Journal of King Saud University - Computer and Information Sciences 36 (2024) 101898

HOSTED BY

Contents lists available at ScienceDirect

Journal of King Saud University - Computer and Information Sciences

journal homepage: www.sciencedirect.com

Full length article

An empirical study of problems and evaluation of IoT malware classification label sources

Tianwei Lei [a,*], Jingfeng Xue [a], Yong Wang [a], Thar Baker [b], Zequn Niu [a,c]

[a] School of Computer Science and Technology, Beijing Institute of Technology, Beijing, 100081, China
[b] School of Architecture, Technology and Engineering, University of Brighton, Brighton BN2 4GJ, United Kingdom
[c] The Open University of China, Beijing, 100039, China

ARTICLE INFO

Keywords:
Malware classification
Malware label
IoT malware analysis
Anti-Virus engine label
Label tool

ABSTRACT

With the proliferation of malware on IoT devices, research on IoT malicious code has also become more mature. Most studies use learning models to detect or classify malware. Therefore, ensuring high-quality labels for malware samples is crucial to maintaining research accuracy. Researchers typically submit malware samples to Anti-Virus (AV) engines to obtain labels, but different engines have varying rules for detecting maliciousness and variants. This study aims to improve future IoT malware research accuracy by investigating label quality. We address three label-related issues, including Anti-Virus detection technology, naming rules, and label expiration. Additionally, we examine multiple sources of malware labels, including 63 studies on IoT, Windows, and Android malware, as well as commonly used tools such as AVClass and Anti-Virus engines. To evaluate and recommend label sources, we construct classification models using an IoT malware dataset obtained from VirusShare, which is labeled with common tools and Anti-Virus engines and classified based on ELF features. For family classification, we investigate four methods and tools, and for variant hierarchy classification, we compare label overlaps with sample clustering from 12 Anti-Virus engines. Based on our findings, we recommend AVClass for obtaining labels for IoT family classification. For small-scale malware families at the variant level, we recommend using the labels from Ad-Aware, BitDefender, and Emsisoft engines. For large-scale malware families, we advise employing labels from Jiangmin, NANO-Antivirus, and Avira engines, serving as a valuable guide for future IoT malware research.

1. Introduction

In recent years, while enjoying various conveniences brought by computer networks, people have faced increasingly serious network security problems, and malicious code is undoubtedly one of them (Ye et al., 2017) Malware has seriously threatened computer and network security since its inception. Moreover, the scale and complexity of malware have increased significantly, and due to the large-scale and rapid development of the Internet of Things in the world, its focus has also shifted to devices in the IoT environment (Clincy and Shahriar, 2019).

Faced with the explosive development of malicious code on IoT devices, security researchers have conducted in-depth research on malicious code analysis technology (Tahir, 2018). With the rapid development of machine learning in recent years, many machine learning technologies have achieved good results in combination with malicious code technology. High-quality labels are a prerequisite for building an adequate model, but unfortunately, we do not have an excellent way to obtain high-quality labels in malware. The primary cause of this problem lies in the inconsistent labeling standards used by Anti-Virus engines (Bailey et al., 2007). And various Anti-Virus engines may differ in their assessments of the harmfulness of a given malicious code sample and often employ distinct naming conventions for families and

* Corresponding author.
E-mail address: lei_tianwei@bit.edu.cn (T. Lei).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2023.101898
Received 3 May 2023; Received in revised form 19 December 2023; Accepted 19 December 2023
Available online 28 December 2023

variants (Canto et al., 2008). These discrepancies present significant challenges for subsequent model development.

At present, researchers of malware usually use the following methods to obtain the labels of malware, and the advantages and disadvantages of each method are as follows:

- **Public dataset:** Using samples from public datasets and annotated labels as the dataset for building the model. The advantage of this approach is that it avoids obtaining labels from Anti-Virus engines and improves research efficiency. However, the quality of labels cannot be guaranteed, and it entirely relies on the creator of the public dataset. Moreover, there are few public datasets of IoT malware.
- **Anti-Virus engine voting:** It is a commonly used method for obtaining labels, especially in detecting whether the sample is malicious. Usually, researchers upload the sample to a virus detection website (such as VirusTotal VirusTotal, 2023) containing multiple Anti-Virus engines, select multiple engines to obtain the detection results for voting, and use the voting results as the sample labels. The advantage of this method is that it can obtain judgments from multiple engines, but there is a problem with different naming rules for each engine in terms of subdivision class granularity. Furthermore, since no single Anti-Virus engine can detect all malware samples with perfect accuracy, researchers often need to choose classification labels through majority voting among multiple Anti-Virus engines (Sebastián et al., 2016). However, this requires researchers to have strong expertise since some Anti-Virus engines may excel at detecting certain virus families but perform poorly for others (Mohaisen and Alrawi, 2014). It is difficult to achieve a majority consensus in Anti-Virus engine voting without expert assistance.
- **Label tool:** It is a well-known method for obtaining labels in classifying malware. These types of labeling tools mostly rely on reports from virus detection websites, which perform text extraction and voting analysis on the reports, to some extent reducing the difficulty of label acquisition in classification work and improving work efficiency. However, it still cannot avoid the problem of Anti-Virus engines providing different labels, and there is currently no good labeling tool at the fine-grained level of variants.
- **Labeling of a particular Anti-Virus engine:** In certain studies, researchers opt to use labels provided by specific Anti-Virus engines due to their knowledge of malware families and AV engines, or to maintain the confidentiality of their data sources (Ronen et al., 2018), and have achieved good results. This method heavily relies on the prior expert knowledge of researchers as well as their understanding of the engines, which is very difficult for the average researcher, but it is undeniable that this approach is effective for fine-grained classification and specific malware research.
- **Label given by expert knowledge:** Similar to the previous method, few researchers are familiar with the sample and its source and will directly use expert knowledge or particular sources to label the sample (Hu et al., 2009). This method has the highest accuracy but requires the highest level of knowledge from researchers and is expensive. Only a very few studies have adopted this method.

The purpose of this paper is to explore the issues of label source and quality in IoT malicious code research. Our motivation is: which label sources are more effective for IoT malware classification? Specifically, the variants of IoT malware are complex, we believe that obtaining high-quality labels at the variant level can help improve the accuracy of supervised learning. Given the different naming rules of existing engines, which Anti-Virus engine should we choose for fine-grained classification? Based on the aforementioned questions and insights, we embarked on a multi-faceted empirical study: First, we investigated three problems associated with malware labels. Then, we explored the sources of these labels, covering 63 studies on IoT, Windows, and Android malware. After that, we further examined label quality specifically for IoT family classification tasks and variant-level classification tasks, built classification models, and evaluated and recommended the quality of commonly used label sources based on the model results. We constructed a malware dataset of IoT samples from the VirusShare (VirusShare, 2023) website. Using common labeling tools and Anti-Virus engines, we labeled samples at the family and variant levels and classified them based on commonly used static features. At the family level, we investigated four methods and tools. At the variant level, we compared the overlap between labels provided by 12 Anti-Virus engines and sample clustering. In conclusion, based on our findings, we recommend using the AVClass tool to obtain labels at the IoT malware family level. For small-scale families at the variant level, we recommend using labels from the Ad-Aware, BitDefender, and Emsisoft engines. For large-scale malicious code families at the variant level, we suggest using labels provided by the Jiangmin, NANO-Antivirus, and Avira engines as references for future IoT malware researchers.

The main contributions of this paper are summarized as follows:

1. We initiated an instance investigation focusing on the label quality of IoT malware, exploring the issues with malware labels, the sources of these labels, and the classification at both family and variant levels, recommending sources for labels. To our knowledge, this represents a novel study within the scope of IoT malware.
2. We identified three problems associated with low-quality labels in the current domain of IoT malware and illustrated them with case studies, providing the corresponding reasons. In addition, we surveyed the sources of labels used in 63 malware studies from 2009 to 2023, tracing the labels involved in public datasets for future researchers to consult and reference.
3. We tried to recommend sources for the labels used in the classification of families and variants. To achieve this, we first constructed a dataset of IoT malware and extracted static features of malware based on related studies of IoT samples to create a family classification model. We utilized commonly known label sources from our survey as training labels for supervised learning and observed the model performance, thereby providing recommendations for label sources at the IoT family level.
4. At a more fine-grained variant level, we build a clustering model based on the Control-Flow-Graph features, compare the labels and clustering results of famous Anti-Virus under their respective naming rules, and give a fine-grained classification of IoT malware recommendations on label sources.

The structure of the following sections is as follows: Section 2 introduces the work related to malware research and labeling; Section 3 introduces research questions, experimental methods, dataset and evaluation indicators; Section 4 presents a holistic discussion on problems and sources of malware labels, then moves to analyze and evaluate IoT family and variant label source. Section 5 concludes the paper with a summary and outlook for future work.

2. Related works

In this section, we started by introducing papers based IoT malware classification. Then, we summarized the label problems and malware label sources.

A. IoT Malware

Compared with the malware research on PE (Portable Executable) files under Windows system and Apps under Android system, the research on ELF (Executable and Linkable Format) files under Linux system is less and still in the stage of development. Researchers such as Shahzad and Farooq (2012), Shalaginov and Verlier (2021), Asmitha

and Vinod (2014) and Kumar and Handa (2021) etc. used the analysis method under Windows system to extract static and dynamic features from malicious code samples and input them into machine learning model for binary detection or family detection. Cozzi et al. (2018) conducted a detailed analysis and discussion on malware under Linux system in 2018, described the threats faced by devices under IoT operating system with Linux kernel, and constructed a malware dataset for classification and analysis. Carrillo-Mondejar et al. (2020) used the dataset constructed by Emanuele to extract static and dynamic features from the samples, using machine learning classifier and clustering method to analyze the threat and possible evolution direction of malware of different architectures under Linux system. Cozzi et al. (2020), who also proposed a large-scale measurement of IoTs malware in the latest research, systematically reconstructed the pedigree of IoTs malware family by using binary code similarity, tracked its relationship, evolution, and variation, and systematically analyzed and combed the relationship of malware.

B. Malware Label Problems

The problems of labeling malware has attracted some research attention. Mohaisen and Alrawi (2014) committed to proving the inconsistency of Anti-Virus engine labels from one engine to another. Zhu et al. (2020) measured the label dynamics and studied the scan result changes along with the timeline with the label problems. Kantchelian et al. (2015) estimated how long it will take for the initial Anti-Virus engine label to stabilize, and introduced a series of machine learning technologies to detect malware, focused on reducing the inconsistencies in malware labels. Yan et al. (2022) conduct an empirical study based on two PE malware sample datasets to focus on the label bias problems and existing solutions. Ge et al. (2021) found that performance of ML-based methods dramatically relies on datasets and labels.

C. Malware Label Sources

In terms of label sources, researchers have developed some label tools to alleviate the difficulties caused by inconsistent label naming. Using labels directly from public datasets is a common research method. This approach not only saves time by avoiding re-inputting samples into Anti-Virus engines but also enables the comparison of experiments with other research utilizing the same dataset. Kato et al. (2021) had been operating and updating honeypots to analyze and capture malware targeting IoT devices since 2015, and made a dataset in the field of IoT. Arp et al. (2014) and Allix et al. (2016) established a valuable malicious code dataset in the Android field, and guaranteed the label of the data family through AV engine voting. In addition, the labeling tool also provides convenience for researchers to obtain clear labels. Seshagiri et al. (2016) introduced an automatic marking tool named AVClass, which provided label for a large number of potential malware samples, and outputs the most likely last name for each sample. Later, AVClass2 (Sebastian and Caballero, 2020) was introduced. It uses and helps to build an open taxonomy, which organizes concepts in engines' labels, but is not limited to a set of predefined labels. Hurier et al. (2017) analyzed the associations between all labels given by different vendors and proposed EUPHONY to systematically unify common samples into family groups. Kim et al. (2022) provided a fully automated labeling tool that assigns each file a family name based on AV labels without prior knowledge

3. Research questions and methods

In this section, we introduced our research questions and our model in different level. And we proposed our dataset and evaluation indicators in the last section.

3.1. Research questions

Our research is motivated by the goal of identifying the most suitable label sources for IoT malware research. Therefore, we need to understand the label sources used by most of the current research, take them as the investigation object, and compare the impact of these label sources on the classification of IoT datasets. The evaluation method we choose is to use common features to establish a classification model on the same IoT dataset and select labels given by different label sources for classification accuracy statistics. Therefore, we set the following research questions:

RQ1: What are the problems with the current malicious code labels?

This question aims to illustrate the problem with malicious code labels and explore possible reasons. The question has been discussed with detailed answer and an example in Section 4.1.

RQ2: What are the sources of labels in current malicious code research?

This question aims to explore what kind of label sources the researchers chose in previous studies, including the name of the Anti-Virus engine used, label tools, and public dataset label sources, to confirm the survey objects for follow-up questions. The question has been discussed combining statistics from multiple papers in Section 4.2.

