sensors

MDPI

Article

CIPHER: Cybersecurity Intelligent Penetration-Testing Helper for Ethical Researcher

Derry Pratama [1], Naufal Suryanto [2], Andro Aprila Adiputra [1], Thi-Thu-Huong Le [3], Ahmada Yusril Kadiptya [1], Muhammad Iqbal [1] and Howon Kim [1,*]

[1] School of Computer Science and Engineering, Pusan National University, Busan 46241, Republic of Korea; derryprata@gmail.com (D.P.); androaprila@pusan.ac.kr (A.A.A.); yusril@pusan.ac.kr (A.Y.K.); iqbal@pusan.ac.kr (M.I.)
[2] IoT Research Center, Pusan National University, Busan 46241, Republic of Korea; naufalsuryanto@gmail.com
[3] Blockchain Platform Research Center, Pusan National University, Busan 46241, Republic of Korea; lehuong7885@gmail.com
* Correspondence: howonkim@pusan.ac.kr

Abstract: Penetration testing, a critical component of cybersecurity, typically requires extensive time and effort to find vulnerabilities. Beginners in this field often benefit from collaborative approaches with the community or experts. To address this, we develop Cybersecurity Intelligent Penetration-testing Helper for Ethical Researchers (CIPHER), a large language model specifically trained to assist in penetration testing tasks as a chatbot. Unlike software development, penetration testing involves domain-specific knowledge that is not widely documented or easily accessible, necessitating a specialized training approach for AI language models. CIPHER was trained using over 300 high-quality write-ups of vulnerable machines, hacking techniques, and documentation of open-source penetration testing tools augmented in an expert response structure. Additionally, we introduced the Findings, Action, Reasoning, and Results (FARR) Flow augmentation, a novel method to augment penetration testing write-ups to establish a fully automated pentesting simulation benchmark tailored for large language models. This approach fills a significant gap in traditional cybersecurity Q&A benchmarks and provides a realistic and rigorous standard for evaluating LLM's technical knowledge, reasoning capabilities, and practical utility in dynamic penetration testing scenarios. In our assessments, CIPHER achieved the best overall performance in providing accurate suggestion responses compared to other open-source penetration testing models of similar size and even larger state-of-the-art models like Llama 3 70B and Qwen1.5 72B Chat, particularly on insane difficulty machine setups. This demonstrates that the current capabilities of general large language models (LLMs) are insufficient for effectively guiding users through the penetration testing process. We also discuss the potential for improvement through scaling and the development of better benchmarks using FARR Flow augmentation results.

Keywords: penetration testing; large language model; pentesting LLM; AI penetration testing assistant; domain specific LLM; LLM evaluation; vulnerabillity detection

check for updates

Citation: Pratama, D.; Suryanto, N.; Adiputra, A.A.; Le, T.-T.-H.; Kadiptya, A.Y.; Iqbal, M.; Kim, H. CIPHER: Cybersecurity Intelligent Penetration-Testing Helper for Ethical Researcher. *Sensors* **2024**, *24*, 6878. https://doi.org/10.3390/s24216878

Academic Editor: Alexios Mylonas

Received: 27 July 2024
Revised: 9 October 2024
Accepted: 22 October 2024
Published: 26 October 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Penetration testing, a fundamental aspect of cybersecurity, involves probing computer systems [1], networks [2], or web applications [3] to identify vulnerabilities that malicious actors could exploit. This proactive approach helps to uncover weaknesses before they can be manipulated, safeguarding sensitive data and maintaining system integrity. Despite its critical importance, penetration testing presents significant challenges, particularly for beginners. These challenges include understanding the complexity of modern IT infrastructures, mastering a wide range of tools, and gaining the necessary technical expertise to identify and exploit vulnerabilities effectively. Traditional penetration testing methods,

which primarily rely on manual techniques, can be inefficient and labor-intensive [4], making them especially difficult for newcomers. Moreover, scaling these methods to meet the complexity and rapid evolution of modern IT systems further complicates the process. As cyber threats continue to grow in sophistication, there is a growing need for penetration testing methodologies that are not only efficient and comprehensive but also accessible to beginners, helping them overcome the steep learning curve.

Automated tools and frameworks, such as Metasploit [5] and OpenVAS [6], are designed to assist beginners by streamlining the penetration testing process. However, even with automation, users still need a certain level of expertise to effectively utilize these tools in dynamic pentesting situations, especially when interpreting and acting on the collected information. Meanwhile, the application of large language models (LLMs) in penetration testing represents a cutting-edge area of research with promising results. Advanced models such as GPT-4 [7] have demonstrated significant potential to automate and improve various penetration testing tasks, including Linux privilege escalation and the identification of file-based vulnerabilities [8]. However, implementing a straightforward LLM pipeline presents challenges, particularly in maintaining long-term memory for generating consistent and accurate commands throughout extended testing sessions [9]. Recent innovations like PentestGPT [9] and AutoAttacker [10] have addressed these limitations. These systems leverage existing LLMs and open-source models, incorporating specialized frameworks to emulate human penetration testers' workflow and decision-making processes. While these advancements mark significant progress, it is important to note that these general-purpose models are not explicitly fine-tuned for the nuances of penetration testing. This lack of domain-specific optimization can potentially limit their effectiveness in handling complex, context-dependent scenarios often encountered in real-world penetration testing environments.

Recognizing these limitations, we developed Cybersecurity Intelligent Penetration-testing Helper for Ethical Researchers (CIPHER) as a chatbot assistant specifically designed to support penetration testing tasks. CIPHER is a large language model fine-tuned on a specialized penetration testing dataset to assist ethical hackers by providing insights and recommendations. Trained on over 300 write-ups of vulnerable machines, hacking techniques, and open-source tools, CIPHER mimics expert reasoning and enhances the penetration testing process. It offers detailed explanations and guidance, particularly benefiting users with limited penetration testing experience. CIPHER is designed to be used specifically to guide beginner proceed with penetration testing a vulnerable machine for ethical research.

Building on this foundation, we introduced the Findings, Action, Reasoning, and Results (FARR) Flow, a novel method for the automated evaluation of large language models in penetration testing scenarios. FARR Flow acts as a structured benchmark, simulating realistic pentesting tasks to rigorously assess the model's technical knowledge, reasoning abilities, and practical effectiveness. By focusing on automated evaluation, FARR Flow provides a robust and realistic standard for measuring LLM performance in penetration testing, ensuring models are thoroughly tested in practical, real-world contexts.

Our assessments indicate that CIPHER demonstrates a significant improvement in providing helpful responses for penetration testing over other models, highlighting the limitations of traditional cybersecurity evaluations in capturing the nuances of practical pentesting skills. Specifically, CIPHER showed a notable improvement, underscoring its potential to transform penetration testing practices. This paper presents several key contributions:

- The methodology for CIPHER's development, utilizing penetration testing write-ups structured as beginner- and expert-level conversations to train a large language model specifically for practical penetration testing assistance.
- A novel augmentation technique called Findings, Action, Reasoning, and Results (FARR) Flow that condenses these write-ups into structured formats for more efficient evaluation.

- The development and evaluation of an automated pentesting simulation benchmark based on FARR Flow, providing a rigorous and realistic framework for evaluating LLM performance in practical penetration testing guidance.
- A comprehensive evaluation demonstrating CIPHER's effectiveness and superior performance over existing LLMs in providing accurate guidance in FARR Flow and MITRE ATT&CK capabilities from PurpleLlama.

These contributions bridge the gap between AI advancements and practical cybersecurity applications, offering a new penetration testing domain-specific language model and reproducible benchmark for realistic penetration testing scenarios with consistent results.

2. Background and Related Works

2.1. Large Language Models in Cybersecurity

2.1.1. Large Language Models

Overview of Large Language Models: Large language models (LLMs), exemplified by GPT-4, have revolutionized natural language processing through deep learning and extensive datasets, enabling them to understand and generate human-like text. These models predict the next word sequentially, capturing complex linguistic patterns and producing contextually relevant responses. Their applications include translation, summarization, and content creation, surpassing traditional NLP systems [11].

Transformers: Introduced by Vaswani et al. [11] in "Attention is All You Need" (2017), Transformers have reshaped NLP by employing self-attention mechanisms to process entire sequences concurrently. They excel at capturing long-range dependencies, outperforming sequential models like RNNs. Transformers comprise self-attention layers, feed-forward networks, and positional encoding to maintain word order. Variants such as BERT and GPT have set benchmarks in NLP and extended into computer vision tasks through Vision Transformers (ViTs).

Open-Source LLMs: Open-source LLMs like the Llama family by Meta AI (Toronto, Canada) [12], Mistral (Paris, France) [13], Yi 1.5 by BAAI (Beijing, China) [14], and Qwen 1.5 by Alibaba Group (Hangzhou, China) [15] democratize access to powerful language models. Evolving from Llama 1 to Llama 3, these models increase token size and context length to enhance language understanding. Innovations like the mixture of experts in Mixtral by Mistral (Paris, France) [16] and unique training methods in Yi 1.5 push the boundaries further. Qwen 1.5 excels in multilingual contexts and incorporates agentic frameworks [17], releasing smaller and powerful models [18], making them versatile tools [12,19].

Reasoning Enhancements in LLMs: LLMs benefit from enhancements like Chain of Thought (CoT) prompting and Self-Consistency (SC). CoT prompting encourages explicit reasoning steps, while SC verifies response correctness. Techniques like instruction fine-tuning (FLAN [20]) and model mimicry (Orca [21]) further improve reasoning capabilities. Despite advancements, challenges in LLM reasoning persist, motivating ongoing research [7,22].

LLM-Based Chatbots: LLM-based chatbots like ChatGPT excel in customer support, education, and complex problem-solving by synthesizing large volumes of information into detailed responses. However, they lack the specialized knowledge required for offensive penetration testing [23].

Supervised Fine-Tuning: Supervised fine-tuning enhances model performance on domain-specific datasets, particularly in areas like penetration testing, ensuring accurate application of specialized language. Unlike Retrieval-Augmented Generation (RAG), fine-tuning improves the model's domain comprehension [24].

