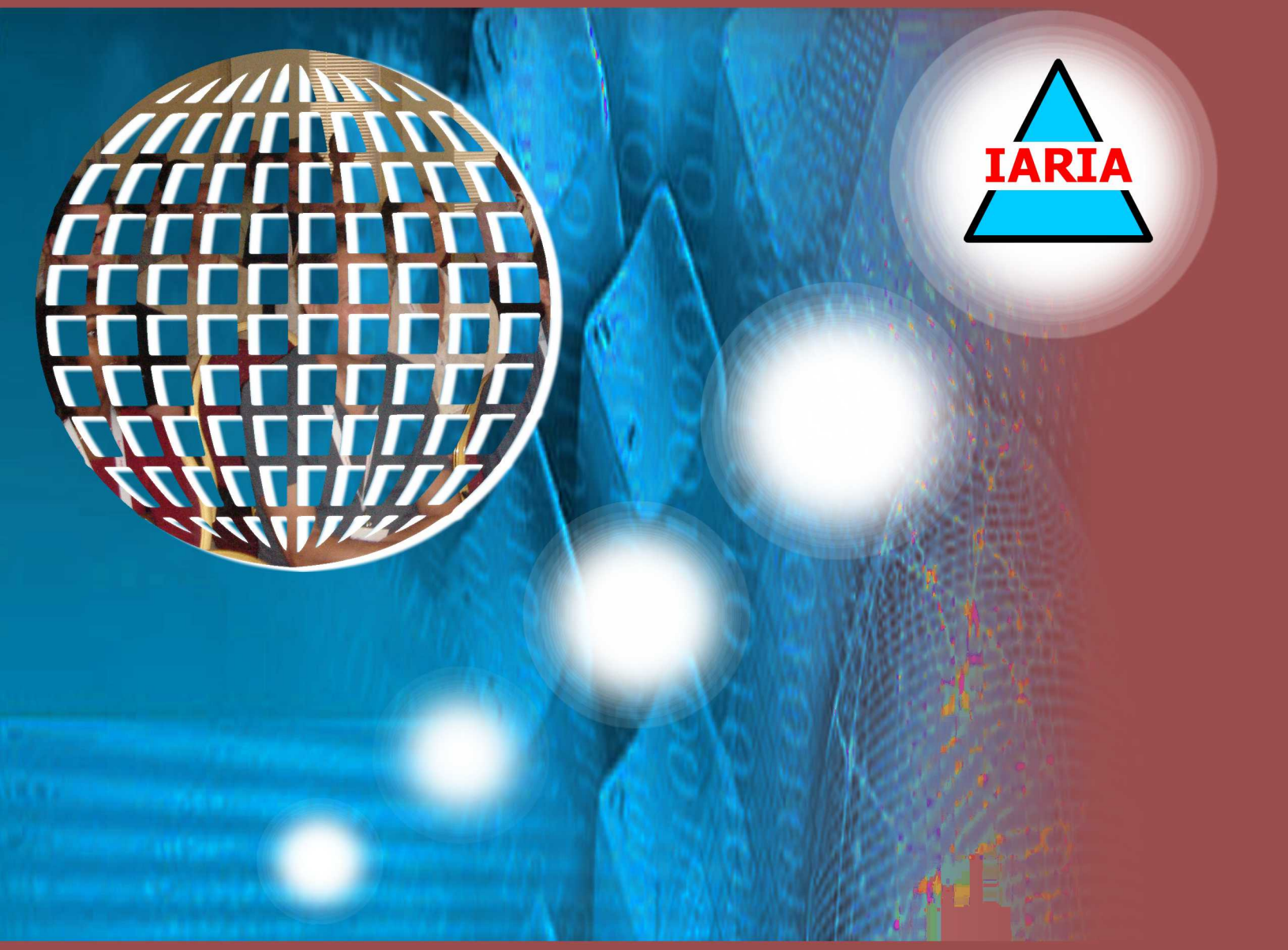
International Journal on Advances in Security



2023 vol. 16 nr. 3&4

The *International Journal on Advances in Security* is published by IARIA.
ISSN: 1942-2636
journals site: http://www.iariajournals.org
contact: petre@iaria.org

Sponsored by IARIA
www.iaria.org

International Journal on Advances in Security
Volume 16, Number 3 & 4, 2023

Sylvia Kierkegaard, IAITL-International Association of IT Lawyers, Denmark
Hyunsung Kim, Kyungil University, Korea
Geir M. Køien, University of Agder, Norway
Ah-Lian Kor, Leeds Metropolitan University, UK
Evangelos Kranakis, Carleton University - Ottawa, Canada
Lam-for Kwok, City University of Hong Kong, Hong Kong
Jean-Francois Lalande, ENSI de Bourges, France
Gyungho Lee, Korea University, South Korea
Clement Leung, Hong Kong Baptist University, Kowloon, Hong Kong
Diego Liberati, Italian National Research Council, Italy
Giovanni Livraga, Università degli Studi di Milano, Italy
Gui Lu Long, Tsinghua University, China
Jia-Ning Luo, Ming Chuan University, Taiwan
Thomas Margoni, University of Western Ontario, Canada
Rivalino Matias Jr ., Federal University of Uberlandia, Brazil
Manuel Mazzara, UNU-IIST, Macau / Newcastle University, UK
Catherine Meadows, Naval Research Laboratory - Washington DC, USA
Carla Merkle Westphall, Federal University of Santa Catarina (UFSC), Brazil
Ajaz H. Mir, National Institute of Technology, Srinagar, India
Jose Manuel Moya, Technical University of Madrid, Spain
Leonardo Mostarda, Middlesex University, UK
Jogesh K. Muppala, The Hong Kong University of Science and Technology, Hong Kong
Syed Naqvi, CETIC (Centre d'Excellence en Technologies de l'Information et de la Communication),Belgium
Sarmistha Neogy, Jadavpur University, India
Mats Neovius, Åbo Akademi University, Finland
Jason R.C. Nurse, University of Oxford, UK
Peter Parycek, Donau-Universität Krems, Austria
Konstantinos Patsakis, Rovira i Virgili University, Spain
João Paulo Barraca, University of Aveiro, Portugal
Sergio Pozo Hidalgo, University of Seville, Spain
Yong Man Ro, KAIST (Korea advanced Institute of Science and Technology), Korea
Rodrigo Roman Castro, University of Malaga, Spain
Heiko Roßnagel, Fraunhofer Institute for Industrial Engineering IAO, Germany
Claus-Peter Rückemann, Universität Münster / DIMF / Leibniz Universität Hannover, Germany
Antonio Ruiz Martinez, University of Murcia, Spain
Paul Sant, University of Bedfordshire, UK
Peter Schartner, University of Klagenfurt, Austria
Alireza Shameli Sendi, Ecole Polytechnique de Montreal, Canada
Dimitrios Serpanos, Univ. of Patras and ISI/RC ATHENA, Greece
Pedro Sousa, University of Minho, Portugal
George Spanoudakis, City University London, UK
Vladimir Stantchev, Institute of Information Systems, SRH University Berlin, Germany
Lars Strand, Nofas, Norway
Young-Joo Suh, Pohang University of Science and Technology (POSTECH), Korea
Jani Suomalainen, VTT Technical Research Centre of Finland, Finland
Enrico Thomae, Ruhr-University Bochum, Germany

International Journal on Advances in Security
Volume 16, Numbers 3 & 4, 2023

CONTENTS

Evaluation of Threat Information Quality Provided by Twitter

Ryu Saeki
Department of Computer Science and Engineering
Fukuoka Institute of Technology
Fukuoka, Japan
mfm22105@bene.fit.ac.jp

Kazumasa Oida
Department of Computer Science and Engineering
Fukuoka Institute of Technology
Fukuoka, Japan
oida@fit.ac.jp

***Abstract*—Twitter (currently being re-branded as "X") is widely used as a tool for collecting and disseminating information about the latest security incidents. However, the quality of threat information provided by Twitter (earliness, detailedness, and reliability) has not been studied sufficiently so far. Fresh information is required by both security experts and non-experts. Detailedness and reliability are also important criteria in two aspects. First, there are many accounts on Twitter, and anyone is free to post fake news. Second, true information that is not detailed is inconsequential to experts. This study compares the quality provided by Twitter and a security news site, which is expected to be very trustworthy. Because Emotet is having a serious impact in Japan, this study verified the earliness and detailedness by measuring when and how often words characterizing Emotet variants have appeared on Twitter and the news site in the past. Experiments revealed that Twitter alerted far earlier and more frequently about more diverse malware types, malicious attachment extensions, and malicious subject lines. Reliability was assessed based on two criteria: website reliability and text reliability. The news site was superior in terms of website reliability. In terms of text reliability, on the other hand, their difference was insignificant. The text reliability was derived from discrepancies between articles about the same security incidents, which were detected by humans and a state-of-the-art machine learning model. Overall, the quality of information on Twitter is higher than on the news site.**

Keywords*—*cyber security; Twitter; cyberthreat intelligence; threat information quality; ChatGPT.

I. Introduction

This study extends our original conference paper [1] in three aspects. First, the quality of threat information has been assessed with the addition of 2023 data. Second, web page structure containing unrelated text (e.g., advertisements, recommendations, related articles, etc.) was analyzed to collect body texts of web pages successfully. Third, an approach using Chat Generative Pre-trained Transformer (ChatGPT) [2] that automatically detects discrepancies between two descriptions was added to verify the reliability of web page content.

Twitter is a place where diverse topics are transmitted in real-time among many users. Because cybersecurity topics are also actively exchanged, many security experts are trying to extract useful Cyber Threat Intelligence (CTI) from Twitter [3]. One of the most attractive features of Twitter is its real-time nature. Because attack tactics continuously evolve and new attack techniques could suddenly emerge, cybersecurity practitioners, who want to proactively defend their organizations, are forced to retrieve timely information from Twitter.

Twitter is currently regarded as one of the major Open-Source INTelligence (OSINT) sources [4] [5]. However, except for a few studies on extracting Indicators of Compromises (IoCs) (i.e., forensic artifacts or remnants of an intrusion), quality evaluation focusing on Twitter CTI information has not been conducted [3]. Niakanlahiji et al. demonstrated the earliness of IoCs in Twitter by extracting many malicious URLs from Twitter that are not in blacklist databases [6]. Shin et al. showed the excellency in earliness, uniqueness, and accuracy of Twitter IoCs by comparing them with public CTI feeds [7].

Fake news, i.e., false or misleading information, often spreads over Twitter, especially when big events, such as the COVID-19 pandemic [8] or natural disasters [9], occur. Twitter may not always provide reliable information because anyone is free to post their own opinions. Additionally, although many security incidents are posted every day, it is not clear whether they are useful professional knowledge. The reliability and detailedness of Twitter information may not fully meet the requirements of experts and non-experts.

This study compares the earliness, detailedness, and reliability of CTI information provided by Twitter and Security NEXT [10], a major Japanese cybersecurity news site. The news site publishes daily free-access articles on security incidents and new vulnerabilities, with coverage by trusted security experts. As such, the site surely provides sufficiently reliable information with some level of detail and speed.

This study collected Japanese tweets and articles related to Emotet as a case study; Emotet is a Trojan horse that is spreading worldwide. The most common Emotet attack is to infect computer systems with various types of malware using malicious attachments in spam emails. Japan has been a major Emotet target since 2019 [11]. This study quantitatively reveals that Twitter excels in terms of detailedness and earliness but not in terms of reliability.

The remainder of this paper is organized as follows. Section II presents research related to this paper. Section III outlines security news, CTIs, and Emotet. Section IV describes our experiment and program to collect and analyzes website data. Section V presents experimental results regarding the informa-

tion quality of the two media types. Section VI discusses the possibility of automatic discrepancy detection using the state-of-the-art artificial intelligence (AI) technology ChatGPT. Section VII discusses some aspects of Twitter threat information from the perspective of our findings. Finally, the conclusion of this study and future directions are presented in Section VIII.

II. Related Work

The idea of collecting CTI information from Twitter dates back more than a decade [12]. Because it is a time-consuming task due to the enormous volume of data generation, many recent studies have proposed more efficient extraction mechanisms. Some studies make use of advanced machine learning technologies [13] [5] [14] [15], some utilize semantic knowledge bases [4], and some focus on active Twitter account tracing [16] and detection [17]. Recent approaches improve the accuracy of IoCs by comparing threat reports from six sources, including Twitter [18], or by collecting tweets from non-experts who discovered or encountered attacks [19].

The term IoC is used in the computer security domain to refer to specific patterns, markers, or evidence that indicate a security incident or breach. IoCs can be represented in different forms and types depending on attack techniques. IoCs may be IP addresses, domain names, file hashes, malware behavior patterns, or network traffic characteristics. These can help detect, investigate, and respond to attacks and breaches on computer systems and networks.

In contrast, limited studies have investigated the quality of CTI information provided by Twitter. Table I compares our approach with prior studies [6] [7] [20]. All existing studies in the table extracted IoCs and compared them with blacklists or CTIs, while this study extracted the entire text that describes Emotet attacks in order to capture characteristic attack patterns and their evolution. Moreover, these studies evaluated Twitter IoCs based on earliness or timeliness. In addition to earliness, this study evaluated Twitter information based on detailedness and reliability. For comparison, we selected Security NEXT, a cybersecurity news site, which is expected to be highly reliable.

