFileMappingW",
    "CreateFileW",
    "CreateProcessW",
    "CreateWindowExW",
    "RegCreateKeyExW"
  ],
```

```
"DestroyWindow",
"EndPaint",
"MessageBoxA",
"GetForegroundWindow",
"TextOutW",
"RegCloseKey",
"RegCreateKeyExW",
"RegSetValueExW",
```

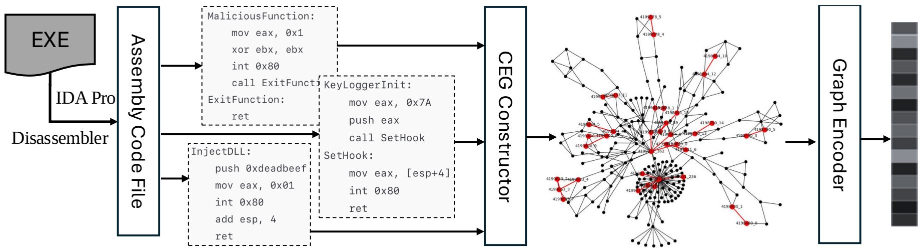
```
"AllocateAndInitializeSid",
"FreeSid",
"CryptExportKey",
"CryptAcquireContextW",
"CryptGetKeyParam",
"CryptReleaseContext",
"CryptImportKey",
"CryptEncrypt",
"CryptGenKey",
"CryptDestroyKey",
"GetUserNameW",
```

```
[5.5, 5.1, …, 9.9]
[5.3, 1.2, …, 6.9]
…
[1.6, 8.1, …, 1.5]
```

```
[2.5, 5.1, …, 9.9]
[5.3, 5.2, …, 6.9]
…
[3.6, 8.8, …, 5.5]
```

```
[7.6, 9.1, …, 9.9]
[2.3, 3.2, …, 6.5]
…
[3.6, 8.8, …, 5.5]
```

Fig. 2. Encoding mechanism for list of import/export and printable strings.



```
MaliciousFunction:
    mov eax, 0x1
    xor ebx, ebx
    int 0x80
    call ExitFuncti
ExitFunction:
    ret
```

```
KeyLoggerInit:
    mov eax, 0x7A
    push eax
    call SetHook
SetHook:
    mov eax, [esp+4]
    int 0x80
    ret
```

```
InjectDLL:
    push 0xdeadbeef
    mov eax, 0x01
    int 0x80
    add esp, 4
    ret
```

Fig. 3. Encoding mechanism for assembly code extracted from binaries.

Intra-Modality Attention For each feature modality, specialized neural architectures are used to capture intra-relations among similar data features.

Given a modality input $\mathbf{X}$, intra-modality attention computes the weighted representation $\mathbf{H}$ as:

$$\mathbf{H} = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}, \tag{1}$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are the query, key, and value matrices, respectively, and d_k is the dimensionality of the keys.

Inter-Level Dependency Learning After obtaining feature representations from each modality, an inter-modality attention mechanism is applied to capture relationships across different data types:

$$\mathbf{Z} = \sum_{i=1}^{M} \alpha_i \mathbf{H}_i, \tag{2}$$

where $\mathbf{H}_i$ represents the feature vector from the i-th modality, M is the number of modalities, and α_i are the learned attention weights.

This cross-modality attention captures dependencies such as the significance of printable string patterns when specific PE header attributes are present. The final fused representation $\mathbf{Z}$ can be obtained through bilinear pooling:

$$\mathbf{Z}_{\text{fused}} = \phi(\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_M), \tag{3}$$

where ϕ denotes the bilinear transformation.

3.3 GPT Architecture Integration

The fused representation $\mathbf{Z}_{\text{fused}}$ is input into a GPT for explanation generation. The transformer uses attention mechanisms to model dependencies and generate textual explanations:

$$\text{Output}_t = \text{softmax}\left(\text{Linear}(\text{Decoder}(\mathbf{Z}_{\text{fused}}))\right), \tag{4}$$

where Decoder represents the transformer decoder block. The generated attention weights from intra- and inter-modality layers enable interpretable explanations, highlighting which features or interactions influenced the final prediction.