LLM Alignment: Efforts to align language models with human values focus on ensuring these models exhibit traits such as helpfulness and truthfulness. Reinforcement Learning with Human Feedback (RLHF) fine-tunes large language models (LLMs) to achieve this alignment. This process involves training a reward model to predict human preferences and then using algorithms like Proximal Policy Optimization (PPO) for fine-

tuning. Although PPO generally yields better results, Direct Preference Optimization (DPO) simplifies the process by fine-tuning directly with human-labeled preferences [25,26].

Incorporating Domain-Specific Knowledge: Domain-specific knowledge enhances LLM accuracy in specialized fields like medicine [27] and cybersecurity. Techniques such as Domain-Adaptive Pretraining (DAPT) and adaptive fine-tuning (AdaptLLM) are crucial for developing specialized models tailored for specific tasks, leveraging domain-specific datasets for improved insights [24,28].

2.1.2. Applications and Challenges in Cybersecurity

In cybersecurity, large language models (LLMs) such as GPT-4 [7], PentestGPT [9], hackingBuddyGPT [29], and HackerGPT [30] have been deployed to assist with penetration testing and other security tasks. These models can analyze and synthesize large volumes of information, aiding in identifying vulnerabilities. However, using such advanced models poses significant challenges. The high cost and privacy risks associated with proprietary models like GPT-4, which handle sensitive vulnerability data, are significant concerns. Moreover, general-purpose LLMs often lack the specialized knowledge required for effective penetration testing.

2.2. Existing Tools and Methodologies for Penetration Testing

2.2.1. Manual and Automated Tools

Manual tools such as Netcat, Wireshark, and Burpsuite Community Edition offer robust vulnerability discovery and exploitation frameworks. These tools, however, heavily rely on user expertise to interpret results and often require manual intervention to execute complex attack scenarios.

Meanwhile, automated tools have emerged to streamline the penetration testing process by automating vulnerability scanning and initial exploitation attempts. Examples including Metasploit [31], OpenVAS [32], and Nessus [33] use predefined algorithms to detect and exploit common vulnerabilities. While these tools enhance efficiency, they may overlook subtle vulnerabilities that require human intuition and context to identify.

Combining manual and automated tools optimizes penetration testing effectiveness. Manual methods leverage human expertise to craft targeted attacks and interpret results, while automation accelerates routine tasks and broadens the scope of vulnerability detection. This hybrid approach ensures comprehensive security assessments addressing known vulnerabilities and emerging threats.

2.2.2. Advancements in AI for Penetration Testing

Recent advancements have seen the development of AI-driven tools designed to enhance penetration testing. Research efforts like PentestGPT [9], AutoAttacker [10], hackingBuddyGPT [29], ReaperAI [34], PenHeal [35], Pentest Copilot [36], and BreachSeek [37] leverage LLMs for various penetration testing tasks. Despite their potential, these tools face several limitations:

- Context Loss and Memory Retention: LLMs struggle with retaining long-term conversational memory, crucial for linking vulnerabilities across services to develop an exploitation strategy [9,10].
- Testing Strategy Limitations: LLMs often adopt a depth-first search approach, which may overlook other potential attack surfaces [9].
- Inaccurate Operations and Commands: LLMs may generate inaccurate commands or misinterpret outputs, leading to ineffective testing [38,39].
- Limited Social Engineering and Image Interpretation: LLMs lack capabilities in physical social engineering techniques and interpreting visual hints [10].
- Ethical Concerns: The potential misuse of LLMs for automating attacks raises ethical issues [40].

2.2.3. Leveraging LLMs for Penetration Testing

There are distinct advantages to using native penetration testing LLMs versus general LLMs for cybersecurity tasks as shown in Table 1. Native penetration testing LLMs offer superior performance, specialization, ease of use, efficiency, and security. These models are optimized explicitly for penetration testing tasks, resulting in higher accuracy and efficiency. Their deep specialization and out-of-the-box readiness make them more effective and more accessible to implement for specific penetration testing needs without extensive customization. Moreover, native models ensure better security by handling sensitive data locally, thus reducing the risk of data breaches.

On the other hand, general LLMs [7] used as penetration testing agents [9,29,38,39] provide greater versatility and scalability. These models can perform a broader range of tasks beyond penetration testing and can be scaled easily by integrating additional modules or upgrading to more advanced general models. However, they often require complex setups involving various frameworks, modules, and pipelines, which can be resource-intensive and challenging to manage. The increased complexity can introduce vulnerabilities and make troubleshooting more difficult. Without significant adjustments and configurations, general LLMs may not achieve the same performance and specialization as native penetration testing models.

Table 1. Comparison of current penetration testing LLMs research. Notes (✓: available; ✗: not available).

Name	Native LLM	Chat Assistant	Foothold	Priv. Escalation	Multiple Tools
LLM as Hackers [8]	✗	✗	✗	✓	✗
PentestGPT [9]	✗	✓	✓	✓	✓
AutoAttacker [10]	✗	✗	✓	✓	✗
hackingBuddyGPT [29]	✗	✗	✓	✓	✓
ReaperAI [34]	✗	✗	✓	✓	✓
PenHeal [35]	✗	✗	✓	✓	✓
Pentest Copilot [36]	✗	✓	✓	✓	✓
BreachSeek [37]	✗	✓	✗	✓	✓
LLM Agents Can Autonomously [38,39]	✗	✗	✗	✓	✗
CIPHER (Ours)	✓	✓	✓	✓	✓

Given these considerations, we develop CIPHER as a native LLM tailored for penetration testing chatbot assistant. This approach ensures our model is optimized for the unique requirements of penetration testing tasks, thereby achieving higher accuracy and efficiency. A specialized model integrates domain-specific knowledge and tools more effectively, simplifying deployment and reducing the need for extensive customization. Additionally, handling sensitive data locally enhances security and minimizes the risk of exposure to vulnerable information.

2.3. Specialized Cybersecurity Large Language Models

2.3.1. Development of Domain-Specific LLMs

Specialized LLMs trained on domain-specific data, such as cybersecurity, are necessary to address the shortcomings of general-purpose models. Techniques like Domain-Adaptive Pretraining (DAPT) and adaptive fine-tuning improve performance by incorporating specialized knowledge and tools tailored for tasks like penetration testing [24,27,41–43]. These approaches enable models to understand and apply domain-specific language more effectively, enhancing their utility in specialized fields.

2.3.2. Evaluation and Benchmarking Challenges

Current evaluation frameworks for cybersecurity models, such as PurpleLlama CyberSecEval [44] with MITRE ATT&CK, primarily focus on predefined scenarios and known attack patterns [45]. These do not adequately measure a model's ability to discover novel vulnerabilities or adapt to dynamic situations encountered in real-world penetration testing. Previous works like PentestGPT and AutoAttacker lack reproducible benchmarks, relying on manual testing and human scoring, which introduces bias and limits reproducibility [9,10]. Developing standardized, automated benchmarks that reflect real-world conditions remains a significant challenge for advancing the evaluation of cybersecurity LLMs.

2.3.3. Reproducible Benchmark for Penetration Testing Scenario

To address these challenges, we introduce the FARR Flow augmentation method, which automates the generation of penetration testing scenarios using high-quality write-ups. Unlike other existing penetration testing benchmarks for LLM, FARR Flow evaluation covers a wide range of tool usage capabilities, from reconnaissance to exploitation tools. FARR Flow evaluation results are also easily reproducible due to the scoring automation by using judge LLM. We also release our code on GitHub. This approach provides a dynamic and reproducible benchmark, assessing the LLM's ability to reason and act like an expert penetration tester.

As shown in Table 2, our research demonstrates that FARR Flow evaluation fills the existing gap in current penetration testing benchmark for an LLM that is easily reproducible, open-source, and covers a wide range of tools.

Table 2. Comparison of penetration testing evaluation approaches. Notes (✓: yes; ✗: not provided).

Name	Multiple Tools	Consistent Reproducibility	Open Source Availability
LLM as Hackers [8]	✗	✓	✓
PentestGPT [9]	✓	✗	✗
AutoAttacker [10]	✗	✓	✗
hackingBuddyGPT [29]	✓	✗	✓
ReaperAI [34]	✓	✗	✓
PenHeal [35]	✓	✓	✗
Pentest Copilot [36]	✓	✗	✗
BreachSeek [37]	✓	✗	✓
LLM Can Autonomously [38,39]	✗	✗	✗
FARR Flow (Ours)	✓	✓	✓

In Section 4.3, we propose FARR Flow augmentation and evaluation, which utilize existing high-quality penetration testing write-ups as a base for penetration testing flow. Therefore, creating the correct step of penetration testing of a specific vulnerable machine is no longer needed, enabling us to measure the performance of the model on existing vulnerable machines. Unlike traditional human scoring, which cannot be consistently reproducible, in FARR Flow we can change the evaluator model to avoid and minimize bias using multiple SOTA models as evaluators.

3. Methodology

CIPHER development focused on two key contributions, the first is where we develop our specific chatbot assistant model that is highly focused on suggesting the correct next step to be taken based on currently obtained information. Secondly, we provide an automated

benchmarking standard for measuring the penetration testing advice from the LLM that is easily reproducible and has consistent results.

Unlike other penetration testing LLMs that rely solely on Retrieval-Augmented Generation (RAG) or agent-based pipelines dependent on the OpenAI model, CIPHER's knowledge is primarily drawn from highly curated penetration testing write-ups. These resources provide a rich foundation of both successful and failed penetration tests, offering invaluable insights into the reasoning and decision-making processes of experienced penetration testers. We believe that merely relying on the literature is insufficient for teaching a model the nuances of penetration testing; the real expertise comes from understanding how professionals adapt and learn through hands-on experience.

Building on this core, CIPHER's methodology Figure 1 leverages publicly available penetration testing resources and write-ups to form the backbone of its knowledge base. The curated mixture of resources, detailed in Table 3, ensures that CIPHER learns from real-world scenarios that capture both successes and failures. This knowledge is further augmented with expert-guided conversations (explained in Section 3.2.3), which help the model understand the context and decision-making paths within the penetration testing process. Additionally, to retain a well-rounded conversational ability, CIPHER integrates general world knowledge through resources like OpenHermes 2.5 [46], ensuring it can handle both technical penetration testing tasks and general discussions as a chatbot.

