SRDC: Semantics-based Ransomware Detection and Classification with LLM-assisted Pre-training

Ce Zhou[1*], **Yilun Liu**[2 ✉], **Weibin Meng**[2], **Shimin Tao**[2], **Weinan Tian**[2], **Feiyu Yao**[2]
Xiaochun Li[2], **Tao Han**[2], **Boxing Chen**[3], **Hao Yang**[2]

[1]Northeastern University, China
[2]Huawei, China
[3]Huawei Canada, Canada

Abstract

In recent years, ransomware has emerged as a formidable data security threat, causing significant data privacy breaches that inflict substantial financial, reputational, and operational damages on society. Many studies employ dynamic feature analysis for ransomware detection. However, these methods utilize neither the internal semantic information (semantic information inherent in the features), nor external semantics (the wealth of existing knowledge and expert experience with regard to ransomware detection). Moreover, conventional methods rely on training data from known ransomware families, while zero-day ransomware often has unknown data distribution patterns, posing detection challenges. In this paper, we propose a **S**emantics-based **R**ansomware **D**etection and family **C**lassification (SRDC) framework that can utilize both internal and external semantics of software. To bolster semantic analysis in zero-day attacks, we also design a procedure called LLM-assisted task-adaptive pre-training (LATAP). In LATAP, ransomware semantics from human experts and LLMs are employed to pre-train the detection model (GPT-2). By fully utilizing semantics, the proposed SRDC framework outperforms the SOTA methods by 12.15% for ransomware family classification tasks, and by 4.03% for zero-day ransomware detection tasks. SRDC also exhibits excellent data efficiency, requiring only two ransom families for training, which is only 35% of the data required by existing methods, to achieve a 90%+ accuracy of zero-day ransomware detection in nine unseen ransom families.

Code — https://github.com/Michael-zhouce/SRDC
Datasets — https://github.com/Michael-zhouce/SRDC/tree/main/Pretraining_Corpus

1 Introduction

Ransomware is a type of malware that primarily aims to extort victims by encrypting their files or systems, subsequently demanding payment for decryption keys or data restoration. The rise and spread of this software pose a serious threat to societal security and data privacy. Ever since 1989 when AIDS Trojan, the first ransomware emerged, ransomware has diversified significantly (Bridges 2008). The

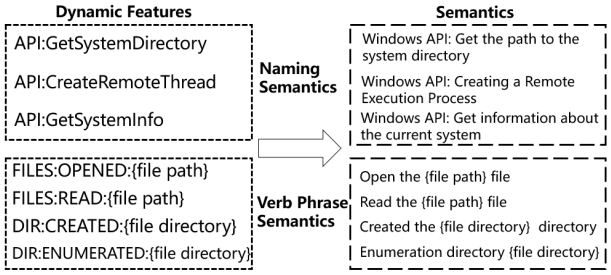

Figure 1: Internal semantics in dynamic features of ransomware. Our method utilizes internal semantics by reconstructing original features (left) to enhance semantic richness (right).

Internet has accelerated the spread of ransomware. In the past three decades, ransomware attacks have evolved into a significant threat, affecting economies and societies worldwide. Toyota Motor Corp. halted 28 production lines across 14 plants in Japan due to an attack on a supplier, impacting about one-third of its global production and resulting in a loss of approximately 13,000 vehicles. In February 2024, a ransomware gang that had infiltrated Hamilton's city government in Canada for months launched a large-scale attack on its IT system. This attack disrupted key IT facilities, including transportation, healthcare, public services, and emergency services, leaving the city in a state of shutdown (itworldcanada 2024). As ransomware attacks increasingly impact society, enhancing detection and classification mechanisms has become urgent.

Windows remains widely used by many companies and individuals (StatCounter 2024). Numerous studies have proposed solutions for detecting ransomware on Windows systems. Using natural language processing (NLP) algorithms to analyze dynamic software features is a promising approach. However, current methods have not fully leveraged the rich internal and external semantics in these features for ransomware detection and classification. This limitation increases dependency on training data and restricts the practical application of these methods in real-world scenarios. Specifically:

1) Inadequate use of the internal semantics of ransomware dynamic features. As shown in Fig. 1, the dy-

∗ Work done during an internship at Huawei.
✉ Corresponding author (liuyilun3@huawei.com).

Corpus	Details	Item
Windows system APIs	Functional description of the Windows system APIs, and the potential security issues that may arise from calling the APIs.	2133
Windows system registry	Functional description of the registry, including description of the registry keys and the system services associated with them.	8385
Introduction to Ransomware	Ransomware analysis reports written by security bloggers as well as security experts.	59

Table 1: External semantics that serve as pre-training corpus for LATAP.

namic features of software runtime are rich in semantics. For example, some API names seem like a combination of symbols ("GetSystemInfo"), but actually represent a meaningful sentence ("Get information about the current system"). These shorthand naming styles are convenient for human experts to use, but they increase the difficulty for AI models in understanding these semantics in ransomware detection and classification tasks (Lin et al. 2024; Liu et al. 2024). Some existing methods merely use NLP modules as feature encoders for subsequent classification tasks but fail to grasp the intent behind these software behaviors or the security risks they may pose (Li et al. 2022).