RQ3: In terms of classifying IoT malware families, what is the performance of each label source?

This question mainly explores the impact of using the survey labels mentioned above as training labels on the classification accuracy of the machine learning model when classifying IoT malicious code families. It infers the quality of the labels through accuracy. The question has been discussed with supervised model in Section 4.3.

In addition, IoT malicious code also has the characteristics of various variants and rapid evolution. In researching IoT malware, we should also pay attention to the labels of variant granularity. Therefore, we also explored the performance of label sources at the variant level. Since most of the current labeling tools do not support variant-level identification, we use multiple Anti-Virus engines that give IoT variant labels as the investigation objects for comparison at this stage. At this stage, we adopt a clustering method to build a classification model, cluster several important IoT families based on the best-performing family labels in the previous stage, and compare the label performance of the Anti-Virus engine. To sum up, we propose our last research question RQ4.

RQ4: At the level of IoT malicious code variants, what is the performance of the labels provided by each Anti-Virus engine?

This question mainly explores the difference between the labels provided by the Anti-Virus engine and the variant clustering and finally provides the label source recommendation in the fine-grained research of several famous IoT families. We discussed the question in Section 4.4 with sample clustering.

3.2. Classification models and features

We study the quality of labels by building classification models, focusing on the classification results given by the classification models under different Anti-Virus engines. We believe that under the same well-trained model, the label quality will directly affect the classification accuracy, so under the same conditions, the accuracy given by the same model under different labels can reflect the quality of the label to a certain extent, and our subsequent The experimental results also proved this point. We will explore the quality of labels given by different label sources at the family and variant levels, respectively, and give corresponding recommendations for different levels to provide references for subsequent research. Fig. 1 shows the exploration process and model framework of our article.

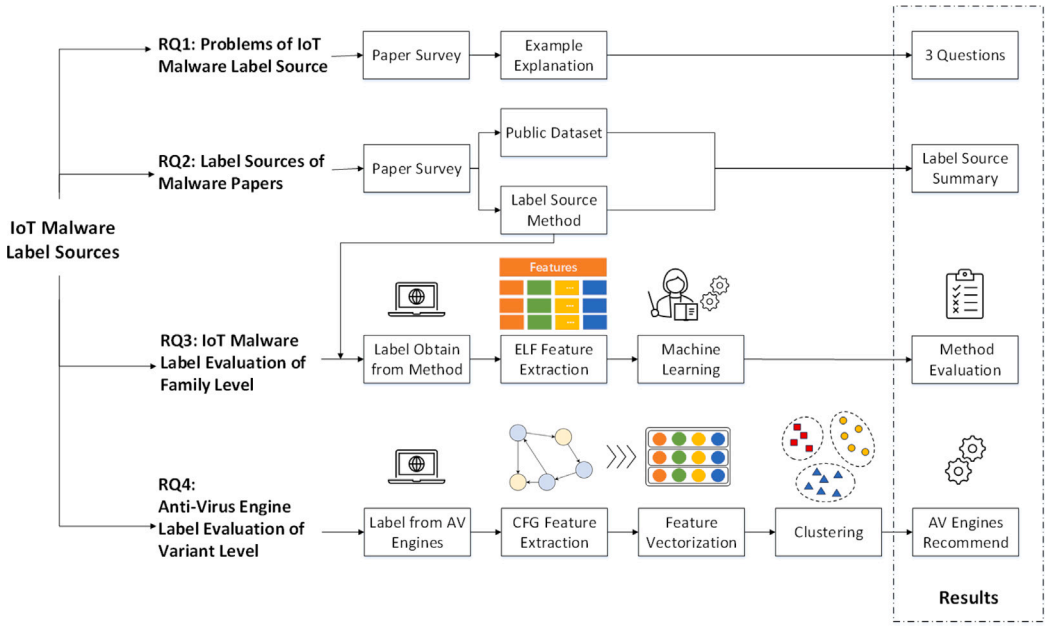

Fig. 1. The research framework of the paper.

Table 1
The features of family classifier.

Type	Features	Description
General features of ELF files	Maximum Raw Size of Sections	The largest size of all sections in the ELF file.
	Minimum Raw Size of Sections	The smallest size of all sections in the ELF file.
	Mean Raw Size of Sections	The average size of all sections in the ELF file.
	Section Nums	The total number of sections in the ELF file.
Entropy of ELF files	Entropy of File	A measure of randomness or unpredictability in the entire ELF file.
	Maximum Entropy of Sections	The highest entropy value among all sections in the ELF file.
	Minimum Entropy of Sections	The lowest entropy value among all sections in the ELF file.
	Mean Entropy of Sections	The average entropy value across all sections in the ELF file.
Size of ELF files attributes	Size	The total size of the ELF file.
	Size of This Header	The size of the ELF header.
	Size of Program Headers	The cumulative size of all program headers in the ELF file.
	Size of SectionHeaders	The cumulative size, in bytes, of all section headers in the ELF file.
	Section Header String Table Index	The index pointing to the section header string table, which holds names of other sections.
	Mean Size of Symbols	The average size of all symbols in the ELF file.
	Min Size of Symbols	The smallest size of all symbols in the ELF file.
	Max Size of Symbols	The largest size, in bytes, of all symbols in the ELF file.
Numbers of important attributes	Number of Section Headers	The count of section headers in the ELF file.
	Number of Symbols	The count of symbols in the ELF file.
	Number of File	The count of files referenced in the ELF file.
	Number of Object	The count of object within the ELF file.
	Number of notype	The number of symbols in the ELF file that do not have a specified type.
	Number of func	The number of function symbols in the ELF file.
	Number of others	The count of symbols or components in the ELF file that do not fit into other categories.

Family classification model

At the family level, we compared labels assigned by various commonly used sources, constructed supervised learning models, and assessed their performance on the dataset. We aimed to use relatively simple features and models for classification because our focus was on evaluating the quality of the labels rather than achieving highly ideal classification results with complex features and models. Therefore, we mainly collected 23 types of ELF static features from malware using IDA Pro scripts, focusing on capturing the intrinsic and overall features of the sample files, such as the number of sections, file headers, flags, and other statistical features, as well as sample size and entropy. Although these features are low-level, they are convenient to extract, not limited by different architectures, and we believe that these characteristics of different samples can differentiate between families. This has also been proven by the research of some scholars, such as Cozzi et al. (2018) and Carrillo-Mondejar et al. (2020). The details of features are listed in Table 1.

To ensure the effectiveness of our model, we employed a variety of machine learning algorithms for feature training. This included classical supervised learning models such as Random Forest, KNN, Naive Bayes, and Support Vector Machine for model construction and training. Additionally, we compared the accuracy across these different algorithms. In our subsequent discussion, we will select the best-performing model from each method for family-level evaluation, in order to deduce the effectiveness of each labeling approach. Table 2 lists the hyperparameters set for each machine learning algorithm model. Finally, to mitigate biases caused by the randomness of training and testing data, we obtained the average value of performance indicators through 10-fold cross-validation to eliminate potential impacts on the validation.

Table 2
The hyperparameters of machine learning model we used.

Algorithm	Hyperparameter	Settings
Random Forest	n_estimators	200
	criterion	gini
	max_depth	None
	max_features	auto
	min_samples_split	2
	min_samples_leaf	5
SVM	C	1
	kernel	linear
	max_iter	1000
	class_weight	balanced
	loss	squared_hinge
	multi_class	ovr
Naive Bayes	priors	None
	var_smoothing	1.00E−09
KNN	algorithm	auto
	leaf_size	30
	metric	minkowski
	n_neighbors	15
	p	2
	weights	uniform

Variant classification model

At the variant level, we aim to explore the overlap between the clustering results we give and the labels given by each Anti-Virus engine and give the recommended label source for the corresponding family. We aim to perform fine-grained clustering of samples from large families by capturing the similarity between samples and comparing the group differences between the clusters and the labels given by the Anti-Virus engine to give the Anti-Virus at the variant level Source recommendation for the engine label column. It is worth noting that due to the vast differences in the naming rules of each Anti-Virus engine at the variant level and the lack of actual labels, our method is an exploratory experiment and cannot quantitatively evaluate the factual correctness of the labels. We first select the label column with the best performance at the family level as the basis for family classification, classify the dataset by family, and retain family samples with a sample size of more than 1%. This is followed by extracting information to measure intra-family similarity from malicious code samples. In correlation clustering work, researchers choose to extract opcodes, flags, and graphs of samples for distance measurement. We choose the CFG graph to obtain file information. CFG is a commonly used function in binary analysis, which can capture the semantic information of samples, such as branches and loops. Algorithm 1 shows our cluster process. We use Angr (Wang and Shoshitaishvili, 2017) as the extraction tool to extract the CFG graph as we mentioned in Step 1. Then in Step 2, we need to vectorize these graph features for subsequent distance measurement. Graph embedding is a technology that uses low-dimensional vectors to represent each node in the graph. The encoded vector can reflect the structure in the graph. The more adjacent nodes shared by two nodes (that is, the more similar the context of the two nodes), the closer their embedding vectors are to Fang et al. (2020). We choose the Graph2Vec (Narayanan et al., 2017) technique for graph embedding. Compared with some other graph embedding algorithms, such as HOPE (Ou et al., 2016), SDNE (Wang et al., 2016) and Node2vec (Grover and Leskovec, 2016), the advantage of Graph2Vec is that it can generate a vector to reflect the characteristics of the entire graph, while most other technologies can only each node of generates a vector, and the eigenvector of the entire graph is the average value of all nodes. We use one family of architectures at a time for graph vector generation and take each vector as a dimension of 32. After generating graph vectors for each sample, we use Cosine similarity to calculate the distance between vectors in Step 3 to 5. Compared with Euclidean distance, cosine similarity measures the difference between two individuals by the cosine value of the angle between two vectors in the vector space. The closer the cosine value is to 1, the closer the included angle is to 0 degrees, that is, the more similar the two vectors are. We calculate the similarity between each vector as a distance matrix and perform clustering based on this distance matrix. After generating graph vectors for each sample, we use cosine similarity to calculate the distance between different vectors in Step 6.

Table 3
The Dataset details and composition.

Architecture	Amount	Percent
ARM 32-bit	19572	43.14%
MIPS	9862	21.74%
Intel 30386	8751	19.29%
PowerPC	4516	9.95%
AMD x86–64	2672	5.89%
Sum	45373	

Algorithm 1: Variant Classifier

Input: Samples divided into families T
Output: The clustering results with variants

1: Abstract CFG by Angr from T
CFG(T)=$V_1, V_2, \ldots, V_n = cfg(T_1), cfg(T_2), \ldots, cfg(T_n)$
2: Generate the vector of CFG.
$G(T) = Graph2Vec(CFG(T))$
3: Calculate the distance of each vector.
for each $G_i \in G$
4: for each $G(T_j) \in G, i \neq j$
5: $D(i,j)$=cosine$(G(T_i), G(T_j))$
6: Clustering with the distance matrix D.
7: Return Cluster(D).

For our sample clustering, the variants in the malware family were evolved by malware writers who made minor modifications to continue their intentions. We believe that the semantic information and control flow information of samples between the same variants is similar, and there may be cases of copying and pasting code. Therefore, CFG vectors covering program semantic information may differ among different kinds of samples, which has been confirmed by family research (Nguyen et al., 2020). Therefore, raising the distance judgment threshold can better deal with the work of various classifications among families.

3.3. Datasets and evaluation indicators

We used an Intel i7-9700k CPU with 16G of memory. Apart from the feature extraction stage, all other scripts were executed using Python version 3.7. We used PyCharm (PyCharm, 2023) as the IDE, and the family-level machine learning algorithms were implemented by calling the sklearn (Feurer et al., 2015) package.

Our dataset is malicious code in ELF format collected by the VirusShare website. VirusShare provides three compressed packages that collect malicious code in ELF format. The period of the samples is from 2016 to 2020. We merged the three sample collections and tested the samples with the commonly used ELF unpacker. At the same time, all samples were uploaded to the VirusTotal website to obtain sample reports, and all packaged and Android malicious codes were excluded; then, we excluded malicious samples with a family size too small (less than 10) and samples that were not detected as malicious by all engines. For the sake of discussion and summarization, we selected samples from the top five architectures in terms of size, which account for 94% of the original sample size. In the end, a total of 45 373 samples were selected. Table 3 presents the architectural composition of our samples.

On the evaluation indicators of the supervised classification model, we used the accuracy ACCURACY, the precision rate Precision, the recall rate Recall, and F1-Score. Its formula is as follows:

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (1)$$

Table 4
Analysis results of ten samples by different Anti-Virus engines.