Collectively, the CHAiN encoder and the GPT-based decoder constitute the MalGPT framework. MalGPT takes a binary executable as input, processes it through CHAiN to generate numerical embeddings, and then utilizes the GPT decoder to translate these embeddings into natural language explanations.

4 Preprocessing and Evaluation

4.1 Dataset Details

The final dataset comprises approximately 23.296 GB of data distributed across 1,702 files, categorized into five distinct groups: four malware families (`Gamarue`, `DownloadAdmin`, `Emotet`, `Firseria`) and a set of benign samples. Feature extraction was augmented through analyst reports retrieved from VirusTotal[4], which served as the foundation for generating human-readable explanations using the ChatGPT API[5].

To overcome the lack of labeled natural language explanations for malware behavior, we used ChatGPT to generate analyst-style descriptions based on

[4] `https://www.virustotal.com/`

[5] OpenAI, ChatGPT API, accessed November 05, 2024, `https://platform.openai.com/`

VirusTotal reports retrieved via MD5 hashes of known malware and benign binaries. These reports included behavioral features such as API calls and registry edits, and were verified to ensure accuracy and minimize hallucinations. Prompt templates were crafted to guide ChatGPT's output in producing consistent and interpretable explanations. For details, refer to the GitHub repository[6]. This constitutes the first dataset pairing malware binaries with human-readable explanations, used solely to pretrain the MalGPT language module. The CHAiN-based classification relied exclusively on real features and labels, with all synthetic outputs reviewed to avoid bias and preserve model integrity.

The dataset was split into 80% training and 20% testing data. The training data was further divided into an 80-20 split for training and validation purposes, ensuring robust evaluation and minimal overfitting during model development.

4.2 Parameter Tuning

The MalGPT model was trained using a manually tuned encoder-decoder architecture optimized for malware explanation generation. Feature values were normalized using `StandardScaler`, missing data were mean-imputed, and explanation texts were tokenized and padded to a maximum length of 200. The encoding approach follows Section 3.1. Table 1 summarizes the core hyperparameters.

Table 1. Key Hyperparameters for MalGPT. LR: Learning Rate, CE: Crossentropy.

Parameter	**Value**
Vocab Size	10,000
Max Seq. Length	200
Transformer Layers	32
Attention Heads	32
Embedding Size	1024
Dropout Rate	0.1
Batch Size	64
Optimizer	Adam (adaptive LR)
Loss	Sparse Categorical CE
Epochs	50

Manual tuning was chosen over automated methods due to the architecture's complexity and computational constraints, balancing generalization and efficiency.

4.3 Training Process of MalGPT

The MalGPT model is trained in an end-to-end manner using binary executables and their corresponding ground truth explanations. Each binary is pro-

[6] `https://github.com/McGill-DMaS/MalGPT`

cessed through a multi-modal encoding pipeline that extracts heterogeneous feature types—such as static headers, import/export functions, printable strings, and CEGs. These features are encoded using intra-modality attention, fused via inter-modality attention, and transformed into a unified numerical embedding. This embedding is then passed to a GPT-style decoder, which learns to generate human-readable explanations. The model is optimized by minimizing the difference between the generated and actual explanations.

Algorithm 1 Compact Training Procedure of MalGPT

```
Require: Binary set $\mathcal{B} = \{b_i\}$ and explanations $\mathcal{Y} = \{y_i\}$
1  for all $b_i \in \mathcal{B}$ do
2      Extract features: $f_i = \{f_i^{static}, f_i^{imp}, f_i^{str}, f_i^{ceg}\}$
3      Encode: $e_i^m \leftarrow \text{Model}_m(f_i^m)$ for $m \in$ {static, imp, str, ceg}
4      Intra-attend: $h_i^m \leftarrow \text{IntraAttn}(e_i^m)$
5      Refine: $r_i^m \leftarrow \text{MLP}(h_i^m)$
6      Fuse: $z_i \leftarrow \text{InterAttn}(\{r_i^m\})$
7      Decode: $\hat{y}_i \leftarrow \text{GPTDecoder}(z_i)$
8      Loss: $\mathcal{L}_i \leftarrow \mathcal{L}(\hat{y}_i, y_i)$
9  end for
10 Update model to minimize $\sum_i \mathcal{L}_i$
```

4.4 Performance Evaluation

The proposed model's performance was assessed in two aspects: its discriminative power and the merits of its explanations. For discriminative power, standard metrics such as precision, recall, F1 score, and accuracy were employed to measure the model's ability to correctly classify malicious and benign binaries.

