Exploring the Impact of LLM Assisted Malware Variants on Anti-Virus Detection

Zhewei Huang*, Yin Minn Pa Pa*, Katsunari Yoshioka*
*Yokohama National University, Japan

***Abstract*—In this study, we develop 400 malware variants of four different types of test malware samples by rewriting their original source code with OpenAI's gpt-3.5-turbo model [1], and investigate if these samples can be detected by anti-virus. From this investigation, we find that about half of the malware variant function correctly and 37.6±33.6 % of them evade anti-virus detection for each malware type and anti-virus.**

***Index Terms*—Malware, Variant, Anti-Virus, ChatGPT**

I. Introduction

With the rapid social implementation of large language models (LLMs) due to recent advances in computing and deep learning technologies, the exploitation of LLMs has become a new threat in cybersecurity. It is known that malware source code can actually be generated by LLMs [2]. This study focuses on the impact of malware variant development by LLMs on anti-virus detection. In this study, we develop 400 malware variants of four different types of test malware samples by rewriting their original source code with OpenAI's gpt-3.5-turbo model [1]. Out of these, 211 can be compiled into executable files and retain the same functionality as the original test samples. We investigate six anti-virus software using 160 variants from these samples, and find that 37.6±33.6 % of the samples evade detection by the anti-virus for each malware type and anti-virus.

II. Methodology and Results

A. Creating Malware Variants Using ChatGPT

We utilize the gpt-3.5-turbo model via the OpenAI API [3] to generate malware variants. Starting with four original test samples—worm, keylogger, ransomware, and fileless malware—targeting Windows, we use the LLM to create these in different languages: Python for the worm, Go for the ransomware and keylogger, and C++ for the fileless malware. Each original sample is then used to generate 100 variants, with prompts instructing the model to maintain the original functionality. The worm variants are converted to executables using PyInstaller [4], while the other malware types are compiled. We manually verify if the generated executables retain the original functionality and evaluate the similarity of the source code and binaries between the variants and the original samples.

Table I presents the number of variant samples that retain the same functionality as the original samples. For the source codes that cannot be converted into executable files, the issues encountered include syntactical errors, incomplete code generation, the use of libraries not installed in the development environment, and modifications to hard-coded constants. Table II shows the BLEU scores [5] of each variant sample compared to the original sample, the increase or decrease in the number of tokens matching the regular expression "*\b[a-zA-Z0-9_]+\b*", as well as the similarity scores calculated using ssdeep [6] and TLSH [7]. Fileless malware samples are similar at the textual level but show significant differences in their binaries. In contrast, Worm, Keylogger, and Ransomware samples exhibit similarities in both their source code and binaries. These differences are likely due to factors such as the programming language used, the implemented logic, and the compiler.

TABLE I
ChatGPT generated samples that retain the same functionality as the original

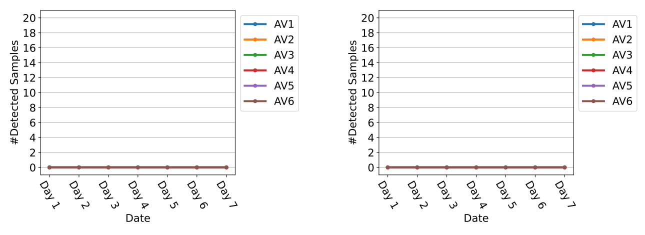
Type	#Samples
Worm	58 / 100
Ransomware	51 / 100
Keylogger	46 / 100
Fileless	56 / 100

TABLE II
Similarity to the original sample [mean±std.]

Type	Text		Binary	
	BLEU	Token	ssdeep	TLSH
Worm	0.83 ± 0.08	-2.7 ± 3.3	96 ± 0.0	1.5 ± 0.95
Keylogger	0.96 ± 0.06	-2.6 ± 5.2	4.3 ± 16	8.1 ± 4.2
Ransomware	0.87 ± 0.05	-1.8 ± 3.9	11 ± 23	12 ± 5.1
Fileless	0.59 ± 0.09	-19 ± 9.6	0.0 ± 0.0	93 ± 55

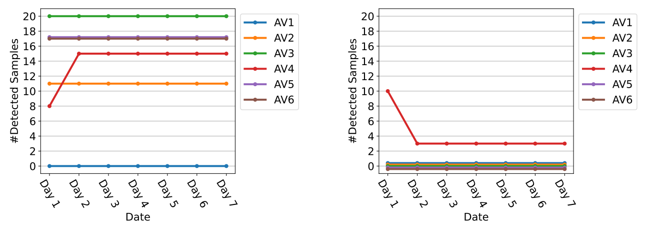
B. Investigating on Anti-Virus detection

The variants are divided into two groups, each containing 80 samples, with 20 samples per malware type across the four different malware types. This division allows for an investigation into anti-virus detection of both unknown and known test samples. The analysis involves executing samples in a local environment and uploading them to VirusTotal [8]. Six anti-virus software, chosen based on popularity and market share, are tested in a Windows 11 Enterprise Evaluation virtual environment connected to the Internet for updates. Four of these products include both file scan and behavior monitoring, while two have only file scan capabilities. The specific software names are withheld for ethical reasons. Test samples are introduced via a shared folder. The analysis proceeds as follows.



