The Potential of LLMs for Generating Malicious Domain Names

Lim Kit Michael Ye[1], Kaijian Zheng[1,2], N. F. Law[1] and Jianping Li[2]
[1] The Hong Kong Polytechnic University
[2] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
{michael.ye, kaijian.zheng}@connect.polyu.hk, ngai.fong.law@polyu.edu.hk, jp.li@siat.ac.cn

***Abstract*— Detecting domains generated by Domain Generation Algorithms (DGAs) is important to maintain network security. Studies have found that dictionary-based DGAs are harder to identify than random DGAs. Despite this, existing dictionary-based DGAs are generated from dictionaries with a limited set of words. This study investigates the feasibility of using large language models (LLMs) to enhance the diversity of generated domain names. Our findings demonstrate that LLM-generated domains exhibit strong semantic coherence, presenting significant challenges for existing DGA detection techniques. This highlights the existing gap in current detection methods and requires improved strategies in response to the evolving landscape of cyber threats.**

I. INTRODUCTION

Domain Generation Algorithms (DGA) have often played a role in network attacks, particularly in the context of botnets. In such attacks, malicious software is installed on a group of Internet-connected devices, granting attackers control over a network of compromised entities, including Internet-of-Things (IoTs) devices and computers. This control facilitates remote manipulation and the execution of specific tasks without the owner's awareness, such as launching attacks or stealing data [5]. Traditionally, attackers directed commands to botnets via a centralized Command and Control (C&C) server. In the past, fixed Internet addresses or domain names were used by the C&C server. Hence, defensive mechanisms like blacklists could easily disrupt communication by obstructing access to known malicious addresses. However, the advent of DGA has enabled attackers to dynamically generate an extensive array of domain names, with only a small subset being registered for genuine C&C server use. This approach ensures secure communication between C&C servers and bots, preventing communication from being blocked by blacklisting method [1].

Traditional DGAs generated domain names by concatenating characters together, resulting in domains that appear random and differ from legitimate domain names. This distinctiveness makes them relatively easy to identify [2-3]. In response to this, cyber attackers have developed dictionary-based DGAs to produce domain names that closely mimic legitimate ones. In this approach, a dictionary is first established. Words from the dictionary are then concatenated to form domain names. Such DGAs exhibit semantic coherence and pose challenges for DGA detection methods. Studies found that it is more difficult to detect dictionary-based DGAs than their randomly generated counterparts [4-6].

The field of natural language processing is experiencing rapid development, driven by the emergence of large pre-trained language models such as BERT and LLaMA [7-8]. In view that existing dictionary-based DGAs rely on a small-sized dictionary, this study explores the feasibility and potential of using large language models (LLMs) to generate DGAs with a vast vocabulary. It results in a set of domain names with a wide range of variations. Our study also aims to evaluate the effectiveness of existing DGA detection methods in identifying DGAs generated by the LLMs.

The primary contributions of this study include the development of an AI agent that leverages LLMs to produce DGAs; and an evaluation of the performance of existing DGA detection methods in detecting these language model-generated DGAs. Our research validates the feasibility of employing LLMs to generate semantically coherent domain names that closely resemble legitimate ones. Furthermore, our findings show the difficulty of existing detection methods in identifying LLM-generated domains.

II. BACKGROUND

Both convolutional neural networks (CNNs) and long short-term memory networks (LSTMs) have been applied for DGA detection [9, 10]. LSTMs are effective for acquiring sequential relationships while CNNs use filter kernels of various sizes to characterize patterns. Previous studies have shown that combining CNNs and LSTMs can enhance detection capabilities [6]. While most methods perform well on detecting random DGAs, they face challenges in detecting dictionary-based DGAs.

To address this issue, n-gram models [11] and subword tokenization techniques [5] have been explored. The n-gram method analyses domain names by breaking them down into n-grams, which are continuous sequences of characters. Subword tokenization allows a more granular token-level analysis. Besides, transformer-based models such as Bidirectional encoder representations from transformers

979-8-3315-7206-8/25/$31.00 ©2025 IEEE

(BERT) [12-14] and large language models like LLaMA can also be employed to analyze the linguistic characteristics of the domain names [15-16]. All these methods have shown strong capabilities in detecting dictionary-based DGAs.