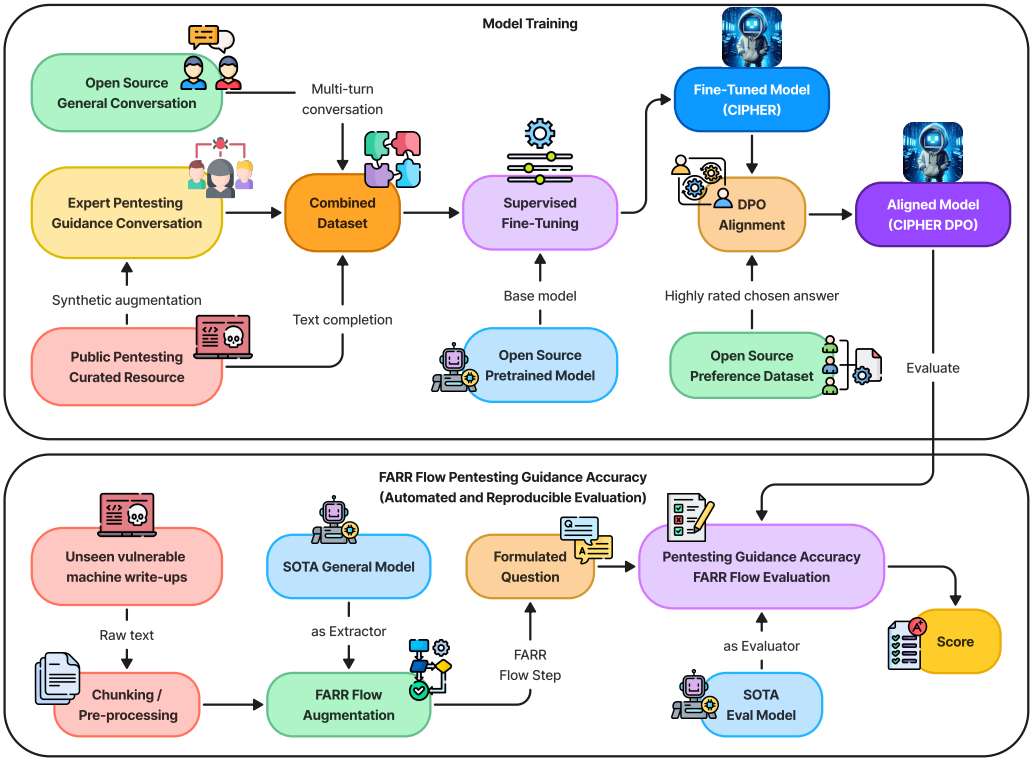


Figure 1. CIPHER development methodology overview for training and novel automated evaluation.

To further refine CIPHER's capabilities, a supervised fine-tuning process is performed using the Axolotl framework [47], enabling quick and reproducible training with hyperparameters detailed in Section 3.3.1. Following fine-tuning, an alignment process is conducted using openly available preference datasets to ensure that CIPHER aligns with human preference, as discussed in Section 3.3.2.

For evaluation, we utilize the latest, unseen vulnerable machine write-ups as references. These write-ups are processed into chunks and extracted using FARR Flow augmentation, which integrates the strengths of a state-of-the-art (SOTA) general LLM model, described in Section 3.4. Once these flows are gathered, we evaluate CIPHER's performance by formulating questions based on the extracted knowledge, as outlined in Section 4.3.

3.1. Architecture Design

The main architecture of CIPHER is shown in Figure 2. CIPHER's primary objective is to simplify identifying system vulnerabilities by providing expert guidance throughout the penetration testing workflow. Recognizing the challenges beginners face in using specialized tools and grasping core concepts, CIPHER offers the following:

- Explanation of the question.
- Intuitive, step-by-step instructions with reasoning.
- Insightful response that resembles expert penetration tester reasoning.
- Additional practical examples for the action.

In order to enhance the accuracy in suggesting command line usage on deployment, we added advanced Retrieval Augmented Generation (RAG) [48] in full architecture as seen in Figure 2. Initially, user question will be processed by the mxbai-embed-large-v1 embedding model with 335M parameters [49,50] to find the similar hacking technique and command line documentation. Then similar documentation will be reranked using reranker model [51,52] to find the best document chunks. CIPHER will use the related chunks as in-context learning [53] material to answer the user question with a suggestion as accurate as possible. CIPHER utilizes BGE-reranker-base (278M parameter) that came from BGE family embedding model [52]. The embedding model will convert the text into vectors to convert all pentesting-related content. To query the content, we retrieve top-3 closest content based on the inputted prompt using Cosine Similarity to reduce unrelated content. This research will discuss the core development of the language model to support this environment architecture.

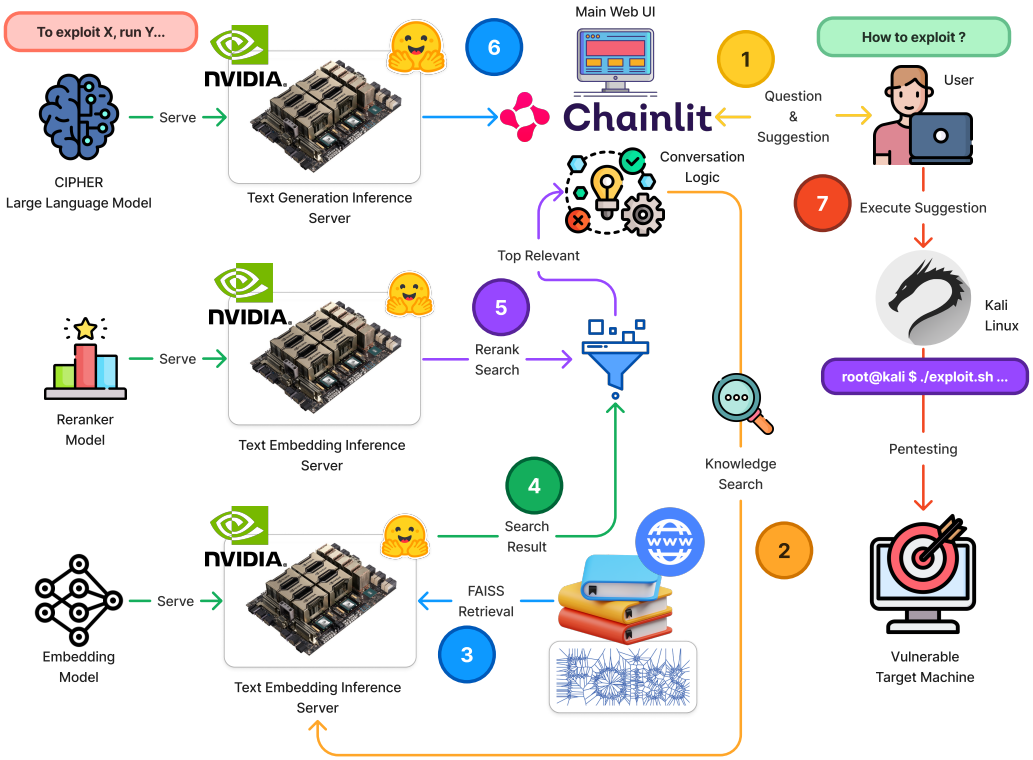


Figure 2. The architecture of CIPHER deployed as chat assistant. (1) The user submits a query and known information, which is converted into text embeddings (2). FAISS VectorDB performs cosine similarity matching with the knowledge database (3). The reranker filters and reorders results based on relevance (4). The top-ranked response is then used to build reference (5), leading to the generation of the best suggestion for penetration steps (6). The user executes this suggestion on the attacker machine (7).

CIPHER focuses on facilitating penetration testing tools and emphasizes developing the user's ability to reason and make informed decisions during the testing process. By offering detailed explanations and reasoning for each step, CIPHER helps users understand the underlying principles and methodologies experienced penetration testers use. This dual focus on tool usage and expert reasoning equips users with the skills and confidence to conduct thorough and effective penetration tests, ultimately enhancing the security of their systems.

The development of CIPHER aims to bridge the gap between novice and expert, equipping users with the technical skills and reasoning abilities necessary to improve their system security posture significantly.

3.2. Dataset

Developing a pentesting chat assistant requires two fundamental capabilities: general chat assistance and domain-specific pentesting expertise. We utilize distinct datasets to address each of these capabilities.

3.2.1. General Assistant Capability

We leverage the OpenHermes 2.5 dataset for our general assistant capability, which is currently recognized as one of the best open-source conversation collections. This dataset was originally used to fine-tune the OpenHermes 2.5 model sourced from various origins, including Orca, Glaive, and other GPT-4 responses in diverse questions and instructions [46]. When used to fine-tune a base model, this conversational dataset has improved performance on multiple general-use benchmarks such as MMLU [54], HumanEval [55], and TruthfulQA [56].

Note that this dataset primarily enhances the model's ability to formulate responses rather than expanding its knowledge base. The dataset provides examples of responding to greetings, questions, and instructions, effectively teaching the model appropriate "answering styles." The varying performance of models fine-tuned on this dataset suggests that the underlying knowledge comes from pretraining rather than from examples of answering questions.

The dataset already includes patterns for responding to user greetings, questions, and instructions. However, the effectiveness of this dataset in improving model performance can vary depending on the base model used, further supporting the notion that true knowledge stems from pretraining rather than from examples of question-answering.

The OpenHermes 2.5 dataset is a comprehensive collection comprising over 1.6 million entries, designed to cover a wide range of conversational scenarios and specialized knowledge domains. Key components include the following (with rough estimations):

- Creative prompts from Airoboros 2.2 (44.8 K);
- Domain-specific expertise from CamelAI in MATH, Physics, Chemistry, and Biology (50 K, 20 K, 20 K, and 20 K, respectively);
- Real-world complex prompts from Chatbot Arena (7 K);
- Up-to-date data from Collective Cognition (as of 9 November 2023) (156);
- Chain-of-thought examples from the Alpaca dataset, produced by GPT-4 (52 K);
- Evolving complexity prompts from Evol Instruct (70 K and 140 K versions);
- Tool usage scenarios from Glaive Code Assistant (136 K);
- General-purpose conversational data from GPT4-LLM (54.6 K) and GPTeacher (35.5 K);
- Specialized task-oriented datasets like Medical Tasks (4.6 K) and MetaMath (40 K);
- Reasoning-focused entries from SlimOrca (550 K) and Platypus (24.9 K);
- Real conversation enhancements from ShareGPT (356 K);
- Versatility-boosting prompts from Unnatural Instructions (9 K).