"Earliness," "detailedness," and "reliability" are the essential criteria that should not be missing in any one of these. It is easy to understand that earliness and reliability are indispensable; detailedness is also necessary because an easy way to increase reliability is to avoid detailed descriptions so as not to increase incorrect expressions. The earliness, uniqueness, and accuracy criteria in [7] were also treated as a set. In our opinion, Shin et al. of [7] emphasize the aspect of Twitter information diversity, while this study focuses on Twitter information volume. Reference [20] extended the work in [7] using similar quality criteria.

This study collected body text on each web page and detected discrepancies among texts using ChatGPT. The study also evaluated earliness and detailedness from three categories: malware types, attachment extensions, and email subject lines. This is because many words that appear frequently in the texts and are not always classified as IoCs fall into one of these three categories.

Table I. Comparison of existing approaches that evaluate threat information quality of Twitter.

	This study	IoCMinor [6] 2019	Twiti [7] 2021	[20] 2023
Quality criteria	earliness detailedness reliability	earliness	earliness uniqueness accuracy	timeliness overlap correctness
Extracted data	texts on Emotet	IoCs	IoCs	IoCs
Baseline	news site	blacklists	public CTIs	CTIs
Data analysis	text analysis ChatGPT	graph theory etc.	machine learning	machine learning

III. Background

A. Threat Information

Security companies and experts provide the latest threat information and advice on security measures via Twitter. They share information and engage in discussion with the hashtag #Infosec. The hashtag #Cybersecurity is also widely used and is a keyword for sharing security news, vulnerability information, and best practices.

In addition to Twitter, there are various other ways to gather threat information [18]. Security vendors and information-sharing organizations, such as Computer Emergency Response Team (CERT) and Computer Security Incident Response Team (CSIRT), provide CTI information and access to threat databases. They also provide useful early warning and countermeasure information and services to automatically collect and analyze threat information. Meanwhile, security news sites and blogs present articles and analyses on the latest threat information and attack trends regularly.

B. News vs. CTI

Both security news and CTI are important components of the cybersecurity landscape. In general, they have the following different characteristics:

- Scope:
  Security news covers a broad range of topics and provides a general awareness of current happenings in the field of security, such as the latest events, incidents, breaches, vulnerabilities, and developments. While analysis and insights may be included, they are usually limited due to the nature of news reporting. Meanwhile, CTI provides in-depth analysis, context, and actionable insights about cyber threats. It involves the collection, analysis, and dissemination of information about potential or ongoing cyber threats to help understand the tactics, techniques, and procedures (TTPs) used by attackers.
- Timeliness:
  Security news typically reports real-time or near real-time information on noteworthy security incidents, data breaches, emerging vulnerabilities, and other relevant topics. CTI focuses on long-term trends, emerging threats, and potential risks, which are not immediately visible in security news.

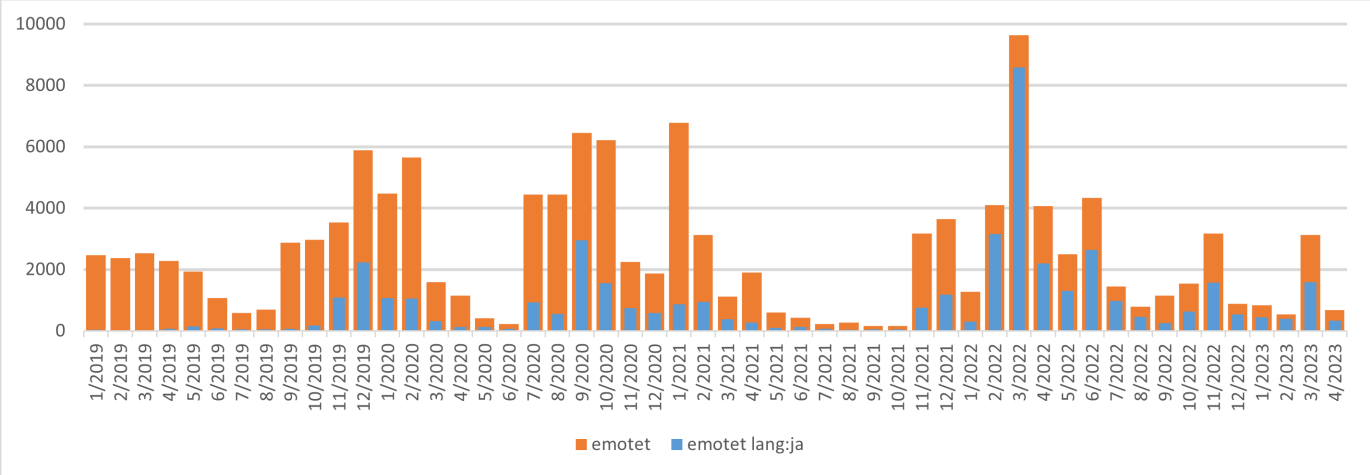


Figure 1. Number of tweets per month collected from Twitter APIs using search strings “emotet” (orange bars) and “emotet lang:ja” (blue bars). Japan is the primary target country and Emotet infection has two phases: dormant and spreading.

- Audience:
  Security news caters to a broad audience, including general users, businesses, and professionals in the cybersecurity industry. CTI is primarily targeted at cybersecurity professionals, threat intelligence analysts, incident responders, and security operations teams.

C. Emotet

Emotet is a type of banking Trojan first discovered in 2014. Emotet hijacks user computers through emails and opens a backdoor to download and install various malicious programs. Infected computers may be used as part of a botnet to send spam emails and spread various types of malware. The goal of the attack is to steal personal information, such as credit card information and passwords.

Figure 1 shows the monthly numbers of tweets collected with the search strings “emotet” and “emotet lang:ja,” where lang:ja limits the language to Japanese. The figure shows that Emotet has heavily affected Japan and that there were several major Emotet outbreaks, partly because of its attack strategy changes. Detailed and reliable Emotet variant information soon after its outbreak is indispensable.

Early Emotet used spam emails containing Microsoft Office files with malicious macros. Later, around 2016-17, Emotet replaced macros in spam emails with links through which users downloaded malicious files. In 2018-20, Emotet often downloaded TrickBot, a type of banking Trojan. Recently, it has been downloading ransomware Ryuk, which encrypts files on infected systems and demands a ransom.

Subjects of Emotet emails vary widely, but they generally have the following characteristics. They may disguise the content of invoices or orders, such as ”Invoice Payment Due,” contain subjects related to banks and financial institutions, such as ”Transaction Notification,” and use themes related to important documents, such as ”Urgent: Read Immediately.” Emotet can disguise file types and extensions and changes them frequently, making it difficult to identify Emotet by a specific extension alone.

IV. Software Structure

A. Program Overview

Figure 2 shows the structure of our Python program. Similar tools and ideas to extract IoCs from Twitter were discussed in Section II. The uniqueness of our program is to automate the process of parsing Japanese words describing Emotet threats and outputting graphs. The correctness of the output results was verified by comparing sampled outputs with manually calculated results.

As shown in Figure 2, input CSV files containing Emotet tweets are processed sequentially, characteristic Emotet words are extracted, and bar graphs are output. The following describes detailed steps 1)-6) executed by the modules in the figure (because the news site program is roughly a subset of the Twitter program in Figure 2, the two programs are described simultaneously):

1) Tweet collection: Collect all Japanese tweets containing string “emotet” using Twitter APIs. The numbers of such tweets per month are shown in Figure 1. A Google Chrome extension [21] that compiles tweets satisfying search criteria into a CSV file was used.
2) URL collection: In the case of the Twitter analysis program, collect all shortened URLs (`http://t.co/`) in the tweet, and then convert all the shortened URLs to the original URLs. Next, exclude all duplicate sites by checking whether they have the same URL, title, or text. In the case of the news site analysis program, collect URLs of all Japanese articles available on the news site.
3) Text collection: Collect text bodies (the areas enclosed by tags `<p>`) of web pages specified by the URLs mentioned above through scraping if the page titles include “emotet.” The reason why the areas enclosed by tags `<p>` are used for text body extraction is explained later in this section.

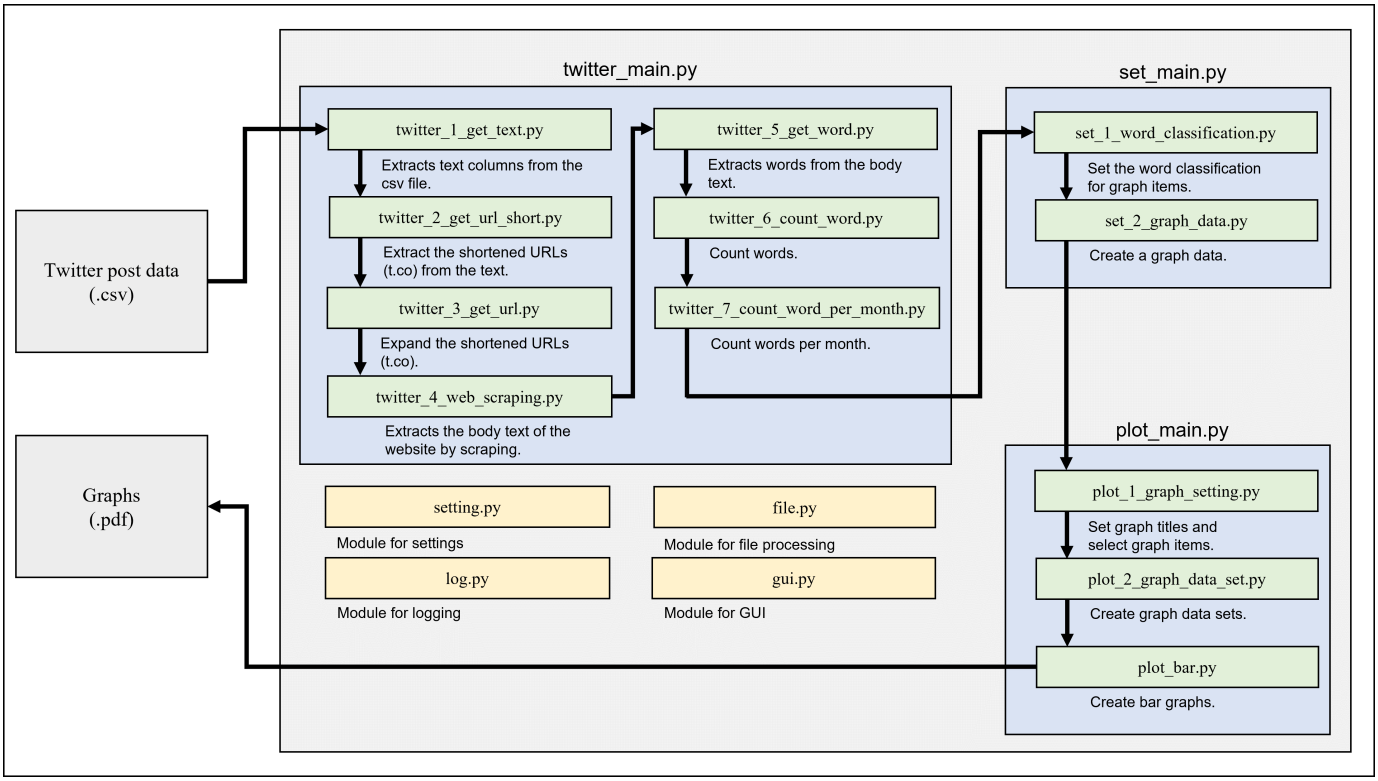


Figure 2. Twitter analysis program written in Python designed to collect words about Emotet threat from Japanese tweets and output graphs.

4) Word collection: Extract nouns or compound nouns from the texts obtained above using janome [22], a morphological analysis library.
5) Word classification: Classify all collected words into several categories based on the meanings of the words. Collected words are diverse; they include a variety of synonyms and contractions.
6) Graph generation: Create graphs (e.g., histograms) that show the frequencies of characteristic Emotet words for each category.

B. Web Page Analysis

In HyperText Markup Language (HTML) [23] documents, the `<p>` tags are generally used to group text bodies into paragraphs to provide appropriate styling and semantic separators for each paragraph. Meanwhile, the `<div>` tags are used for semantic content grouping and styling and can group different types of content, so that elements other than text bodies (e.g., advertisements) can be placed within the `<div>` tags. Although the `<p>` tag is generally not used directly for advertising, it may be used, for example, to represent a caption for an image or chart or to indicate a whole or part of a quote.

Program `twitter_4_web_scraping.py` in Figure 2 extracts body texts of web pages from text areas enclosed by `<p>` tags in the HTML codes. Although `<div>` tags can also be used to enclose text bodies, they tend to be used for areas other than the body text. Table II compares the occupancy rates (how much the body is enclosed) and unrelated rates (how much the text area is not related to the body) of `<p>` and `<div>`. For comparison, the ten most frequently tweeted web pages and ten randomly selected Security NEXT news articles were used. As can be seen from the table, the mean occupancy rate is always high, independent of the tags and the media types. On the other hand, the mean unrelated rates of `<p>` for Twitter and the news site are not greater than 12.4%, whereas those of `<div>` exceed 85%. Thus, tags `<p>` were used for text body extraction. Most unrelated texts include advertisements, recommendations, related articles, etc., so statistical results obtained using `<div>` tags are strongly influenced by such texts.

C. Word Collection

Emotet spreads via spoofed emails with malicious file attachments, and installs a variety of malware on the infected devices. Accordingly, extensions of the attachments, subject lines of the spoofed emails, and types of Emotet malware are words that characterize currently popular Emotet variants and are very valuable threat information.