2) Inadequate use of the external semantics in pre-existing knowledge and experience related to ransomware detection tasks. Besides the internal semantics present in dynamic features, external semantics (Table 1) related to ransomware are also crucial. The reason why a human security expert can distinguish the existence of a ransomware from normal software, even when the captured dynamic features are from zero-day attacks, is precisely because he has a wealth of external knowledge and experience related to ransomware (*e.g.*, the purpose of the software's call to the API, or whether opening a system registry is a suspicious operation). However, existing methods (Lin et al. 2024; Liu et al. 2024) only use task-specific training samples to fit the patterns of ransomware, without leveraging relevant external semantics to guarantee the generalization ability of models.

3) Existing methods heavily depend on training data. Existing approaches rely solely on fitting to existing data. Although these approaches can perform well in known distributions, zero-day ransomware usually has an unseen underlying data distribution, making it challenging for these models to capture its features and patterns, and effectively address zero-day attacks.

In this paper, we propose an SRDC framework that fully utilizes internal semantics in software runtime features and external ransomware-related semantics. For internal semantics, as shown in Fig. 1, we design a feature processing procedure that reconstructs symbol combinations in the original features to make them more natural and semantically rich. For external semantics, as shown in Table 1, we first curated a corpus of ransomware-related semantic information with the assistance from both human experts and powerful LLMs. Using the collected corpus, we further design a pre-training stage that adapts the base language model (GPT-2) with task-related external semantics. In SRDC, since the model is pre-trained with external semantics, it can understand the internal semantics embedded in software runtime features more efficiently (which are reconstructed by our procedure to contain more semantics). Thereby, the reliance on labeled training samples for fine-tuning ransomware-related tasks can be relieved. Experimental results show that SRDC achieves SOTA results in both ransomware family classification and zero-day ransom detection tasks. Our contributions are summarized as follows:

- We propose LLM-assisted task-adaptive pre-training (LATAP), a novel unsupervised procedure for adapting language models with external semantics in ransomware tasks. Through LATAP, SRDC-GPT, the model trained from GPT-2, shows an 18.6% decrease in perplexity on unseen ransomware samples, and an improvement of up to 7.03% in performance on zero-day ransomware detection tasks.
- We propose a semantics-based ransomware detection and family classification (SRDC) framework that utilize both internal and external semantics of software features at runtime. By fully utilizing semantics, SRDC outperforms existing best detection methods by 12.15% for ransomware family classification tasks, and by 4.03% for zero-day ransomware detection tasks.
- We propose a novel few-shot training paradigm aimed at addressing the challenge of insufficient ransomware samples for zero-day attacks in real-world scenarios. Our approach achieves superior performance by utilizing only 35% of the training data compared to existing methods. Furthermore, using only two ransomware families as training data, SRDC can achieve an accuracy of over 90% in detecting nine unseen zero-day ransomware families.

2 Related Work

2.1 Traditional Approaches to Ransomware Detection and Classification

The signature-based approach is the most basic and traditional method of identifying potential ransomware threats by comparing a file or program to a specific pattern or fingerprint (signature) of a known virus (Sathyanarayan, Kohli, and Bruhadeshwar 2008). Static analysis methods include examining file characteristics such as PE header, file size, entropy, and opcode frequency without executing the file (Han et al. 2019; Manavi and Hamzeh 2020). Dynamic analysis involves running malware in a secure environment to extract behavioral features. These features are then analyzed using machine learning and deep learning techniques