Md5	Malicious level detection				Number of species detected					
	Detected	Undetected	Invalid	Percent	Platform categories	MostcCommon	Type categories	Most common	Family categories	Most common
c6739e0a1b38bcdebbf22ede76f059a9	40	22	19	65.57%	4	Android/AndroidOs	6	Exploit	6	GingerBreak
428111c22627e1d4ee87705251704422	41	20	14	67.21%	2	Linux	3	Trojan	4	Gafgyt
3059aeb2ab71eb3416d2a02366e3df25	39	23	14	62.90%	3	Linux	3	Backdoor/ Trojan	6	Agent
a3752b4eb910db34ace444fe33f0ffa5	45	16	12	73.77%	3	Linux	4	Backdoor	6	Gafgyt
33987c397cefc41ce5e269ad9543af4c	43	19	14	69.35%	3	Linux	4	Trojan	4	Mirai
bb96c4e5d3711e9ca1f439ba694712b1	43	19	13	69.35%	3	Linux	4	Trojan	4	Gafgyt
d781a95b3cffab57f432dbe23c232197	36	24	13	60.00%	3	Linux	6	Trojan	8	Ganiw
402adb32fc4b87431ced951fd0db56ed	28	19	28	59.57%	3	Linux	5	Backdoor	4	Mirai
bc835f87221b93fa909f0d436511e00d	38	22	15	63.33%	3	Linux	4	Trojan	6	Dofloo
720a5423a45ae1248b2d9c9ca8cfd03a	41	20	14	67.21%	3	Linux	8	Trojan	7	Dnsamp

Among them, "Precision" represents the accuracy of the positive predictions made by the model, while "Recall" refers to the proportion of actual positive cases that were correctly identified by the model. The formulas for these metrics are as follows:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

4. Experiment and research

In this section, we discussed our research problems which mentioned in Section 3.1, and gave the analysis and discussion in the end of the section.

4.1. Problems with malware labels

This subsection aims to answer the inquiry questions of RQ1. Ground-Truth labels are crucial for supervised learning in any field, and the quality of the labels will directly affect the model's accuracy and the algorithm's feedback. Real labels are tough to obtain, and manual labeling of datasets using domain expertise is the most convincing source of labels, but its high cost prevents it from being used in all datasets. The same applies to malware; we cannot bear the overhead of manually labeling each malware sample. Researchers often refer to VirusTotal as an "expert" in malware to circumvent this problem. The VirusTotal website is a detection platform integrating more than 70 Anti-Virus engines. Researchers can upload samples to this website to obtain reports from all Anti-Virus engines and download reports to obtain labels and other information. Although the VirusTotal has alleviated the difficulty of label acquisition to a certain extent, the following problems still exist in malware labels:

A. Detection level problem Anti-Virus engine detection technology is different. Each engine has a different method of detecting samples. Different Anti-Virus engines may have completely different cores and detection algorithms to detect samples, such as heuristic algorithms, Artificial Intelligence (AI) algorithms, Yara rule matching, etc., which will directly lead to different sample results and the quality of information given. It all depends on the expert knowledge of the manufacturer of the engine, which directly leads to the different labels given.

Ten randomly selected samples were extracted from the dataset and individually uploaded to the VirusTotal website for thorough analysis. The objective of this exercise was to examine the detection of the samples' maliciousness by all engines on VT. The results are tabulated in Table 4, with the first column depicting the MD5 hash of each sample. The second to fourth columns provide statistical insights into the malware nature of question A. Within the 'Detected' column, the number of engines that flagged the sample as malware is depicted. Similarly, the 'Undetected' column displays the number of engines that classified the sample as benign. 'Invalid' represents the number of engines unable to analyze the sample. Lastly, the 'Percent' column indicates the percentage of engines that identified the sample as malicious out of all worked engines. Taking the malicious code whose MD5 is "c6739e0a1b38bcdebbf22ede76f059a9" as an example, among the 76 Anti-Virus engines, 62 have done the detection work, and the other 19 engines cannot detect the ELF files for detection. Among the engines participating in the detection, 40 reported the sample as a malicious sample, and the remaining 22 reported it as "undetected", meaning no malicious was detected. It can be seen that the samples detected by VirusTotal are different, and different engines give inconsistent results for the same sample due to technical and other reasons, which has a significant impact on the subsequent analysis.

Table 5
Detection results of the example.

Level	Token	Frequncy
Platform	Linux	12
	ELF	4
	Win32	1
Type	Trojan	10
	Backdoor	4
	Malware	3
	DDoS	1
Family	Dofloo	9
	Generic	6
	Mrblack	2
	Others	7

B. Family naming issues: Anti-Virus engine's respective naming rules The malware labels are artificially defined, although there are some naming conventions in terms of naming CARO (1991) and Bontchev, but judging from the results, the Anti-Virus engine needs to follow these rules better. On the contrary, different engines have their own vocabularies, and these different label keywords may point to the same malware family or behavior, which makes the label more confusing and difficult to understand.

We conducted a more detailed analysis of the 10 malware samples selected in Section 4.1. The analysis focused on the keywords in the reports that identified them as malicious. We categorized these keywords according to platform, virus type, and family. The results are shown in columns 5 to 10 of Table 4. The columns "Platform Categories", "Type Categories", and "Family Categories" represent the number of keywords provided by all engines that detected the malware at each respective detection level. The three "Most Common" columns correspond to the most frequently occurring token at each level. Still taking the sample mentioned above as an example, we counted the keywords in the 40 reports that detected it as malware by platform, virus type, and family, and the results are shown in Table 5. We can see that the label information given by different Anti-Virus engines is very different at each stage. On the platform, 12 engines reported that

the sample was an Linux sample, 4 engines directly indicated that the sample was an ELF executable file, and 1 sample each believed that the sample was a file under Win32 systems. Regarding virus types, 10 engines considered the sample to be Trojan, 4 considered it malware using Backdoor, and 3 engines only judged malware. Finally, the Anti-Virus engine gave various information at the most critical classification level. Nine engines believed that the sample belonged to the Dofloo family, 10 believed that the sample belonged to the Generic family, 2 engines gave name with Mrblack, and 7 engines gave their unique keywords. Such a result will undoubtedly increase the trouble for subsequent label acquisition and make subsequent taxonomic research difficult.

C. Label timeliness In addition to the above two problems, the detection results of VT also have problems over time, resulting in label noise, which makes it more difficult to speculate on the true labels of samples and analyze the evolutionary relationship between families. Aiming at the timeliness of label, Wang et al. (2023) conducted experiments in the field of Android malicious codes with a time span of 2 years. They first obtained the labeled dataset created by Androzoo (Allix et al., 2016) in 2016, and the creator guaranteed that these samples were detected as malware by at least one Anti-Virus engine. They uploaded these malicious samples to the VirusTotal website to obtain labels in August 2019. Upload the sample again for scanning in August 2021 for a third label snapshot. Finally, comparing the three scan results, they found that in the 2021 results, at least 21% of the samples were detected as benign samples, which means that all engines in the 2021 Virustoal detection considered these samples as "not malicious". In addition, comparing the scan results in 2019 and 2021, about 64% of the samples received different labels; in 2019, 78.8% of the samples in the dataset were marked as malicious by at least one engine, and by 2021, This percentage dropped to 65.9%. Comparing the first and third scans, the number of apps with label changes is higher (34.1%). This also adds excellent difficulty to label acquisition.

RQ1 results: At present, malware have problems such as different Anti-Virus engine detection technologies that lead to inconsistent detection results, Anti-Virus engine naming rules that lead to family label confusion, and timeliness in obtaining labels from VirusTotal, which seriously affects the efficiency of label acquisition of malware and the quality of labels.

4.2. Sources of labels in malware research

For the RQ2 problem, we collected 63 malware studies and explored the source of labels in the papers. The papers mainly focus on Windows and IoT platforms; their statistics are shown in Table 6.

We have counted the data and label sources of malicious codes in recent years, 26 of which have the main task of detecting whether samples are malicious samples, and the remaining 37 have the main task of classification. In terms of data sources, 6 papers used self-built datasets, including downloading samples from websites and self-built honeypots to obtain samples, etc., accounting for 9.2% of the total; 18 papers chose from public virus databases download samples or sample sets, including VirusTotal Intelligence, VirusShare, VxHeavens, VirusChaser, Malware-Bazaar, and Mal-Share, accounting for 27.7% of the total; 15 papers used datasets contributed by papers or institutions, accounting for 23.1% of the total, The remaining 30 papers used a combination of virus databases and public datasets to ensure data diversity, accounting for 46.2% of the total papers.

In the research of 26 detection tasks, we counted 4 kinds of label sources, of which 13 papers used VirusTotal detection and voting mechanism, accounting for 50% of the total detection tasks. The voting mechanism is to pass samples through all VirusTotal platforms. For engine detection, if the number of engines exceeding the threshold detects that the sample contains malicious code, the sample is judged to be malicious code, otherwise, it is considered a benign sample. However, no research specifying the threshold value was provided. 8 studies are using the labels that come with the public dataset, accounting for 30% of the total, 3 using the labels given by the VXHeaven community, 1 using the labels given by the Windows Defender engine, and 1 article does not give the label source.

In the 37 classification tasks, we pay more attention to the source of the researchers' acquisition of the family-level labels of the samples. We have counted 10 methods in total, and the most commonly used source of labels is the direct use of labels from public datasets, involving a total of 15 studies, accounting for 38.5%, followed by the most famous labeling tool AVClass, a total of 8 articles using AVClass to obtain family labels, accounting for 23.1% of all classification tasks. AVClass is a label extraction tool. When using it, the user will input the VirusTotal report of malicious code and output the most likely family name of each sample extracted from the Anti-Virus label. It can also be used as a family hierarchy to cover all VirusTotal website engines. Voting majority results. In addition, 4 research teams (Lee et al., 2020; Tien et al., 2020; Wan et al., 2020a; Hoang et al., 2020) used the VirusTotal engine voting method to obtain labels, and they respectively used 3 to 5 trusted engines for voting and statistics. There are 2 articles (Jang et al., 2015; Mitsuhashi and Shinagawa, 2022) respectively using the Kaspersky engine and Microsoft engine as label sources. In addition, 1 research team (Toan et al., 2022) used the Symantec label engine, 1 article (Shang et al., 2010) used the labels provided by VXHeaven, 1 article (Hu et al., 2009) used expert knowledge to obtain labels, and 1 article (Sánchez-Fraga and Acosta-Bermejo, 2022) directly used multi-label as for the processing method, there is still 1 article (Nataraj et al., 2011a) that does not indicate the source of the label.

Finally, we also investigate the source of labels for the public datasets involved in the article. Here we mainly focus on public projects and datasets with the high frequency of occurrence, such as IoT-POT (Kato et al., 2021), Malimg (Nataraj et al., 2011b), Drebin (Arp et al., 2014) and BIG 2015 (Ronen et al., 2018) datasets, which have been used in our survey for more than 2 times. Table 7 shows the basic situation and label sources of these datasets. We can see that, except for the IoTPOT dataset that uses the AVClass tool, the other 3 datasets all use the information given by the engine trusted by the creator as the endorsement of the label source.

Based on our investigation, we found that in malware detection tasks, the most commonly used label sources are VirusTotal voting and directly using public dataset labels, while in classification tasks, using public dataset labels, AVClass label tool and engines vote. When using a specific Anti-Virus engine for direct labeling, the most trusted engines are Kaspersky and Microsoft Security.

Finally, we used statistics to identify observations at the family taxonomic level. Our selection criteria are commonly used label sources and methods or tools that can accurately and quickly give the labels determined by malicious code at the family level. The most commonly used sources for classification tasks use public dataset labels, which are not applicable since we chose to build the evaluation model on the sample package given by VirusShare. The second most used labeling tool, AVClass, will also be our observation. The third most used method is to use several trusted engines to vote, including the labels of the Drebin dataset, so we will also include this method in the next step of observation. We selected 5 engines for voting based on the surveyed papers and the recommendations of AV-Test (AV-TEST2022), namely: ESET (ESET, 2023), Kaspersky (kaspersky, 2023), Avira (Avira, 2023), Avast (Avast, 2023), and Symantec (Symantec, 2023), and named the method 5-Engines-Vote. Finally, we found that when using a certain engine directly as a label source, which is what we usually call a trusted engine, the two engines of Kaspersky and Microsoft Security (Microsoft Security, 2023) have a high frequency of occurrence, and we also use these two engines as the next object of observation. These four observation objects can quickly obtain unique and definite labels of malicious codes. By the way, AVClass2, an upgraded version of AVClass, was launched by the same team in 2020. It pays more attention to the

Table 6
Statistics on the data and label sources in recent years for malware research.