Existing dictionary-based DGAs have certain limitations. Table 1 lists the sizes of dictionaries used in generating dictionary-based DGAs. It is evident that each dictionary in existing dictionary-based DGAs contains a limited number of words, ranging from a few hundred to a few thousand. With the restricted vocabulary sizes within these dictionaries, different DGA detection methods may effectively capture the limited word selection of these dictionary-based DGAs.

To introduce more diversities and challenges, one can use a larger dictionary so that a wider variety of dictionary-based DGAs can be generated. Since each botnet needs to generate the same set of dictionary-based DGAs, having a larger dictionary size may result in increased computations for botnets, which may not be desirable. Thus, we aim to investigate if it is feasible to have a greater variety of words selection while computationally feasible.

Table 1: Dictionaries used in dictionary-based DGAs.

DGA family	Number of words in dictionaries
GOZI	4379 (GLP), 1537 (LUTHER), 558 (NASA), 2460 (RF)
MATSNU	878 (verb dictionary) 1008 (noun dictionary)
NYMAIM	2450 (the first dictionary) 4387 (the second dictionary)
PIZD	384
ROVNIX	About 1458 (US declaration of independence)
SUPPOBOX	384

III. PROPOSED LLM-DGA

The aim of this study is to explore the capability of large language models (LLMs) in producing domain names that closely resemble legitimate domains. Unlike existing dictionary-based DGA methods, LLMs are trained in extensive data, leading to a significantly larger vocabulary size.

LLM agents are designed to automatically perform tasks and produce content based on text prompts. Our objective is to guide and regulate the outputs of these agents to generate domain names that exhibit strong linguistic context. To ensure validity, the generated domain names must adhere to the rules governing valid domain names, including a maximum length of sixty-three characters and the use of permissible characters exclusively. Additionally, the domains must be unregistered. As the generated domain names closely resemble legitimate domains, we can consider them as a new type of dictionary-based DGAs, which we refer to as LLM-DGAs in subsequent discussions.

To explore this approach, we construct an LLM-based agent using the LLaMA 3.2-3b-instruct model. The agent is structured through a node-edge framework, which allows it to maintain an internal state that tracks necessary information for domain name generation. Key parameters include the number of domain names to be generated, the specific topic guiding the domain name generation, and previous model interactions.

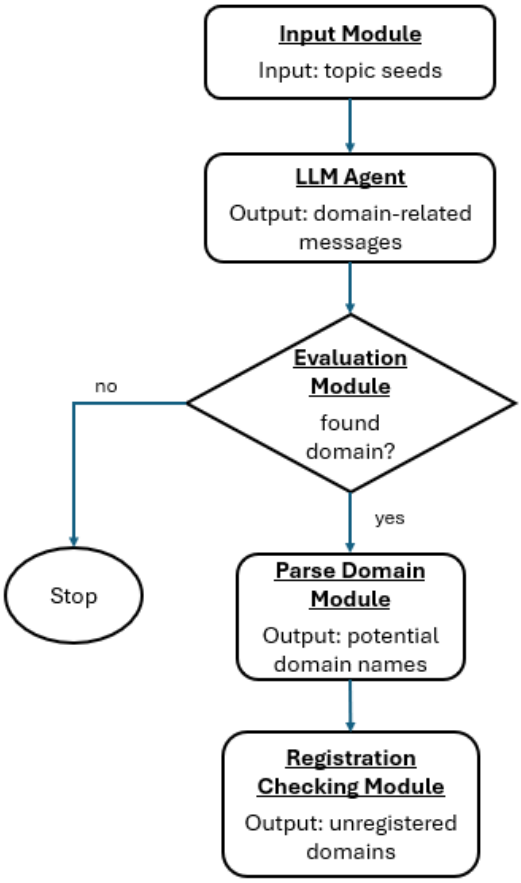

Fig. 1 The proposed LLM agent for language model-based domain name generation (LLM-DGA).

Figure 1 shows the architecture of the LLM agent for language model-based domain name generation (LLM-DGA). It comprises the following key components:

(1) **Input module**: topic seeds are derived from the "google-10000-english" word list [17]. This seed provides a foundation for generating contextually relevant domains related to a specific topic.