The program `set_1_word_classification.py` in Figure 2 collects characteristic Emotet words using regular expressions, which specify match patterns in the body texts. Table III exemplifies regular expressions for malware type TrickBot and attachment extension ONE. As shown in the table, Microsoft OneNote files correspond to those that include either one or onenote.

Table II. Number of words in the body texts, occupancy rates (percentages of the text bodies that are enclosed by <p> or), and unrelated rates (percentages of text areas of <p> or that are not related to the body text). The <p> tags can extract text bodies with greater accuracy than the tags.

URLs in tweets	#words in	<p>			
	body text	occupancy rate	unrelated rate	occupancy rate	unrelated rate
cybersecurity-info.com	795	100.0%	0.0%	100.0%	97.9%
cybersecurity-jp.com	6,359	95.7%	10.9%	0.0%	100.0%
htn.to	6,908	93.3%	18.1%	99.8%	78.4%
b.hatena.ne.jp	5,401	87.3%	0.5%	99.4%	57.0%
www.itmedia.co.jp	2,397	95.6%	8.0%	100.0%	93.6%
news.mynavi.jp	3,770	95.5%	10.7%	100.0%	84.2%
newsrelea.se	1,415	63.1%	3.3%	99.8%	84.0%
xtech.nikkei.com	1,755	90.0%	21.6%	99.3%	85.4%
ameblo.jp	783	37.3%	0.0%	99.5%	79.8%
hash1da1.hatenablog.com	464	4.7%	38.6%	100.0%	94.6%
Mean	3,005	76.3%	12.4%	89.8%	85.5%
Article numbers	#words in	<p>			
in Security NEXT	body text	occupancy rate	unrelated rate	occupancy rate	unrelated rate
141138	1,068	100.0%	2.3%	100.0%	92.0%
144322	445	100.0%	6.8%	100.0%	92.7%
144577	570	100.0%	1.7%	100.0%	91.5%
136146	730	100.0%	1.8%	100.0%	100.0%
144644	311	100.0%	3.2%	100.0%	94.3%
141546	1,271	100.0%	2.0%	100.0%	90.1%
144656	354	100.0%	2.5%	100.0%	93.9%
136167	1,308	95.7%	2.1%	100.0%	90.7%
136270	1,061	100.0%	2.4%	100.0%	91.6%
144089	523	100.0%	2.1%	100.0%	91.4%
Mean	764	99.6%	2.7%	100.0%	92.8%

Table III. Regular expressions for malware type TrickBot and attachment extension ONE.

Search word	Regular expression
TrickBot	1. ^trickbot$
	2. ^trickbot[^a-zA-Z]+
	3. [^a-zA-Z]+trickbot$
	4. [^a-zA-Z]+trickbot [^a-zA-Z]+
ONE	1. ^one$
	2. ^one[^a-zA-Z]+
	3. [^a-zA-Z]+one$
	4. [^a-zA-Z]+one[^a-zA-Z]+
	5. ^onenote$
	6. ^onenote[^a-zA-Z]+
	7. [^a-zA-Z]+onenote$
	8. [^a-zA-Z]+onenote [^a-zA-Z]+

Table IV. Dataset sizes and execution times.

	Twitter	Security NEXT
Number of Websites	1,895	102
Number of words (different words)	313,814 (46,099)	6,860 (2,280)
Execution time (h)	128	8

V. Calculation Results

A. Data Size

We collected Japanese tweets and news articles describing Emotet threats posted in the period from January 1, 2019 to April 30, 2023. The numbers of Japanese tweets during this period can be seen in Figure 1. Table IV shows the number of websites we observed, the number of words (different words) in all websites, and the execution times for characteristic word extraction. The table indicates that Twitter provides larger dataset sizes; the numbers of websites and words on Twitter are 19 and 46 times greater than those of Security NEXT, respectively. The table also indicates that the number of words per website on Twitter is 2.5 times greater than that on Security NEXT. According to Table II, no news articles include more than 2,000 words, while some websites referenced from tweets exceed 6,000 words.

The execution time for Twitter was approximately 16 times longer than that for the news site. Much of the execution time was spent collecting URLs and texts (steps 2)-3) in Section IV). Therefore, the ratios of the execution times and the numbers of websites between the two media types are roughly the same.

B. Earliness

Emotet tactically distributes malware using spam emails with malicious attachments. Therefore, malware types, extensions of attachments, and subject lines of spam emails are important threat trends. Let us first verify how quickly these trends were announced via Twitter and the news site. Table V compares the dates when the two media reported each of the ten Emotet malware types [24] [25] [26] for the first time. As shown in the table, Twitter reported at least 100 days earlier for all malware types. In addition, there are four types that have not appeared on the news site yet. It is clear that security experts can more quickly share information about malware variants via Twitter.

Tables VI and VII compare the dates when the two media reported the file name extensions and email subject lines used for Emotet infection for the first time. Although Twitter provides quicker reports for these categories as well, the two categories show greater variability in delays than the malware type category. Moreover, Twitter did not issue an alert for the extension “ONE” and subject line “fire inspection” earlier than

Table V. First published dates of Emotet malware types and delays (days) of Security NEXT.

Malware	Twitter	Security NEXT	Delay
TrickBot	Apr. 13, 2019	Nov. 28, 2019	229
QakBot	Apr. 13, 2019	Oct. 8, 2020	553
Ryuk	Apr. 22, 2019	Nov. 28, 2019	220
IcedID	Apr. 13, 2019	Nov. 10, 2020	577
Zloader	Sep. 7, 2020	Dec. 22, 2020	106
Ursnif	Nov. 1, 2019	Feb. 10, 2022	101
ZeusPandaBanker	Apr. 13, 2019	unpublished	-
Gootkit	Jul. 29, 2020	unpublished	-
Conti	Mar. 22, 2021	unpublished	-
Cobalt Strike	Nov. 16, 2019	unpublished	-

Table VI. First published dates of attachment extensions used for Emotet infection and delays (days) of Security NEXT.

Extension	Twitter	Security NEXT	Delay
ZIP	May 15, 2019	Sep. 4, 2020	567
DOC	Feb. 6, 2019	Nov. 28, 2019	295
PDF	Oct. 18, 2019	Feb. 5, 2020	110
XLS	Nov. 29, 2019	Feb. 10, 2022	804
RTF	Aug. 29, 2020	unpublished	-
LNK	Jan. 23, 2020	Apr. 26, 2022	824
ONE	Mar. 16, 2023	Mar. 16, 2023	0

Table VII. First published dates of email subject lines used for Emotet infection and delays (days) of Security NEXT. A negative delay indicates that the news site published earlier.

Subject	Twitter	Security NEXT	Delay
Reply	Nov. 1, 2019	Nov. 28, 2019	27
COVID-19	Nov. 12, 2019	Feb. 5, 2020	85
Invoice	Feb. 6, 2019	Dec. 25, 2019	322
Bonus	Dec. 12, 2019	Dec. 25, 2019	13
Conference	Jan. 21, 2020	Jul. 31, 2020	192
Questionnaire	Dec. 20, 2019	Sep. 4, 2020	259
Fire inspection	Sep. 8, 2020	Sep. 4, 2020	-4
Christmas	Dec. 6, 2019	Dec. 25, 2020	385

Table VIII. Website reliability scores based on eight measurements. Security NEXT is distinctly superior to Twitter.

Measure item	Twitter	Security NEXT
1. Writer name	20/20	20/20
2. Writer's contact info.	17/20	20/20
3. Published/updated date	19/20	20/20
4. SSL certificate	17/20	20/20
5. Information sources	16/20	2/20
6. Privacy policy	17/20	20/20
7. No link errors	13/20	20/20
8. No misspellings	18/20	20/20
Total score	$\frac{120}{160} = 0.75$	$\frac{142}{160} = 0.89$

the news site. This result indicates that Twitter is not always the first to announce current Emotet threats.

C. Detailedness

Figure 3 illustrates the numbers of websites that described each of the ten Emotet malware types per year from 2019 to the first four months of 2023. Figs. 3 (left) and (right) correspond to Twitter and Security NEXT, respectively. Note that the vertical axis scale of Twitter is more than 10 times larger than that of the news sites. Note also that four malware types have not been reported on the news site yet. From the figure, Twitter has more websites reporting Emotet malware and more malware types than the news site for all years. Thus, Twitter provides more detailed malware information each year than the news site.

Figure 4 shows the numbers of websites that described each of the seven malicious file extensions. Again, Twitter has more detailed Emotet attachment information. For example, Figure 4 (left) indicates that Twitter has been alerting malicious XLS attachments every year since 2019, whereas from Figure 4 (right), Security NEXT reported them only in 2022.

Figure 5 shows the numbers of websites that described Emotet email subject types. Figure 5 (left) shows that at least four subject types appeared every year, while Figure 5 (right) shows that more than three types appeared only in 2020. Malware types are shared among experts, while malicious subject lines are sent to alert email users. Thus, information on Twitter is useful to non-experts as well. In summary, Twitter consistently provides far more detailed attack trends than Security NEXT in terms of malware type, extension, and subject line.

D. Reliability

This study evaluates the reliability of information from two perspectives: reliability of websites and reliability of text on the web pages. Table VIII compares eight measurements of twenty websites randomly selected from those referenced by Twitter and Security NEXT. The first six measurements are whether the website specifies the writer's name, writer's contact information, published/update date, Secure Sockets Layer (SSL) certificate [27], information source, and privacy policy. The last two verify whether the site has link errors and misspellings.

Table VIII concludes that Security NEXT is more reliable. The news site is perfect except that it barely specifies the source of the news articles. Although Twitter also provides a high score, reliability varies from one website to another. It should be noted that Twitter is more likely to give link errors and misspellings because, as shown in Table II, its text size is larger in most cases.

Let us next see the reliability of text on the web pages. Because it is difficult to verify whether text includes fake news, study detected discrepancies between two web pages and considered that their descriptions are not fake if there are no discrepancies. We detected discrepancies between two pages describing the same security incident, where the number of incidents we used was 53. Figure 6 shows the number of pairs of web pages describing the same security incident and the number of page pairs that include discrepancies in their descriptions. From the figure, there are eight discrepant pairs among 108 pairs referenced from different media, and there is one discrepant pair among 110 pairs referenced from Twitter. (The news site has one article per incident, so there were no discrepancy checks between news articles.) Thus, the percentage of discrepancies that occurred between the two media (between the pages referenced in tweets) is approximately 8% (1%).

Table IX shows discrepancies (highlighted in red) of the

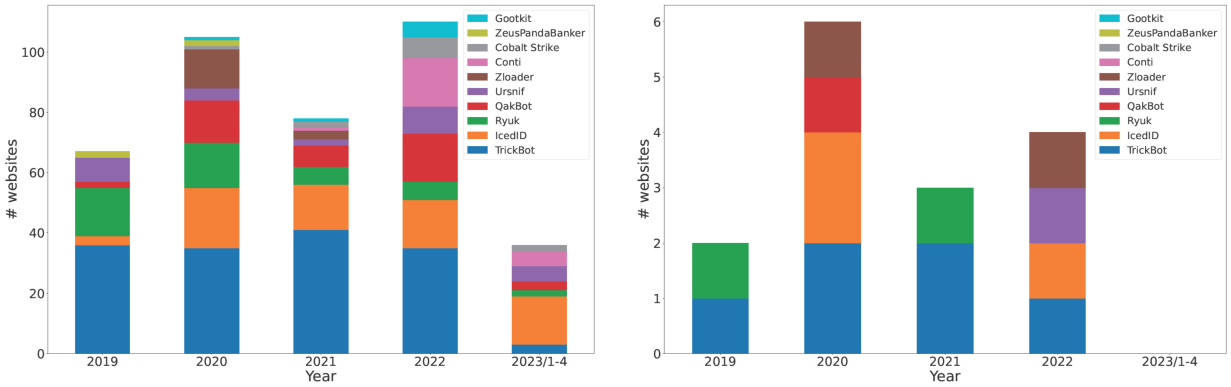


Figure 3. Yearly frequencies of Emotet malware types described on (Left) Twitter and (Right) Security NEXT.

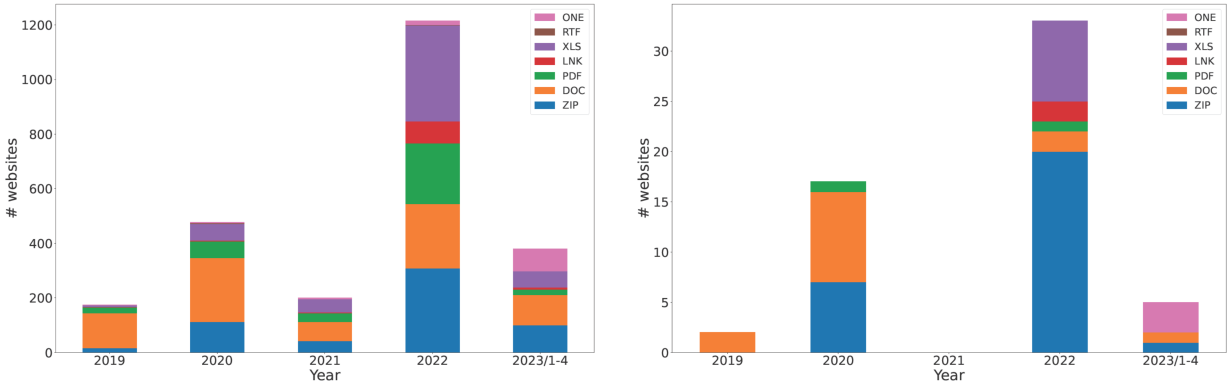


Figure 4. Yearly frequencies of Emotet attachment extensions described on (Left) Twitter and (Right) Security NEXT.