Table 2. Comparison of Precision (P), Recall (R), F1 Score, False-Positive Rate (FPR), False-Negative Rate (FNR), and Accuracy for each baseline. A: Only static features, B: Without import export,, and C: Without CEG.

Features	P	R	F1 Score	Acc	FPR	FNR	ROUGE-L	BERTScore
A	0.920	0.812	0.764	0.9002	0.0266	0.2028	0.0544	0.7701
B	0.932	0.804	0.772	0.9208	0.0198	0.1982	0.0549	0.7741
C	0.984	0.970	0.984	0.9794	0.0058	0.0120	0.0527	0.7752
CHAiN	0.988	0.984	0.988	0.9882	0.0048	0.0026	0.0820	0.8140

To evaluate explanation correctness quantitatively, we employed ROUGE-L[7] and BERTScore [20]. ROUGE-L was selected for its ability to capture long-sequence dependencies and align key segments between generated and reference

[7] `https://huggingface.co/spaces/evaluate-metric/rouge`

explanations. In contrast, BERTScore uses contextual embeddings to measure semantic consistency, providing a robust evaluation of nuanced relationships and achieving strong correlation with human judgment, even in paraphrased scenarios [20]. These complementary metrics ensured a comprehensive assessment of both technical precision and semantic alignment in the model's explanations.

For a qualitative assessment, metrics such as False Positive Rate (FPR) and False Negative Rate (FNR) were employed alongside manual inspection of hallucinated content in the generated explanations. This approach allowed for a detailed evaluation of areas where the model misinterprets or invents information, ensuring a comprehensive analysis of the model's explanation reliability and its interpretative consistency.

5 Results and Discussions

5.1 Ablation Study

To assess the contribution of each feature set, we conducted an ablation study by evaluating detection and explanation performance with different combinations of features. Initially, we performed experiments using only static or numerical features, followed by a combination of static features and CEG information (excluding import/export features), and static features with import/export features (excluding CEG). The findings revealed that the combination of static features and import/export features achieved satisfactory detection rates. However, explanation metrics remained relatively consistent across ablation settings until all feature types were combined (Table 2).

The integration of all feature sets—static, import/export, and CEG—yielded superior detection performance, marked by significantly reduced error rates (FPR and FNR). Additionally, the explanation metrics (e.g., ROUGE-L and BERTScore) improved notably, indicating that CHAiN successfully leverages both syntactical and semantic information to generate meaningful explanations for malware binaries. This comprehensive approach highlights CHAiN's robustness in both detection and explainability tasks.

5.2 Discriminative Power

We evaluated the discriminative power of the proposed CHAiN model across various malware families, including benign samples, and compared it against benchmark algorithms GAGE and CFGExplainer, which are state-of-the-art in malware detection. The results are summarized in Table 3.

The findings underscore CHAiN's ability to integrate syntactical and semantic information effectively, enhancing both detection and explanation capabilities, particularly for challenging malware families such as Gamarue. The proposed model achieves the highest accuracy (98%), followed by GAGE (94.13%) and CFGExplainer (90.03%). These results demonstrate its robustness and superiority over traditional models.

Table 3. Performance metrics comparison

Model	Precision	Recall	F1 Score	FPR	FNR	Accuracy
CHAiN	0.9880	0.9860	0.9860	0.0048	0.0026	0.9800
GAGE	0.9300	0.9140	0.9200	0.0156	0.0902	0.9413
CFGExplainer	0.8720	0.8180	0.8320	0.0378	0.1830	0.9003

5.3 Novelty and Plausibility

The proposed MalGPT framework demonstrates significant novelty in generating plausible malware explanations, as evidenced by the table. The comparison between actual and MalGPT-generated explanations highlights both the strengths and areas for improvement in the system. MalGPT exhibits an advanced ability to replicate key malicious attributes, integrating domain-specific knowledge from malware families, such as `Firseria`.