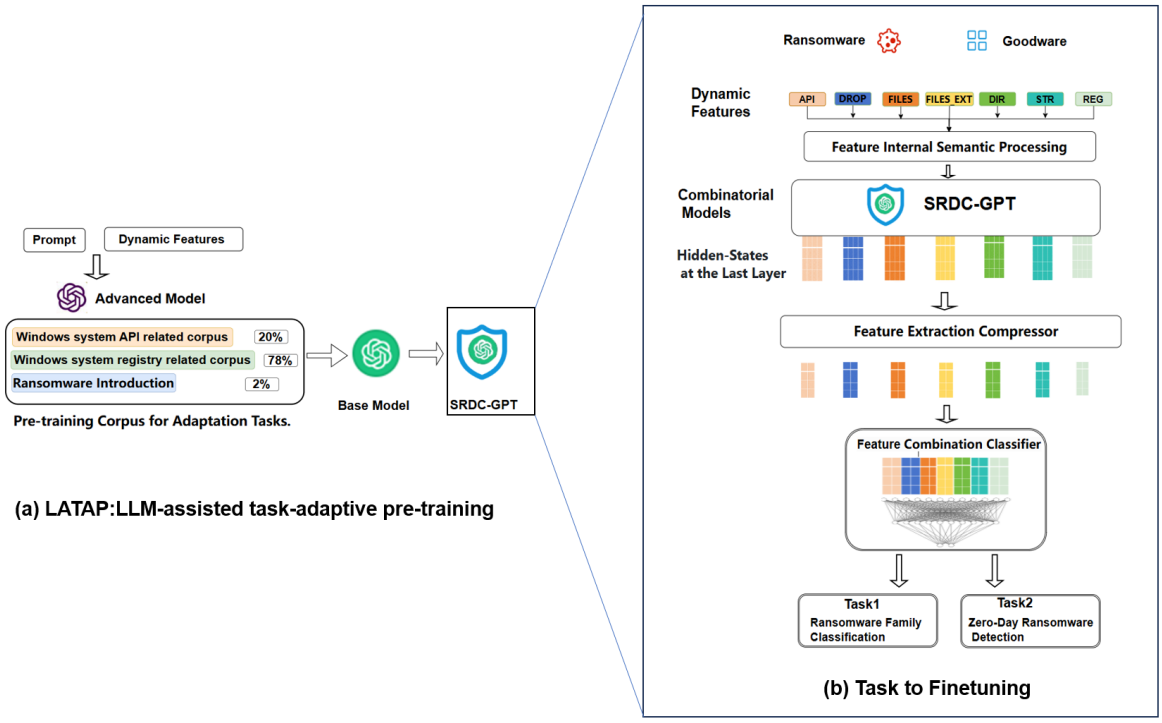

Figure 2: Overall workflow of SRDC. (a) the LATAP process. (b) the task fine-tuning process.

for ransomware detection and classification (Razaulla et al. 2023).

Signature-based methods rely on known malware fingerprint databases, which cannot effectively protect against new or variant ransomware (i.e., zero-day attacks). Several ransomware variants use sophisticated packing techniques, making them difficult to analyze statically. In contrast, dynamic analysis offers a clear advantage (Demetrio et al. 2021). We also use dynamic analysis methods to collect and analyze ransomware features.

2.2 Semantic-Based Approach

The earliest proposed method for semantic malware detection was a formal semantics-based approach. This method uses mathematical and logical techniques to precisely define the behavior and attributes of a program, while still relying on string-based template matching (Christodorescu et al. 2005; Preda et al. 2007). With the development of NLP, several machine learning and deep learning algorithms have been applied to this field. Van Nhuong et al. (2014) used N-gram models to encode the semantics of program code, while Zhang et al. (2023) employed the word embedding algorithm Skip-Gram to encode the extracted API sequence features. Li et al. (2022) used the raw sentences of API calls and semantic chain features to detect them using a Bi-LSTM approach. Qin, Wang, and Ma (2020) used TextCNN to encode text features. However, these methods merely serve as a means of encoding features and do not truly understand the behavioral intentions behind the internal characteristics of the software, nor do they grasp the potential security risks they pose.

With the advent of pre-trained large language models, the field of NLP has undergone an unprecedented transformation. Owing to their powerful semantic understanding and analysis abilities, LLMs perform very well in a variety of tasks and have spread widely across verticals, including cybersecurity. Some researchers have used LLMs like BERT, but mainly as basic encoders to analyze raw feature semantics (Lin et al. 2024; Liu et al. 2024). Our research also uses large models but differs in the following ways: (1) In addition to using internal semantic information from ransomware features, we incorporate external semantics, including expert knowledge on ransomware detection. (2) We also focus on evaluating our model's performance in zero-day ransomware detection.

3 Methodology

3.1 LLM-assisted Task-adaptive Pre-training

Training Data Generation Numerous experiments in this field have shown that task-adaptive pre-training (TAP), used with only a small corpus, can be very effective in improving the model's performance on specific tasks (Gururangan et al. 2020; Liu et al. 2023; Tao et al. 2023). In our study, we designed an LLM-assisted task-adaptive pre-training (LATAP) scheme, as shown in Fig. 2(a).

We first analyzed the task and capability boundaries that our model is intended to address, and constructed a high-quality dataset for the detection task and the knowledge that the base model lacks.

While analyzing the ransomware features collected in the dataset (Sgandurra et al. 2016), we noticed that the dataset includes the call information of Windows system APIs and details the ransomware's behavior of opening, reading, and modifying the Windows system registry. To better understand these behaviors, we collected official API descriptions from Microsoft (Microsoft 2024). We also searched for information on the Windows system registry. However, very little information could be found. To improve the corpus's efficiency and quality, we used a more advanced model (Ouyang et al. 2022) to generate additional data for augmenting our dataset. The specific prompt designs and the generated corpus have been open-sourced in our GitHub repository. To address potential model hallucinations, the generated corpus was manually reviewed by cybersecurity experts before being used for training. Four experts spent three days correcting errors and ambiguities, finding that 1.8% of the data was irrelevant or incorrect. The final corpus we used is detailed in Table 1. The corpus consists of 12% manually collected data and 88% data generated by more advanced models.

Note: We did not add samples of ransomware attacks (*i.e.*, ransom detection datasets) to the pretraining data to avoid unfair testing.

Pre-training We used GPT-2 (Radford et al. 2019) as the base model (Embedding Size = 768, Decoder layer = 12, Parameter number = 124M). There are several reasons why we choose a model with fewer parameters:

- Given the specificity of our tasks and domains, a simple, lightweight model can effectively accomplish the task.
- Smaller models are more cost-effective and easier to deploy and update, enabling faster real-world applications.
- High demand for AI applications has strained graphics card supply, increasing prices and causing instability. Smaller models, which are easier to pre-train, fine-tune, and deploy on consumer-grade hardware, help mitigate these challenges (Liao et al. 2024).

3.2 Task-specific Fine-tuning System

Our proposed system is shown in Fig. 2(b), where the captured dynamic features are first preprocessed by the Feature Semantic Processing module for feature internal semantic parsing and verb phrase construction, and then different kinds of processed features are fed into the SRDC-GPT respectively, and the output feature vectors are compressed and extracted by the feature extraction compressor for feature compression and then fed into a classifier for classification according to task.