This rich mixture of datasets aims to create a well-rounded foundation for the AI assistant, covering various aspects of knowledge, reasoning, and interaction styles. The diversity of sources enhances the model's ability to handle various queries and tasks effectively.

3.2.2. Penetration Testing Capability

Domain-specific knowledge enhancement has been extensively researched [27,42,43]. The most effective approach to improve domain knowledge is to enrich the corpus with domain-specific text. Our focus is on basic cybersecurity and penetration testing knowledge.

Penetration testing is a broad field that requires a strong foundation before identifying specific services or systems vulnerabilities. A penetration tester must understand how various services and systems operate and common vulnerabilities. While most public internet data provide basic knowledge of systems, which is reflected in current LLMs, the same is not true for vulnerability data and exploitation techniques. Due to safety concerns, most LLMs are biased towards protecting vulnerabilities or preventing exploitation techniques rather than suggesting how to exploit them.

To focus CIPHER's knowledge as a red-teamer, we scraped high-quality and frequently referenced hacking techniques, as shown in Table 3. This includes open public collections of OSCP notes and popular GitHub repositories. We incorporated compact documentation of popular tool command-line usage, using Cheatsheets [57] and TLDR [58] to improve general command-line argument knowledge. The Kali Tools [59] dataset strengthens the model's knowledge of penetration testing tools in Kali Linux, which we chose as the target environment for CIPHER. The Identity dataset augments CIPHER's self-awareness based on its purpose and specifications.

Table 3. Penetration testing essentials mixture.

Category	Content
Fundamental Knowledge	Cheatsheets [57], TLDR [58], Identity, Kali tools [59]
Pentesting Knowledge	OSCP Notes [60], OSCP Playbook [61]
Privilege Escalation	GTFOBins [62], LOLBAS [63]
Hacking Techniques	Hacktricks [64], PayloadAllTheThings [65]
Practical Knowledge	0xdf Hack The Box Write-ups [66]

We added advanced penetration testing knowledge from Gabbar OSCP notes [60] and OSCP Playbook [61], chosen for their structured and high-quality content. For privilege escalation, we included GTFOBins [62] (Linux) and LOLBAS [63] (Windows), considering misconfiguration as a common attack vector.

To deepen CIPHER's understanding of hacking and exploitation techniques, we incorporated the open-source Hacktricks [64] books, covering reconnaissance to undocumented exploitation techniques. We further enhanced exploitation skills by adding PayloadAllTheThings [65], which provides a rich collection of payload examples and usage instructions.

While fundamental knowledge, hacking techniques, and payloads can equip a medium-level penetration tester, achieving objectives with limited resources is the core challenge. This requires understanding situational context, prioritizing attack vectors, and developing intuition through experience. Such knowledge is rarely documented and is typically gained through practice.

To address this, we utilized a collection of high-quality write-ups from over 300 Hack The Box [67] machines, sourced from the 0xdf blog [66]. We selected these write-ups for their expert-level content and text-based format, which facilitates straightforward processing as a dataset. The write-ups provide detailed descriptions of visual elements, ensuring that the content remains comprehensive even without accompanying images.

While there are other collections, such as those from TryHackMe [68] or VulnHub [69], we found that the quality and quantity of the 0xdf blog's write-ups surpass these alternatives. One potential downside of this decision is the introduction of bias, either towards Hack The Box machines or the author's methodology. However, this is not a significant

issue, as Hack The Box is widely regarded as a leader in cybersecurity skills and training platforms. These are the currently best available data to use.

3.2.3. Augmentation

We process the dataset through several steps:

(1) Raw Context Dataset: For open-source datasets available in markdown format, no post-processing is needed. Web-formatted datasets are converted to markdown to preserve content structure. We split the datasets into chunks of 500, 1 K, and 2 K tokens, maintaining parent headers for nested sections to preserve context. This approach, providing the same information at different lengths, aids in better model generalization.

(2) Conversation Dataset: While raw context datasets expand domain-specific knowledge like DAPT [24], they do not fully prepare the model to answer and suggest like an expert. We augment and prioritize Hack The Box write-ups of vulnerable machines into conversations, simulating exchanges between novices and experts. While conversations do not add new knowledge, they help the model mimic expert answering styles, reducing mistakes and improving response quality [21].

(3) Conversation Generation Pipeline: We designed CIPHER to assist with penetration testing tasks, focusing on scenarios where a novice reports findings and situations and an expert provides suggestions, definitions, reasoning, and examples for the next steps. As seen in Figure 3, our pipeline generates conversations from 500-token chunks of raw text, ensuring focused discussions on specific topics instead of broader context.

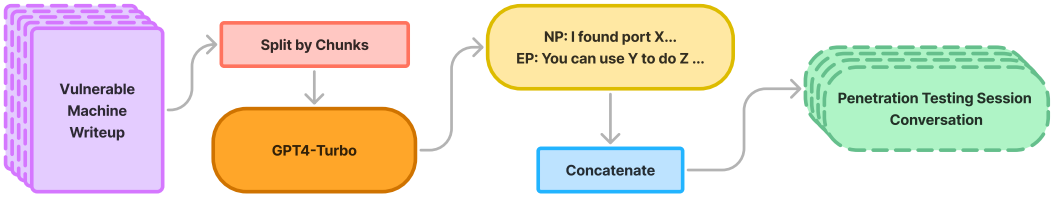


Figure 3. Each machine write-up is chunked into smaller pieces to extract multiple conversations, concatenated into a complete penetration testing session dialogue.

(4) Self-sufficient and Generalized Conversation: Table 4 shows our prompt for generating synthetic conversations. We emphasize **self-sufficient** and **generalized** conversations to create independent scenarios that convert write-ups into current situations, avoiding direct references to the source material.

Table 4. Prompt for conversation generation, with concise instructions where each word is crucial for generation quality.

Prompt Format
<500 tokens of write-up chunk> — Convert the write-up above into a self-sufficient generalized conversation without referring to this context. The conversation is a question from a novice pentester and a helpful answer from an expert pentester. The newbie always asking what to do next. The experts always provide reasoning explanations, then followed by examples. The conversation is multiple turns, step by step.

(5) Newbie and Expert Roles: We explicitly define Newbie and Expert roles to generate beginner-level questions and helpful, detailed expert responses.

(6) Next Step Prediction: We incorporate questions like "What should I do?" or "What is the next step?" to improve CIPHER's ability to suggest appropriate actions based on different scenarios.

(7) Helpful Response Structure: Expert responses in our dataset follow a structure of explanation, reasoning, and examples, aiming to develop the model's ability to provide comprehensive assistance.

(8) Multiple Turn Conversation: We generate and concatenate multiple-turn conversations to capture the dynamic nature of penetration testing scenarios, creating over 300 long conversations covering entire machine penetration testing sessions.

The real dataset sample in Figure 4 demonstrates how our generated conversations effectively capture real-world penetration testing experiences. The expert's response shows depth of knowledge and practical problem-solving, reflecting the expertise typically gained through extensive experience rather than textbook learning. Our conversation conversion method captures the expert problem solving skills such as that related to a WinRM firewall issue, converts it into users problems, then answers with Chisel, which is the preferred tool chosen by the expert.

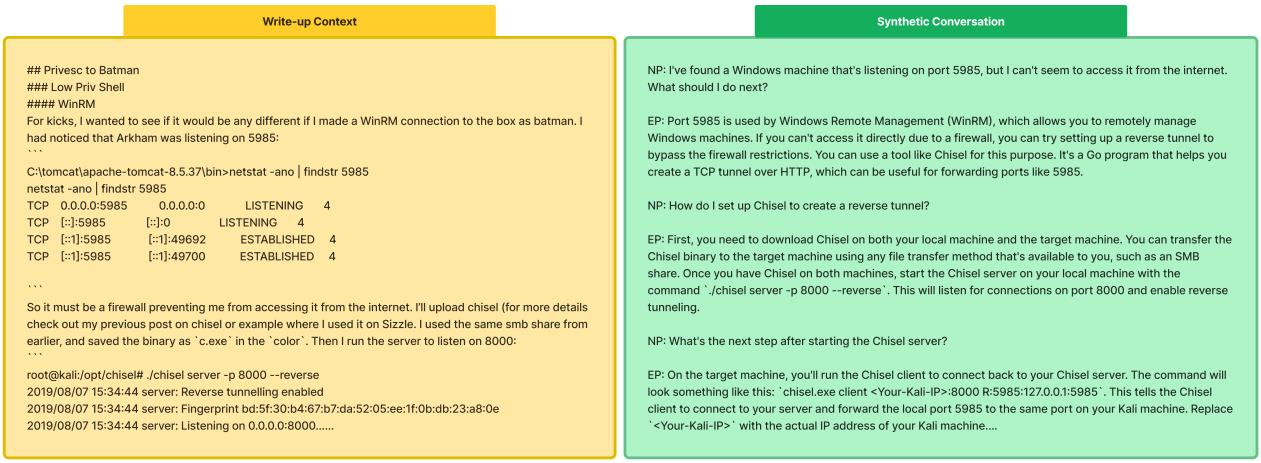


Figure 4. Synthetic conversation between Newbie Pentester (NP) and Expert Pentester (EP) generated from real write-up chunk. The expert's experience is reflected in the specific solution, demonstrating knowledge typically gained through practice rather than textbook learning.

3.2.4. Dataset Structure and Pre-Processing

CIPHER trained with two format of datasets. In Figure 5, for the conversation dataset, ChatML format is used, while for base knowledge, the raw context dataset is formatted as markdown with nested header information to prevent loss of information after chunking. All of the context datasets are scraped into markdown files and pre-processed by removing images to reduce hallucinations and token usage cost in training.

3.3. Training

This section details CIPHER's training process, datasets, and parameters, which leverage the Axolotl framework [47] for efficient and easily reproducible training capabilities, as shown in Figure 6.