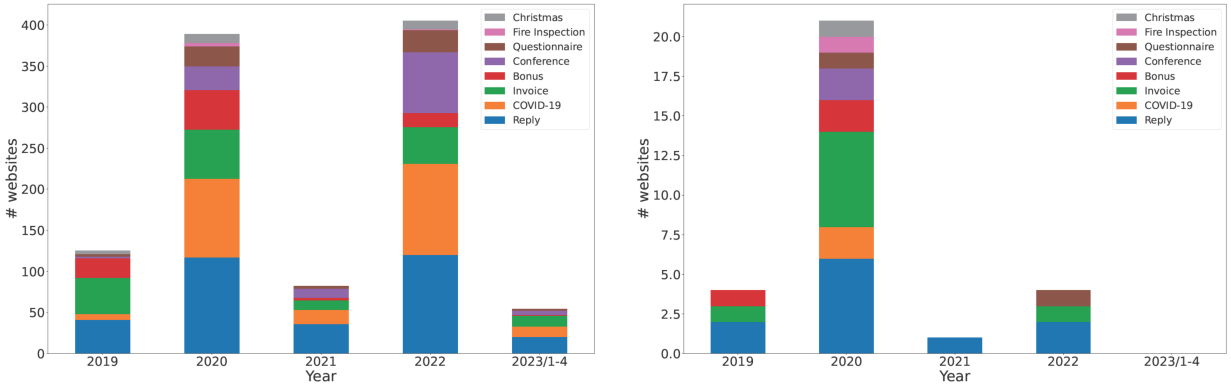


Figure 5. Yearly frequencies of Emotet subject lines described on (Left) Twitter and (Right) Security NEXT.

nine pairs. The discrepancies are all small differences in numbers and dates, except for incident 4, where two pages reported different infection routes. In other words, excluding minor numerical differences, the semantic discrepancy rate between the two media is less than 1%. Thus, the rate at which Twitter provides serious incorrect information is 1% at most.

VI. Automatic Discrepancy Detection

In the previous section, discrepancies in the descriptions of two web pages were detected by humans. When large amounts of text must be inspected, humans can make mistakes. Therefore, this section verifies whether ChatGPT can be applied to discrepancy detection. OpenAI, an American AI research laboratory, provides APIs [28] for easy access to various ChatGPT models [29] for AI developers. It is well known that slight differences in question wording could significantly alter ChatGPT's answers. Thus, we decided to develop a program that includes an OpenAI API to feed many similar questions into a ChatGPT model.

Figure 7 shows a part of our Python program. The program asks a question `question` about discrepancies between two texts: `text1` and `text2`. We prepared 144 questions with almost the same meaning. Some of them are as follows:

Table IX. Discrepancies (in red) between reports that described security incidents 1-7.

ID	Twitter	Security NEXT
1	Eighteen units were found to be infected with malware.	Eight units were found to be infected and ten were suspected.
2	On February 8, a malware infection was discovered.	On February 9, a malware infection was discovered.
3	Emails were sent to an unspecified large number of people.	Emails were sent to multiple parties.
4	The email text includes a link to an external page.	The email had an Excel file attached in macro format.
5	Emotet infection was discovered on March 15.	It was confirmed on March 16 that spoofed emails were sent after March 15.
6	2,311 leaks nationwide.	2,312 emails were leaked from infected terminals.
7	On July 11, Emotet infection was found in another terminal. On July 11, one new staff member's computer was infected.	On July 9, another employee's terminal was damaged.
ID	Twitter	Twitter
6	2,311 leaks nationwide.	A total of 2,312 emails were leaked.

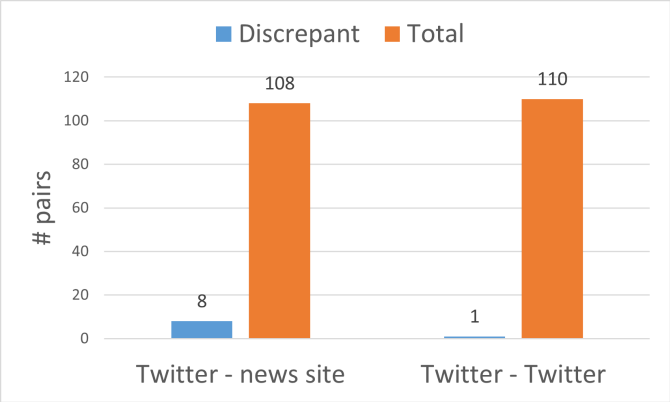


Figure 6. Among pairs of web pages describing the same incidents, there were nine pairs in which some of the descriptions do not agree. The number of security incidents was 53.

```
response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    temperature=0.5,
    messages=[
        {"role": "user",
         "content": f"{question} 1. {text1} 2. {text2}"},
    ]
)
```

Figure 7. Python code for receiving an answer to question `question` about `text1` and `text2` from `gpt-3.5-turbo`. A smaller `temperature` reduces output randomness (the default is 1.0).

- Are there any contradictions between the two texts?
- Please point out any discrepancies between the two texts.
- Are there any differences in the information in the two texts?
- Are there any inconsistencies in the information provided by the two texts?

Tables X and XI show the outputs of the AI program for text pairs with and without discrepancies, respectively. In the tables, TP and TN (FP and FN) denote the number of outputs detecting discrepancies correctly (incorrectly) and the number of outputs indicating no discrepancies correctly (incorrectly), respectively, and "else" denotes the number of unintelligible answers. The tables also present the percentages of these numbers among all 144 answers. Table X shows that the percentage of including correct answers (TP and TP & FP) is 17.2%. Note that the sum of TP and TP & FP for

Table X. AI program output for three text pairs (incidents 1, 2, and 7) that include discrepancies.

ID	TP	TP & FP	FP	FN	else	total
1	4	7	88	39	6	144
2	2	5	106	28	3	144
7	21	35	37	43	8	144
mean	6.3%	10.9%	53.5%	25.5%	3.9%	
total						

Table XI. AI program output for three text pairs (incidents 8-10) that do not include discrepancies.

ID	TN	FP	else	total
8	25	113	6	144
9	17	124	3	144
10	82	59	3	144
mean	28.7%	68.5%	2.8%	
total				

incident 2 is only 7. From this result, the 144 questions is not necessarily too many. Meanwhile, Table XI shows that the percentage of correctly answering no discrepancies is 28.7%. Thus, GPT tends to present more correct answers for text pairs without discrepancies than with discrepancies. This section has revealed that even the latest AI technology still cannot detect discrepancies automatically. In our opinion, GPT could be used as an auxiliary tool that helps reduce false positive and false negative errors.

VII. Discussion

We have learned that it is possible to gather large amounts of up-to-date threat information about Emotet through Twitter. Furthermore, we have learned that the information is not considerably less reliable than that of the security site used for comparison. However, we also found that the following considerations must be made before collecting information via Twitter:

- Twitter alone is not enough.
  Twitter tends to provide information on Emotet malware and malicious attachment extensions much earlier than the news site. However, there was a case where the news site was quicker to warn about the subject line used in Emotet emails.
- Diverse website structures.
  Compared to the news site operated by a single organization, information on Twitter is disseminated by a diverse set of people. Therefore, when collecting information via Twitter, it is necessary to consider the structure and operating policies of the sites (e.g., update frequency,

scraping, and API). For instance, as shown in Table II, Twitter cannot always provide body text successfully using only <p> tabs, while the news site can.

- Text reliability verification.
  Table VIII suggests that Security NEXT is trustworthy, but not all Twitter users are. In other words, Twitter information should be verified in most cases. As discussed in Section VI, automatic discrepancy detection is not possible at this time, so human verification is definitely required. According to the approach in Section VI, we must consider how many questions we need to prepare depending on the GPT version and parameters. Another issue is how and how often we should find sources reporting the same security incident to perform discrepancy detection. Currently, reliability is dependent on the results of verification by some valuable sources [30].

VIII. Conclusion and Future Work

Today, many people retrieve threat information through Twitter. Because there are so many accounts on Twitter and anyone can freely transmit information, it is important to know the quality of the information provided by Twitter in advance. However, there have been insufficient studies dealing with this topic. This study measured quality based on earliness, detailedness, and reliability, which are different from the quality criteria used by previous studies.

To evaluate the quality of Twitter information, this study collected tweets about Emotet threat from January 1, 2019 to April 30, 2023, of which many tweets regarding Emotet were shared in Japan. The quality was compared with that of news articles provided by Security NEXT, a major cybersecurity news site in Japan, which is expected to be very reliable.

Our results revealed that the quality of Twitter information was far superior in terms of earliness and detailedness. However, in terms of reliability of websites, the news site achieved a better score. The discrepancy rate between descriptions of the same incidents provided by the two media was less than 1% after minor numerical differences were excluded. Accordingly, we concluded that the two media rarely contain incorrect information.

In the future, we plan to develop a fake threat news monitoring system that will regularly collect texts about the same security incidents from Twitter and news sites and publish discrepancies on an ongoing basis.

Acknowledgment

This study was supported by the Fukuoka Prefectural Police. The authors would especially like to thank Mr. Jun Koga for his helpful comments over the years.

References

[1] R. Saeki and K. Oida, "Security information quality provided by news sites and twitter," in *7th International Conference on Cyber-Technologies and Cyber-Systems (CYBER 2022)*. IARIA, 2022, pp. 42–44.

[2] "Introducing chatgpt," https://openai.com/blog/chatgpt, online; accessed 22 November 2023.

[3] M. R. Rahman, R. Mahdavi-Hezaveh, and L. Williams, "What are the attackers doing now? automating cyberthreat intelligence extraction from text on pace with the changing threat landscape: A survey," *ACM Computing Surveys*, 2021.

[4] S. Mittal, P. K. Das, V. Mulwad, A. Joshi, and T. Finin, "Cybertwitter: Using twitter to generate alerts for cybersecurity threats and vulnerabilities," in *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2016, pp. 860–867.

[5] L.-M. Kristiansen, V. Agarwal, K. Franke, and R. S. Shah, "Cti-twitter: Gathering cyber threat intelligence from twitter using integrated supervised and unsupervised learning," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 2299–2308.

[6] A. Niakanlahiji, L. Safarnejad, R. Harper, and B.-T. Chu, "Iocminer: Automatic extraction of indicators of compromise from twitter," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 4747–4754.

[7] H. Shin, W. Shim, S. Kim, S. Lee, Y. G. Kang, and Y. H. Hwang, "# twiti: Social listening for threat intelligence," in *Proceedings of the Web Conference 2021*, 2021, pp. 92–104.

[8] P. Patwa, S. Sharma, S. Pykl, V. Guptha, G. Kumari, M. S. Akhtar, A. Ekbal, A. Das, and T. Chakraborty, "Fighting an infodemic: Covid-19 fake news dataset," in *Combating Online Hostile Posts in Regional Languages during Emergency Situation: First International Workshop, CONSTRAINT 2021, Collocated with AAAI 2021, Virtual Event, February 8, 2021, Revised Selected Papers 1*. Springer, 2021, pp. 21–29.

[9] H. Kanoh, "Why do people believe in fake news over the internet? an understanding from the perspective of existence of the habit of eating and drinking," *Procedia Computer Science*, vol. 126, pp. 1704–1709, 2018.

[10] "Security next provides you with the latest security news daily," https://www.security-next.com/, online; accessed 22 November 2023.

[11] "The ever-changing malware "emotet" is causing more damage in japan," https://blog.trendmicro.co.jp/archives/22959, 2019, online; accessed 22 November 2023.

[12] A. Marcus, M. S. Bernstein, O. Badar, D. R. Karger, S. Madden, and R. C. Miller, "Twitinfo: aggregating and visualizing microblogs for event exploration," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2011, pp. 227–236.

[13] U. Tekin and E. N. Yilmaz, "Obtaining cyber threat intelligence data from twitter with deep learning methods," in *2021 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*. IEEE, 2021, pp. 82–86.

[14] N. Dionísio, F. Alves, P. M. Ferreira, and A. Bessani, "Cyberthreat detection from twitter using deep neural networks," in *2019 international joint conference on neural networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[15] V. Behzadan, C. Aguirre, A. Bose, and W. Hsu, "Corpus and deep learning classifier for collection of cyber threat indicators in twitter stream," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 5002–5007.

[16] A. Bose, S. G. Sundari, V. Behzadan, and W. H. Hsu, "Tracing relevant twitter accounts active in cyber threat intelligence domain by exploiting content and structure of twitter network," in *2021 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2021, pp. 1–6.

[17] M. I. Mahaini and S. Li, "Detecting cyber security related twitter accounts and different sub-groups: a multi-classifier approach," in *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2021, pp. 599–606.

[18] J. Caballero, G. Gomez, S. Matic, G. Sánchez, S. Sebastián, and A. Villacañas, "The rise of goodfatr: a novel accuracy comparison methodology for indicator extraction tools," *Future Generation Computer Systems*, vol. 144, pp. 74–89, 2023.