Feature Internal Semantic Processing To enable better utilization of feature semantics by the model, we analyzed the collected dynamic features. We adjusted the features, excluding some computer-specific symbols, to better align with natural language grammar and clearly express their meaning in English. The processing steps include:

(1) *Removing Special Abbreviations.* Function names with capitalized "W" and "A" suffixes, such as "OpenServiceW" and "WriteConsoleA", are commonly seen in Windows APIs. These suffixes denote different encoding methods. Typically, two versions of a function are provided: a narrow character version (ending in "A") and a wide character version (ending in "W"). Importantly, adding or removing these suffixes does not impact the API functionality. To maintain consistent semantics, we omit the "W" or "A" suffixes when using these functions.

Original Features	Semantically Enhanced Features
REG:OPENED:******	Opened registry ******
REG:DELETED:******	Deleted registry ******
DIR:ENUMERATED:******	Enumerated directory ******

Table 2: Semantic Enhancement of Features: The original features (left) compared with the semantically enhanced features (right).

(2) *Converting Camel-cased Function Names to Verb Phrases.* Camel case is a naming convention commonly used for naming identifiers, such as variables, functions, classes, and properties. It is the practice of writing phrases without spaces, where the first letter of each word is capitalized, except for the first letter of the entire compound word, which may be either upper or lower case. This naming convention gets its name from its resemblance to the humps on a camel's back. In our dataset, numerous APIs adhere to the Camel Case naming convention, with examples like "GetFileSize" and "GetNativeSystemInfo". To enhance clarity, we split these camel-cased function names into normal phrases with correct grammar. For instance:

"GetFileSize" $\rightarrow$ "get file size"

"GetNativeSystemInfo"$\rightarrow$"get native system information"

(3) *Enhancing Features Semantically.* Many feature records in the dataset do not follow standard syntax, for example, ''REG:OPENED:HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services" means opening the registry ''HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services", and ''FILES:OPENED:C:\ProgramFiles\Launch4j\manifest" means opening the file ''C:\ProgramFiles\Launch4j\manifest". To assist the model better understand these features, we applied a simple "verb + subject" structure to semantically enhance the collected features, as shown in Table 2.

Combinatorial Models GPT-2 is a decoder-only transformer model (Vaswani et al. 2017), using the last token of the input sequence to predict the next token (Radford et al. 2019). This implies that the last token encapsulates all the information needed for prediction. Considering this, we can utilize the last token representation for prediction in the classification task.

In large-scale language models, a token is the basic unit of text processing. Tokenization, the process of splitting input text into tokens, is crucial in NLP tasks. GPT-2 (Radford et al. 2019) has a context size limit of 1024 tokens (about 600 words). If all features are directly spliced as in Tu-Liang Lin (Lin et al. 2024), 54% of the data will exceed this limit, resulting in the loss of exceeding features.

To solve this problem (Sun et al. 2019), we use a hierarchical method by inputting 11 types of features into the model separately for feature understanding. This approach ensures the semantic coherence of the same type of sequence, making it easier for the model to understand while also preventing the loss of information due to excessive token length.

Feature Compression Extractor After processing each feature's text, the model outputs a word vector matrix of max_seq_len * hidden_size. We construct a feature compression extractor to fuse and compress these features, reducing the model's computational complexity while enhancing its robustness and generalization. Three different feature compressors are experimented: Maxpooling, Avgpooling and LSTM.

Feature Combination Classifier We concatenated the seven compressed feature vectors into a combined vector, then applied a linear classification header to transform this vector into probability scores for class membership. Finally, our SRDC model was fine-tuned by maximizing the probability of correct classifications.

4 Social Impact of Tasks

Ransomware Classification. Different ransomware families vary in distribution methods, encryption algorithms, and attack strategies. Classifying them aids in understanding these differences and informs defense strategies, as some ransomware families are linked to specific cybercriminal organizations. Analyzing these families assists law enforcement in tracking and identifying criminal behavior. Additionally, commercial security software like Endpoint Detection and Response (EDR) and Threat Detection and Response (TDR) display ransomware families to aid users in threat analysis.

Zero-day ransomware detection refers to attacks exploiting undisclosed vulnerabilities in cybersecurity. This type of attack is harder to detect, more covert, and typically has a higher infection success rate than traditional ransomware. Its rising prevalence is driven by economic incentives, Ransomware-as-a-Service (RaaS) platforms, and advanced tools. Improving zero-day ransomware detection accuracy can protect data security, reduce economic losses, and ensure societal functions.

5 Experimentation

We evaluate SRDC, focusing on answering the following research questions:

RQ1: Does SRDC-GPT, after LATAP pre-training, improve zero-day ransomware detection compared to the base model GPT-2?

RQ2: How effective is SRDC in zero-day ransomware detection and family classification tasks?

RQ3: Are the proposed components effective in enhancing SRDC's performance?

Model	PGPCODER	Reveton	Average
GPT-2	8.3423	5.0663	5.0082
SRDC-GPT	**5.2901**	**4.3109**	**4.0765**

Table 3: The perplexity of language models on dynamic features of ransomware in the dataset.

5.1 Task Dataset

We used the dataset constructed by Sgandurra et al. (2016), which captures the dynamic behavior of executing the samples in a secure environment. It includes 582 ransomware and 942 goodware instances. The ransomware samples are categorized into 11 families. These samples represent the most prevalent versions and variants currently found in the wild. The goodware samples are sourced from reliable sources and include file utilities, browsers, emulators, drivers, word office devices, gaming applications, etc. These applications were executed for 30 seconds each in a sandbox environment.