3.3.1. Supervised Fine-Tuning

CIPHER's initial training phase uses Mistral v0.3 as a base model combined with the OpenHermes 2.5 dataset with a specialized pentesting dataset. The Axolotl framework manages tokenization and multipacking. We employ a standard text completion approach without specific raw text pretraining data formatting. The OpenHermes 2.5 and pentesting datasets utilize the ChatML format, supporting system, user, and assistant prompts. The pentesting dataset uses a default system prompt of `"You are a helpful penetration testing assistant."` while the general dataset's prompt remains unmodified. The combined dataset totals approximately 782 million tokens, comprising 500,606,758 tokens of general data and 281,516,236 tokens of supervised data.

Context

```
# 7z
## File read
It reads data from files, it may be used to do privileged
reads or disclose files outside a restricted file system.
```
LFILE=file_to_read
7z a -ttar -an -so $LFILE | 7z e -ttar -si -so

```
## Sudo
If the binary is allowed to run as superuser by `sudo`, it
does not drop the elevated privileges and may be used to
access the file system, escalate or maintain privileged
access.
```
LFILE=file_to_read
sudo 7z a -ttar -an -so $LFILE | 7z e -ttar -si -so

```…
```

Conversation

```
<|im_start|>system
You are a helpful penetration testing assistant
<|im_end|>
<|im_start|>user
How can 7z be used with sudo to access the file system?
<|im_end|>
<|im_start|>assistant
If the 7z binary is allowed to run as superuser by using
`sudo`, it can be used to access the file system. By
running the command with sudo privileges, the 7z command
does not drop the elevated privileges and can be used to
access, escalate, or maintain privileged access to the file
system.\n\n\Example:\n```\nLFILE=file_to_read\nsudo 7z a -
ttar -an -so $LFILE | 7z e -ttar -si -so\n```\nIn this
example, the \"file_to_read\" is the file that we want to
access. By running the command with sudo privileges, the 7z
command can access the file system and perform operations
on the specified file.
<|im_end|>
```

Figure 5. Two CIPHER training dataset structures: context and conversation type. Note that the '#' represents header in markdown format while the other is in ChatML format.

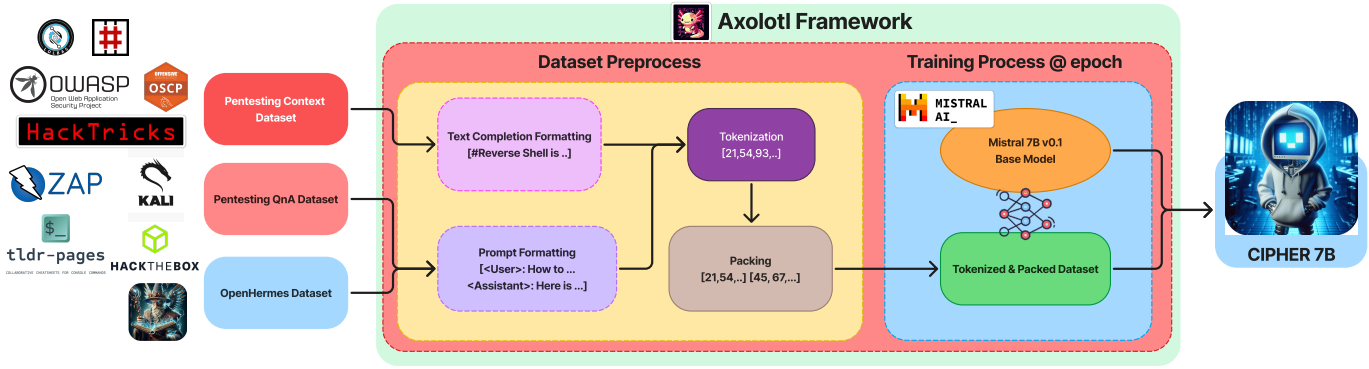


Figure 6. CIPHER training pipeline.

Fine-tuning is performed on an 8 × H100 compute cluster. Training the 782 million token dataset, combining pentesting and general data, requires approximately 12 h per epoch. Evaluation reveals optimal loss between epochs 5 and 7, approaching 0.1 loss. We train at a 32 K context length for maximum performance, utilizing a gradient accumulation size of 4 and a micro-batch size of 3. The adamw_bnb_8bit optimizer is employed with a cosine scheduler. Full training spans 10 epochs with a 2×10^{-5} learning rate without weight decay. Deepspeed is enabled with Zero3 CPU offload optimization.

3.3.2. Direct Preference Optimization

Following supervised fine-tuning, we implement Direct Preference Optimization (DPO) to refine model responses using the Argilla DPO Mix 7K [70] open-source dataset, which is already pre-formatted in ChatML. DPO training is conducted via the Axolotl framework and extends to 4 epochs, surpassing typical epoch counts due to consistently decreasing loss.

3.4. FARR Augmentation

Currently, evaluating LLMs for penetration testing faces significant challenges:

- No automated benchmarks: There are no established automatic penetration testing benchmarks for LLMs.

- Reliance on manual testing: The best available method involves human testers performing manual penetration testing while using an LLM chatbot for guidance.
- Time-consuming: This manual approach requires approximately 1–4 h per task per evaluator.
- Inconsistent results: Outcomes can vary significantly based on the evaluator's expertise, condition, and interpretation of the LLM's guidance.

These factors make it difficult to efficiently and consistently evaluate LLM performance in penetration testing scenarios, highlighting the need for more standardized and scalable assessment methods in this field.

To address the lack of automated benchmarks for evaluating LLMs in penetration testing, we have developed a novel approach:

- Benchmark Creation: We have designed a benchmark to measure the accuracy of an LLM's first suggestion in a penetration testing scenario.
- Data Augmentation: Unlike our previous synthetic conversation data generation method for supervised fine-tuning (SFT), this benchmark augments write-ups into compact, dense lists of information.
- FARR Penetration Testing Flow: We introduce a new method to augment penetration testing write-ups into a Findings, Action, Reasoning, Result (FARR) Flow. This structure reflects the typical phases of a penetration test, capturing the following: Findings: information discovered; Action: steps taken based on the findings; Reasoning: explanation for the chosen action; and Result: outcome of the action.
- Rationale: We observed that penetration testing write-ups consistently follow this ordered structure, providing a comprehensive view of the testing process and vulnerable points in a system.

This approach allows for a more standardized and detailed evaluation of LLM performance in penetration testing scenarios, addressing the limitations of manual evaluation methods.

The Findings, Action, Reasoning, Result (FARR) augmentation process systematically extracts the core vulnerabilities of a target machine from penetration testing write-ups. This structured method produces a FARR Flow, which provides a detailed overview of the penetration testing experience for a specific machine.

A FARR Flow is composed of multiple FARR Steps arranged sequentially. Each FARR Step contains a single set of FARR information, extracted from one specific section of the write-up. As illustrated in Figure 7, the FARR Flow Generation Pipeline transforms raw write-ups into a structured format that captures the essence of real-world penetration testing:

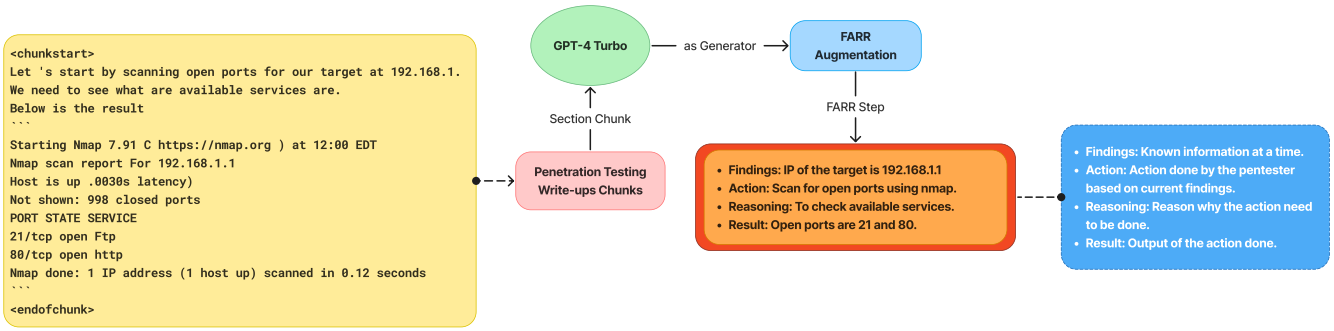


Figure 7. Findings, Action, Reasoning, and Result extraction from the real write-ups to construct the real penetration testing experience.

The FARR augmentation method offers versatility in evaluating different aspects of a model's knowledge:

1. Findings experience: Assessing the model's understanding of the required findings for a specific action.

2. Reasoning: Evaluating the model's ability to explain why an action is taken given certain findings.
3. Result prediction: Measuring the model's capability to predict the outcome of an action.

Figure 8 illustrates the augmentation process of transforming write-ups into FARR Steps, where each FARR component contains valuable information. Before the information is augmented, it is scraped and pre-processed into markdown format with nested headers to preserve context when split.