(a) detection by file scan (b) detection by behavior

Fig. 1. Keylogger in Group1 detection result in procedure 1)



(a) detection by file scan (b) detection by behavior

Fig. 2. Keylogger in Group1 detection result in procedure 3)

S.1) **Local:** Group 1 samples are scanned and executed daily in a Windows virtual environment with anti-virus software installed. The anti-virus response is observed over 7 days when an unknown sample appears on one anti-virus product.

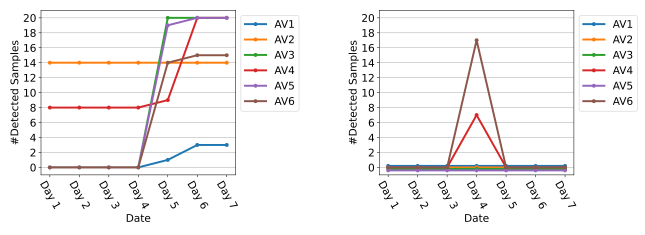
S.2) **VT:** Group 1 samples are uploaded to VirusTotal, and the number of anti-virus engines detecting them is recorded, aiming to increase endpoint detections.

S.3) **Local:** Group 1 samples are scanned and executed again 92 days later, with anti-virus responses observed over 7 days to monitor any increase in endpoint detections.

S.4) **Local:** Group 2 samples are scanned and executed, with anti-virus responses observed over 7 days.

S.5) **VT:** Group 2 samples are uploaded to VirusTotal, and the number of anti-virus engines detecting them is recorded.

Steps 2), 3), 4), and 5) are scheduled strategically to assess the impact of VirusTotal uploads and anti-virus database updates on detection rates, with observations continuing for 7 days. The results vary depending on the type of malware sample. For the worm, five anti-virus fail to detect the worm samples in Group 2. For the keylogger, four anti-virus fail to detect the keylogger samples in Group 2, while two anti-virus detect these samples consistently from the outset. For the ransomware, the variants are detected by the behavior. For fileless malware, detection rates are low, indicating that it may not be considered a significant threat. The samples that evades detection by anti-virus's signature-based methods demonstrate that the source code modification is effective in avoiding detection. Fig. 1 shows the detection results for the 20 keylogger samples in Group 1 during S.1), both through file scanning (a) and behavior monitoring (b). Fig. 2 presents the detection results for the keyloggers in Group 1 during S.3),



(a) detection by file scan (b) detection by behavior

Fig. 3. Keylogger in Group2 detection result in procedure 4)

while Fig. 3 illustrates the detection results for the keyloggers in Group 2 during S.4). These results indicate that not all variants are detected, as some samples were able to evade detection. The increase in the number of detected samples after the submission of Group 1 and Group 2 to VirusTotal suggests that VirusTotal is being utilized as an intelligence source by anti-virus to update their signatures.

III. Conclusion

In this study, we employ LLM to modify the source code of test malware samples, thereby generating malware variants, and investigate the detection capabilities of anti-virus software against these variants. The results indicate that approximately half of the generated variant samples from the 100 samples created for each type of malware can be converted into executable files while retaining the functionality of the original samples. Our analysis of the impact of these variants on anti-virus detection reveals specific tendencies in how anti-virus software handle both unknown and known malware samples.

Ethic: To prevent misuse, experiment prompts are not disclosed and we have refrained from mentioning any particular anti-virus, focusing instead on their general impact.

Acknowledgment: This work was supported by JSPS KAKENHI Grant Number 21H03444 and 21KK0178. This paper is based on results obtained from a project, JPNP24003, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

[1] OpenAI, "GPT-3.5 Turbo." https://platform.openai.com/docs/models/gpt-3-5-turbo.

[2] Yin Minn Pa Pa, S. Tanizaki, T. Kou, M. Van Eeten, K. Yoshioka, and T. Matsumoto, "An attacker's dream? exploring the capabilities of chatgpt for developing malware," in *Proceedings of the 16th Cyber Security Experimentation and Test Workshop*, pp. 10–18, 2023.

[3] OpenAI, "OpenAI API." https://openai.com/blog/openai-api.

[4] PyInstaller. https://pyinstaller.org/.

[5] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.

[6] J. Kornblum, "Identifying almost identical files using context triggered piecewise hashing," *Digital investigation*, vol. 3, pp. 91–97, 2006.

[7] J. Oliver, C. Cheng, and Y. Chen, "Tlsh–a locality sensitive hash," in *2013 Fourth Cybercrime and Trustworthy Computing Workshop*, pp. 7–13, IEEE, 2013.

[8] VirusTotal. https://www.virustotal.com.