Explanation 1
MD5: C0A5DE8A89C236DD6071A6014065CAE8 **Malware Family:** Firseria **File Name:** origin_15301_ _ **Indication:** Red: Malicious intent or behavior. Magenta: Behavioral descriptions relevant to malware families.
Actual: The file is identified as part of the ransomware family, a notorious malware category impacting systems with sophisticated registry manipulations and evasive behaviors. Antivirus engines, including Microsoft, Bitdefender, and BKAV, confirm its presence, highlighting its ability to persist through browser modifications, bundled associations, and delivery mechanisms. Notable tactics include the exploitation of Solimba and Kryptik infrastructures, avoiding detection via sandbox analysis, and manipulating user libraries and APIs. The malware demonstrates domain persistence, alterations to whitelisted platforms, and leveraging vulnerabilities for propagation. Andromeda.exe links, 64-bit patched alterations, and HFSAdware integrations underscore its risks. It actively evades countermeasures by disabling zero-day detection systems, network functionality, and adopting downware techniques. Frequent administrative updates, . . . and regular updates to maintain system integrity against this persistent threat.
MalGPT: The file "andromeda.exe" is infected with a high-risk 64-bit adware named HFSAdware. This malware, which has some 49 distinct signs highlighting its presence, has visibly altered the system's policies for harmful purposes. This Andromeda variant propagates itself and notably supports domain persistence despite attempts to remove it or contain it. There are indications suggesting that this file is also associated with the invasive malware, Solimba. As part of its comprehensive preventive measures, it regularly checks its presence and updates itself,

further underscoring its persistence in infected systems. Certain interesting anomalies such as invalid downware policies and the presence of API confirmations, have emerged from recent discussions in various security platforms. Moreover, the malware is known to infect systems through misleading advertisements, displaying as essential updates. Identified as a part of the Kryptik family, the malware has shown a significant number of warning flags such as modifying critical system files and displaying unauthorized popup ads. Bias analysis from the 62 whitelisting results, major delivery programs, and several other indications highlight Andromeda's sophisticated evasion strategies. One such strategy includes disabling zero-day detection systems. These actions are likely triggered by the Firseria infection, which further stresses the need for categorizing and scanning files, programs, and links continuously. A comprehensive ... and injection prevention are vital for maintaining system integrity.

The `Firseria` malware family is a type of ransomware known for its ability to manipulate system registries and evade detection [6]. It often employs sophisticated techniques, such as API abuse, sandbox evasion, and persistent file modifications. MalGPT extends its explanatory depth by replacing browser-related references (as noted in the actual explanation) with more specific behaviors, such as `'domain persistence'` and API-focused activities. Notably, MalGPT highlights the involvement of the `Kryptik family` and `Andromeda.exe`, effectively capturing key attributes of `Firseria`'s infection strategies, including evasion techniques and API manipulation (see Explanation 1). This highlights `Firseria`'s use of sophisticated frameworks for evasion and malicious propagation. Such detailed interrelations demonstrate MalGPT's robust understanding of the binary's syntactical and semantic features.

The plausibility of MalGPT explanations is highlighted by its ability to match the key evidence in actual explanations. For example, in the `Firseria` family, MalGPT connects critical tactics such as `'persistence through browser manipulations'` and `'delivery mechanisms'`, which align closely with the ground truth descriptions. Similarly, in the benign sample, MalGPT accurately emphasizes secure characteristics, such as `'reliable systems'` and `'safe browsing policies'`, further bolstering its credibility.

6 Limitations

6.1 Hallucination and Error Analysis

We assess MalGPT's capabilities using both quantitative and qualitative analyses. The error metrics in Table 3 quantitatively highlight MalGPT's performance.

Explanation 2
MD5: 976BD72651F83785CA69FB74C3C5BBAE **Malware Family:**

Downloadadmin **File Name:** origin_31932_ _ **Indication:** Red: Malicious intent or behavior. Green: Benign characteristics. Magenta: Behavioral descriptions relevant to malware families. Yellow: Ambiguous or conflicting content blending malicious and benign contexts. Cyan: Hallucinated or unrelated content.