[19] H. Nakano, D. Chiba, T. Koide, N. Fukushi, T. Yagi, T. Hariu, K. Yoshioka, and T. Matsumoto, "Canary in twitter mine: Collecting phishing reports from experts and non-experts," *arXiv preprint arXiv:2303.15847*, 2023.

[20] D. R. Arikkat, P. Vinod, R. K. A. Rafidha, A. D. Sorbo, C. A. Visaggio, and M. Conti, "Can twitter be used to acquire reliable alerts against novel cyber attacks?" *arXiv preprint arXiv:2306.16087*, 2023.

[21] "Tweet export," https://chrome.google.com/webstore/, online; accessed 22 November 2023.

[22] "Japanese morphological analysis engine written in pure python," https://github.com/mocobeta/janome/, online; accessed 22 November 2023.

[23] D. Connolly and L. Masinter, "Rfc2854: The'text/html'media type," 2000.

[24] "Triple threat: Emotet deploys trickbot to steal data & spread ryuk," https://www.cybereason.co.jp/blog/cyberattack/3613/, online; accessed 22 November 2023.

[25] "Threat actor profile: Ta542, from banker to malware distribution service," https://www.proofpoint.com/us/threat-insight/post/threat-actor-profile-ta542-banker-malware-distribution-service/, online; accessed 22 November 2023.

[26] "Threat spotlight: Panda banker trojan targets the us, canada and japan," https://blogs.blackberry.com/en/2018/10/threat-spotlight-panda-banker-trojan-targets-the-us-canada-and-japan/, online; accessed 22 November 2023.

[27] A. Freier, P. Karlton, and P. Kocher, "Rfc 6101: The secure sockets layer (ssl) protocol version 3.0," 2011.

[28] "Openai api," https://openai.com/blog/openai-api, online; accessed 22 November 2023.

[29] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[30] A. Tundis, S. Ruppert, and M. Mühlhäuser, "A feature-driven method for automating the assessment of osint cyber threat sources," *Computers & Security*, vol. 113, p. 102576, 2022.

Malware Self-Supervised Graph Contrastive Learning with Data Augmentation

Yun Gao*, Hirokazu Hasegawa†, Yukiko Yamaguchi*, and Hajime Shimada*
*Information Technology Center
Nagoya University, Furo-cho, Chikusa-ku, Nagoya, 464-8601, Japan
Email: gaoyun@net.itc.nagoya-u.ac.jp, yamaguchi@itc.nagoya-u.ac.jp, shimada@itc.nagoya-u.ac.jp
†Center for Strategic Cyber Resilience
National Institute of Informatics, Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430, Japan
Email: hasegawa@nii.ac.jp

***Abstract*—Traditional malware detection methods struggle to quickly and effectively keep up with the massive amount of newly created malware. Based on the features of samples, machine learning is a promising method for the detection and classification of large-scale, newly created malware. The current research trend uses machine-learning technologies to rapidly and accurately learn newly created malware. In this paper, we propose a malware classification framework based on Graph Contrastive Learning (GraphCL) with data augmentation. We first extract the Control-Flow Graph (CFG) from portable executable (PE) files and simultaneously generate node feature vectors from the disassembly code of each basic block through MiniLM, a large-scale pre-trained language model. Then four different data augmentation methods are used to expand the graph data, and the final graph representation is generated by the GraphCL model. These representations can be directly applied to downstream tasks. For our classification task, we use C-Support Vector Classification (SVC) as a classification model. To evaluate our approach, we made a CFG-based malware classification dataset from the PE files of the BODMAS Malware Dataset, which we call the Malware Geometric Multi-Class Dataset (MGD-MULTI), and collected the results. In addition, we added a new public malware graph dataset called MALNET-TINY. We also employed Local Degree Profile (LDP) to generate node representations for the graph data. The evaluation results on two malware graph dataset show that our proposal has great potential. According to our experimental evaluation, the self-supervised learning method can also achieve superior results, even surpassing the supervised learning method.**

Keywords*—malware classification; graph contrastive learning; data augmentation; self-supervised learning.

I. Introduction

In our previous work [1], we implemented self-supervised representation learning on the self-built MGD-Multi dataset, and used the representation to carry out the downstream malware classification task. In this work, we added a new public dataset to further verify the effectiveness of self-supervised contrastive learning, and evaluated different data augmentation combination.

Fueled by the progress of software technology and the internet's development, thousands of malware are created every day due to the proliferation of malware creation and obfuscation tools. Such a massive flood of data poses a considerable challenge to malware analysts and security response centers (SOCs). Traditional malware detection methods cannot continue to quickly and effectively detect such a massive amount of newly created malware. In past decades, machine learning has played an important role in information security, especially in malware detection and classification tasks. It is also a promising method to detect and classify large-scale newly created malware using the features of samples.

In the field of static malware detection, the feature extraction method of portable executable (PE) files used in the Endgame Malware Benchmark for Research (EMBER) dataset [2] has been widely applied. This feature extraction method directly provides consistent feature vectors to researchers, allowing individuals in the same field to compare their respective proposed methods. The information related to software structure, such as CFG, is rarely extracted, and most methods are based on surface analysis for extracting statistical information as features. In addition, in most malware detection and classification scenarios, the model is supervised for end-to-end training.

Supervised learning requires manual labeling of a large amount of data, and the model effect depends on the quality of the labels. Therefore, the future research trend, which is exploring unsupervised learning methods, is critical for malware detection and classification. In recent years, Graph Neural Networks (GNNs) have made remarkable progress. We can exploit their powerful representation ability to better represent malware and improve the effectiveness of its detection and classification. However, one remaining difficulty is how to represent malware in graphical form. CFG is a natural graph structure, we can generate the graph structure data of malware by extracting CFG. Therefore, we seek to classify malware by constructing a graph dataset and using unsupervised learning. Since no publicly available graph classification dataset exists for malware classification, we started by creating such a dataset.

Our contributions can be summarized as follows:

- We propose a malware classification framework based on graph contrastive learning under self-supervised learning.
- We retain the structural information of the samples extracted from CFG and embed the text features of each node with a pre-trained language model.
- We create a special graph dataset for malware classification that can be used directly on GNNs.
- Our pre-trained model can effectively represent malware in low dimensions, and this representation can be used for various downstream tasks.
- We have achieved promising results in the classification task of two malware graph datasets, and compared the effectiveness of different methods.

The remainder of this paper is organized as follows. Section II reviews related researches and highlights their methodological differences. In Section III, we describe the background knowledge of our research, as well as the methods of graph feature extraction and graph data construction. In Section IV, we discuss the principles of our proposed data augmented GraphCL-based static malware PE classification system and its application to malware classification. In Section V, we briefly discuss the implementation details of our proposal. In Section VI, we describe the corresponding experiments and evaluate their feasibility as well as the advantages and limitations of our proposal. Finally, we discuss our conclusion and describe future work in Section VII.

II. Related Work

Static malware detection allows a sample to be classified as malicious or benign without executing it. In contrast, dynamic malware detection is based on its runtime behavior and as well as its analysis, including time-dependent system call sequences [3]–[5]. A program that precisely discerns a virus from any other program by examining its appearance is infeasible, so static detection is not generally deterministic [6]. Compared with dynamic detection, the advantages of static detection are also obvious. Static detection can identify malicious files before the samples are executed. Since 1995, various machine-learning-based methods for static PE malware detection have been proposed [7]–[9].

A. Supervised-learning-based Methods

Saxe used histograms through byte-entropy values as input features and multilayer neural networks for classification [8]. Raff et al. showed that fully connected and recursive networks can be applied to malware detection problems [10]. They also used the raw bytes of PE files and built end-to-end deep learning networks [9]. Chen proposed robust PDF malware classifiers with verifiable robustness properties [11]. Coull explored malware detection byte-based deep neural network models to learn more about malware and examined the learned features at multiple levels of resolution, from individual byte embeddings to the end-to-end analysis of models [12]. Rudd proposed ALOHA, which uses multiple additional optimization objectives to enhance the model, including multi-source malicious/benign loss, count loss on multi-source detections, and semantic malware attribute tag loss [13].

B. Unsupervised-learning-based Methods

Yang presented a novel system called CADE, which can detect drifting samples that deviate from existing classes, and explained detected drift [14].

C. Graph Representation Learning

The goal of graph representation learning is to preserve as much topological information as possible when mapping nodes into vector representations. We can use its to represent malware, and do the downstream malware classification task.

Graph classification assigns a label to each graph to map the graph to the vector space. A graph kernel is dominant in history. It uses the kernel function to measure the similarity between graph pairs and maps graphs to a vector space with a mapping function. In the context of graph classification, GNNs often employ readout operations to obtain a compact representation at the graph level. GNNs have attracted a lot of attention and demonstrated amazing results in this task.

1) Supervised Learning: The Dynamic Graph Convolutional Neural Network [15] (DGCNN) uses K nearest neighbors (KNN), builds a subgraph for each node based on the node's features, and applies a graph convolution to the reconstructed graph. The Graph Isomorphism Network (GIN) [16] presents a GIN that adjusts the weights of the central nodes by learning, theoretically analyzes the GIN's expressiveness better than such GNN structures as the Graph Convolutional Network (GCN), and achieves state-of-the-art accuracy on multiple tasks.

2) Unsupervised Learning: Graph2vec [17] uses a set of all the rooted subgraphs around each node as its vocabulary through a skip-gram training process. Infograph [18] applies contrastive learning to graph learning, which is carried out in an unsupervised manner by maximizing the mutual information between graph-level and node-level representations.

3) Self-Supervised Learning: Recently, graph contrastive learning has received much attention. It has also been applied in the field of malware detection and classification. EVOLIoT [19] is a novel approach that combats "concept drift" and the limitations of inter-family IoT malware classification by detecting drifting IoT malware families and examining their diverse evolutionary trajectories. This robust and effective contrastive method learns and compares semantically meaningful representations of IoT malware binaries and codes without expensive target labels. Graph contrastive learning (GraphCL) [20] framework for learning self-supervised representations of graph data. GraphCL design four types of graph augmentations to incorporate various priors. GraphMAE [21] is a generative self-supervised graph learning method, which achieves competitive or better performance than existing contrastive methods on tasks including node classification, graph classification, and molecular property prediction. Wiener Graph Deconvolutional Network (WGDN) [22], an augmentation-adaptive decoder empowered by graph wiener filter to perform information reconstruction.

III. Background

A. Raw Graph Generation

Most malware samples exhibit polymorphic characteristics, indicating that even slight alterations in the original malware's source code can lead to substantially different compiled code [23], [24]. Cybercriminals often exploit this phenomenon to evade signature-based detection, which is a prevalent method used for malware detection [25]. Fortunately, these subtle source code changes have minimal effect on the CFG and FCG of the executable.

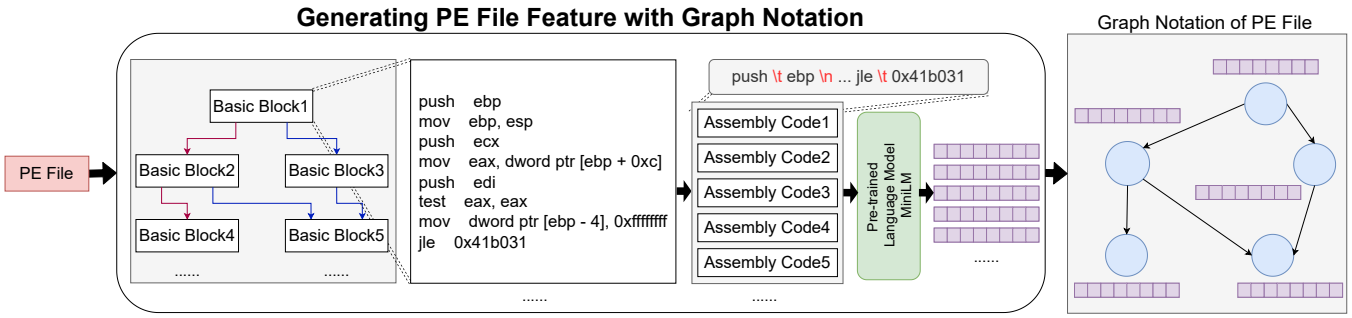


Figure 1. Raw Graph Generation for Proposal

1) CFG Structure and Disassembly Code: The CFG information is extracted from the original PE file samples, the structure information of the basic blocks is retained, and the disassembly code of each basic block is extracted. Each basic block of CFG has a corresponding disassembly code, and the relationship between each basic block is directional. Disassembly codes need to be transformed into feature vectors of specific dimensions to train GNNs. Malware CFG is usually a very large graph, so it is very time-consuming to extract CFG. Since the disassembly code in each basic block of CFG contains rich semantic information, we need to completely exploit that information and suitably embed it, for example, using a large pre-trained language model.

To train the GNNs, we need to produce graph datasets, and the main task of this module is to convert PE files into raw graphs. The overview of raw graph generation is shown in Figure 1.

2) FCG Structure and Function Call: A Function Call Graph (FCG) is a type of graph representation that captures the relationships between functions in a software program or system. It is commonly used in software analysis and program understanding to model how functions in a program interact with each other through function calls. In an FCG, each function is represented as a node, and the edges between nodes indicate function calls. If function A calls function B, there will be a directed edge from node A to node B in the FCG. The FCG provides a visual and structural representation of the flow of control and data between functions within the program.