(2) **LLM agent**: it generates domain-related messages based on the context provided, ensuring that the output maintains linguistic coherence. As shown in Figure 2, the prompt contains the aim of the agent, detailed

instructions regarding the task, and examples to show the expected input and output (e.g., “Generate 3 domain names related to cooking with the TLD .com”).

(3) **Evaluation module**:

a. If the generated message contains domain names, the process proceeds to the Parse Domain module.

b. If no domain names are found, the workflow stops.

(4) **Parse Domain module**: prompt engineering techniques are used to convert unstructured generated messages into structured domain name outputs. As shown in Figure 3, the prompt contains simple instructions and examples to show the expected output.

(5) **Registration Checking module**: the agent checks the generated domain names to determine whether they are already registered. As shown in Figure 4, the checking is done using Python’s socket library. The final output list contains only unregistered domain names.

Table 2 shows examples of domain names generated by the domain name LLM agent. These examples show that the generated domains (i.e., LLM-DGAs) closely resemble legitimate domains in terms of context and linguistic coherence.

```
#domain generation agent
system_prompt_node1 = \
"""You are required to generate domain names based  \
on a given topic. For every request, you will be \
provided with a topic, and your task is to generate \
domain names that include the word 'DOMAIN' before \
each suggested name. Below are examples of how the \
input request and your response should look like: \
Example:\
Request: Please help me generate 3 domain names about \
"cooking\" with tld .com\
Response:\
DOMAIN: TastyHub.com\
DOMAIN: RecipeRush.com\
DOMAIN: TheCookingCoach.com """

user_prompt_node1 = "Please help me generate {number}  \
domains related to \"{topic}\" with tld \"{tld}\" and \
please do not generate domain that already exisit"
```

```
def domain_generate(agent_state: Agent_State):

    messages = agent_state["state"]["messages"]

    node1_prompt = PromptTemplate(
        template=user_prompt_node1
    )
    prompt_input = node1_prompt.format(
    number = agent_state['number'],
    topic = agent_state['topic'],
    tld = agent_state['tld'],
    )

    output = get_model_response(prompt_input, system_prompt_node1)

    updated_state = MessagesState(messages=new_messages)
```

Fig. 2 LLM agent to generate domain related messages.

```
def parse_domain(agent_state: Agent_State):
    messages = agent_state["state"]["messages"]
    text_to_analyze = message[1]

    domain_schema = ResposneSchema(
        name = "domain",
        description = """\
        Extract domain names mentioned in the text. For example, \
        if the text contains: "The domains are hello.com and \
        byebye.net", the output should be a list like: \
        ["hello.com", "byebye.net"]."""
    )
    response_schemas = [domain_schema]
    output_parser = StructureOutputParser.from_response_schemas(
        response_schemas)
    review_prompt = PromptTemplate(template=review_template1)
    prompt_input = review_prompt.format(text=text_to_analyze)
    output = get_model_response(prompt_input)
```

Fig. 3 Parse domain module to extract domain names.

```
def is_domain_registered(domain):
    try:
        # Try to get the domain's IP address
        socket.gethostbyname(domain)
        return True  # Domain is registered
    except socket.gaierror:
        return False  # Domain is not registered
```

Fig. 4 Registration checking module to check if a domain name has been registered.

Table 2. Example domain names in LLM-DGAs.

Topic seed	**Generated domains**
September	septinspire.com
available	availablesolutions.com
department	departmenthub.com
conditions	medicalconditionguide.com

Repeatability in DGA generation is important for facilitating secure communication between the C&C server and the botnet. Several parameters play significant roles in the LLM-DGA generation, namely “do_sample”, “temperature”, and “top_k”. The “do_sample” parameter relates to how the most probable next word is selected. To promote consistency and predictability in the generated text, it should be set to False. The parameter “temperature” influences the randomness in text generation. It modifies the probability distribution of the next word. A small temperature value implies a conservative model, favoring words with higher likelihoods and reducing randomness. The “top-k” parameter defines the number of top labels returned. Setting it to one ensures only the most probable outputs are returned. Moreover, it is important to avoid quantization. Besides, for maintaining numerical consistency and preventing lossy compression, FP32 should be used as the computation data type. Through these settings, the generated domains have

minimal variability, ensuring that the same list of DGAs is generated consistently by both botnets and the C&C server.