Citation	Dataset source	Label source	Task	Year
Cozzi et al.	Self-Built	AVClass	Family	2018
Nguyen et al.	IoTPOT, Virusshare	VirusTotal	Malware	2020
Shalaginov and Verlier	Self-Bulit	Microsoft Anti-Virus	Family	2021
Cozzi et al.	Self-Built	AVClass	Family	2020
Ravi and Alazab	Malimg Dataset	Public Dataset Label	Malware	2023
Alhanahnah et al.	IoTPOT	Public Dataset Label	Malware	2018
Torabi et al.	IoTPOT	AVClass	Family	2021
Lee et al.	VirusTotal	VirusTotal	Family	2020
Tien et al.	IoT HoneyPot	VirusTotal 3 Avs vote	Family	2020
Nguyen et al.	IoTPOT	Public Dataset Label	Malware	2018
HaddadPajouh et al.	VirusTotal	VirusTotal	Malware	2018
Vasan et al.	VirusShare, IoTPOT, Detux	Public Dataset and VirusTotal	Malware	2020
Nguyen et al.	Katsunari Yoshioka	Public Dataset Label	Malware	2018
Alasmary et al.	CyberIOC	VirusTotal	Malware	2019
Bai et al.	Vxheaven	Vxheaven apply	Malware	2013
Shahzad and Farooq	Vxheaven	Vxheaven apply	Malware	2012
Xiao et al.	Vxheaven	Vxheaven apply	Malware	2019
Carrillo-Mondejar et al.	[1] Dataset	AVClass	Family	2020
Karanja et al.	IoTPOT	VirusTotal 50% AV to confirm malware, dataset label for family	Malware+family	2020
Phu et al.	VirusShare, IoTPOT, Detux	VirusTotal	Malware	2019
Wan et al.	VirusTotal	VirusTotal vote	Malware+family	2020
Darabian et al.	VirusTotal	VirusTotal	Malware	2020
Su et al.	IoTPOT	Public Dataset Label	Family	2018
Hu et al.	Symantec Corporation,	VID and Expert knowledge	Family	2009
Shang et al.	Vxheaven	Vxheaven apply	variant	2010
Jang et al.	VirusShare, MalwareBlacklist	Kaspersky	Family	2015
Landman and Nissim	VirusTotal	Self-bulit Maltag based on VirusTotal	Family	2021
Hwang et al.	KISA, VirusShare	Public Dataset and VirusTotal	Malware	2020
Samantray and Tripathy	Kaggle	Windows Dender	Malware	2020
Musgrave et al.	theZoo	Public Dataset and VirusTotal	Malware	2022
Pham et al.	VirusSign	AVClass	Family	2021
Sánchez-Fraga and Acosta-Bermejo	VirusTotal, VirusShare, Vx-Heaven, Drebin, BIG2015	VirusTotal multi-label	Family	2022
Yuan et al.	3 IoT Dataset include Drebin	Public Dataset and VirusTotal	Family	2022
Sharma et al.	BIG 2015	Public Dataset Label	Family	2022
Xiao et al.	Malimg, Microsoft Malware Classification Challenge	Public Dataset and VirusTotal	Family	2020
Lee et al.	unkown	unkown	Malware	2021
Kumar and Handa	Vxheaven	VirusTotal	Malware	2019
Catalano et al.	VirusShare	VirusTotal	Malware	2022
Chen et al.	BIG 2015	Public Dataset Label	Family	2021
Deng et al.	MalShare, Hybrid-Analysis,VirusSign	Public Dataset and VirusTotal	Family	2020
Chaganti et al.	TWISC	Public Dataset and VirusTotal	Family	2022
Samantray and Tripathy	TWISC	VirusTotal	malware	2021
Hoang et al.	VirusShare, Detux	4 AV vote:Kaspersky, Symantec, Avast, Avira	Family	2020
Vouvoutsis et al.	Malware Bazaar	Public Dataset and VirusTotal	Family	2022
Conti et al.	Malimg, BIG 2015, Malware Bazaar	Public Dataset and VirusTotal	Family	2022
Yoo et al.	VirusChaser	VirusTotal	Malware	2020
Ahmed et al.	Malimg Dataset	Public Dataset Label	Family	2023
Aurangzeb et al.	[51] Dataset, VirusShare	Public Dataset and VirusTotal	Malware	2022
Sgandurra et al.	VirusShare	VirusTotal and manual	Family	2016
Ghouti and Imam	Malimg , Ember, BIG 2015	Public Dataset and VirusTotal	Family	2020
Qiao et al.	VirusShare	VirusTotal	Malware	2023
Wan et al.	VirusTotal	VirusTotal	Malware	2020
Qiao et al.	BIG 2015	Public Dataset and VirusTotal	Family	2021
Toan et al.	IoTPOT, VirusShare, VirusTotal, Detux	Symantec label	Family	2022
Zhao and Kuerban	VirusTotal	VirusTotal	Malware	2023
Belguendouz et al.	Mal-Share, VirusShare, Malware-Bazaar	AVClass	Family	2022
Cui et al.	Vision Research Lab[60]	Public Dataset and VirusTotal	Family	2018
Nataraj et al.	Self-Built	unkown	Family	2011
Dib et al.	VirusTotal, VirusShare, IoTPOT	AVClass	Family	2021
Naeem et al.	Malimg, Malheur, VirusShare and Android Dataset	Microsoft Anti-Virus	Family	2019
Li et al.	Malware capture facility project, VX Heavens and Virus Share	Public Dataset Label	Family	2021
Mitsuhashi and Shinagawa	Malimg, Drebin	Kaspersky	Family	2021
Wang et al.	Self-Built	AVClass	Family	2021

Table 7
The details of 4 datasets.

Dataset name	Field	Sample amount	Families amount	Year	Label source
Malimg	PE	9342	25	2011	Microsoft Security Essentials
IoTPOT	ELF	124 799	Multiple versions	2016	AVClass/AVClass2
Drebin	Android	5560	179	2012	10 Anti-Virus engines vote
BIG2015	PE	10 868	9	2015	Microsoft

Table 8
Performances of AVClass, 5-Engines-Vote, Kaspersky and Microsoft Security.

Method	Algorithm	Accuracy	Recall	Precision	F1-Score
AVClass	Random Forest	**0.9711**	**0.9209**	**0.9318**	**0.9263**
	SVM	0.8863	0.7643	0.7098	0.7360
	Naive Bayes	0.5902	0.5346	0.6469	0.5854
	KNN	0.9667	0.8560	0.9016	0.8782
5-Engines-Vote	Random Forest	**0.9375**	**0.9101**	**0.8925**	**0.9012**
	SVM	0.8852	0.8586	0.8224	0.8401
	Naive Bayes	0.6368	0.6886	0.6561	0.6719
	KNN	0.9132	0.8670	0.8446	0.8556
Kaspersky	Random Forest	0.9027	**0.8837**	0.8919	0.8875
	SVM	0.8664	0.8297	0.8344	0.8320
	Naive Bayes	0.6881	0.6382	0.6126	0.6251
	KNN	**0.9153**	0.8703	**0.9144**	**0.8918**
Microsoft security	Random Forest	**0.6721**	**0.6667**	**0.6098**	**0.6370**
	SVM	0.6641	0.6186	0.5951	0.6066
	Naive Bayes	0.4382	0.3979	0.3901	0.3939
	KNN	0.5332	0.5938	0.4455	0.5091

update and expansion of labels and gives the keywords and occurrence frequencies of all engines for a certain sample. The focus of AVClass2 is separate from the label quality we are concerned about, and the team also gave similar suggestions in the description, so we did not list AVClass2 as our further observation object.

We finally determined that the source of the labels investigated in the RQ3 question is: AVClass (also the result of voting by all VT engines), 5-Engines-Vote, and labels given by two different Anti-Virus engines, Kaspersky and Microsoft, as our follow-up investigation objects, comparing the quality of the labels they give at the family level.

RQ2 results: We surveyed 63 malware papers and found that the most commonly used label sources for malicious code detection tasks are VirusTotal voting and direct use of public dataset labels, while for classification tasks, public dataset labels and AVClass label tools are used is the most commonly used label source. We will conduct further quality investigations on family labels with AVClass, 5-Engines-Vote and engine Kaspersky and Microsoft.

4.3. IoT family label source evaluation

For the RQ3 question, we compared the classification performance of the four survey subjects above at the family level to explore their labeling quality. We use the dataset mentioned in Section 3.3, divide them according to ISA, and extract the static of the samples. Afterward, 4 methods were used to obtain family-level labels. For AVClass, Kaspersky, and Microsoft Security, we uploaded samples to the VirusTotal website, obtained reports, and extracted labels. For 5-Engines-Vote, we counted the family labels of 5 engines and extracted the token with the most frequent occurrence as a sample. After data feature extraction and labeling, we build and train the samples of each architecture separately and perform enough iterations for each model to ensure the accuracy of the model. We use the four machine learning models mentioned in Section 3.2 to model the collected feature data and the label data provided by each method.

Table 8 presents the overall performance of the 4 methods we investigate on all data. Although different machine learning algorithms produce varying data, the rankings provided for the four methods are almost consistent. We can see that AVClass has the best performance in all indicators on the overall data, which verifies the quality of the labels it gives on IoT malicious codes and why it is popular in classification tasks. This is followed by 5-Engine-Vote, which ranks second across all metrics. Kaspersky's performance is slightly weaker than the voting results of the 5-Engines-Vote, but it also has a relatively good performance. Among the four machine learning methods, Random Forest performs the best, consistently showing the best performance in combination with any label method. This is likely due to Random Forest's proficiency in handling high-dimensional feature data. Linear SVM and KNN perform similarly, delivering results akin to Random Forest in combination with the AVClass, 5-Engine-Vote, and Kaspersky methods. Naive Bayes shows poor performance, which may be related to the number of label categories and feature dimensions.

Since the Random Forest performs the best on the overall data, we chose it as the machine learning algorithm to analyze on each individual architecture. Fig. 2 shows the performance of the four methods on each ISA. On different ISAs, the accuracy given by different label source models fluctuates slightly, but the overall difference is not significant. AVClass has the best performance on all 4 architectures, And the performance is relatively stable, with both indicators exceeding 0.9 in terms of architecture except for PowerPC. Both 5-Engines Note and Kaspersky performed well in terms of accuracy by 90%, reaching above 0.85 in F1 values. 5-Engines-Vote is slightly more stable than Kaspersky, but the difference in performance between the two is not significant, which is consistent with their performance in all samples. Microsoft Security performed slightly lower in the sub architecture experiment, with an average accuracy of around 0.64, while the F1-Score average was only 0.59. We checked the labels given by the engine and found two possible reasons for the low accuracy: first, the granularity given by the engine on IoT samples needs to be more general and rough. For many samples, only Backdoor, The label of Trojan's virus type ignores the finer granularity; the second is that the engine may focus more on detecting and classifying malicious code of other kernels, and there are a large number of samples detected by the other three methods, the engine reports no malicious detections.

Finally, we compare the difference and detection rates of the 4 methods. AVClass has 0.2% of the samples in the dataset giving the "undetected" label, the lowest among the four methods. The undetected rates of the other three methods are 0.7%, 2.7%, and 6.6%, respectively. Regarding sample differences, we judged that the families given by the two methods were utterly consistent. AVClass and 5-Engines-Vote gave consistent labels on 96.4% of the samples, which was entirely consistent with Kaspersky on 90.4%. However, only 31.1% of the samples from Microsoft gave the same judgment. Likewise, the 5-Engines-Vote method had a 90.7% agreement with Kaspersky, which dropped to 32.2% with Microsoft. Finally, Kaspersky and Microsoft gave the same family label in 33.4% of the samples. Except for the Microsoft engine, the samples of the other three engines have a high degree of similarity.