CFG nodes contain disassembly code, while FCG nodes only include API function names or the starting memory addresses of functions. Although FCG node information is less abundant compared to CFG, it tends to focus more on modeling the graph's structural aspects. It can even generate node features without relying on the existing node information by using graph centrality measures. As a result, compared to CFG, it can efficiently and swiftly extract the graph information from binary files.

B. Non-attributed Graph Classification

Although the CFG and FCG extracted from binary files contain node information, the extraction of node features and embeddings is a matter that requires consideration. Furthermore, it is necessary to consider whether these node functions are effective.

For the task of graph classification, we need to generate node feature vectors for the graph data. For CFGs, basic blocks contain abundant disassembly code, which can be leveraged to generate node feature vectors. However, for FCGs, each node represents a function call, and there is limited semantic information available, often only comprising API names or function memory addresses. Therefore, we can directly use graph centrality measures and the structural information of the graph itself to generate corresponding node feature vectors.

Below, we introduce two types of node feature generation methods that we use.

1) Pre-trained Language Model MiniLM: MiniLM is a method released by Microsoft based on reducing large-scale transformer pre-trained models into smaller models [26]. This Deep Self-Attention Distillation (DSAD) method uses large-scale data for pre-training. The model we use is called "all-MiniLM-L12-v2," which has a 1-billion-sized training set and is designed as a general-purpose model. MiniLM model is a 12-layer transformer with a 384 hidden size and 12 attention heads that contain about 33 M parameters. It maps sentences and paragraphs to a 384-dimensional dense vector space and can be used for tasks like clustering or semantic search. This model is the fastest generation of related studies and still provides good quality. In this step, a 384-dimensional dense vector is generated for each CFG node using the pre-trained model. This vector is added to the corresponding nodes of the directed graph to generate complete graph data with node feature vectors. These directed graphs are used as our raw graph data.

2) Graph Centrality: Graph centrality refers to a set of measures used in graph theory and network analysis to determine the importance or influence of individual vertices (nodes) within a graph. Centrality measures aim to identify nodes that play crucial roles in the network's structure, functioning, and communication. Several centrality measures exist, each capturing different aspects of a node's importance. Some of the most common centrality measures include:

a) Degree Centrality: Degree centrality is the simplest centrality measure, and it is based on the number of edges incident to a node (its degree). Nodes with higher degrees are considered more central as they have more connections in the network.

b) Eigenvector Centrality: Eigenvector centrality measures a node's importance based on the centrality of its

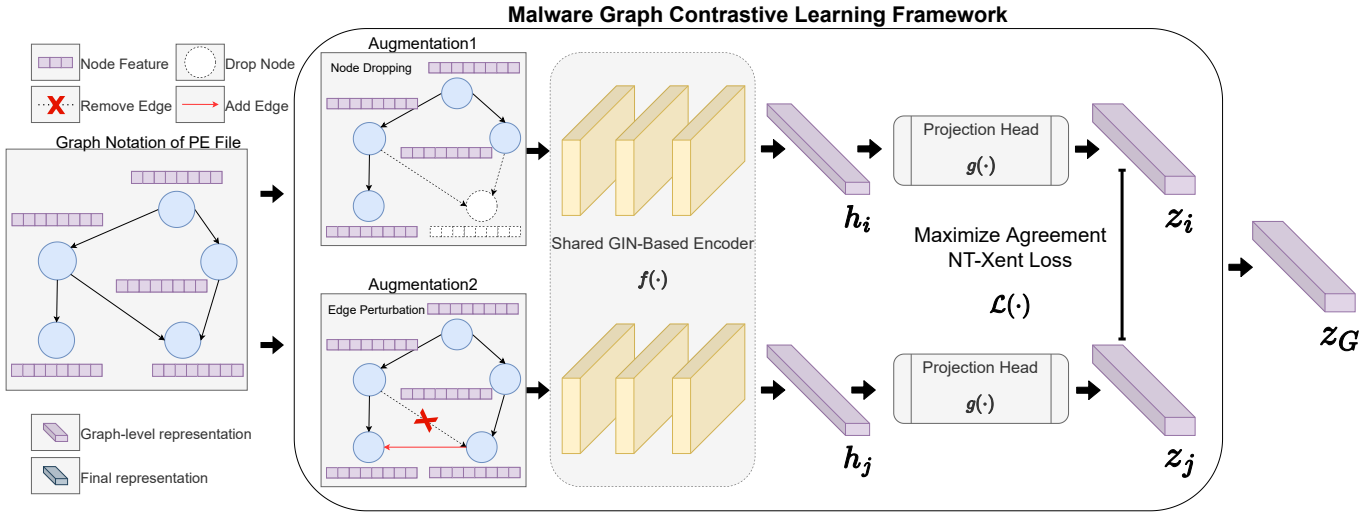


Figure 2. Proposed Malware Graph Contrastive Learning Framework for Graph Representation Generation

neighbors. Nodes connected to other central nodes have higher eigenvector centrality themselves.

c) PageRank: Originally developed by Google, PageRank is a variant of eigenvector centrality and is used to rank web pages in search engine results. It assigns a score to each node based on the number and quality of incoming links.

d) Local Degree Profile (LDP): LDP [27] is a simple representation scheme, each node feature summarizes 5 degree statistics information of this node and its 1-hop neighborhood. We selected LDP method as graph centrality measures.

Different centrality measures are applicable in various contexts and provide insights into different aspects of a network's structure and dynamics. Researchers and analysts choose centrality measures based on the specific questions they want to answer and the characteristics of the network under study.

IV. Proposed Data Augmented GraphCL-Based Static Malware PE Classification

Our proposal is a data augmented GraphCL-based static malware PE classification framework, which can obtain a graph-level representation from malware. We directly extract malware CFG from PE files and through graph contrastive learning obtain a representation of the malware with a vector notation. Finally, malware representations can be performed downstream for various tasks. Graph-level representation shows good performance on malware classification tasks. Next we scrutinize the framework.

A. Data Augmentation for Graphs

We used the following four data augmentation methods. As shown in Figure 2, our proposal uses two of them as a combination.

1) Node Dropping: Randomly discard some parts of the vertex and its connections. The underlying prior enforced by it is that missing part of vertices does not affect the semantic meaning of the graph.The dropping probability of each node follows a default Bernoulli uniform distribution (or any other distribution).

2) Edge Perturbation: Randomly add or remove a certain ratio of edges so that the learned representation is consistent under edge perturbation. The prior information of the representation is that adding or removing some edges does not affect the semantics of the graph. The dropping probability of each node follows a default Bernoulli uniform distribution. We only used Edge Removing in this evaluation.

3) Attribute Masking: Randomly removing the attribute information of some nodes motivates the model to use other information to reconstruct the masked node attributes. The masking probability of each node feature dimension follows a default uniform distribution. We only used simple Feature Masking.

4) Subgraph Sampling: Use random walk subgraph sampling [28] to extract subgraphs from the original graph. The basic assumption is that a graph's semantic information can be preserved in its local structure.

Table I overviews the data augmentation for graphs. The default augmentation (dropping, perturbation, masking) ratio is set to 0.1, and the walk length is set to 10.

TABLE I. OVERVIEW OF DATA AUGMENTATION FOR GRAPHS

Data Augmentation	Type	Default Setting
Node Dropping	Nodes, Edges	Bernoulli distribution (ratio = 0.1)
Edge Perturbation	Edges	Bernoulli distribution (ratio = 0.1)
Attribute Masking	Nodes	Uniform distribution (ratio = 0.1)
Subgraph Sampling	Nodes, Edges	Random Walk (length = 10)

B. Graph Contrastive Learning

Motivated by recent developments in graph contrastive learning, we propose a graph contrastive learning framework for malware classification. As shown in Figure 2, in graph contrastive learning, pre-training is performed by maximizing the agreement between two augmented views of the same graph by contrastive loss in the potential space. The framework consists of the following four main components:

1) Graph Data Augmentation: Throughout the GraphCL framework, given graph data G, two related augmented graphs, $\hat{G}_i, \hat{G}_j$, are generated as positive sample pairs by data augmentation.

2) GIN-based Encoder: GIN-based encoder $f(\cdot)$ is used to generate graph-level vector representation. There are three layers in the GIN-based encoder, and the hidden layer has 64 dimensions. Through the readout function, the embedding of all the nodes is summed to obtain initial graph representation h_i, h_j for augmented graphs $\hat{G}_i, \hat{G}_j$. Graph contrastive learning does not apply any constraint to the GIN-based encoder.

3) Projection Head: Nonlinear transformation $g(\cdot)$, called a projection head, maps the augmented representations to another latent space. Contrast loss is computed in the latent space, and z_i, z_j are obtained by applying a two-layer perceptron (MLP).

4) Contrastive Loss Function: Contrastive loss function $\mathcal{L}(\cdot)$ is defined to enforce the maximum consistency between positive pairs z_i, z_j and negative pairs. Here we exploit the normalized temperature-scale cross-entropy loss (NT-Xent) [29] [30] and obtain a graph-level final representation of z_G.

C. Graph Classification

By pre-training with GraphCL, we can obtain a valid graph representation z_G. To further verify the effectiveness of our method, different classification models can be chosen for the process, such as random forest, logistic regression, SVM, etc. We chose C-Support Vector Classification (SVC) as the algorithm to validate our pre-trained model's effectiveness.

V. Implementation Details

We verified the effectiveness of our proposed contrastive learning framework by implementing it with open-source libraries. The implementation details are introduced in this section.

A. Malware Graph Datasets

The publicly available dataset of malicious software graphs is extremely scarce and does not directly provide the raw binary files. In order to extract CFG, we have constructed our own dataset. Additionally, some publicly available malicious software graph datasets, such as MALNET dataset [31], do not include node features in their graph data themselves, requiring us to generate them based on our needs.

1) MGD-MULTI Dataset:

a) PE Files Source: Our PE file sample was obtained from the BODMAS Malware Dataset [32]. The software types of all the PE file samples used in our dataset are executable files under an x86-architecture Windows platform without any Dynamic Link Library (DLL) type.

b) Dataset Description: From the BODMAS dataset, we selected eight families of malware and took 500 samples from each family, for a total of 4000 samples in our dataset. Our dataset is named MGD-MULTI. The malware family distribution information is shown in Table II.

TABLE II. MALWARE FAMILY DISTRIBUTION OF MGD-MULTI

Family	Category	Origin Count	Selected Count
sfone	worm	4729	500
upatre	trojan	3901	500
wabot	backdoor	3673	500
benjamin	worm	1071	500
musecador	trojan	1054	500
padodor	backdoor	655	500
gandcrab	ransomware	617	500
dinwod	dropper	509	500
Total	-	16209	4000

Due to the difficulty of collecting benign samples and the imbalanced data problem, we did not include benign samples in our multi-class dataset. In our previous malware detection work [33], the MGD-BINARY dataset contained benign samples. We used almost the same GIN model to represent the PE samples, with a slightly different operation of the READOUT layer this time compared to the GIN model in our previous work, giving the final representation a higher vector dimensionality. Based on our previous research, we believe that the GIN model can effectively distinguish benign samples from malicious ones.

Among the different types of malware, we chose families that are more common and have a relatively large number in BODMAS. Due to some limitations of the CFG extraction tool for the PE files we used, many samples couldn't be recognized, causing extraction failure. In addition, for large PE file samples, the process of extracting CFG is very time-consuming. Since the extraction of some samples will fail, we selected a family with more than 500 samples in BODMAS and relatively small original PE files. We further improved the efficiency by only selecting successful samples whose total extraction time is less than 20 seconds in which the total extraction time includes the time of the feature vectors generated by the pre-trained language model.

c) Dataset Splitting: We split 4000 pieces of data in MGD-MULTI into training, validation, and testing sets of 50%, 20%, and 30%, respectively. Since the results of the validation set and the test are similar, only the testing set results are shown.

2) MALNET-TINY Dataset:

a) APK Files Source: The dataset's authentic APK files were acquired from the renowned AndroZoo repository [46, 44]. AndroZoo is a growing collection of Android Applications collected from several sources, including the official Google Play app market.

b) Dataset Description: MalNetTiny contains 5,000 malicious and benign software FCGs across 5 different types. Each graph contains at most 5k nodes.

c) Dataset Splitting: The dataset is stratified and divided into three sets: training, validation, and test, with a split ratio of 70/10/20, respectively. To thoroughly assess both the model's performance and the dataset's quality, we conduct 10-fold cross-validation on MALNET-TINY, which is distinct from the MGD-MULTI dataset.

B. Graph Node Feature Generation

For our self-constructed dataset MGD, it is essential to transform the disassembly code within CFG basic blocks into embedded vectors. To achieve this, we opted to utilize a pre-trained language model to generate node embedding vectors. However, for the MALNET-TINY dataset, since the graph data itself does not include node features, we need to extract them directly from the graphs. Therefore, we selected the Local Degree Profile (LDP) method to generate node feature vectors. Next, we will provide a detailed explanation of these two approaches.

1) Pre-trained Language Model MiniLM: SentenceTransformers is a python framework for state-of-the-art sentence, text, and image embeddings. The initial work was described in a paper from the Sentence-Bidirectional Encoder Representations from Transformers (Sentence-BERT) [34]. We used the MiniLM model provided by the SentenceTransformers library with the model name, all-MiniLM-L6-v2. The model details used in this paper are shown in Table III.