IV. Simulation Results of LLM-DGA

The proposed LLM agent for LLM-DGAs has been configured to produce domain names that resemble legitimate domain names. Setting the parameters as number=20 and topic= “beatles”, the LLM agent (Figure 2) is instructed to produce the following twenty domain names related to the topic “Beatles”:

> Here are 20 domain name suggestions related to "Beatles" with the .com extension:
>
> 1. DOMAIN: FabFourBeatles.com
> 2. DOMAIN: LennonLegacy.com
> 3. DOMAIN: TheBeatlemania.com
> 4. DOMAIN: SgtPeppersDomain.com
> 5. DOMAIN: MagicalMysteryTour.com
> 6. DOMAIN: YellowSubmarineDomain.com
> 7. DOMAIN: AbbeyRoadDomain.com
> 8. DOMAIN: TheBeatlemania.com
> 9. DOMAIN: JohnLennonLegacy.com
> 10. DOMAIN: PaulMcCartneyDomain.com
> 11. DOMAIN: GeorgeHarrisonHub.com
> 12. DOMAIN: Ringo StarrDomain.com
> 13. DOMAIN: TheBeatlesOfficial.com
> 14. DOMAIN: BeatlesFanClub.com
> 15. DOMAIN: LetItBeDomain.com
> 16. DOMAIN: AllYouNeedIsLove.com
> 17. DOMAIN: StrawberryFieldDomain.com
> 18. DOMAIN: PennyLaneDomain.com
> 19. DOMAIN: TheBeatlesClassics.com
> 20. DOMAIN: ComeTogetherDomain.com<|eot_id|>

With the use of the Parse Domain module in Figure 3, potential domain names are extracted as follows,

"FabFourBeatles.com", "LennonLegacy.com", "TheBeatlemania.com", "SgtPeppersDomain.com", "MagicalMysteryTour.com", "YellowSubmarineDomain.com", "AbbeyRoadDomain.com", "TheBeatlemania.com", "JohnLennonLegacy.com", "PaulMcCartneyDomain.com", "GeorgeHarrisonHub.com", "Ringo StarrDomain.com", "TheBeatlesOfficial.com", "BeatlesFanClub.com", "LetItBeDomain.com", "AllYouNeedIsLove.com", "StrawberryFieldDomain.com", "PennyLaneDomain.com", "TheBeatlesClassics.com", "ComeTogetherDomain.com"

These twenty domain names are checked in Registration Checking Module (Figure 4) to return only those that are unregistered as in Figure 5.

```
Unregistered domains:

[32]: ['fabfourbeatles.com',
       'lennonlegacy.com',
       'thebeatlemania.com',
       'sgtpeppersdomain.com',
       'yellowsubmarinedomain.com',
       'abbeyroaddomain.com',
       'thebeatlemania.com',
       'johnlennonlegacy.com',
       'paulmccartneydomain.com',
       'georgeharrisonhub.com',
       'beatlesfanclub.com',
       'letitbedomain.com',
       'strawberryfielddomain.com',
       'pennylanedomain.com',
       'thebeatlesclassics.com',
       'cometogetherdomain.com']
```

Fig. 5 Results of LLM-DGAs related to the topic of “Beatles”.

Source codes are available on https://github.com/zdrgb622/llm_generate_dga. We will also provide an interface for readers to select their desired topic to generate LLM-DGAs.

V. Evaluation of various Model performances on LLM-DGAs

In order to assess the quality of LLM-DGAs, three state-of-the-art methods have been selected to evaluate their effectiveness in detecting LLM-DGAs. These methods are the n-gram Transformer method [11] with n=3, the BERT model [12], and the LLaMA-based method [16], all known for their excellent DGA detection performance. The 3-gram Transformer model has been fully trained, the BERT model has undergone full fine-tuning, and the LLaMA-3.2-3b-instruct model has been fine-tuned using QLoRA [18]. The number of trainable parameters in each model is given in Table 3.

Table 3. Number of trainable parameters.