Summarizing the above experiments and statistics while answering the RQ3 question, we also give the following conclusions that help select the source of family classification labels: First, AVClass is widely used in family classification. Our experiment also proves that this tool is the best choice among the 4 comparison methods and has the best performance in accuracy and detection rate; secondly, choosing a trusted Anti-Virus engine vote is also a good choice for label sources, but its disadvantages It is necessary for the researchers who construct the vote to have sufficient prior knowledge of each engine; otherwise the accuracy of the vote will be affected; thirdly, different engines have

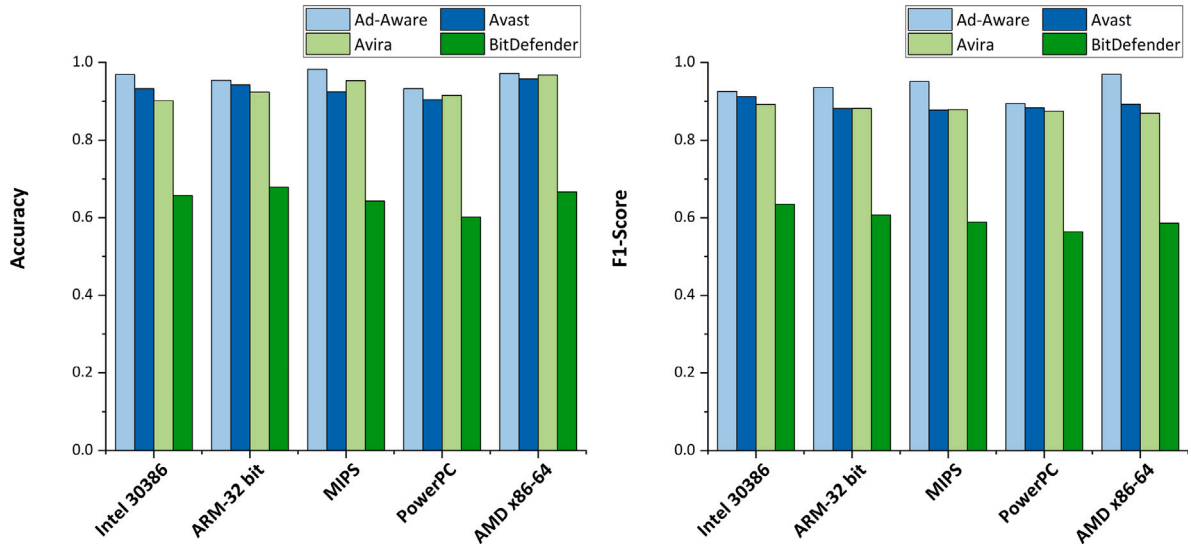

Fig. 2. The accuracy and F1-Score of 4 methods.

their areas of focus on detection, and it is necessary to be cautious when choosing a single engine as a label source, such as in In our experiments, the accuracy and detection rate of Microsoft is shallow, but it performs well on PE files and is a label source of well-known public datasets; fourth, the performance of different methods on different architectures in the IoT field may be affected by Influence, attention can be paid to the architecture when selecting the label source.

RQ3 results: We compared the performance of the 4 label sources at the family level across different machine learning methods, and the AVClass tool performed the best in both detection rate and accuracy, followed by 5-Engines-Vote, Anti-Virus engine Kaspersky and Microsoft. In addition, the first three methods give at least 90% consistency on the label, while Microsoft only gives the same result on about 30% of the samples. Combined with our experiments, we recommend using the AVClass tool on the family classification task, and we found that when IoT malware selects a label source, both the focus area of Anti-Virus engine and the ISA architecture of the sample may affect the quality of the label, which needs to be treated with caution.

4.4. IoT variant label source evaluation

For the problem of RQ4, we conduct exploratory experiments on the dataset. We used AVClass, the best-performing label source in RQ3, as the family label, labeled the dataset, and selected samples with a size of more than 1% in each architecture for clustering experiments of variants. We carried out experiments using samples from eight prominent families. Given the absence of precise variant-level labeling tools and the lack of definitive sources for variant labels in our survey, we specifically chose the result directly from several Anti-Virus engines. We verified their effectiveness in detecting IoT malicious code through VirusTotal (VT) report analyses. Notably, these engines not only demonstrated high detection rates but also consistently provided detailed labels for different variants. We selected 12 Anti-Virus engines, they are Ad-Aware, Avast, Avira, BitDefender, Emisoft, ESET-NOD32, GData, Jiangmin, Kaspersky, McAfee, NANO-Antivirus and Tencent. The primary reasons for selecting these 12 engines are twofold. Firstly, these engines demonstrate effective detection of IoT malware, rarely encountering cases of non-detection or operational failures. Secondly, in contrast to some engines that tend to classify samples into broad categories (such as maliciousness or family), these engines provide detection results at the variant granularity level. They also possess their unique labeling rules, aligning with our fundamental interest in exploring the quality of labels.

We collect their variants labels, compared with the clusters we mentioned in Section 3.2, we used cosine similarity as the distance matrix, recorded the cluster labels of each sample, and established a co-occurrence matrix with the labels given by the engines. Since we do not actually know how many variants there are in a family, we can only make reasonable assumptions about the number and nature of the variants: We expect higher similarity within the variants of the same family and greater differences between different categories; regarding the number of variants, we do not want it to be too high: this indicates that the model tends to classify samples as individuals, which would lead to an explosion in the number of variants in large families; nor do we want it to be too low, as this implies that the model tends to consider all samples as the same type, losing the significance of fine-grained exploration. Therefore, after experiments and hypothesizing, we finally determined the threshold for clustering: For large families Gafgyt, Mirai, and Tsunami, to limit the number of variants generated, we set a clustering distance threshold of 0.8, while for other families, we set a distance threshold of 0.9. We will traverse all possible mapping results of generated labels and Anti-Virus labels, record and rank the best PMI value performance of each engine in different families, and finally give our recommendation results.

We use the co-occurrence matrix to describe the similarity between our clustering results and the labels given by the engines, and name it the Pointwise Mutual Information (PMI) value. We label the clusters of variants generated by each sample by 0, 1, 2... n, defined as $Label_C$, collected the variant labels of the Anti-Virus engine to be compared, and also randomly number 0, 1, 2...n according to the type as $Label_{AV}$. We use the co-occurrence matrix to count the number of times that the classification labels appear at the same time, and the definition of the PMI value is as follows.

$$PMI(A,B) = \log_2 \frac{P(A,B)}{P(A) * P(B)} \tag{4}$$

Where $P(A,B)$ represents the frequency of the common occurrence of label A and label B on the sample set, $P(A)$ represents the frequency of label A, and $P(B)$ represents the frequency of label B. And here, Label A and B are the results of the clustering we mentioned above, $Label_C$ and $Label_{AV}$, we substitute the clustering result label and the label of the AV to be evaluated into the formula. The basic idea is to count the probability of two classification labels appearing in the text simultaneously. If the probability is greater, the correlation is closer, and the degree of correlation is higher.

Please note that there currently needs to be a more straightforward public dataset due to the lack of research at the variant level. At the

Table 9
The PMI of all Anti-Virus engines.

Architecture	Family	Anti-Virus engines											
		Ad-Aware	Avast	Avira	BitDefender	Emsisoft	ESET-NOD32	GData	Jiangmin	Kaspersky	McAfee	NANO-Antivirus	Tencent
Intel 30386	Ddostf	**0.9688**	0.5000	0.5917	**0.9688**	**0.9688**	0.7500	**0.9688**	0.7859	0.4942	**0.9688**	0.7578	0.5000
	Dofloo	**0.9984**	0.9804	0.8525	**0.9984**	0.9134	0.4988	0.9856	0.4011	0.6981	0.7266	0.8176	0.8333
	Elknot	**1.0000**	0.5524	0.4894	**1.0000**	**1.0000**	0.6667	0.9841	0.6047	0.3852	0.6481	0.9167	0.7778
	Gafgyt	0.1301	0.1669	0.0446	0.1302	0.1294	**0.4069**	0.2039	0.2165	0.3106	0.2161	0.0772	0.3122
	Ganiw	0.8167	0.5774	0.4459	0.8167	0.8000	0.1962	0.2153	0.4834	0.1644	0.4663	**0.8611**	0.2889
	Generica	**0.8802**	0.7656	0.4557	**0.8802**	0.8802	0.5010	**0.8802**	0.7521	0.3910	0.3688	0.7510	0.7510
	Mirai	0.3718	0.0649	0.6046	0.3718	0.3719	0.1142	0.3072	0.5689	0.2541	0.1966	**0.6216**	0.1502
	Tsunami	0.3386	0.2550	0.3232	0.3386	0.3384	0.2329	0.2800	**0.5616**	0.1890	0.3291	0.5087	0.2752
ARM-32 bit	Dofloo	**0.8981**	0.6065	0.3313	**0.8981**	**0.8981**	0.4837	0.8853	0.7148	0.4737	0.3426	0.4517	0.4854
	Gafgyt	0.3116	0.0748	0.1438	0.3103	0.3080	0.1960	0.1629	0.6111	0.0876	0.1652	0.4908	0.2788
	Mirai	0.3661	0.2454	0.4530	0.3661	0.3670	0.2206	0.2912	**0.6218**	0.0666	0.1638	0.5514	0.1022
	Tsunami	0.1728	0.1806	0.4869	0.1728	0.1792	0.1316	0.1985	**0.7091**	0.3213	0.2184	0.6712	0.1970
MIPS	Dofloo	**0.9804**	0.6667	0.2119	**0.9804**	**0.9804**	0.3232	0.7881	0.7406	0.3232	0.7881	0.2119	0.5000
	Gafgyt	0.2017	0.1727	0.5032	0.2017	0.2043	0.3645	0.1333	**0.5752**	0.1591	0.3343	0.4979	0.4229
	Mirai	0.3891	0.2247	0.4973	0.3871	0.3848	0.2709	0.1816	**0.6363**	0.0719	0.3440	0.6176	0.1168
	Tsunami	0.1434	0.3289	0.3809	0.1390	0.1390	0.2749	0.1753	**0.4650**	0.1785	0.3122	0.4387	0.2644
PowerPC	Gafgyt	0.1671	0.1240	0.4683	0.1671	0.1681	0.3284	0.1027	**0.5970**	0.1788	0.1599	0.5060	0.3801
	Mirai	0.2978	0.1627	0.5176	0.2972	0.2981	0.2896	0.2302	**0.6217**	0.0579	0.2434	0.5834	0.1041
	Tsunami	0.2439	0.3943	0.4950	0.2439	0.2439	0.3284	0.3471	**0.7441**	0.1550	0.4636	0.7020	0.4364
AMD x86-64	Gafgyt	0.1572	0.1245	0.5175	0.1573	0.1573	0.2917	0.0917	**0.6028**	0.1746	0.1509	0.4900	0.3490
	Mirai	0.3414	0.2622	0.5428	0.3436	0.3412	0.2927	0.2588	**0.5727**	0.1567	0.2135	0.4848	0.3316
	Tsunami	0.1741	0.3642	0.5629	0.2034	0.1882	0.2874	0.2860	0.6511	0.2040	0.2577	**0.6631**	0.2938

same time, as mentioned in 4.1, the respective naming rules of different Anti-Virus engines lead to considerable differences in the labels of the variant level. We cannot guarantee that our clustering or any label an engine gives is "correct". Here we only give relative recommendations based on the clustering results and try to analyze the performance of Anti-Virus engines on different ISAs to provide a reference for subsequent fine-grained research. The PMI of all Anti-Virus Engines are shown in Table 9, and we will discuss different architectures separately next.

The Intel 30386 architecture has the most sample families in our dataset, although the number of samples collected on this architecture is smaller. We have collected samples from 8 families on this architecture. Fig. 3 shows the performance of the PMI values of all engines. The abscissa in the figure is the large-scale family on the architecture, and the ordinate is the PMI value representing the similarity between the label of each engine and our clustering result. We can see from the figure that, except for the three families of Gafgyt, Mirai, and Tsunami, the variant labels given by the engine have a high similarity with our clustering results, and the maximum PMI values are all above 0.85. Even on the Dofloo and Elknot families, our clustering results are almost entirely consistent with the variant types given by several engines, which verifies the validity of our clustering results. The performance of different engines once again proves that each engine has different classification and naming rules in terms of variant granularity classification, which undoubtedly increases the difficulty of fine-grained research. The Gafgyt, Mirai, and Tsunami families are the most prominent families in the IoT field, and their average PMI values are low, which is related to the fact that there are too many variants in large families, the interior is intricate, and many minor variants have only a few samples. Regarding Anti-Virus engine performance, Ad-ware, BitDefender, and NANO-Antivirus are the three engines with the highest average PMI values on this architecture, and the variant categories they give are similar to our clustering results. Although the Jiangmin engine performs poorly in small families, it ranks first in the three complex families. The possible reason is that the Jiangmin engine directly classifies samples into variant levels, such as "Linux.aaa", "Linux.bbb" formats.

The ARM 32-bit architecture has the most significant number of samples in our dataset, accounting for nearly 40%. It is the first RISC microprocessor with low power consumption and cost designed by Acorn Co., Ltd. in the UK. It has high performance and low energy

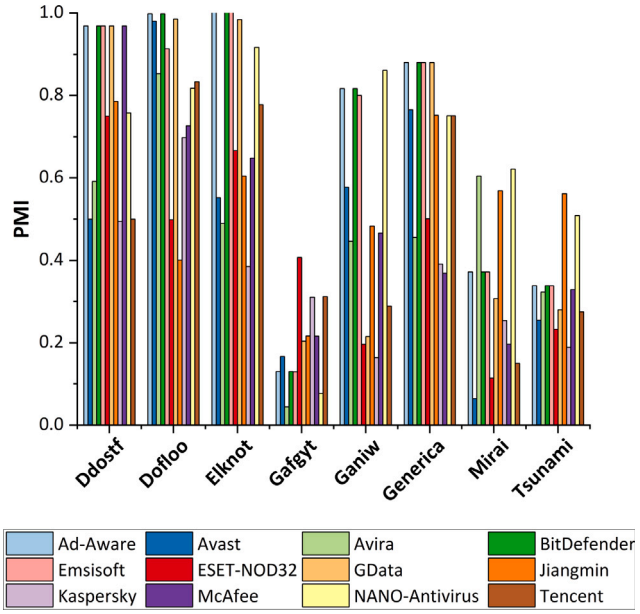

Fig. 3. The PMI of Intel 30386.

consumption. It is also a widely used architecture in the current IoT architecture. We have collected 4 types of families on this framework with relatively large scales. Fig. 4 is a histogram of PMI values under this framework. The PMI value on ARM-32 bit is generally lower than that of Intel 30386, which may be related to its sample family composition, mostly Gafgyt and Mirai. In addition, we can see results similar to those of Intel 30386 on this architecture. The classification of Jiangmin and NANO-Antivirus on the three large families is more similar to our clustering results, while Ad-Aware, Bit-Defender and Emsisoft are highly similar to the clustering results on Dofloo.