Actual: The file exhibits behavior characteristic of "DownloadAdmin" malware, known for its intrusive and deceptive activities. It employs obfuscation tactics to evade detection while distributing malicious payloads through misleading campaigns. Notable actions include intrusion attempts, unauthorized certificate usage, and deploying Potentially Unwanted Applications (PUAs) such as Firseria. The malware reinforces its presence via remote downloader capabilities, often altering system configurations to maintain persistence. Organizations must implement preventive defenses, such as Bitdefender or Kaspersky, to counteract its spread. Regular verification, awareness programs, and robust classification mechanisms remain crucial for mitigating this threat.

MalGPT: The explanation discusses several key aspects of a malware family or benign file characterized by its notable status and the comprehensive cyber campaigns it has been linked to. It's stated that this entity employs obfuscation and alterations as part of its deployment strategy. Also, it regularly reinforces its defenses and maintains a prominent presence by distributing itself through various channels. Moreover, it leverages ""Firseria"" and ""Dowadmin"" sections, along with a "Win32 downloader," to spread its influence. This can lead to an intrusion into Windows applications, including those associated with major organizations. Conventional defense mechanisms seem ineffective against it. It's known for being capable of evading detection by the likes of Bitdefender and Kaspersky. Though posing as a 'goodware', it has been classified as a PUA ('Potentially Unwanted Application') due to its intrusive behavior. Regularly educating users . . . this entity seems to belong under the 'intrusive' classification but remains notable within the cybersecurity landscape for its persistent behavior.

Explanation 3

MD5: 68AE5602E48B5457E22340D09F34BE23 **Malware Family:** Benign **File Name:** 0ad73180...52d629b3 **Indication:** Same as previous.

Actual: The file in question has been assessed by antivirus engines and classified as benign due to its lack of malicious indicators. It is identified as a PE32 executable for Windows systems, with no evidence of malicious activities, C2 communications, encryption, or known IoCs (Indicators of Compromise). Verified by engines e.g., Kaspersky and Symantec, the file does not demonstrate suspicious behaviors such as trojan-like persistence, data exfiltration, or anomalous DLL injections. However, alternate

analyses indicate some potential for exploitation in 'Fareit'-like mechanisms, including manipulation of DLLs, intrusive system libraries, or suspicious communication chains. Despite this, no confirmed malicious actions or tactics have been linked to the file, and its legitimacy remains supported by majority consensus. It is noted in whitelisting databases, lacks significant threats or anomalies, and remains flagged as benign in the absence of further evidence.
MalGPT: This explanation pertains to the intricate workings of malware and benign files. It presents an analysis of the disruptive AVG expired botnet, and the intrusiveness it displays. The explanation demonstrates the lack of notorious analysis in dealing with the cybersecurity threats posed by malware. The cornerstone is ensuring an effective response to the anomalous activities conducted by malware through a prominent mechanism, such as NVIDIA. In this discussion, sensitive common lists of suspicious cyber communications play a crucial role, with a particular focus on benign files. . . . The report incorporates various statistics and references to 'dll', or Dynamic Link Library sections, which are active in combating malware threats. With an increasing misuse of the DLLs, there is a need to enhance blocking mechanisms to prevent their manipulation. The document goes on to emphasize the significance of identifying, categorizing, and addressing threats, particularly those characterized by the Andromeda address. The explanation also . . . and the 'dowadmin' mechanisms. Mention of URLs indicates the network aspect of these threats. In conclusion, the review presents the sobering statistics regarding malware infections despite the many cybersecurity measures in place. However, it also maintains trust in sophisticated mechanisms and clean programs, such as the 'sobrab' and 'hfsadware', in maintaining cybersecurity around the globe.