5.2 Effectiveness of LATAP

Experiment Setting To validate the effectiveness of LATAP, we evaluate it from two perspectives: the conventional language modeling metrics and the impact on real-world tasks. We use perplexity as the evaluation metric for language modeling. The perplexity can be used to quantitatively measure a model's semantic understanding ability related to the dynamic feature of ransomware. (Jang et al. 2022). We calculate the perplexity on the main features (*i.e.*, API and registry characteristics) of the ransomware samples from the dataset introduced in Section 5.1. As shown in Table 4, we also compare the performance of the two models on the zero-day ransomware detection task (*i.e.*, training on the seen classes of ransomware and testing on the unseen classes as shown in Table 7, the specific and detailed experimental settings can be found in Section 5.4).

Result and Analysis As shown in Table 3, we evaluated the perplexity of samples from two ransomware families, PGPCODER and Reveton, known for low detection accuracy (Abbasi et al. 2022). SRDC-GPT significantly reduced perplexity compared to GPT-2, with a 58% decrease for PGPCODER. The model achieved an 18.6% reduction in average perplexity across the entire dataset, which includes 11 ransomware families and goodware samples. Note that LATAP did not involve ransomware sample features; thus, the features in the dataset are unseen to SRDC-GPT. A lower perplexity in such cases indicates that the model's predicted probability distribution is closer to the true distribution, re-

Model	Accuracy	Recall	F1-score
GPT-2	0.955	0.94	0.955
SRDC-GPT	0.955	**0.97**	**0.956**

Table 4: Comparison of model performance in zero-day ransomware detection before and after LATAP preprocessing.

Methods[a]	Goodware	Critroni	CL	CW	KOLLAH	Kovter	Locker	MATSNU	PGPCODER	Reveton	TC	TR
VarLenPSO	95.80	64.50	68.02	49.54	39.52	57.97	49.09	34.27	0.00	63.99	43.33	14.28
SRDC	**97.45**	**80.28**	**72.08**	36.78	**40.48**	**81.28**	44.38	**52.38**	**25.00**	**79.95**	12.50	**35.4**

[a] For a better representation of the table, CryptLocker, CryptoWall, TeslaCrypt and Trojan-Ransom are denoted by **CL**, **CW TC** and **TR**,respectively.

Table 5: Class-wise average accuracy (%) with baseline method and our method.

flecting a better understanding of zero-day ransomware features. This improvement is expected to enhance performance in downstream tasks as well.

As shown in Table 4, we also compared the two models' performance on the zero-day ransomware detection task, where models were trained on seen classes and tested on unseen classes (see Section 5.4 for experimental settings). Recall increased by 3.19% after LATAP (see Fig. 3 for a larger improvement), indicating that additional external semantics enhance the model's efficiency in detecting unseen ransomware samples.

The experimental results address RQ1. Our model leverages data patterns from previous fine-tuning to detect zero-day ransomware. For instance, if a prior ransomware sample deleted a specific file and invoked a Windows API, a test sample with similar features is likely ransomware. This mirrors how people use past experiences to address current challenges.

However, if zero-day ransomware exhibits entirely new features, past patterns may not help and could lead to errors. In such cases, cybersecurity experts analyze the intent behind actions like file deletions and system queries, and whether typical software accesses such data. By synthesizing these details, experts can determine if the software behaves anomalously. The deeper the understanding and the more knowledge experts possess, the better they can address emerging ransomware.

To enable our model to analyze and assess features autonomously, akin to cybersecurity experts, we conducted incremental pre-training through LATAP. This equips language models with an understanding of ransomware feature-related knowledge and context (external semantics). As illustrated in Table 3, our model exhibits a universal decrease in perplexity in feature texts across ransomware families. This indicates an enhancement in the model's comprehension of features from unseen ransomware, allowing our system to analyze features akin to human experts in discerning newly emerging ransomware.

5.3 Ransomware Family Classification Task

Experiment Setting For the ransomware family classification task, we use the dataset presented in Section 5.1, which contains 11 ransomware families and one goodware category, resulting in a total of 12 classes. We adhere to the experimental design in the current literature (Abbasi et al. 2022), using balanced accuracy (shorten as balanced-acc) as shown in Eq. (1) to compute the classification error due to the large variation in the number of samples between categories. Where n denotes the total number of categories in the dataset, and $correct_i$ is the correctly categorized instance from the total number of instances denoted by $total_i$ for category i. To ensure a fair comparison, all the above experimental settings are the same as existing methods (Abbasi et al. 2022; Tran, Xue, and Zhang 2017; Xue, Xue, and Zhang 2019).

Model	Balanced Accuracy (%)
LSTM	43.23
KNN	48.56
VarLenPSO	48.89
BERT	50.25
SRDC	**54.83**

Table 6: Comparison of our method SRDC with other methods in ransomware family classification accuracy.

$$\text{Balanced Accuracy} = \frac{1}{n}\sum_{i=1}^{n}\frac{\text{correct}_i}{\text{total}_i} \tag{1}$$

.