The MIPS architecture is a processor architecture that adopts a simplified instruction set. It was developed and authorized by MIPS Technology Corporation in 1981. It is widely used in many electronic products, network equipment, personal entertainment, and commercial devices. It is the second-largest architecture in our dataset. Fig. 5 shows

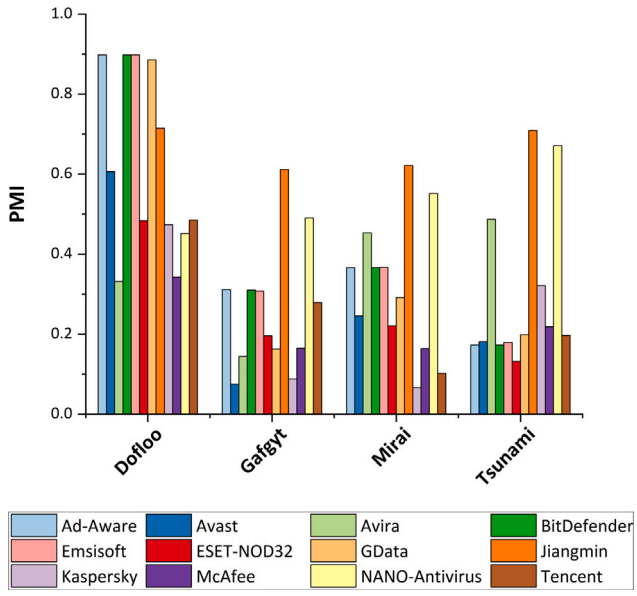

Fig. 4. The PMI of ARM-32 bit.

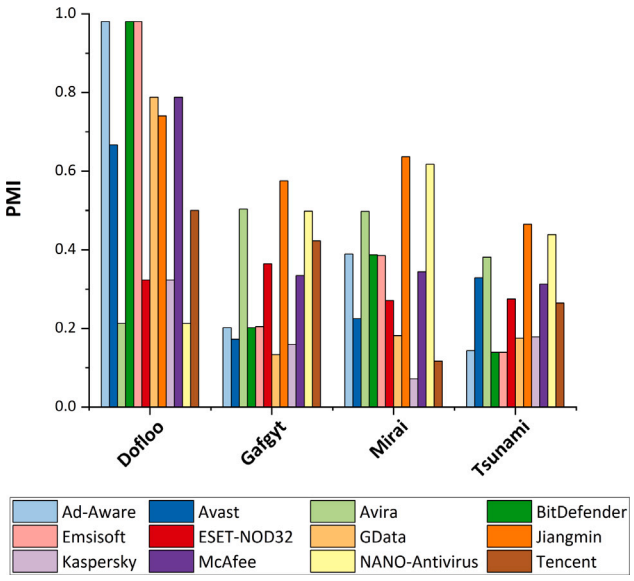

Fig. 5. The PMI of MIPS.

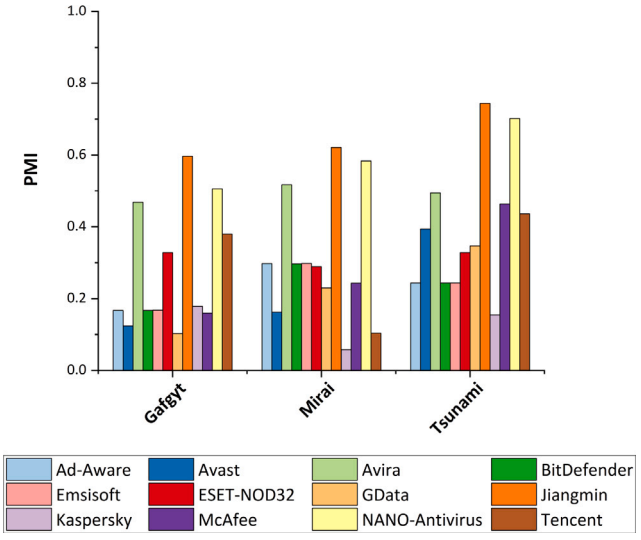

Fig. 6. The PMI of PowerPC.

the PMI values of each Anti-Virus engine in this architecture. Compared with the same family of ARM architecture, the clustering of variants on MIPS architecture is more prominent. There are also several apparent variants in large families. The number of individual variants is less than that of ARM architecture, which is also why its average PMI value is higher than that of the ARM architecture. Regarding PMI value ranking, the performance of the MIPS architecture is similar to that of the ARM architecture. The clustering results of Jiangmin and NANO-Antivirus are the most similar to ours. On the Dofloo family, our clustering results are similar to those of Ad-Aware and Bit-Defender. The labels given by Emsisoft are the closest, and PMI values close to 1 indicate that they almost give labels that are the same as our clustering results.

PowerPC is a microprocessor architecture developed by the AIM Alliance formed by Apple, IBM, and Motorola in 1991. The AMD X86-64 is an extension of the x86 architecture to 64 bits launched by AMD. These two architectures are widely used in IoT and account for about 15% of our dataset. We mainly focus on the clustering performance of three large families on these two architectures. Figs. 6 and 7 show the PMI values on the PowerPC architecture and the AMD x86-64 architecture, respectively. There is not much difference with other architectures; the Jiangming, NANO, and Avira engines still have the most similar performance. The PMI value of the PPC architecture is higher than that of the AMD architecture, which means that it is closer to our clustering results.

Finally, we conducted experiments on the entire dataset, the results of which are shown in Fig. 8 We can observe that, through clustering and comparison of different architectures, on five families except for Gafgyt, Mirai, and Tsunami, our clustering results are highly similar to those of multiple Anti-Virus engines. For families Gafgyt, Mirai, and Tsunami, we have the highest PMI value performance of about 0.74. The low value is related to the three prominent families' internal complexity and the Anti-Virus engine's inconsistent labeling rules. At the same time, the clustering results of variants also verified the necessity of variant classification. Our dataset has at least two variants in both large and small families. The larger the family size, the more internal variants, each greater the difference between variants.

On the five different architectures, the performance rankings of each engine fluctuate little. Among all the small families, the Anti-Virus engines with the best PMI value are Ad-Aware, BitDefender, and Emsisoft. They often give the same results as our clustering at a threshold of 0.9, and in some families, the three engines give entirely consistent labels. We infer that the similarities in labeling among different engines could be attributed to either the adoption of similar classification rules or the utilization of same detection mechanisms by these Anti-Virus companies.

On the three large families, the three engines with the best performance are Jiangmin, NANO-Antivirus and Avira. The label given by Jiangmin skips the family level. For example, the malicious backdoor code of a sure ELF will be labeled "Backdoor.Linux.aaaa" by the engine, where aaaa represents the variant name. NANO-Antivirus and Avira give family and variant labels. These three engines have a high degree of coincidence with our clustering results on large families, which is also related to the samples with complex variants, many separate clusters, or difficult to classify in the three large families. Finally, it is worth mentioning that when comparing variant labels, we did not require the engine to give consistent labels such as AVClass, which is the family label we used in the family classification stage. This is because many engines directly give variant labels to Linux malicious codes.

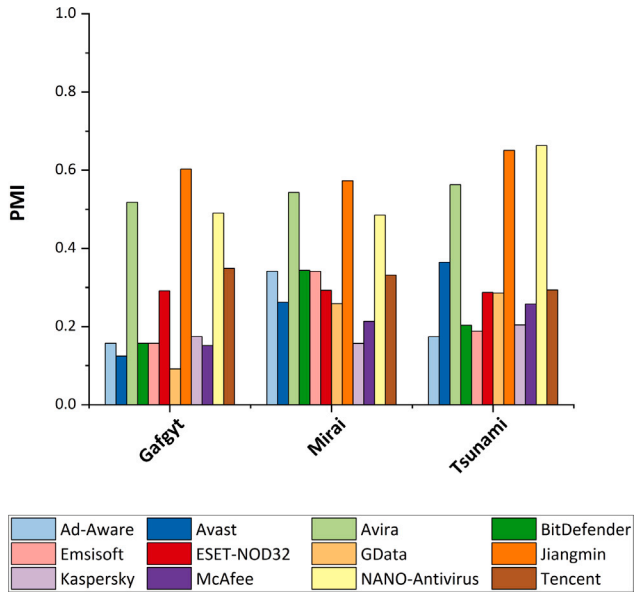

Fig. 7. The PMI of AMD x86-64.

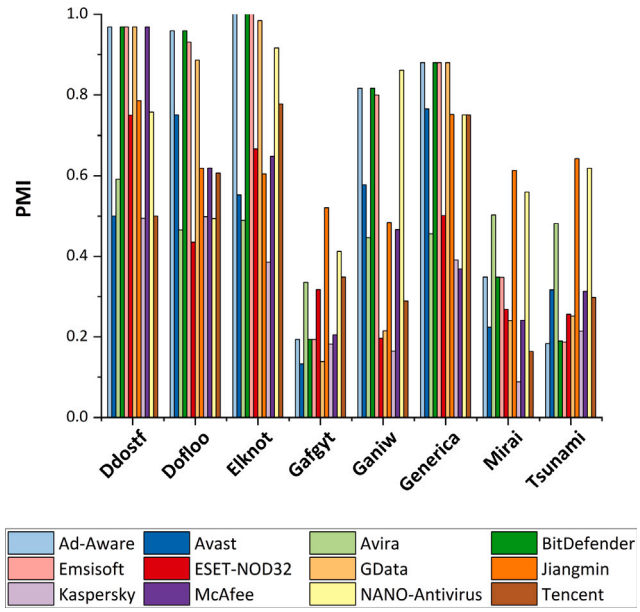

Fig. 8. The PMI of all samples.

Therefore, we recommend using the label types provided by Ad-Aware, BitDefender and Emsisoft engines for fine-grained classification label acquisition for small and straightforward families. For the three types of large malicious code families such as Mirai, Gafgyt and Tsunami, we recommend using the labels given by Jiangmin, NANO-Antivirus, and Avira. However, it should be noted that some engines hide the family-level labels, such as Jiangmin.

RQ4 results: We conducted a clustering experiment on families with a more than 1% scale in the dataset and compared the variant level labels of 12 Anti-Virus engines. Based on the comparison of the PMI values for the clustering outcomes and various label results, we provide recommendations for label sources for families of different sizes: for small and simple families, we recommend using the labels given by Ad-Aware, BitDefender, and Emsisoft engines. They may use the same classification and naming rules, which are consistent with our clustering results. For the three sizeable malicious code families of Mirai, Gafgyt, and Tsunami, we recommend using the labels provided by the engines Jiangmin, NANO-Antivirus, and Avira. However, it should be noted that some engines hide the family-level labels, such as Jiangmin.

4.5. Analysis and discussion

We hope that through our experiments and analysis, we can give some suggestions for subsequent IoT classification researchers to choose the source of the label. Although the label is a tiny part of the research, the quality of labels will directly affect the reliability of subsequent models and analysis. Therefore, we conducted experiments and analyzed the problems in the current label sources, the commonly used label sources of malicious code, and Anti-Virus engines and label tools recommended by families and variants for classification tasks in the IoT field.

Returning to the original intention of our research, our motivation is to find a suitable and quality-assured source of labels for IoT malware during our study. To this end, we conducted an empirical study on malware labels around four research questions. First, we investigated the existing problems with malware labels. In RQ1, we explored the issues of detection levels, family naming conventions, and the timeliness of labels in current malware. In this paper, we mainly addressed the first two issues by avoiding or mitigating them through methods of label acquisition and AV engine selection, while the third issue was not further explored, which may also be a potential risk. Then we carried out an investigation on the common sources of malware labels around RQ2. We conducted a survey of 63 malware articles from 2009 to 2023, including papers on Windows, Android, and IoT. The 63 articles we selected all clearly mentioned their methods of label acquisition, or the public datasets used, and we annotated the methods used to obtain labels for the 4 types of public datasets involved. In RQ1 and RQ2, we did not discuss the IoT field separately for two reasons: firstly, we believe that it is meaningful to discuss these two issues that are prevalent in the malware field comprehensively, and secondly, research on IoT malware started later and fewer articles are available, or many articles do not publicly mention their sources of labels. Therefore, we also hope to find commonalities in the problems of label sources in the entire field of malware by researching label sources widely used in the entire field, so as to discuss more specifically in the IoT field.