By analyzing both the *Actual Explanation* and *MalGPT Explanation*, it is evident that while the model captures key malicious behaviors effectively, it also introduces hallucinated information and occasionally misclassifies intent (Explanation 2, and 3). Using the highlighted text in the examples, we can systematically evaluate these issues:

Distinguishing contextual relevance: MalGPT demonstrates the ability to identify malicious activities such as obfuscation and downloading behavior for `DownloadAdmin`. However, it introduces hallucinated elements, such as references to `"Firseria sections"`, which are not explicitly relevant in the actual explanation (Explanation 2). This misattribution might stem from the presence of the term `"Firseria"` in the actual explanation, where it was used in a different context. This highlights a limitation in distinguishing contextual relevance during explanation generation.

Distinguishing intention relevance: In the `benign` example, hallucinated content introduces contradictions. For instance, phrases, `"workings of malware and benign files"` create confusion by blending opposing concepts (Explanation 3). While MalGPT correctly highlights benign characteristics, such as `"with a particular focus on benign files,"` it simultaneously incorporates unrelated malicious terms, such as `"botnet,"` `"Andromeda,"` and `"dowadmin mechanisms."` This inconsistency likely arises from contextual overlaps in the training data. The intermixing of benign and malicious indicators underscores the need for enhanced precision when distinguishing intent categories to ensure coherent explanations.

6.2 Data Dependency

A key limitation of the MalGPT framework lies in its dependency on the quality and diversity of the input data. Although the dataset includes malware binaries and corresponding explanations generated with the assistance of the ChatGPT API, the reliability of these explanations is contingent upon the accuracy of the underlying analyst reports sourced from VirusTotal. While efforts were made to minimize hallucinations by extracting verified malware descriptions using MD5 hashes, the dataset remains limited in size and scope, potentially affecting generalization to unseen or novel malware families.

Additionally, some of the explanation patterns exhibit redundancy, which may introduce noise into the training process. Future work should focus on curating a larger, more diverse dataset by incorporating alternative sources—such as Amazon Trūk or hybrid threat intelligence platforms—and ensuring balanced coverage of malware behaviors. Enhancing dataset robustness will further improve the model's ability to handle previously unseen features and reduce the risk of explanation artifacts.

7 Conclusion

Explaining malicious binaries remains a challenging domain due to the complexity and diversity of malware behaviors. The proposed model, MalGPT, represents a significant step forward as the first approach to generate detailed explanations for malware binaries from scratch, focusing on both their behaviors and intentions. MalGPT outperforms state-of-the-art methods, achieving high accuracy in malware detection with exceptionally low error rates. Beyond detection, it excels in generating high-quality explanations, quantitatively and qualitatively validated, while also contributing a novel dataset tailored for this task.

Although hallucinations remains a concern, leading to occasional inaccuracies in explanations, this limitation can be mitigated in future iterations. Enhancements such as improved contextual embeddings and fine-tuning on more extensive labeled datasets will help refine the model's interpretability and reliability. The results demonstrate MalGPT's potential to transform the field of explainable AI for cybersecurity, bridging gaps in malware analysis and enabling more informed decision-making.

Supplementary Information The dataset, source code and appendix containing detailed implementation information are available in the Data Mining and Security (DMaS) Lab GitHub repository[8].