Result and Analysis To assess the efficacy of our system, we conducted a comparative analysis with four different approaches. VarLenPSO (Tran, Xue, and Zhang 2017) is a ransomware classification system that uses a blend of optimization algorithms and machine learning techniques to exploit dynamic features. We also compared our method with KNN (Daku, Zavarsky, and Malik 2018), a traditional machine learning method. LSTM (Maniath et al. 2017) and BERT (Lin et al. 2024) are methods that address features from a semantic perspective. The results in Table 6 show that our method achieved the best performance.

We analyzed the experimental results in detail, comparing our model with the top-performing method for each ransomware family (see Table 5). Our model achieves 1.65% higher accuracy than VarLenPSO in classifying goodware. It outperforms VarLenPSO in eight of the 11 ransomware families. Notably, VarLenPSO fails to correctly identify any PGPCODER samples due to their limited size, while our method achieves 0.25 accuracy. This shows our model's superior ability to recognize key features of ransomware families, even with sparse data. Additionally, LATAP pre-training leverages external semantics, reducing perplexity on PGPCODER features by 58% compared to the base model and significantly improving its ability to distinguish PGPCODER from other ransomware families.

We further analyzed the causes of misclassification cases, finding that one primary reason is insufficient features, as misclassified samples often lack adequate features, hindering semantic-based classification. In addition, some ransomware families are extremely rare; for instance, PGPCODER has only 4 samples, while TeslaCrypt has 6 sam-

Seen classes	Unseen classes
Citroni	PGPCODER
CryptLocker	Reveton
Cryptowall	TeslaCrypt
Kollah	Trojan Ransom
Kovter	
Locker	
Matsnu	

Table 7: Train/test splitting (seen/unseen) of ransomware families for simulating zero-day ransomware attacks.

Methods	Accuracy	Recall	F1-score
SVM	0.88	0.79	0.87
GoogleNet	0.85	0.81	0.80
LSTM	0.85	0.85	0.86
Inception V3	0.89	0.84	0.86
ResNet50	0.88	0.91	0.84
BERT	0.89	0.89	0.89
DCAE-ZSLHVE	0.93	0.95	0.92
SRDC	**0.96**	**0.97**	**0.96**

Table 8: Comparison of our method SRDC with other methods in zero-day ransomware detection.

ples.

5.4 Zero-day Ransomware Detection Task

Experiment Setting For zero-day ransomware detection, we follow the approach in (Zahoora et al. 2022). We divide ransomware families into seen and unseen classes to develop a robust detection model (see Table 7). The seen classes include 7 ransomware families representing known samples. The unseen class includes 4 ransomware families representing zero-day attacks. We use 134 unseen ransomware samples and an equal number of good software samples for the test set. The training set consists of 448 seen ransomware samples and 808 good software samples.

We fine-tune the SRDC-GPT model using the training dataset as outlined in Section 3. We then evaluate our system with the test set, measuring Accuracy, Recall, and F1-score. We compare SRDC with machine learning methods (SVM) and deep learning methods (GoogleNet, Inception V3, ResNet50) to demonstrate its performance. Note that DCAE-ZSLHVE (Zahoora et al. 2022) combines deep and machine learning for zero-day ransomware detection.

Result and Analysis As shown in Table 8, our method achieves the best performance across all three metrics in zero-day ransomware detection. We further explore few-shot learning for SRDC to address rapidly evolving zero-day ransomware with limited samples and new attack methods. Unlike Zahoora et al. (2022), we reduce the number of seen ransomware families during fine-tuning to better simulate real-world scenarios, while using increasing unseen families for testing. As shown in Fig. 3, with only four seen ransomware families and testing against seven, our F1-score exceeds that of DCAE-ZSLHVE (trained on seven families).

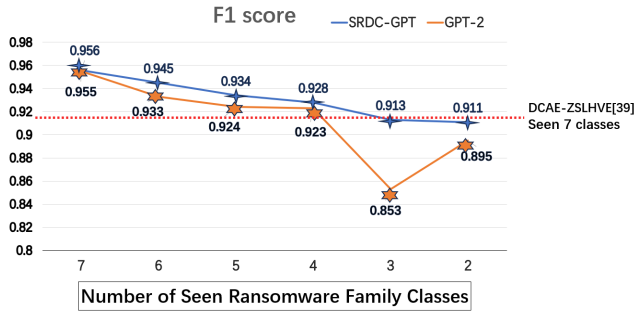

Figure 3: Evaluation of the Few-Shot Learning Capabilities of SRDC and GPT-2. The horizontal coordinate represents the number of seen ransomware families during training. We gradually reduce the number of seen classes, using fewer ransomware families for training, and testing on the rest.

Even with training reduced to two categories, the F1-score remains above 90%, demonstrating SRDC's effectiveness in real-world zero-day ransomware detection through comprehensive feature analysis.

We compared SRDC-GPT with GPT-2 (Fig. 3). Without LATAP, GPT-2's F1-score drops more sharply than SRDC-GPT when training samples are reduced. With three seen families, GPT-2's F1-score dropped by 10.68% compared to using four families, whereas SRDC-GPT's F1-score decreased by only 4.50%, outperforming GPT-2 by 7.03%. This shows that LATAP pre-training enhances SRDC-GPT's few-shot learning capability, ensuring greater stability with fewer samples for zero-day ransomware detection.

Sections 5.3 and 5.4 address our second question by evaluating SRDC's performance using various metrics and analyzing the system's advantages. Our system shows the potential to outperform existing methods in both zero-day ransomware detection and ransomware family classification tasks.