The research purpose of RQ3 and RQ4 is, at the family and variant levels in the IoT field, what kind of label source is more conducive to classification work? For these two questions, we constructed a dataset of IoT malware and based our subsequent experiments on this. For the family level, we extracted commonly used and previously validated ELF features as model inputs and then used common machine learning algorithms for family-level classification. The labels used were the four common methods investigated in RQ2. Under the same conditions, we judged the quality of the labels based on the final classification results. For the variant level, we extracted the results given by 12 commonly used AV engines for IoT malware detection as the label column and performed fine-grained clustering of malware using CFG as features and Cosine similarity, with the PMI value comparing the co-occurrence degree of AV engine results with clustering outcomes. In the end, we gave different recommendation results for the family and variant levels: we recommend researchers use the AVClass tool when classifying IoT malware families, and at the variant level, we recommend using Ad-Aware, BitDefender, and Emsisoft for small families, and for large families, using Jiangmin, NANO-Antivirus, and Avira.

There are several validity threats to our experiments. The first validity threat is related to the dataset. The dataset we use is the three ELF compressed packages given by VirusShare. We have checked whether it is compressed and whether it is an Android program to ensure that our dataset is completely aimed at the IoT field. However, it must be noted that the files in this compressed package are from 2016 to 2020. As we mentioned before, labels are time-sensitive. The basis of

our label research is the VT website. There may be label deviations in the samples due to time. The second validity threat is our model based on static features and CFG graph clustering may affect the answer to the inquiry question. In response to this problem, we used the suggestions of Carrillo (Chakraborty et al., 2017) and Cozzi (Cozzi et al., 2020) for reference, and the model was also verified by Yan (Yan et al., 2022) and Yang (Yang et al., 2021). We use 10-fold cross-validation for all supervised experiments to ensure that the models are fully trained.

In addition, we excluded smaller samples from the variant-level experiments; to guarantee the results of the experiments and large families usually have more variant types. However, we believe that minority families are also critical. From our experiments, we can see that the detection rate of many engines in minority families is low, which is also one of the focuses of future research. Finally, some ground-truth public datasets have appeared in Windows malware (Joyce et al., 2023), which is of great significance for improving the credibility of Anti-Virus labels and model accuracy. Unfortunately, obtaining samples in the IoT field is complex, with few public datasets. Building a credible ground-truth IoT dataset may be the key to improving the quality of IoT labels.

We provide some gentle and effective suggestions for the label source of IoT malware in this paper, including what label source to use for the family and variant levels. In the future, we will study how to obtain further higher-quality label columns based on these suggestions.

5. Conclusions and future work

In recent years, IoT malware has received more and more attention from researchers, and high-quality labels are a prerequisite for practical classification tasks. For the current problem of label sources, we conduct empirical research. We analyzed the problems in the current IoT label sources, investigated the label sources of 63 studies and four public datasets, identified the critical observation objects, and constructed the differences in family and variant levels on the 45,373 samples given by the VirusShare classification model. At the family level, we performed supervised learning using static and dynamic features and compared the quality of four label sources. At the variant level, we analyzed the performance of 12 Anti-Virus engines for families of different sizes, and the scale recommends the corresponding Anti-Virus engine. We recommend that future IoT researchers use AVClass tools at the family level. At the variant level, for small families, we recommend using the label types given by Ad-Aware, BitDefender, and Emsisoft engines, and for Mirai, Gafgyt, and Tsunami for these three types of sizeable malicious code families, we recommend using the labels given by Jiangmin, NANO-Antivirus, and Avira.

Looking to the future, our core objective is to obtain higher-quality labels for IoT malware to enhance the reliability of classification models. We will continue to focus on issues related to label quality and fine-grained classification, as well as potential methods to mitigate these issues, including labeling tools, more effective engine rules, and possible variant-level classification approaches. We aim to provide further analysis and discussion on these methods. Through our research into label sources, we hope to improve researchers' confidence in current classification models and the models' inherent reliability. Next, we plan to experiment with more and newer publicly available IoT datasets, explore their data sources, and verify whether our conclusions remain reliable with new data, offering suggestions for researchers.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by Major Scientific and Technological Innovation Projects of Shandong Province (2020CXGC010116) and the National Natural Science Foundation of China (No. 62172042).

References

Ahmed, I., Anisetti, M., Ahmad, A., Jeon, G., 2022. A multilayer deep learning approach for malware classification in 5G-enabled IIoT. IEEE Trans. Ind. Inform. 19 (2), 1495–1503.

Alasmary, H., Khormali, A., Anwar, A., Park, J., Choi, J., Abusnaina, A., Awad, A., Nyang, D., Mohaisen, A., 2019. Analyzing and detecting emerging Internet of Things malware: A graph-based approach. IEEE Internet Things J. 6 (5), 8977–8988.

Alhanahnah, M., Lin, Q., Yan, Q., Zhang, N., Chen, Z., 2018. Efficient signature generation for classifying cross-architecture IoT malware. In: 2018 IEEE Conference on Communications and Network Security. CNS, pp. 1–9. http://dx.doi.org/10.1109/CNS.2018.8433203.

Allix, K., Bissyandé, T.F., Klein, J., Le Traon, Y., 2016. Androzoo: Collecting millions of android apps for the research community. In: Proceedings of the 13th International Conference on Mining Software Repositories. pp. 468–471.

Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., 2014. DREBIN: Effective and explainable detection of android malware in your pocket. In: Network and Distributed System Security Symposium.

Asmitha, K.A., Vinod, P., 2014. A machine learning approach for Linux malware detection. In: 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques. ICICT, pp. 825–830.

Aurangzeb, S., Anwar, H., Naeem, M.A., Aleem, M., 2022. BigRC-EML: Big-data based ransomware classification using ensemble machine learning. Cluster Comput. 25 (5), 3405–3422.

AV-TEST2022, AV-TEST2022, https://www.av-test.org.

Avast, 2023. https://www.avast.com/.

Avira, 2023. https://www.avira.com/.

Bai, J., Yang, Y., Mu, S.G., Ma, Y., 2013. Malware detection through mining symbol table of Linux executables. Inf. Technol. J. 12, 380–384.

Bailey, M., Oberheide, J., Andersen, J., Mao, Z.M., Jahanian, F., Nazario, J., 2007. Automated classification and analysis of internet malware. In: Recent Advances in Intrusion Detection: 10th International Symposium, RAID 2007, Gold Goast, Australia, September 5-7, 2007. Proceedings 10. Springer, pp. 178–197.

Belguendouz, H., Guerid, H., Kaddour, M., 2022. Static classification of IoT malware using grayscale image representation and lightweight convolutional neural networks. In: 2022 5th International Conference on Advanced Communication Technologies and Networking. CommNet, IEEE, pp. 1–8.

Bontchev, V., Current status of the CARO malware naming scheme, https://bontchev.nlcv.bas.bg/papers/naming.html.

Canto, J., Dacier, M., Kirda, E., Leita, C., 2008. Large scale malware collection: Lessons learned. In: IEEE SRDS Workshop on Sharing Field Data and Experiment Measurements on Resilience of Distributed Computing Systems. p. 6.

CARO, 1991. http://www.caro.org/articles/naming.html.

Carrillo-Mondejar, J., Martínez, J., Suarez-Tangil, G., 2020. Characterizing Linux-based malware: Findings and recent trends. Future Gener. Comput. Syst. 110, 267–281.

Catalano, C., Chezzi, A., Angelelli, M., Tommasi, F., 2022. Deceiving AI-based malware detection through polymorphic attacks. Comput. Ind. 143, 103751.

Chaganti, R., Ravi, V., Pham, T.D., 2022. Deep learning based cross architecture Internet of Things malware detection and classification. Comput. Secur. 120, 102779.

Chakraborty, S., Stokes, J.W., Xiao, L., Zhou, D., Marinescu, M., Thomas, A., 2017. Hierarchical learning for automated malware classification. In: 2017 IEEE Military Communications Conference. MILCOM, IEEE, pp. 23–28.

Chen, Z., Brophy, E., Ward, T., 2021. Malware classification using static disassembly and machine learning. arXiv preprint arXiv:2201.07649.

Clincy, V., Shahriar, H., 2019. IoT malware analysis. In: 2019 IEEE 43rd Annual Computer Software and Applications Conference, Vol. 1. COMPSAC, pp. 920–921.

Conti, M., Khandhar, S., Vinod, P., 2022. A few-shot malware classification approach for unknown family recognition using malware feature visualization. Comput. Secur. 122, 102887.

Cozzi, E., Graziano, M., Fratantonio, Y., Balzarotti, D., 2018. Understanding Linux malware. In: 2018 IEEE Symposium on Security and Privacy. SP, pp. 161–175.

Cozzi, E., Vervier, P.-A., Dell'Amico, M., Shen, Y., Bilge, L., Balzarotti, D., 2020. The tangled genealogy of IoT malware. In: Annual Computer Security Applications Conference. Association for Computing Machinery, pp. 1–16.

Cui, Z., Xue, F., Cai, X., Cao, Y., Wang, G.-g., Chen, J., 2018. Detection of malicious code variants based on deep learning. IEEE Trans. Ind. Inform. 14 (7), 3187–3196.

Darabian, H., Dehghantanha, A., Hashemi, S., Homayoun, S., Choo, K.-K.R., 2020. An opcode-based technique for polymorphic Internet of Things malware detection. Concurr. Comput.: Pract. Exper. 32 (6), e5173.

Deng, L., Wen, H., Xin, M., Sun, Y., Sun, L., Zhu, H., 2020. Malware classification using attention-based transductive learning network. In: Security and Privacy in Communication Networks: 16th EAI International Conference, SecureComm 2020, Washington, DC, USA, October 21-23, 2020, Proceedings, Part II 16. Springer, pp. 403–418.

Dib, M., Torabi, S., Bou-Harb, E., Assi, C., 2021. A multi-dimensional deep learning framework for IoT malware classification and family attribution. IEEE Trans. Netw. Serv. Manag. 18 (2), 1165–1177.

ESET, 2023. https://www.eset.com/.

Fang, C., Liu, Z., Shi, Y., Huang, J., Shi, Q., 2020. Functional code clone detection with syntax and semantics fusion learning. In: Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis. pp. 516–527.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F., 2015. Efficient and robust automated machine learning. In: Advances in Neural Information Processing Systems, Vol. 28. 2015, pp. 2962–2970.

Ge, X., Huang, Y., Hui, Z., Wang, X., Cao, X., 2021. Impact of datasets on machine learning based methods in android malware detection: an empirical study. In: 2021 IEEE 21st International Conference on Software Quality, Reliability and Security. QRS, pp. 81–92. http://dx.doi.org/10.1109/QRS54544.2021.00019.

Ghouti, L., Imam, M., 2020. Malware classification using compact image features and multiclass support vector machines. IET Inf. Secur. 14 (4), 419–429.

Grover, A., Leskovec, J., 2016. node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 855–864.

HaddadPajouh, H., Dehghantanha, A., Khayami, R., Choo, K.-K.R., 2018. A deep recurrent neural network based approach for Internet of Things malware threat hunting. Future Gener. Comput. Syst. 85, 88–96.

Hoang, D.K., Vu, D.L., et al., 2020. IoT malware classification based on system calls. In: 2020 RIVF International Conference on Computing and Communication Technologies. RIVF, IEEE, pp. 1–6.

Hu, X., Chiueh, T.-c., Shin, K.G., 2009. Large-scale malware indexing using function-call graphs. In: Proceedings of the 16th ACM Conference on Computer and Communications Security. pp. 611–620.

Hurier, M., Suarez-Tangil, G., Dash, S.K., Bissyandé, T.F., Traon, Y.L., Klein, J., Cavallaro, L., 2017. Euphony: Harmonious unification of cacophonous anti-virus vendor labels for android malware. In: 2017 IEEE/ACM 14th International Conference on Mining Software Repositories. MSR, pp. 425–435.

Hwang, C., Hwang, J., Kwak, J., Lee, T., 2020. Platform-independent malware analysis applicable to windows and Linux environments. Electronics 9 (5), 793.

Jang, J.-w., Woo, J., Mohaisen, A., Yun, J., Kim, H.K., 2015. Mal-netminer: Malware classification approach based on social network analysis of system call graph. Math. Probl. Eng. 2015.

Joyce, R.J., Amlani, D., Nicholas, C., Raff, E., 2023. MOTIF: A malware reference dataset with ground truth family labels. Comput. Secur. 124, 102921.