Acknowledgments. The research is supported in part by NSERC Discovery Grants (RGPIN-2024-04087), NSERC Collaborative Research and Training Experience (CREATE-554764-2021), and Canada Research Chairs Program (CRC-2019-00041).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Ambekar, N.G., Devi, N.N., Thokchom, S., Yogita: Tablstmnet: enhancing android malware classification through integrated attention and explainable ai. Microsystem Technologies pp. 1–19 (2024). https://doi.org/10.1007/s00542-024-05615-0
2. Demirkıran, F., Çayır, A., Ünal, U., Dağ, H.: An ensemble of pre-trained transformer models for imbalanced multiclass malware classification. Computers & Security **121**, 102846 (2022)
3. Herath, J.D., Wakodikar, P.P., Yang, P., Yan, G.: Cfgexplainer: Explaining graph neural network-based malware classification from control flow graphs. In: 2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). pp. 172–184. IEEE (2022)
4. Khan, I.A., Moustafa, N., Pi, D., Sallam, K.M., Zomaya, A.Y., Li, B.: A new explainable deep learning framework for cyber threat discovery in industrial iot networks. IEEE Internet of Things Journal (2021). https://doi.org/10.1109/JIOT.2021.3130156
5. Kinkead, M., Millar, S., McLaughlin, N., O'Kane, P.: Towards explainable cnns for android malware detection. Procedia Computer Science **184**, 959–965 (2021). https://doi.org/10.1016/j.procs.2021.03.118
6. Lever, C., Kotzias, P., Balzarotti, D., Caballero, J., Antonakakis, M.: A lustrum of malware network communication: Evolution and insights. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 788–804 (2017). https://doi.org/10.1109/SP.2017.59
7. Li, M.Q., Fung, B.C., Charland, P., Ding, S.H.: I-mad: Interpretable malware detector using galaxy transformer. Computers & Security **108**, 102371 (2021). https://doi.org/10.1016/j.cose.2021.102371
8. Lu, Z., Thing, V.L.: "how does it detect a malicious app?" explaining the predictions of ai-based malware detector. In: 2022 IEEE 8th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS). pp. 194–199. IEEE (2022). https://doi.org/10.1109/BigDataSecurityHPSCIDS54978.2022.00045
9. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. p. 4768–4777. NIPS'17, Curran Associates Inc., Red Hook, NY, USA (2017)

[8] `https://github.com/McGill-DMaS/MalGPT`

10. Mitchell, J., McLaughlin, N., Martinez-del Rincon, J.: Generating sparse explanations for malicious android opcode sequences using hierarchical lime. Computers & Security **137**, 103637 (2024). https://doi.org/10.1016/j.cose.2023.103637
11. Rahali, A., Akhloufi, M.A.: Malbert: Malware detection using bidirectional encoder representations from transformers. In: 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC). pp. 3226–3231 (Oct 2021). https://doi.org/10.1109/SMC52423.2021.9659287
12. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (11 2019)
13. Ribeiro, M.T., Singh, S., Guestrin, C.: "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. p. 1135–1144. KDD '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939778, `https://doi.org/10.1145/2939672.2939778`
14. Saqib, M., Fung, B.C.M., Charland, P., Walenstein, A.: Gage: Genetic algorithm-based graph explainer for malware analysis. In: Proc. of the 40th IEEE International Conference on Data Engineering (ICDE). pp. 2258–2270. IEEE Computer Society, Utrecht, Netherlands (May 2024)
15. Saqib, M., Mahdavifar, S., Fung, B.C.M., Charland, P.: A comprehensive analysis of explainable ai for malware hunting. ACM Comput. Surv. **56**(12) (Oct 2024). https://doi.org/10.1145/3677374, `https://doi.org/10.1145/3677374`
16. Smmarwar, S.K., Gupta, G.P., Kumar, S.: Xai-amd-dl: An explainable ai approach for android malware detection system using deep learning. In: 2023 IEEE World Conference on Applied Intelligence and Computing (AIC). pp. 423–428. IEEE (2023). https://doi.org/10.1109/AIC57670.2023.10263974
17. To, T.N., Hoang, H.D., Duy, P.T., Pham, V.H.: Maldex: An explainable malware detection system based on ensemble learning. In: 2023 International Conference on Multimedia Analysis and Pattern Recognition (MAPR). pp. 1–6 (2023). https://doi.org/10.1109/MAPR59823.2023.10288922
18. Ullah, F., Alsirhani, A., Alshahrani, M.M., Alomari, A., Naeem, H., Shah, S.A.: Explainable malware detection system using transformers-based transfer learning and multi-model visual representation. Sensors **22**(18), 6766 (2022)
19. Xing, X., Jin, X., Elahi, H., Jiang, H., Wang, G.: A malware detection approach using autoencoder in deep learning. IEEE Access **10**, 25696–25706 (2022). https://doi.org/10.1109/ACCESS.2022.3155695
20. Zhang, T., Kishore, V., Wu, F., Weinberger, K.Q., Artzi, Y.: Bertscore: Evaluating text generation with bert. In: Proceedings of the International Conference on Learning Representations (ICLR). Cornell University and ASAPP Inc., Cornell Tech, New York, NY, USA (2020), `https://doi.org/10.48550/arXiv.1904.09675`