In addition, we evaluated the performance of several state-of-the-art models on two tasks. The detailed experimental results are publicly available in our GitHub repository.

Our model outperforms these models in terms of task accuracy. This is because the state-of-the-art models have limited understanding of specific dynamic features. They rely heavily on general knowledge and reasoning to solve problems, which often leads to hallucinations. Trained on domain-specific knowledge and fine-tuned with supervised task data, our model is better suited to these tasks.

Regarding performance, we focus on the processing time for individual data points, particularly the model's ability to quickly respond to dynamically collected online features. This capability is crucial for deploying security products. By avoiding uncertain network latency and service queueing, our model achieves faster average processing times per data point.

5.5 Ablation on SRDC

Experiment Setting Our proposed SRDC leverages both external and internal semantics of dynamic software fea-

Methods	RFC[a]	ZRD[b]		
	Balanced-acc	Accuracy	Recall	F1-score
N-FISP[c]	0.529	0.914	0.851	0.908
N-PEC[d]	0.528	0.925	0.811	0.922
N-Concatenate[e]	0.439	0.903	0.836	0.896
SRDC	**0.548**	**0.955**	**0.970**	**0.956**

[a] **RFC** denotes the ransomware family classification task.
[b] **ZRD** denotes zero-day ransomware detection task.
[c] **N-FISP**: no feature internal semantic processing module.
[d] **N-PEC** denotes no feature extraction compressor module.
[e] **N-Concatenate** denotes no combinatorial models module. We input seven features directly into the model by splicing them together

Table 9: The ablation experimental results.

tures, with external semantics provided through LATAP. The effectiveness of LATAP was demonstrated in Section 5.2. To utilize internal semantics, we designed a feature processing module, a combinatorial models module, and a feature extraction compressor. We assessed the efficacy of these modules through an ablation study, where components were excluded or replaced, and evaluated the system on zero-day ransomware detection and ransomware family classification.

Result and Analysis As shown in Table 9, we found that omitting any part of the model results in a decline in performance. The experiments in Section 5.5 address our RQ3, confirming the effectiveness and rationality of the modules we designed. The internal semantic processing module enhances the language model's understanding of dynamic feature texts by semantically preprocessing the original feature texts. The combinatorial modeling module improves the system's ability to handle features, avoiding information loss caused by truncating overly long feature texts and thus impacting model accuracy. Our designed feature extraction compressor module aggregates information, reduces feature dimensions, and facilitates accurate classification by the classifier in handling feature vectors.

6 Conclusion

We propose SRDC, a semantics-based ransomware detection and family classification framework that can utilize both internal and external semantics of software. We design a procedure called LLM-assisted Task-Adaptive Pre-training (LATAP), which can utilize the assistance of LLM to pretraining the model with corpus related to ransomware detection, enhancing the model's understanding of the external features related to ransomware. It also demonstrates strong few-shot learning capabilities and can be effectively implemented in industrial settings. This capability can alleviate the challenge of limited ransomware samples for training models, especially in the face of emerging new ransomware attacks. Deploying the SRDC system can enhance the accuracy of ransomware detection and classification, which not only helps reduce the economic losses caused by ransomware for individuals and businesses but also strengthens society's defense against cyber threats.

References

Abbasi, M. S.; Al-Sahaf, H.; Mansoori, M.; and Welch, I. 2022. Behavior-based ransomware classification: A particle swarm optimization wrapper-based approach for feature selection. *Applied Soft Computing*, 121: 108744.

Bridges, L. 2008. The changing face of malware. *Network Security*, 2008(1): 17–20.

Christodorescu, M.; Jha, S.; Seshia, S. A.; Song, D.; and Bryant, R. E. 2005. Semantics-aware malware detection. In *2005 IEEE symposium on security and privacy (S&P'05)*, 32–46. IEEE.

Daku, H.; Zavarsky, P.; and Malik, Y. 2018. Behavioral-based classification and identification of ransomware variants using machine learning. In *2018 17th IEEE international conference on trust, security and privacy in computing and communications/12th IEEE international conference on big data science and engineering (TrustCom/BigDataSE)*, 1560–1564. IEEE.

Demetrio, L.; Coull, S. E.; Biggio, B.; Lagorio, G.; Armando, A.; and Roli, F. 2021. Adversarial exemples: A survey and experimental evaluation of practical attacks on machine learning for windows malware detection. *ACM Transactions on Privacy and Security (TOPS)*, 24(4): 1–31.

Gururangan, S.; Marasović, A.; Swayamdipta, S.; Lo, K.; Beltagy, I.; Downey, D.; and Smith, N. A. 2020. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.

Han, W.; Xue, J.; Wang, Y.; Huang, L.; Kong, Z.; and Mao, L. 2019. MalDAE: Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics. *computers & security*, 83: 208–233.

itworldcanada. 2024. Hamilton confirms ransomware is behind cyber attack. https://www.itworldcanada.com/article/hamilton-confirms-ransomware-is-behind-cyber-attack/560034. Accessed: 2024-08-14.

Jang, J.; Ye, S.; Lee, C.; Yang, S.; Shin, J.; Han, J.; Kim, G.; and Seo, M. 2022. Temporalwiki: A lifelong benchmark for training and evaluating ever-evolving language models. *arXiv preprint arXiv:2204.14211*.