	3-gram transformer	BERT	LLaMA-3b-instruct
Total number of parameters	1,182,229	109,484,547	1,805,431,808
Number of trainable parameters	1,182,229	109,484,547	1,968,128

Two training datasets have been constructed for evaluation:

- **Train-Dataset A** contains 400 samples, each from legitimate domains, random DGAs and dictionary-based DGAs.
- **Train-Dataset B** extends Train-Dataset A by including LLM-DGAs. It comprises 400 samples of legitimate domains, 400 samples of random DGAs, 200 samples of dictionary-based DGAs and 200 samples of LLM-DGAs (LLM-DGAs is regarded as a variant of dictionary-based DGAs).

Three test datasets are constructed for performance evaluation:

- **Test-set 1** contains 60,000 domains, which includes 20,000 domains each from legitimate domains, random DGAs and dictionary-based DGAs.
- **Test-set 2** contains the 16 unregistered LLM-DGAs in Figure 5.
- **Test-set 3** contains17,032 LLM-DGAs.

Note that the aim of Test-set 2 is to demonstrate the potential performance gap of existing detection methods for the LLM-DGAs generated in Figure 5. Test-set 3 contains more LLM-DGAs to confirm the performance gap.

We first consider using Train-Dataset A for training the three models. Table 4 shows the evaluation results on Test-set 1. Due to the small training data size, the 3-gram transformer does not perform as good as the BERT and the LLaMA model.

Table 4. Detection performance on Test-set 1 when trained on Train-Dataset A.

Models	Precision	Recall	F1 score	Overall accuracy
3-gram transformer				73.28%
Legitimate domains	64.32%	68.44%	66.32%	
Random DGAs	76.70%	61.85%	68.48%	
Dictionary-based DGAs	79.29%	89.57%	84.11%	
BERT				95.10%
Legitimate domains	92.94%	92.81%	92.87%	
Random DGAs	97.00%	93.59%	95.26%	
Dictionary-based DGAs	95.41%	98.91%	97.13%	
LLaMA				94.73%
Legitimate domains	90.66%	94.74%	92.66%	
Random DGAs	96.21%	91.49%	93.79%	
Dictionary-based DGAs	97.55%	97.95%	97.75%	

From Table 4, we can see that these models are effective for dictionary-based DGAs. These trained models are then applied for LLM-DGAs (test-set 2), the accuracies are:

- 3-gram Transformer: 3/16 = 18.75%
- BERT: 1/16 = 0.25%
- LLaMA: 0/16 = 0%

We also considered a larger set of LLM-DGAs (test-set 3), the accuracies are:

- 3-gram Transformer: 37.27%
- BERT: 8.67%
- LLaMA: 10.28%

For detecting LLM-DGAs, all models have bad performance with low accuracies. It is in sharp contrast to other types of dictionary-based DGAs. Train-Dataset A is used to assess the capabilities of existing detectors in identifying the language model generated DGAs. In order to have better generalization, it is always important to make the source domain for training be like the target domain for testing. As such, Train-Dataset B is prepared to improve the performance on LLM-DGAs. With models trained on Train-Dataset B, the performance on Test-set 1 is shown in Table 5.

Table 5. Model performance on Test-set 1 when trained on Train-dataset B.

Models	Precision	Recall	F1 score	Overall accuracy
3-gram transformer				73.26%
Legitimate domains	57.32%	52.65%	54.88%	
Random DGAs	73.46%	64.80%	68.86%	
Dictionary-based DGAs	80.32%	88.97%	84.42%	
BERT				90.84%
Legitimate domains	82.51%	82.60%	82.56%	
Random DGAs	97.46%	93.59%	95.49%	
Dictionary-based DGAs	91.89%	93.81%	92.84%	
LLaMA				90.22%
Legitimate domains	80.47%	84.52%	82.45%	
Random DGAs	92.42%	96.74%	94.53%	
Dictionary-based DGAs	94.72%	89.78%	92.18%	

Comparing results in Table 4 and Table 5, the performance on both legitimate domains and dictionary-based DGAs drops slightly by including LLM-DGAs for training. These trained models are then applied for LLM-DGAs (Test-set 2), the accuracies are:

- n-gram Transformer: 7/16 = 43.75%
- BERT: 10/16 = 62.5%
- LLaMA: 4/16 = 25%

We also considered a larger set of LLM-DGAs (Test-set 3), the accuracies are:

- n-gram Transformer: 90.72%
- BERT: 94.15%
- LLaMA: 86.80%

These results suggest that incorporating LLM-DGAs during training (as in Train-Dataset B) improves the models' ability to correctly classify LLM-DGAs. These simulations illustrate that current DGA detection methods cannot accurately identify language model-generated DGAs. It is necessary to integrate LLM-DGAs into the training process to enhance the detection performance.