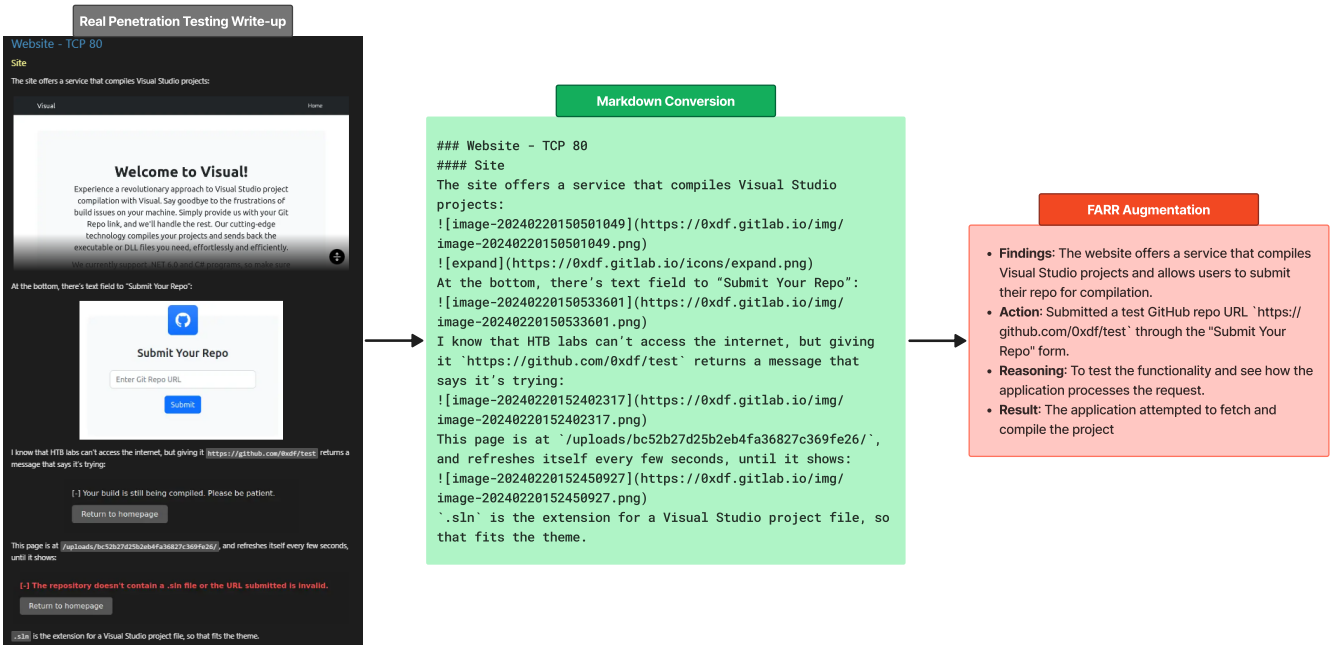


Figure 8. Process of FARR augmentation from write-up chunk into compact valuable information, FARR Step. This process is performed until the end of write-ups then combined together as FARR Flow.

Some information, such as images will be lost but due to the high quality of the write-up, which is very descriptive, the LLM model can still capture and understand the context clearly. Specifically, it recognizes that the current situation involves testing the input form using a fake URL to evaluate the response.

Since CIPHER is designed to assist penetration testers with accurate guidance, our evaluation specifically focuses on predicting the reasoning behind actions using known information. This approach allows us to assess how well the model understands and explains decision-making processes in real penetration testing scenarios. In Section 4.3, we use the FARR Flow to evaluate the model's comprehension of real penetration testing experiences, mainly its ability to process gathered information and determine appropriate next steps focused on reasoning. Please note that we do not include FARR Flow augmentation in CIPHER's training dataset to ensure the model's generalization in providing helpful penetration testing guidance and to allow for a fair comparison with existing models.

4. Experiment Results

In this evaluation, we assess CIPHER's effectiveness in providing accurate guidance for penetration testing. Our assessment consists of several key components:

- General Capabilities: We use the LLM Eval Harness benchmark to measure CIPHER's general task capabilities, examining how integrating general and penetration testing data affects performance.
- Cybersecurity Expertise: We evaluate the model's performance on cybersecurity-specific questions to gauge its domain knowledge.

- Pentesting Guidance Accuracy: Using our novel FARR Flow reasoning evaluation method, we assess CIPHER's ability to guide novice penetration testers through the testing process automatically.
- MITRE ATT&CK Understanding: We employ the PurpleLlama CyberSecEval [44] framework to measure CIPHER's comprehension of the MITRE ATT&CK knowledge base.
- Real-world Beginner Use Case Experiment: To analyze and evaluate the current capabilities of CIPHER in real-world usage assisting a beginner in penetration testing.

This multi-faceted approach allows us to comprehensively evaluate CIPHER's capabilities in providing accurate and effective penetration testing guidance.

4.1. Eval Harness General Capabilities

CIPHER's effectiveness in providing penetration testing guidance depends on maintaining coherent conversations. We assess its general language model capabilities using the EleutherAI LLM Eval Harness benchmark [71]. This section presents the top two CIPHER models, ranked by their average performance across various tasks in the benchmark.

The evaluation is conducted without n-shot prompting, using datasets that measure general-purpose skills (e.g., logical reasoning, commonsense reasoning, world knowledge) and cybersecurity topics. For logical reasoning, critical for context comprehension and step-by-step solutions, we use ARC [72], LogiQA [73], and parts of MMLU (Formal Logic and Logical Fallacies) [54]. Commonsense reasoning, essential for practical decision-making and ethical solutions, is evaluated with PiQA [74], SWAG [75], and HellaSwag [76]. For cybersecurity and complex reasoning, we use MMLU's computer science and security sections [54], along with the Adversarial NLI (ANLI) [77]. Lastly, OpenBookQA [78] evaluates common and subject-specific knowledge.

Table 5 presents model performance on logical reasoning tasks, a key factor for LLMs in producing relevant, well-structured output. A lack of proficiency in this area can result in disorganized or out-of-context content. Compared to OpenHermes 2.5, CIPHER shows a slight performance drop of 0.01 in ARC and 0.03 in LogiQA. However, this minimal degradation is acceptable given the large pentesting dataset, indicating no signs of overfitting. CIPHER continues to generalize well across non-pentesting topics. In addition to logical reasoning, common sense reasoning is essential, involving intuitive judgment about feasibility with support from logical reasoning. We also assess complex adversarial examples and world knowledge, which are both important for evaluating deductive reasoning and preventing incorrect conclusions or hallucinations due to limited knowledge.

Table 5. Model performance on logical reasoning.

Model	ARC Challenge	ARC Easy	LogiQA	Average Score
qwen1.5-7b-chat [79]	0.4352	0.6831	0.2995	0.4726
colibri-8b-v0.1 [80]	0.4838	0.7904	0.2565	0.5102
lily-cybersecurity-7b-v0.2 [81]	0.4974	0.7912	0.2903	0.5263
meta-llama-3-8b-instruct [82]	0.5265	0.8161	0.2719	0.5381
llama-3-whiterabbitneo-8b-v2.0 [83]	0.5247	0.8102	0.2873	0.5407
openhermes-2.5-mistral-7b [46]	0.5631	0.8329	0.2980	0.5647
hermes-2-pro-llama-3-8b [84]	0.5520	0.8342	0.3226	0.5696
mistral-7b-instruct-v0.3 [85]	0.5717	0.8418	0.3195	0.5776
CIPHER 7B DPO (Ours)	0.4838	0.7652	0.2642	0.5044
CIPHER 7B (Ours)	0.5503	0.8232	0.2688	0.5475

In Table 6, CIPHER models are competitive in common sense reasoning benchmarks, trailing the top model in PiQA by only 1.5%, where OpenHermes 2.5 leads with a score of 0.8156. CIPHER, however, excels in ANLI, with cipher-mistralv0.2-chio-32k-v1.6 achieving the highest score (0.5292), showcasing strong logical reasoning and adversarial handling.

OpenHermes models lead in both logical and common sense reasoning benchmarks, though CIPHER's lower score in OpenBookQA (0.38 by Hermes-2-Pro-Llama-3-8B) may reflect a trade-off between its penetration testing focus and general knowledge.

Table 6. Model performance on common sense reasoning, general logic, and world knowledge.

Model	PiQA	Swag	Hellaswag	OpenBook QA	ANLI Average	Average Score
Lily-Cybersecurity-7B-v0.2 [81]	0.7807	0.5610	0.6095	0.3280	0.3955	0.5350
Llama-3-Whiterabbitneo-8B-v2.0 [83]	0.7938	0.5794	0.6093	0.3340	0.3846	0.5402
Meta-Llama-3-8B-Instruct [82]	0.7856	0.5700	0.5773	0.3440	0.4653	0.5484
Qwen1.5-7b-Chat [79]	0.7546	0.5776	0.5881	0.3220	0.5026	0.5490
Colibri-8B-v0.1 [80]	0.7960	0.5866	0.5942	0.3640	0.4676	0.5617
Mistral-7B-Instruct-v0.3 [85]	0.8177	0.5867	0.6475	0.3540	0.4560	0.5724
Hermes-2-Pro-Llama-3-8B [84]	0.8003	0.6026	0.6266	0.3800	0.4776	0.5774
OpenHermes-2.5-Mistral-7B [46]	0.8156	0.5935	0.6310	0.3460	0.5116	0.5795
CIPHER 7B DPO (Ours)	0.7753	0.5682	0.5922	0.2920	0.4903	0.5436
CIPHER 7B (Ours)	0.7965	0.5935	0.6288	0.3260	0.5194	0.5728

In summary, CIPHER models excel in logical reasoning but show a slight trade-off in general knowledge, likely due to their specialization. Future efforts should aim to balance domain-specific expertise with broader knowledge, with continued evaluation on specialized topics like computer science and security.

Table 7 shows the results for specialized MMLU topics in cybersecurity. Despite being trained on penetration testing datasets, CIPHER underperforms compared to base models on general tasks, suggesting that while optimized for penetration testing, it may be less effective on broader MMLU tasks.

Qwen1.5-7B-Chat excels in high school and college computer science, while Llama-3-WhiteRabbitNeo-8B-v2.0 leads in computer security. CIPHER, though not the top performer, is consistent across categories, blending general competency with specialized penetration testing knowledge, meeting LLM assistant standards.

Table 7. Specialized MMLU for cyber security-related topics.