TABLE III. PRE-TRAINED MINILM MODEL DETAILS

Name	all-MiniLM-L12-v2
Base Model	microsoft/MiniLM-L12-H384-uncased
Max Sequence Length	256
Dimensions	384
Normalized Embeddings	true
Size	120 MB
Pooling	Mean Pooling
Training Data	1B+ training pairs

2) LDP: PyTorch Geometric (PyG) [35] is a popular Python library built on top of PyTorch, specifically designed for deep learning on graphs and other irregular data structures. PyG provides a wide range of tools and utilities to simplify the implementation of graph neural networks (GNNs) and other graph-based machine learning models. We use the graph transforms to generate the LDP node features.

C. Graph Contrastive Learning

PyGCL [36] is a PyTorch-based open-source Graph Contrastive Learning library, which features modularized GraphCL components from published papers, standardized evaluation, and experiment management. The Data dataset statistics and hyper-parameters are shown in Table IV.

VI. Evaluation and Discussion

In this section, we apply the GraphCL model and discuss the experiment results and limitations of our method.

A. Evaluation Metric

We used the following evaluation metrics to assess the performance of our proposed models:

- **The Micro-averaged F1 score** is defined as the harmonic mean of the precision and recall:

$$Micro\ F1\text{-}score = 2 \times \frac{\text{Micro-Precision} \times \text{Micro-Recall}}{\text{Micro-Precision} + \text{Micro-Recall}}$$

TABLE IV. DATASET STATISTICS AND HYPER-PARAMETERS

Dataset	MGD-MULTI	MALNET-TINY
# Graphs	4000	5000
# Avg. Nodes	3861.7	1410.3
# Avg. Edges	5494.8	7125.7
# Features	384	5
# Class	8	5
Learning Rate	0.0001	0.0001
Batch Size	128	256
Epoch	100	100
Hidden Size	64	64
Hidden Layers	3	3
Global Pooling	Sum Pooling	Max Pooling

- **The Macro-averaged F1 score** is defined as the average of the class-wise/label-wise F1-scores:

$$Macro\ F1\text{-}score = \frac{1}{N}\sum_{i=0}^{N} F1\text{-}score_i$$

where *i* is the class/label index and *N* is the number of classes/labels.

B. Evaluation Results

Next we apply the GraphCL model and discuss the experiment results of our method. We selected five different data augmentation methods: Identical (*I*), Edge Removing (*ER*), Node Dropping (*ND*), Feature Masking (*FM*), and Random Walk Subgraph (*RWS*).

To compare the different data augmentation approaches on the GraphCL model, we used both data augmentation approaches for the input graph itself *I* + *I* as the GraphCL model baseline. We also tried different combinations of data augmentation, such as *ER* and *ND*, *FM* and *ND*, *FM* and *ER*, *RWS* and *ER*, *RWS* and *ND*, and *RWS* and *FM*. The experimental results on two datasets are shown in Table V.

1) MGD-MULTI Classification Results: The best two data augmentation combinations on MGD-MULTI dataset were *RWS* and *FM*. When the walk length of *RWS* is 10 and the ratio of *FM* is 0.1, we obtained the best Micro-F1 (0.9958) and Macro-F1 (0.9959).

In the previous set of experiments, we found that the best data augmentation combination is *RWS* + *FM*. Based on this combination, we also investigated the results on different ratios on the *FM* side, and the *FM* results on different ratios are shown in Table VI. We set the walk length of *RWS* to 10 and adjusted the ratio of the *FM*. When the ratio of *FM* is less than 0.3, the results gradually improve, and as it exceeds 0.3, the results gradually worsen. The optimal result is achieved when the *FM* is set to 0.3.

The *RWS* + *FM* method is most effective because neither method changes the structural information of the original graph. The *RWS* method samples a subgraph that is smaller than the structure of the original graph, but still retains most of the original graph's structure. For the *FM* method, the original graph structure is not changed at all, but the values of some dimensions of the node feature vectors are masked, which

TABLE V. EXPERIMENTAL RESULTS OF DIFFERENT DATA AUGMENTATION

Augmentation 1	Augmentation 2	MGD-MULTI		MALNET-TINY	
		Micro-F1	Macro-F1	Micro-F1	Macro-F1
I	I	0.9883	0.9883	0.8758 ± 0.0148	0.8753 ± 0.0149
ER[1](0.1)	ND[1](0.1)	0.9925	0.9924	**0.8652 ± 0.0129**	**0.8641 ± 0.0131**
FM[1](0.1)	ND (0.1)	0.9942	0.9942	0.8536 ± 0.0150	0.8520 ± 0.0156
FM (0.1)	ER (0.1)	0.9942	0.9942	0.8394 ± 0.0108	0.8384 ± 0.0111
RWS [2]	ER (0.1)	0.9950	0.9949	0.7766 ± 0.0152	0.7695 ± 0.0155
RWS	ND (0.1)	0.9950	0.9949	0.7834 ± 0.0139	0.7771 ± 0.0152
RWS	FM (0.1)	**0.9958**	**0.9959**	0.7760 ± 0.0212	0.7715 ± 0.0218

[1] Default ratio setting is 0.1.
[2] RWS uses a default walk length setting of 10.

makes the node features more robust. On the contrary, the other two methods (*ER* and *ND*) change the original graph structure more, so the results are lowered.

TABLE VI. BEST COMBINATION WITH DIFFERENT RATIO RESULTS

Augmentation 1	Augmentation 2	MGD-MULTI	
		Micro-F1	Macro-F1
RWS [1]	FM (0.1)	0.9958	0.9959
RWS	FM (0.2)	0.9967	0.9967
RWS	FM (0.3)	**0.9975**	**0.9976**
RWS	FM (0.4)	0.9958	0.9958
RWS	FM (0.5)	0.9942	0.9941

[1] RWS uses a default walk length setting of 10.

2) MALNET-TINY Classification Results: The best two data augmentation combinations on MALNET-TINY dataset were I and I. On the MGD-MULTI dataset, we did not attempt to explore more combinations with the same settings or different ratios of combinations. To further validate different scenarios, we applied more identical combinations, such as *ND + ND*, *ER + ER*, *RWS + RWS* and *FM + FM* on the new dataset MALNET-TINY. Additionally, concerning different ratios of combinations, we experimented with values ranging from 0.1 to 1.0. For the *RWS*, we tested walk length from 10 to 50,000. The optimal results are shown in Table VII. The best results were achieved when the ratio of all augmentation combinations was set to 0.1. The optimal *RWS* walk length was found to be 20000.

For the same augmentation combinations, the best results were obtained with *I + I*. *ND (0.1) + ND (0.1)* and *ER (0.1) + ER (0.1)* also showed good performance, but *FM (0.1) + FM (0.1)* performed poorly. We believe that the reason behind the poor performance of *FM (0.1) + FM (0.1)* is that the MALNET-TINY dataset has a feature dimension of only 5. Applying the *FM* augmentation can lead to the loss of already limited node feature information, making it difficult for the model to effectively distinguish node features and resulting in poor results. In contrast, *ND* and *ER* operations primarily focus on adjusting the graph structure, which is why their results are more favorable.

It seems that among the different data augmentation combinations, the combination of *ER (0.1) + ND (0.1)* yielded the best performance. This result indicates that using *ER (0.1) + ND (0.1)* in combination provided the most effective data augmentation strategy for improving model performance on the MALNET-TINY dataset. It appears that regardless of using the same data augmentation or different data augmentation strategies, the performance of the *RWS* augmentation is not satisfactory.

This could be due to the specific feature and properties of the MALNET-TINY dataset, which might be better suited for the specific augmentation combination rather than other combinations that involve more complex operations like *RWS*. It's essential to consider the nature of the dataset and the impact of different augmentation techniques on the model's ability to learn meaningful patterns and features from the data. Depending on the dataset's characteristics, some augmentation combinations may be more effective than others in improving model performance. Further experimentation and analysis can help to better understand the relationship between data augmentation and model performance on the specific dataset.

TABLE VII. BEST COMBINATION WITH DIFFERENT RATIO RESULTS

Augmentation 1	Augmentation 2	MALNET-TINY	
		Micro-F1	Macro-F1
I	I	0.8758 ± 0.0148	0.8753 ± 0.0149
ND (0.1)	ND (0.1)	**0.8610 ± 0.0173**	**0.8595 ± 0.0178**
ER (0.1)	ER (0.1)	0.8568 ± 0.0110	0.8554 ± 0.0110
RWS[1]	RWS	0.8504 ± 0.0081	0.8476 ± 0.0084
FM (0.1)	FM (0.1)	0.8178 ± 0.0153	0.8138 ± 0.0158
I	ER (0.1)	0.8578 ± 0.0112	0.8565 ± 0.0118
I	ND (0.1)	0.8536 ± 0.0156	0.8520 ± 0.0158
I	RWS	0.8480 ± 0.0116	0.8450 ± 0.0121
I	FM (0.1)	0.8346 ± 0.0118	0.8327 ± 0.0127
ER (0.1)	ND (0.1)	**0.8652 ± 0.0129**	**0.8641 ± 0.0131**
FM (0.1)	ND (0.1)	0.8536 ± 0.0150	0.8520 ± 0.0156
FM (0.1)	ER (0.1)	0.8394 ± 0.0108	0.8384 ± 0.0111
RWS	ER (0.1)	0.8466 ± 0.0110	0.8441 ± 0.0113
RWS	ND (0.1)	0.8350 ± 0.0093	0.8322 ± 0.0091
RWS	FM (0.1)	0.8304 ± 0.0155	0.8262 ± 0.0161

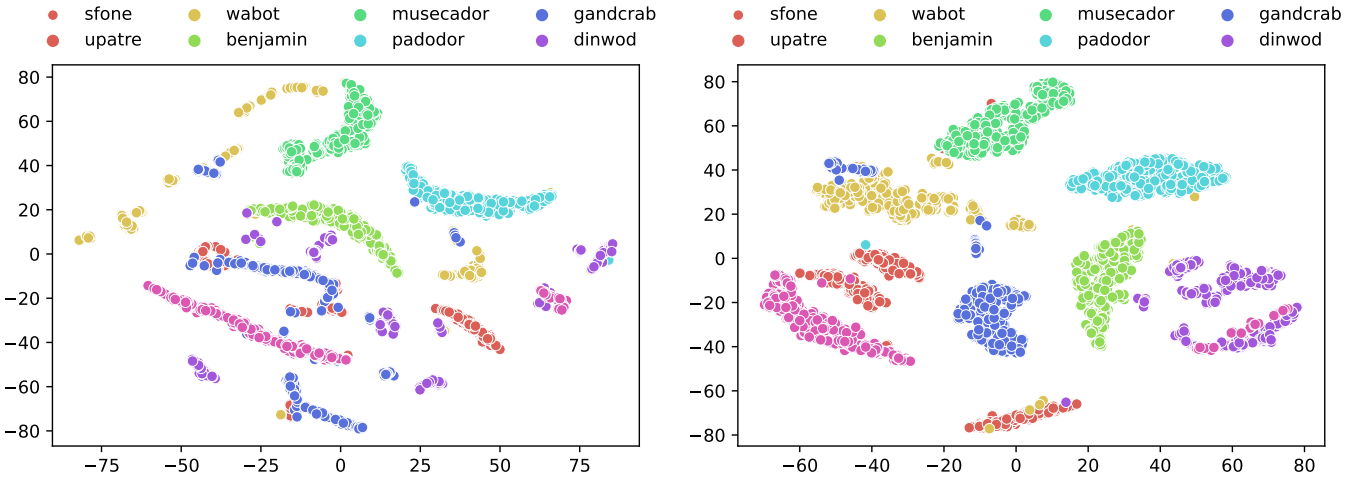
[1] RWS uses the best walk length setting of 2000.

3) Comparison of Results from Different Datasets: As shown in Table IV, the graph data of the two datasets have different statistical characteristics. Compared with MGD-MULTI dataset, MALNET-TINY dataset has fewer average nodes and more average edges. The graph samples of the two datasets are quite different. When the node feature vector dimension is

TABLE VIII. EXPERIMENTAL RESULTS OF DIFFERENT LEARNING METHODS

Name	Method	Type	Classifier	MGD-MULTI		MALNET-TINY	
				Micro-F1	Macro-F1	Micro-F1	Macro-F1
Baseline 1	GIN-Encoder	-	SVC	0.9617	0.9620	0.7060 ± 0.0156	0.7012 ± 0.0149
Baseline 2	GIN (Previous work [33])	SL[1]	MLP	0.9958	0.9957	0.8080	0.8126
Proposal	GraphCL	SSL[2]	SVC	0.9975	0.9976	**0.8758 ± 0.0148**	**0.8753 ± 0.0149**

[1] SL denotes supervised learning.
[2] SSL denotes self-supervised learning.



(a) Baseline 1 on MGD-MULTI dataset: GIN-Encoder

(b) Proposal on MGD-MULTI dataset: GraphCL (*RWS+FM_0.3*)

Figure 3. t-SNE projection of Baseline 1 and Proposal on MGD-MULTI Dataset

high and the number of edges is relatively small, *FM* becomes more effective. Compared with *RWS*, the more average edges of dataset, the more effective *ER* is.

4) Comparison of Different Learning Methods: Our previous studies focused on supervised learning. This study is a graph contrastive learning method in an self-supervised setting. Baseline 1 involves using GIN as the encoder to directly perform graph-level encoding on the input graphs, followed by evaluating the embedding effectiveness using SVC.