VI. Conclusions

Large language models (LLMs) have emerged as powerful tools for different applications. In this study, we have demonstrated that LLMs can be employed to generate dictionary-based DGAs that exhibit greater diversity while maintaining semantic coherence. Experimental results show that these LLM-generated domains pose challenges to existing DGA detection methods, emphasizing the need for enhanced detection strategies to effectively detect this new type of dictionary-based DGAs.

VII. Acknowledgment

Kaijian Zheng acknowledges the support from both the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences and the Hong Kong Polytechnic University. The work was supported by Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University.

References

[1] G. Vormayr, T. Zseby, and J. Fabini, "Botnet Communication Patterns," *IEEE Commun Surv Tutor*, 19: 2768-2796, 2017.

[2] Z. Wang and Y. Guo, "Neural Networks based Domain Name Generation", *J. Inf Secur Appl*, 61:102948, 2021.

[3] M Zago, M. Perez, and G.M. Perez, "UMUDGA: a Dataset for Profiling DGA-based Botnet", *Comput Secur*, 92:101719, 2020.

[4] F. Ren, Z. Jiang, X. Wang, and J. Liu, "A DGA Domain Names Detection Modeling Method based on Integrating an Attention Mechanism and Deep Neural Network", *Cybersecurity*, 3:1-13, 2020.

[5] S.R.C. Liew and N.F. Law, "Use of Subword Tokenization for Domain Generation Algorithm Classification," *Cybersecurity*, 6(1), 49, 2023.

[6] S.R.C. Liew and N.F. Law, "Word Encoding for Word-looking DGA-based Botnet Classification," *APSIPA ASC 2023*, 1816-1821, 2023.

[7] J. Devlin, M.W. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of Deep Bidirectional Transformers for Language Understanding", *arXiv: 1810.04805*, 2019.

[8] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.A. Lachaux, T. Lacroix, B. Roziere, N. Goyal, E. Hambro, F. Axhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "LLaMA: open and Efficient Foundation Language Models", *arXiv: 2302.13971*, 2023.

[9] D.S. Berman, "DGA CapsNet: 1D Application of Capsule Networks to DGA Detection", *Information*, 10:157, 2019.

[10] J. Woodbridge, H.S. Anderson, A. Ahuja, and D. Grant, "Predicting Domain Generation Algorithms with Long Short-term Memory Networks", *arXiv: 1611.00791*, 2016.

[11] A. Cucchiarelli, C. Morbidoni, L. Spalazzi, M. Baldi, "Algorithmically Generated Malicious Domain Names Detection based on n-grams Features", *Expert Systems with Applications*, 170:114551, 2021.

[12] Y. Tian and Z. Li, "Dom-BERT: Detecting Malicious Domains with Pre-training Model", *International Conference on Passive and Active Network Measurement*, 133-158, 2024.

[13] A.E. Mahdaouy, S. Lamsiyah, M.J. Idrissi, H. Alami, Z. Yartaoui, and I. Berrada, "Domurls_bert: Pre-trained BERT-based Model for Malicious Domains and URLs Detection and Classification", *arXiv: 2409.09143*, 2024.

[14] S.R.C. Liew and N.F. Law, "BEAM – an Algorithm for Detecting Phishing Link", *APSIPA ASC 2022*, 598-604, 2022.

[15] O.R. Leyva La, C.A. Catania, T. Parlanti, "LLMs for Domain Generation Algorithm Detection", *arXiv: 2411.033307*, 2024.

[16] M.A. Sayed, A. Rahman, C. Kiekintveld, and S. Garcia, "Fine-tuning Large Language Models for DGA and DNS Exfiltration Detection", *arXiv: 2410.21723*, 2024.

[17] J. Kaufman, "Google-10000-english", GitHub: https://github.com/first20hours/google-10000-english.

[18] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient Finetuning of Quantized LLMs", *arXiv: 2305.14314*, 2023.