Li, C.; Lv, Q.; Li, N.; Wang, Y.; Sun, D.; and Qiao, Y. 2022. A novel deep framework for dynamic malware detection based on API sequence intrinsic features. *Computers & Security*, 116: 102686.

Liao, C.; Sun, M.; Yang, Z.; Chen, K.; Yuan, B.; Wu, F.; and Wang, Z. 2024. Adding NVMe SSDs to Enable and Accelerate 100B Model Fine-tuning on a Single GPU. *arXiv preprint arXiv:2403.06504*.

Lin, T.-L.; Chang, H.-Y.; Chiang, Y.-Y.; Lin, S.-C.; Yang, T.-Y.; Zhuang, C.-J.; Tseng, W.-L.; and Zhang, B.-H. 2024. Ransomware Detection by Distinguishing API Call Sequences through LSTM and BERT Models. *The Computer Journal*, 67(2): 632–641.

Liu, J.; Zhao, Y.; Feng, Y.; Hu, Y.; and Ma, X. 2024. Semalbert: Semantic-based malware detection with bidirectional encoder representations from transformers. *Journal of Information Security and Applications*, 80: 103690.

Liu, Y.; Tao, S.; Meng, W.; Wang, J.; Yang, H.; and Jiang, Y. 2023. Multi-Source Log Parsing With Pre-Trained Domain Classifier. *IEEE Transactions on Network and Service Management*.

Manavi, F.; and Hamzeh, A. 2020. A new method for ransomware detection based on PE header using convolutional neural networks. In *2020 17th international ISC conference on information security and cryptology (ISCISC)*, 82–87. IEEE.

Maniath, S.; Ashok, A.; Poornachandran, P.; Sujadevi, V.; AU, P. S.; and Jan, S. 2017. Deep learning LSTM based ransomware detection. In *2017 Recent Developments in Control, Automation & Power Engineering (RDCAPE)*, 442–446. IEEE.

Microsoft. 2024. Programming reference for the Win32 API. https://learn.microsoft.com/zh-cn/windows/win32/api/. Accessed: 2024-08-14.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744.

Preda, M. D.; Christodorescu, M.; Jha, S.; and Debray, S. 2007. A semantics-based approach to malware detection. *ACM SIGPLAN Notices*, 42(1): 377–388.

Qin, B.; Wang, Y.; and Ma, C. 2020. API call based ransomware dynamic detection approach using textCNN. In *2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, 162–166. IEEE.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Razaulla, S.; Fachkha, C.; Markarian, C.; Gawanmeh, A.; Mansoor, W.; Fung, B. C.; and Assi, C. 2023. The age of ransomware: A survey on the evolution, taxonomy, and research directions. *IEEE Access*, 11: 40698–40723.

Sathyanarayan, V. S.; Kohli, P.; and Bruhadeshwar, B. 2008. Signature generation and detection of malware families. In *Information Security and Privacy: 13th Australasian Conference, ACISP 2008, Wollongong, Australia, July 7-9, 2008. Proceedings 13*, 336–349. Springer.

Sgandurra, D.; Muñoz-González, L.; Mohsen, R.; and Lupu, E. C. 2016. Automated dynamic analysis of ransomware: Benefits, limitations and use for detection. *arXiv preprint arXiv:1609.03020*.

StatCounter. 2024. Operating System Market Share Worldwide. https://gs.statcounter.com/os-market-share. Accessed: 2024-08-14.

Sun, C.; Qiu, X.; Xu, Y.; and Huang, X. 2019. How to fine-tune bert for text classification? In *Chinese computational linguistics: 18th China national conference, CCL 2019, Kunming, China, October 18–20, 2019, proceedings 18*, 194–206. Springer.

Tao, S.; Liu, Y.; Meng, W.; Ren, Z.; Yang, H.; Chen, X.; Zhang, L.; Xie, Y.; Su, C.; Oiao, X.; et al. 2023. Biglog: Unsupervised large-scale pre-training for a unified log representation. In *2023 IEEE/ACM 31st International Symposium on Quality of Service (IWQoS)*, 1–11. IEEE.

Tran, B.; Xue, B.; and Zhang, M. 2017. A new representation in PSO for discretization-based feature selection. *IEEE Transactions on Cybernetics*, 48(6): 1733–1746.

Van Nhuong, N.; Nhi, V. T. Y.; Cam, N. T.; Phu, M. X.; and Tan, C. D. 2014. Semantic set analysis for malware detection. In *Computer Information Systems and Industrial Management: 13th IFIP TC8 International Conference, CISIM 2014, Ho Chi Minh City, Vietnam, November 5-7, 2014. Proceedings 14*, 688–700. Springer.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xue, Y.; Xue, B.; and Zhang, M. 2019. Self-adaptive particle swarm optimization for large-scale feature selection in classification. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(5): 1–27.

Zahoora, U.; Rajarajan, M.; Pan, Z.; and Khan, A. 2022. Zero-day ransomware attack detection using deep contractive autoencoder and voting based ensemble classifier. *Applied Intelligence*, 52(12): 13941–13960.

Zhang, Y.; Yang, S.; Xu, L.; Li, X.; and Zhao, D. 2023. A Malware Detection Framework Based on Semantic Information of Behavioral Features. *Applied Sciences*, 13(22): 12528.