Kantchelian, A., Tschantz, M.C., Afroz, S., Miller, B., Shankar, V., Bachwani, R., Joseph, A.D., Tygar, J.D., 2015. Better malware ground truth: Techniques for weighting anti-virus vendor labels. In: Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security. pp. 45–56.

Karanja, E.M., Masupe, S., Jeffrey, M.G., 2020. Analysis of Internet of Things malware using image texture features and machine learning techniques. Internet of Things 9, 100153.

kaspersky, 2023. https://www.kaspersky.com.cn/.

Kato, S., Tanabe, R., Yoshioka, K., Matsumoto, T., 2021. Adaptive observation of emerging cyber attacks targeting various IoT devices. In: 2021 IFIP/IEEE International Symposium on Integrated Network Management. IM, pp. 143–151.

Kim, S., Jung, W., Lee, K., Oh, H.-G., Kim, E.T., 2022. Sumav: Fully automated malware labeling. ICT Express 8, 530–538.

Kumar, N., Handa, A., 2021. Detection of advanced Linux malware using machine learning. In: Advances in Computational Intelligence and Communication Technology. Springer Singapore, pp. 185–194.

Landman, T., Nissim, N., 2021. Deep-Hook: A trusted deep learning-based framework for unknown malware detection and classification in Linux cloud environments. Neural Netw. 144, 648–685.

Lee, Y.-T., Ban, T., Wan, T.-L., Cheng, S.-M., Isawa, R., Takahashi, T., Inoue, D., 2020. Cross platform IoT-malware family classification based on printable strings. In: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications. TrustCom, pp. 775–784. http://dx.doi.org/10.1109/TrustCom50675.2020.00106.

Lee, S., Jeon, H.-G., Park, G., Kim, J., Youn, J.M., 2021. IoT malware static and dynamic analysis system. J. Hum.-Centric Sci. Technol. Innov.

Li, Q., Mi, J., Li, W., Wang, J., Cheng, M., 2021. CNN-based malware variants detection method for Internet of Things. IEEE Internet Things J. 8 (23), 16946–16962.

Microsoft Security, 2023. https://www.microsoft.com/security.

Mitsuhashi, R., Shinagawa, T., 2022. Exploring optimal deep learning models for image-based malware variant classification. In: 2022 IEEE 46th Annual Computers, Software, and Applications Conference. COMPSAC, IEEE, pp. 779–788.

Mohaisen, A., Alrawi, O., 2014. Av-meter: An evaluation of antivirus scans and labels. In: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, pp. 112–131.

Musgrave, J., Messay-Kebede, T., Kapp, D., Ralescu, A., 2022. A novel feature representation for malware classification. arXiv preprint arXiv:2210.09580.

Naeem, H., Guo, B., Ullah, F., Naeem, M.R., 2019. A cross-platform malware variant classification based on image representation. KSII Trans. Internet Inf. Syst. (TIIS) 13 (7), 3756–3777.

Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., Jaiswal, S., 2017. graph2vec: Learning distributed representations of graphs. arXiv preprint arXiv:1707.05005.

Nataraj, L., Karthikeyan, S., Jacob, G., Manjunath, B.S., 2011a. Malware images: visualization and automatic classification. In: Proceedings of the 8th International Symposium on Visualization for Cyber Security. pp. 1–7.

Nataraj, L., Karthikeyan, S., Jacob, G., Manjunath, B.S., 2011b. Malware images: Visualization and automatic classification. In: Visualization for Computer Security.

Nguyen, H.-T., Ngo, Q.-D., Le, V.-H., 2018a. IoT botnet detection approach based on PSI graph and DGCNN classifier. In: 2018 IEEE International Conference on Information Communication and Signal Processing. ICICSP, pp. 118–122. http://dx.doi.org/10.1109/ICICSP.2018.8549713.

Nguyen, H.-T., Ngo, Q.-D., Le, V.-H., 2020. A novel graph-based approach for IoT botnet detection. Int. J. Inf. Secur. 19 (5), 567–577.

Nguyen, K.D.T., Tuan, T.M., Le, S.H., Viet, A.P., Ogawa, M., Le Minh, N., 2018b. Comparison of three deep learning-based approaches for IoT malware detection. In: 2018 10th International Conference on Knowledge and Systems Engineering. KSE, IEEE, pp. 382–388.

Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W., 2016. Asymmetric transitivity preserving graph embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1105–1114.

Pham, D.-P., Marion, D., Mastio, M., Heuser, A., 2021. Obfuscation revealed: Leveraging electromagnetic signals for obfuscated malware classification. In: Annual Computer Security Applications Conference. pp. 706–719.

Phu, T.N., Hoang, L.H., Toan, N.N., Tho, N.D., Binh, N.N., 2019. CFDVex: A novel feature extraction method for detecting cross-architecture IoT malware. In: Proceedings of the 10th International Symposium on Information and Communication Technology. pp. 248–254.

PyCharm, 2023. https://www.jetbrains.com/pycharm/.

Qiao, Y., Zhang, W., Du, X., Guizani, M., 2021. Malware classification based on multilayer perception and Word2Vec for IoT security. ACM Trans. Internet Technol. (TOIT) 22 (1), 1–22.

Qiao, Y., Zhang, W., Tian, Z., Yang, L.T., Liu, Y., Alazab, M., 2022. Adversarial ELF malware detection method using model interpretation. IEEE Trans. Ind. Inform. 19 (1), 605–615.

Ravi, V., Alazab, M., 2023. Attention-based convolutional neural network deep learning approach for robust malware classification. Comput. Intell. 39 (1), 145–168.

Ronen, R., Radu, M., Feuerstein, C., Yom-Tov, E., Ahmadi, M., 2018. Microsoft malware classification challenge. arXiv:1802.10135.

Samantray, O.P., Tripathy, S.N., 2020. A knowledge-domain analyser for malware classification. In: 2020 International Conference on Computer Science, Engineering and Applications. ICCSEA, IEEE, pp. 1–7.

Samantray, O.P., Tripathy, S.N., 2021. IoT-malware classification model using byte sequences and supervised learning techniques. In: Next Generation of Internet of Things: Proceedings of ICNGIoT 2021. Springer, pp. 51–60.

Sánchez-Fraga, R., Acosta-Bermejo, R., 2022. Toward a taxonomy and multi-label dataset for malware classification. In: 2022 10th International Conference in Software Engineering Research and Innovation. CONISOFT, IEEE, pp. 150–157.

Sebastian, S., Caballero, J., 2020. AVClass2: Massive malware tag extraction from AV labels. In: ACSAC '20: Annual Computer Security Applications Conference. pp. 42–53.

Sebastián, M., Rivera, R., Kotzias, P., Caballero, J., 2016. Avclass: A tool for massive malware labeling. In: International Symposium on Research in Attacks, Intrusions, and Defenses. Springer, pp. 230–253.

Seshagiri, P., Vazhayil, A., Sriram, P., 2016. AMA: Static code analysis of web page for the detection of malicious scripts. Procedia Comput. Sci. 93, 768–773.

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

Shahzad, F., Farooq, M., 2012. ELF-Miner: Using structural knowledge and data mining methods to detect new (Linux) malicious executables. Knowl. Inf. Syst. 30 (3), 589–612.

Shalaginov, A., Verlier, L., 2021. A novel study on multinomial classification of X86/X64 Linux ELF malware types and families through deep neural networks. In: Malware Analysis using Artificial Intelligence and Deep Learning. Springer International Publishing, pp. 437–453.

Shang, S., Zheng, N., Xu, J., Xu, M., Zhang, H., 2010. Detecting malware variants via function-call graph similarity. In: 2010 5th International Conference on Malicious and Unwanted Software. IEEE, pp. 113–120.

Sharma, S., Rama Krishna, C., Sahay, S.K., 2019. Detection of advanced malware by machine learning techniques. In: Soft Computing: Theories and Applications: Proceedings of SoCTA 2017. Springer, pp. 333–342.

Su, J., Vasconcellos, D.V., Prasad, S., Sgandurra, D., Feng, Y., Sakurai, K., 2018. Lightweight classification of IoT malware based on image recognition. In: 2018 IEEE 42Nd Annual Computer Software and Applications Conference, Vol. 2. COMPSAC, IEEE, pp. 664–669.

Symantec, 2023. https://www.broadcom.com/support/symantec.

Tahir, R., 2018. A study on malware and malware detection techniques. Int. J. Educ. Manag. Eng. 8 (2), 20–30.

Tien, C.-W., Chen, S.-W., Ban, T., Kuo, S.-Y., 2020. Machine learning framework to analyze IoT malware using elf and opcode features. Digital Threats: Res. Pract. 1 (1), 1–19.

Toan, N.N., Thang, D.Q., et al., 2022. Static feature selection for IoT malware detection. J. Sci. Technol. Inf. Secur. 1 (15), 74–84.

Torabi, S., Dib, M., Bou-Harb, E., Assi, C., Debbabi, M., 2021. A strings-based similarity analysis approach for characterizing IoT malware and inferring their underlying relationships. IEEE Netw. Lett. 3 (3), 161–165. http://dx.doi.org/10.1109/LNET.2021.3076600.

Vasan, D., Alazab, M., Venkatraman, S., Akram, J., Qin, Z., 2020. MTHAEL: Cross-architecture IoT malware detection based on neural network advanced ensemble learning. IEEE Trans. Comput. 69 (11), 1654–1667. http://dx.doi.org/10.1109/TC.2020.3015584.

VirusShare, 2023. https://www.virusshare.com/.

VirusTotal, 2023. http://www.virustotal.com.

Vouvoutsis, V., Casino, F., Patsakis, C., 2022. On the effectiveness of binary emulation in malware classification. J. Inf. Secur. Appl. 68, 103258.

Wan, T.-L., Ban, T., Cheng, S.-M., Lee, Y.-T., Sun, B., Isawa, R., Takahashi, T., Inoue, D., 2020a. Efficient detection and classification of Internet-of-Things malware based on byte sequences from executable files. IEEE Open J. Comput. Soc. 1, 262–275.

Wan, T.-L., Ban, T., Lee, Y.-T., Cheng, S.-M., Isawa, R., Takahashi, T., Inoue, D., 2020b. IoT-malware detection based on byte sequences of executable files. In: 2020 15th Asia Joint Conference on Information Security. AsiaJCIS, IEEE, pp. 143–150.

Wang, D., Cui, P., Zhu, W., 2016. Structural deep network embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1225–1234.

Wang, F., Shoshitaishvili, Y., 2017. Angr-the next generation of binary analysis. In: 2017 IEEE Cybersecurity Development. SecDev, IEEE, pp. 8–9.

Wang, L., Wang, H., Luo, X., Sui, Y., 2023. MalWhiteout: Reducing Label Errors in Android Malware Detection. Association for Computing Machinery, New York, NY, USA, ISBN: 9781450394758.

Wang, H., Zhang, W., He, H., Liu, P., Luo, D.X., Liu, Y., Jiang, J., Li, Y., Zhang, X., Liu, W., et al., 2021. An evolutionary study of IoT malware. IEEE Internet Things J. 8 (20), 15422–15440.

Xiao, G., Li, J., Chen, Y., Li, K., 2020. MalFCS: An effective malware classification framework with automated feature extraction based on deep convolutional neural networks. J. Parallel Distrib. Comput. 141, 49–58.

Xiao, F., Lin, Z., Sun, Y., Ma, Y., 2019. Malware detection based on deep learning of behavior graphs. Math. Probl. Eng. 2019, 1–10.

Yan, J., Jia, X., Ying, L., Su, P., 2022. Understanding and mitigating label bias in malware classification: An empirical study. In: 2022 IEEE 22nd International Conference on Software Quality, Reliability and Security. QRS, IEEE, pp. 492–503.

Yang, L., Ciptadi, A., Laziuk, I., Ahmadzadeh, A., Wang, G., 2021. BODMAS: An open dataset for learning based temporal analysis of PE malware. In: 2021 IEEE Security and Privacy Workshops. SPW, IEEE, pp. 78–84.

Ye, Y., Li, T., Adjeroh, D., Iyengar, S.S., 2017. A survey on malware detection using data mining techniques. ACM Comput. Surv. 50 (3), 1–40.

Yoo, S., Kim, S., Kim, S., Kang, B.B., 2021. AI-HydRa: Advanced hybrid approach using random forest and deep learning for malware classification. Inform. Sci. 546, 420–435.

Yuan, B., Wang, J., Wu, P., Qing, X., 2021. IoT malware classification based on lightweight convolutional neural networks. IEEE Internet Things J. 9 (5), 3770–3783.

Zhao, Y., Kuerban, A., 2023. MDABP: A novel approach to detect cross-architecture IoT malware based on paas. Sensors 23 (6), 3060.

Zhu, S., Shi, J., Yang, L., Qin, B., Zhang, Z., Song, L., Wang, G., 2020. Measuring and modeling the label dynamics of online anti-malware engines. In: USENIX Security Symposium. pp. 2361–2378.