Model	High School CompSci	Computer Security	College CompSci	Formal Logic	Logical Fallacies	Average Score
colibri-8B-v0.1 [80]	0.5900	0.6800	0.4200	0.4127	0.7055	0.5616
lily-cybersecurity-7B-v0.2 [81]	0.5800	0.6800	0.5600	0.3333	0.7055	0.5718
openhermes-2.5-Mistral-7b [46]	0.7100	0.7200	0.4500	0.4048	0.7607	0.6091
mistral-7B-Instruct-v0.3 [85]	0.6200	0.6700	0.5400	0.4524	0.7730	0.6111
hermes-2-pro-llama-3-8B [84]	0.6700	0.7100	0.4700	0.4762	0.7669	0.6186
qwen1.5-7B-Chat [79]	0.7200	0.7600	0.6100	0.4127	0.6748	0.6355
llama-3-whiterabbitneo-8B-v2.0 [83]	0.6700	0.7800	0.5400	0.4762	0.7239	0.6380
meta-llama-3-8B-Instruct [82]	0.6800	0.7300	0.5200	0.5000	0.7607	0.6381
CIPHER 7B DPO (Ours)	0.6600	0.6800	0.4800	0.3651	0.6933	0.5757
CIPHER 7B (Ours)	0.6100	0.7300	0.5500	0.3651	0.7301	0.5970

4.2. Cybersecurity Knowledge Evaluation

In this section, we conduct a two-part evaluation of CIPHER:

- Pentesting QA Evaluation: CIPHER is developed to bridge the gap between novice and expert. Therefore, the capability to explain penetration testing knowledge is very crucial. We utilize the pentesting evaluation topics from the `preemware/pentesting-eval` Huggingface repository [86]. This dataset consists of question and explanation pairs from GPT-4. We use the questions as prompts and the explanations as ground truth. Model responses are then compared to the ground truth and assessed by the Qwen1.5 72B Chat model to produce a score indicating alignment with the ground truth.

- Comparative Analysis: We compare the results from the LM evaluation harness to identify any performance degradation in general tasks after training various CIPHER models with pentesting datasets.

As shown in Table 8, CIPHER models achieve the highest average score of around 83%, outperforming other LLMs in pentesting-specific evaluations. While models like OpenHermes 2.5 excel in general MMLU tasks, they fall short in comparison to CIPHER on specialized pentesting explanation tasks. Similarly, models like Lily Cybersecurity and WhiteRabbitNeo perform poorly, likely due to overfitting or a decline in explanatory capabilities.

Table 8. Pentesting QA evaluation.

Model	Average
lily-cybersecurity-7B-v0.2 [81]	0.5688
llama-3-whiterabbitneo-8B-v2.0 [83]	0.5759
colibri-8b-v0.1 [80]	0.5887
hermes-2-pro-llama-3-8B [84]	0.6626
qwen1.5-7B-chat [79]	0.6846
meta-llama-3-8B-instruct [82]	0.7008
openhermes-2.5-Mistral-7B [46]	0.7157
mistral-7B-Instruct-v0.3 [85]	0.7896
CIPHER 7B (Ours)	0.8302
CIPHER DPO 7B (Ours)	0.8394

No other cybersecurity model approaches CIPHER's performance. The substantial gap highlights CIPHER's effectiveness in pentesting tasks, despite trade-offs in general knowledge areas seen in previous benchmarks.

The general task performance (Figure 9) shows CIPHER excelling in pentesting tasks while maintaining competitive general task performance. It outperforms cybersecurity-focused models like WhiteRabbitNeo and Lily, which show the lowest results in both general and pentesting-specific tasks. This highlights CIPHER's effective training approach, enhancing pentesting capabilities without sacrificing general knowledge.

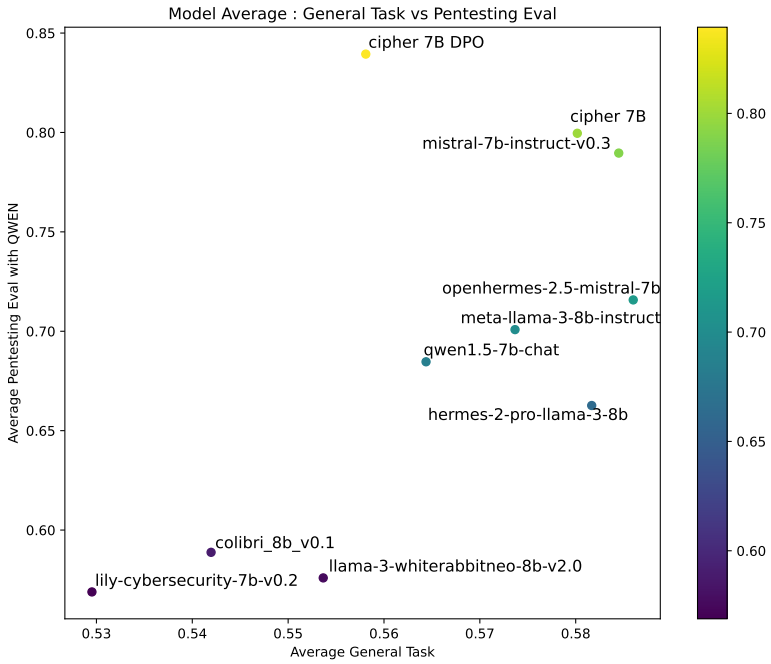


Figure 9. Comparison of models on General Task and Pentesting Evaluation: The plot illustrates the performance of various models, comparing their average scores. Higher placement on both axes indicates better performance across both domains.

The CIPHER DPO model performs best in pentesting but scores lower on general tasks, indicating possible overfitting toward specialized knowledge. This trade-off suggests that further optimization could balance general and pentesting performance more effectively. Despite this, CIPHER consistently outperforms other models in both domains, especially in pentesting, positioning it as a strong pentesting assistant.

In summary, CIPHER non-DPO offers a good balance between general and specialized knowledge, while the DPO version excels in pentesting but requires improvements in general task performance.

4.3. Penetration Testing Guidance Accuracy Evaluation (FARR Flow Reasoning)

The general and cybersecurity knowledge question benchmarks alone are not enough to measure how good the model is in guiding the user in the penetration testing process. We formulated a realistic situation combining much information and a dynamic situation obtained from a real penetration testing process.

In Section 3.4, after the augmentation process and obtaining the full flow of the machine penetration testing steps augmented as FARR Flow, we constructed a guidance reasoning test for the model by asking it what to do next when we have only the Findings. FARR Flow consists of multiple sets of Findings, Actions, Reasonings, and Results, as shown in Figure 10, where F is Finding, A is Action, R is Reasoning, and the last R is Result.

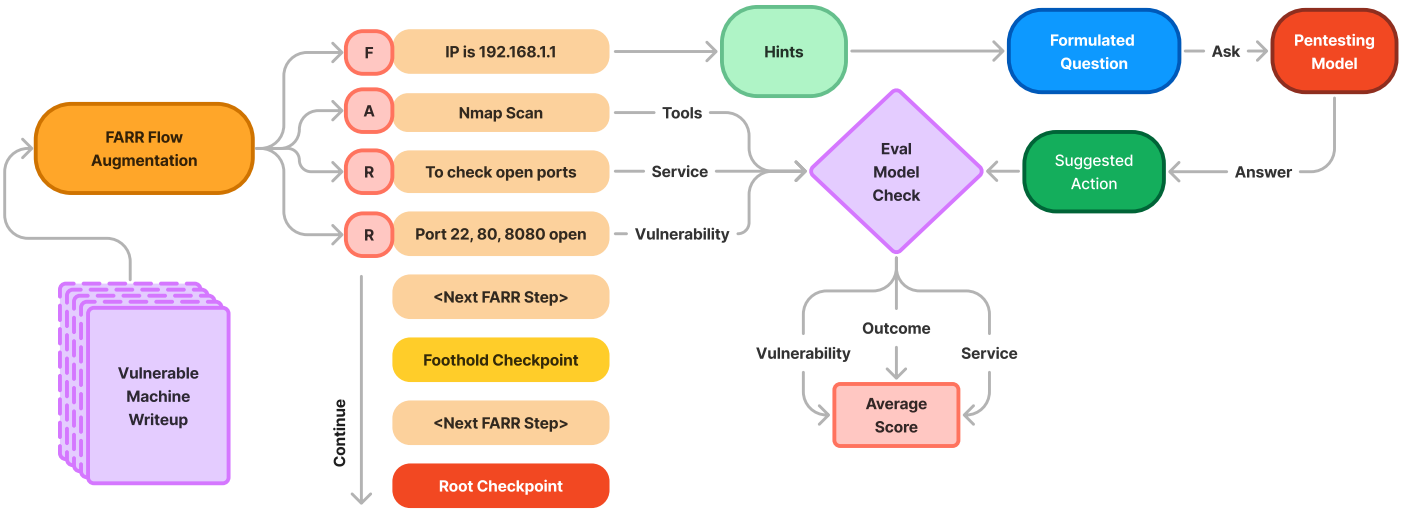


Figure 10. FARR Flow evaluation.

Figure 10 also illustrates how we evaluate the model using FARR Flow. At each step, we utilize the Findings as known information. We formulate questions for the model to predict the most likely action to do based on the current hint, which is constructed from the accumulated previous and current findings. The model's suggested response is then evaluated against Llama 3 70B to assess the relevance of the answer across three aspects: whether the model's answer anticipates the same outcome, targets the same service, and focuses on the same vulnerability type as the reference. Each criterion is rated with up to three points for relevance. Using these three aspects, we measure the average performance of each model on each Hack The Box machine in the FARR Flow evaluation.

Our guidance reasoning test using the FARR Flow evaluation algorithm is shown in Algorithm 1. At the first question, the user does not know about the target machine, often only the IP information or open ports. Therefore, we only provide the current findings. However, regardless of the suggested answer to the following question, we will still provide the model with the result of the previous findings to ensure the model understands the current situation. Section 5.3 discusses the possibility of improvement in this question formulation.