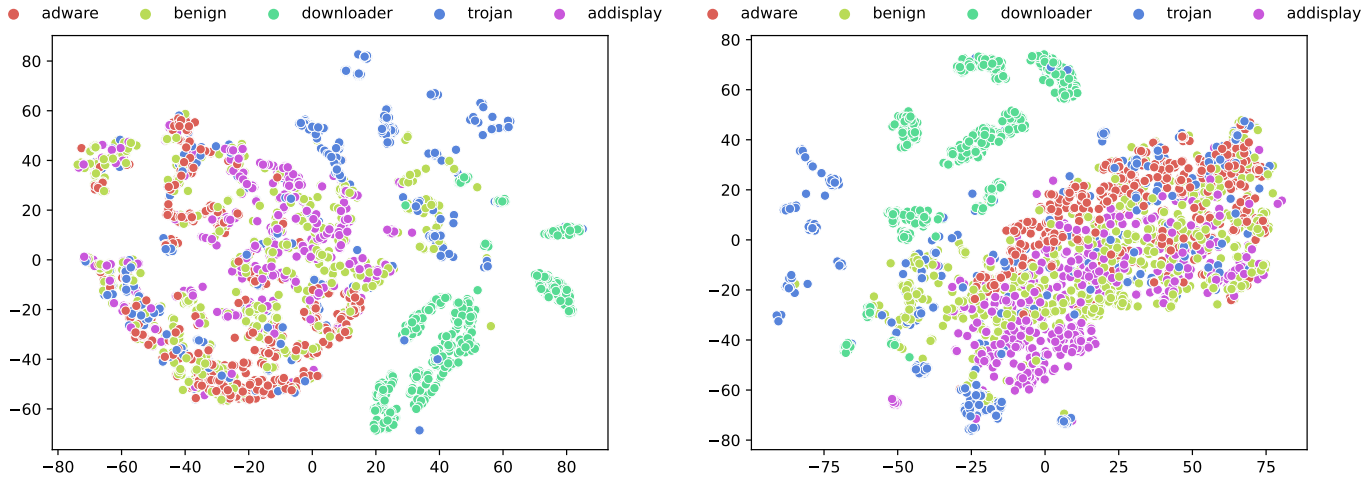
Baseline 2 is our previous work [33] on the malware graph classification task. We utilized the Graph Isomorphism Network for training in a self-supervised setting and directly connected two Multi-Layer Perceptron layers after the GIN readout layer for classification. The GIN model consists of three layers with a hidden layer size of 64, and the MLP has two layers with a hidden layer size of 128. A comparison of different learning type methods is shown in Table VIII. It is worth noting that on the MALNET-TINY dataset, Baseline 2 did not use 10-fold cross-validation, and we only reported the results on the testing set.

On the MGD-MULTI dataset, GraphCL with a setting of *RWS + FM (0.3)* achieved the best classification results. On the MALNET-TINY dataset, GraphCL with a setting of *I + I* achieved the best classification results. The evaluation results indicate that our approach achieved Micro-F1 scores of 0.9975 and Macro-F1 scores of 0.9976 on MGD-MULTI dataset. Similarly, on MALNET-TINY dataset, we obtained Micro-F1 scores of 0.8758 ± 0.0148 and Macro-F1 scores of 0.8753 ± 0.0149. According to our experimental evaluation, the self-supervised learning approach outperformed the supervised learning approach in Graph Neural Networks based on malware classification. We can see that self-supervised learning has great potential.

We employed t-SNE technique to visualize the embeddings of Baseline 1 and our proposed method separately on two datasets: MGD-MULTI and MALNET-TINY. For the MGD-MULTI dataset, as shown in Figure 3, the method of Baseline 1 has already clustered some categories, such as the malware of the “padodor” family, but it cannot cluster the “gandcrab” family well. On the other hand, our contrastive learning model proposal can better cluster different categories in the eight classes, and a large distance between different categories is maintained. For the MALNET-TINY dataset, as shown in Figure 4, Compared to the MGD dataset, the visualization results of the TINY dataset are not satisfactory. We found that among the four classes of malicious software, the performance for the “trojan” type is the worst, while the “downloader” type has the best results. For “benign” samples, it is challenging to distinguish them from “adware” and “addisplay”.

C. Current Limitations

In this study, the two datasets used were relatively small. We achieved promising results on the MGD-MULTI dataset. However, the performance on the MALNET-TINY dataset was not very satisfactory.



(a) Baseline 1 on MALNET-TINY dataset: GIN-Encoder

(b) Proposal on MALNET-TINY dataset: GraphCL ($I + I$)

Figure 4. t-SNE projection of Baseline 1 and Proposal on MALNET-TINY Dataset

1) MGD-MULTI dataset: In the feature engineering stage, there are two steps that are very time-consuming. Firstly, extracting the CFG of the PE file is very time-consuming. Secondly, during the construction of training data, it is also very time-consuming because it requires embedding representation of all nodes' features in the graphs using a pre-trained model. During the training phase, due to the large graph data structures and the high dimensionality of each node in the graph (384 dimensions), the GraphCL model training is slow, even though the dataset itself contains only 4000 samples. We desire a better way to generate node features, such as a lower dimensional in a method that retains its effectiveness.

2) MALNET-TINY dataset: GraphCL ($I + I$) is a combination of two Identical, and the effect is equivalent to turning a training set of N samples into 2N samples. The same data model is learned twice for the same data, so the obtained result naturally outperforms GIN-Encoder. Compared to other data augmentation combinations, the combination of $I + I$ yields the best results, and a reasonable explanation for this has not been found yet.

3) Graph Self-supervised Learning: GraphCL is representative of contrastive methods, but it overly relies on high-quality data augmentation. We explored the effects of various data augmentation methods, such as *FM*, *RWS*, and *ER*. We found that effective data augmentation on graphs often depends on domain knowledge. For instance, *ER* is beneficial for training on the MALNET-TINY dataset, but it has a negative impact on the MGD-MULTI dataset. However, in the context of graph contrastive learning, there is still no universally effective data augmentation method up to now. Generative self-supervised learning can avoid the aforementioned dependencies, as it aims to reconstruct the features and information of the data itself, such as GraphMAE [21] and WGDN [22] models.

VII. Conclusion

We propose using graph contrastive learning for unsupervised learning of different families of malicious software. We employ SVC for multi-class classification of the graph representations and achieve promising results. For our self-built dataset MGD-MULTI, we extract the CFG of malicious software and embed the disassembly code in CFG basic blocks using a pre-trained large-scale language model, MiniLM. For the publicly available MALNET-TINY dataset, as it does not provide node features for the graph data based on FCG, we generate node representations using LDP. Through these two methods, both CFG-based and FCG-based malicious software are transformed into directed graphs with node features. Using graphs to represent malicious software offers significant advantages, as each node carries rich information and preserves the structural details of the samples. Graph models can learn from both global and local perspectives. Furthermore, self-supervised learning eliminates the need for sample annotations, learning from self-supervised signals, making it more efficient to leverage a larger number of samples for pre-training. On our self-built dataset, the graph-based self-supervised method for malicious software classification has outperformed supervised graph learning methods, such as Graph Isomorphism Networks used for graph classification. In future work, our focus will shift towards self-supervised learning.

REFERENCES

[1] Y. Gao, H. Hasegawa, Y. Yamaguchi, and H. Shimada, "Unsupervised graph contrastive learning with data augmentation for malware classification," in *Proceedings of the 16th International Conference on Emerging Security Information, Systems and Technologies, SECURWARE 2022*, pp. 41–47, IARIA, 2022.

[2] H. S. Anderson and P. Roth, "EMBER: an open dataset for training static PE malware machine learning models," *CoRR*, vol. abs/1804.04637, pp. 1–8, 2018.

[3] B. Athiwaratkun and J. W. Stokes, "Malware classification with LSTM and GRU language models and a character-level CNN," *ICASSP 2017*, pp. 2482–2486, 2017.

[4] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu, "Large-scale malware classification using random projections and neural networks," *ICASSP 2013*, pp. 3422–3426, 2013.

[5] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, "Malware classification with recurrent networks," *ICASSP 2015*, pp. 1916–1920, 2015.

[6] F. Cohen, "Computer Viruses: Theory and Experiments," *Computers & security, Vol. 6, No. 1*, pp. 22–35, 1987.

[7] J. O. Kephart, G. B. Sorkin, W. C. Arnold, D. M. Chess, G. Tesauro, and S. R. White, "Biologically Inspired Defenses Against Computer Viruses," *IJCAI 1995, Vol. 2*, pp. 985–996, 1995.

[8] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," *MALWARE 2015*, pp. 11–20, 2015.

[9] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. K. Nicholas, "Malware Detection by Eating a Whole EXE," *AAAI Workshops 2018*, pp. 268–276, 2018.

[10] E. Raff, J. Sylvester, and C. Nicholas, "Learning the PE Header, Malware Detection with Minimal Domain Knowledge," *AISec@CCS 2017*, pp. 121–132, 2017.

[11] Y. Chen, S. Wang, D. She, and S. Jana, "On Training Robust PDF Malware Classifiers," *USENIX Security 2020*, pp. 2343–2360, 2020.

[12] S. E. Coull and C. Gardner, "Activation Analysis of a Byte-Based Deep Neural Network for Malware Classification," *SP Workshops 2019*, pp. 21–27, 2019.

[13] E. M. Rudd, F. N. Ducau, C. Wild, K. Berlin, and R. E. Harang, "ALOHA: Auxiliary Loss Optimization for Hypothesis Augmentation," *USENIX Security 2019*, pp. 303–320, 2019.

[14] L. Yang, W. Guo, Q. Hao, A. Ciptadi, A. Ahmadzadeh, X. Xing, and G. Wang, "CADE: Detecting and Explaining Concept Drift Samples for Security Applications," *USENIX Security 2021*, pp. 2327–2344, 2021.

[15] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic Graph CNN for Learning on Point Clouds," *ACM Transactions on Graphics (TOG), Vol. 38, Issue 5*, pp. 1–12, 2019.

[16] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*, pp. 1–17, 2019.

[17] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "graph2vec: Learning Distributed Representations of Graphs," *CoRR*, vol. abs/1707.05005, pp. 1–8, 2017.

[18] F. Sun, J. Hoffmann, V. Verma, and J. Tang, "Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," in *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*, pp. 1–16, 2020.

[19] M. Dib, S. Torabi, E. Bou-Harb, N. Bouguila, and C. Assi, "Evoliot: A self-supervised contrastive learning framework for detecting and characterizing evolving IoT malware variants," in *Proceedings of ASIA CCS '22: ACM Asia Conference on Computer and Communications Security*, pp. 452–466, 2022.

[20] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

[21] Z. Hou, X. Liu, Y. Cen, Y. Dong, H. Yang, C. Wang, and J. Tang, "Graphmae: Self-supervised masked graph autoencoders," in *Proceedings of KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 594–604, 2022.

[22] J. Cheng, M. Li, J. Li, and F. Tsung, "Wiener graph deconvolutional network improves graph self-supervised learning," in *Proceedings of the 37th AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023*, pp. 7131–7139, AAAI Press, 2023.

[23] I. You and K. Yim, "Malware obfuscation techniques: A brief survey," in *Proceedings of the 15th International Conference on Broadband and Wireless Computing, Communication and Applications, BWCCA*, pp. 297–300, IEEE Computer Society, 2010.

[24] J. Jang, S. Choi, and J. Hong, "A method for resilient graph-based comparison of executable objects," in *Research in Applied Computation Symposium, RACS '12*, pp. 288–289, ACM, 2012.

[25] V. S. Sathyanarayan, P. Kohli, and B. Bruhadeshwar, "Signature generation and detection of malware families," in *Proceedings of 13th Australasian Conference on Information Security and Privacy, ACISP*, vol. 5107 of *Lecture Notes in Computer Science*, pp. 336–349, Springer, 2008.

[26] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers," *CoRR*, vol. abs/2002.10957, pp. 1–15, 2020.

[27] C. Cai and Y. Wang, "A simple yet effective baseline for non-attribute graph classification," *CoRR*, vol. abs/1811.03508, pp. 1–13, 2018.

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

[29] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *CoRR*, vol. abs/1807.03748, pp. 1–13, 2018.

[30] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *CoRR*, vol. abs/2010.13902, pp. 1–12, 2020.

[31] S. Freitas, Y. Dong, J. Neil, and D. H. Chau, "A large-scale database for graph representation learning," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021*, 2021.

[32] L. Yang, A. Ciptadi, I. Laziuk, A. Ahmadzadeh, and G. Wang, "BODMAS: An Open Dataset for Learning based Temporal Analysis of PE Malware," *DLS 2021*, pp. 78–84, 2021.

[33] Y. Gao, H. Hasegawa, Y. Yamaguchi, and H. Shimada, "Malware detection using attributed cfg generated by pre-trained language model with graph isomorphism network," in *Proceedings of the 12th IEEE International Workshop on Network Technologies for Security, Administration and Protection (NETSAP 2022)*, pp. 1495–1501, 2022.

[34] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," *CoRR*, vol. abs/1908.10084, pp. 1–11, 2019.

[35] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *Proceedings of the ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[36] Y. Zhu, Y. Xu, Q. Liu, and S. Wu, "An empirical study of graph contrastive learning," *CoRR*, vol. abs/2109.01116, pp. 1–25, 2021.