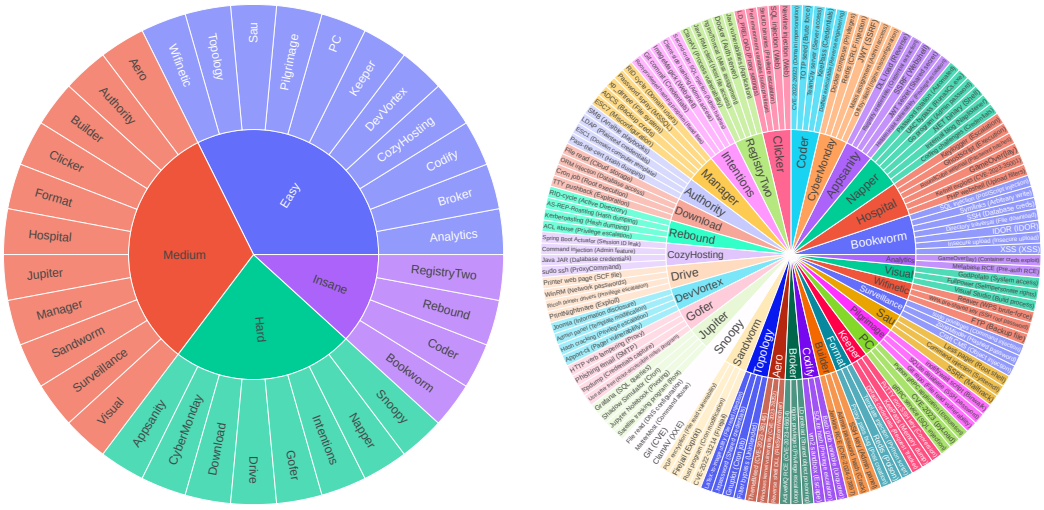
Algorithm 1 Formulating FARR Step Question

Input: JSON file containing a list of findings, actions, reasonings, and results (`Machine_X_FARR_flow.json`)
Output: Suggested next action from inference model

```
Load Machine_X_FARR_flow.json
Parse JSON content into FARRflow list of dictionaries
Initialize all_findings as an empty string
for each flow in FARRflow do
    current_findings ← flow['Findings']
    current_result ← flow['Result']
    question_prompt ← "Below are my current findings:
    all_findings
    current_findings
    What is the most likely action to do next? Answer with one specific action only, not
    more than that."
    model_output ← inference_model(question_prompt)
    all_findings ← all_findings + "
    current_findings, current_result"
end for
Return model_output as the model suggested action
```

CIPHER training data use the 0xdf write-up [66] dataset until early September 2023. We took whole machine write-up samples from mid-September 2023 until the FARR Flow benchmark development started in early May 2024 without cherry-picking. We only excluded incomplete machine write-ups and separated them, resulting in 35 machine write-ups augmented as FARR Flow for the reasoning test, with different difficulties as shown in Figure 11a. Note that these machines are completely unknown to CIPHER. Overall, there are 35 Hack The Box machines with 136 vulnerability topics, all in 2124 formulated questions.

Figure 11b presents different attack vectors in this evaluation. The scope includes diverse vulnerabilities and techniques used to penetrate the machine. Each machine has a different attack vector to achieve a foothold from easy to insane level. Various privilege escalation techniques are needed to achieve better scores, and CVE knowledge is necessary to access some machines. This covers better coverage and penetration testing dynamics than traditional question-and-answer evaluation.



(a) **(b)**

Figure 11. Hack The Box machines: (**a**) Hack The Box machines selected for challenges sorted by difficulties, (**b**) Hack The Box machines selected for evaluation and corresponding attack vector scope.

Figure 12 shows real example of a question augmented from FARR Flow from a Hack The Box Keeper machine, where it contains a list of findings and results to provide a situation of the user's current penetration testing progress. Based on the question, the last situation is the current findings in the current FARR step; the model needs to understand the current progress first and then suggest the most likely action to be taken next. We ask for the most likely action to extract the best action suggested from the model to measure the accuracy of the vulnerability points, the service targeted, and the expected outcome from the action.

The prompt question in Figure 12 provides valuable information about the current status of the penetration testing progress, including open ports, service versions, subdomains, running instances, credentials, and both failed and successful attempts. These details are typically noted by penetration testers. For beginners in penetration testing, however, this information can be overwhelming, making it difficult to determine the appropriate next steps. In real-world scenarios, beginners often turn to forums to ask experts or more experienced penetration testers for hints, or what is commonly referred to as a "nudge" in the community.

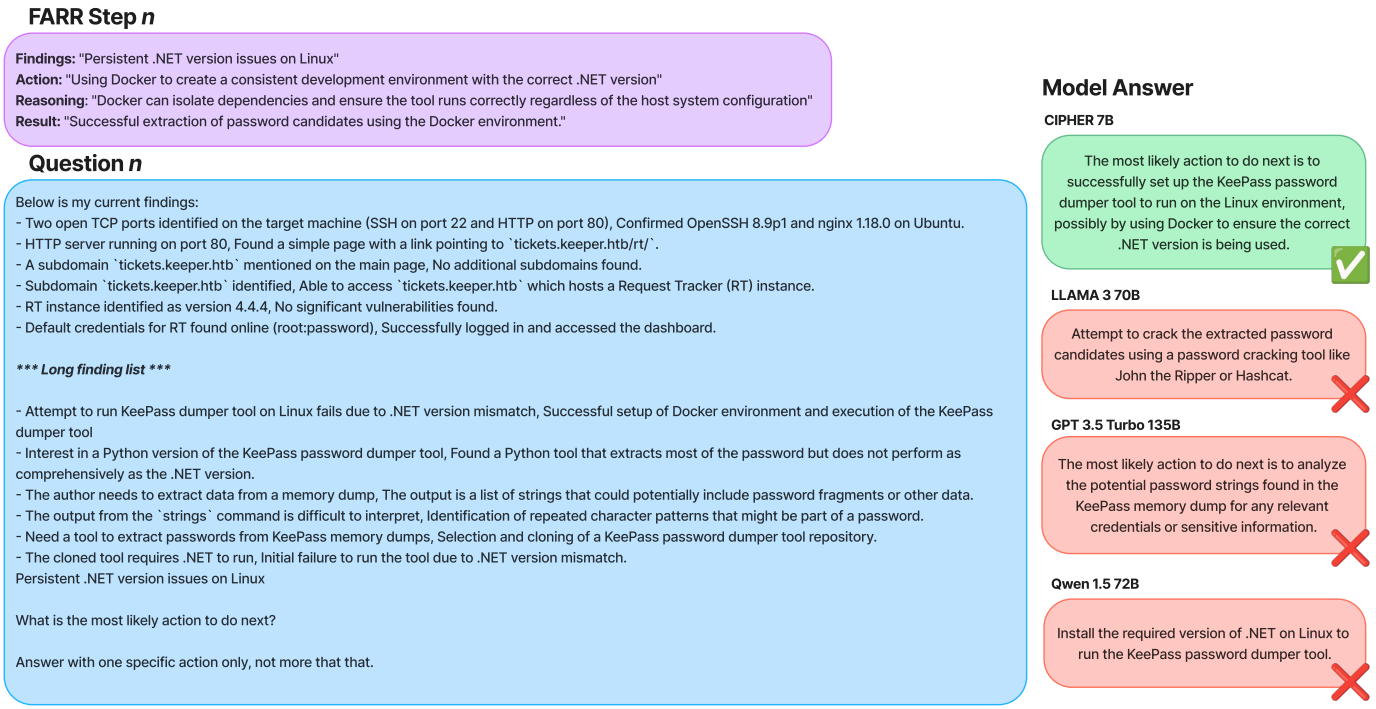


Figure 12. Real FARR Flow augmented question from Hack The Box Keeper machine. Notes (***: the rest of middle part findings.)

This is where CIPHER bridges the gap. According to the write-ups augmented in FARR Step Figure 12, the current finding is "*Persistent .NET version issues on Linux*," and the correct action is to "*Use Docker to create a consistent development environment with the correct .NET version*." However, all three of the other models failed to suggest this. Llama 3 70B, for instance, suggested attempting to extract a password, which is irrelevant. GPT-3.5 Turbo proposed analyzing a memory dump, which does not make sense since it was already stated that the tool could not run. Qwen 1.5 72B suggested installing .NET on Linux, which is also incorrect because the question clearly states that there is a persistent issue with .NET on Linux. CIPHER, however, responded correctly and aligned with the reference by suggesting that the user run the tool in Docker with the correct .NET tools. When provided with complex information from the penetration testing process, CIPHER was able to accurately determine the next step, outperforming even models with ten times the number of parameters, such as Llama 3 70B, GPT-3.5 Turbo, and Qwen 1.5 72B.

The most relevant models are selected for this evaluation, including available small pentesting or cybersecurity models, and we also included the best open-source state-of-the-art (SOTA) model as the judge: Llama 3 70B. To ensure our benchmark is compatible with larger parameter models, we included the Llama 3 70B and Qwen 1.5 72B Chat scores in our benchmark. This allows us to assess how well our model's guidance compares to larger models out of CIPHER's size.

4.3.1. Penetration Testing Guidance Accuracy Performance

Table 9 shows the three criteria measured in our FARR Flow evaluation for each model tested. Our model CIPHER achieves the highest average score compared to other penetration testing models and large SOTA general models like Llama 3 70b and Qwen1.5 72B. Llama 3 70B scores slightly higher (by 0.19) in outcome prediction, but CIPHER guides better in service aimed and vulnerability targeted, with a 0.52 higher score overall at 52.59. However, compared to other penetration testing models similar to CIPHER's size, there is a significant increase in performance, as big as 9.75% from Colibri 0.1 8B.

Since the judge LLM used was Llama 3 70B, there might be some bias in the result causing Llama 3 to score higher on its answer. Conversely, Qwen1.5 72B achieves a score similar to GPT3.5, which seems more accurate and free from judgment bias.

Table 9. Model performance on FARR Flow reasoning test.

Model	Outcome	Service	Vulnerability	Total Avg
whiterabbitneo-7b [83]	25.81	38.78	27.49	30.69
lily-7b [81]	29.35	43.77	35.57	36.23
qwen-7b [79]	36.50	51.18	40.11	42.60
colibri-0.1-8b [80]	40.85	56.61	46.30	47.92
hermes2pro-llama3-8b [84]	41.27	58.13	47.15	48.85
mistral-ins-0.2-7b [85]	41.36	59.91	46.60	49.29
qwen-72b [87]	43.57	59.05	46.84	49.82
gpt-3.5-turbo [26]	43.57	59.31	47.45	50.11
llama-3-70b-ins [82]	44.92	61.67	49.62	52.07
CIPHER DPO 7B (Ours)	44.73	62.65	50.39	52.59

Smaller models with around 7B parameters score below 49.30, led by Mistral Instruct v0.2 7B, which excels at general instruction. Hermes2pro-llama3-8b, known for its agentic and tool usage prowess, still cannot achieve good guidance on penetration testing. Colibri 7B achieves 47.92 overall; despite its strength in QnA evaluation, it fails to correctly guide 50 percent of the questions, except for aiming at the correct service (56.61).

Qwen 1.5 7B only achieves 42.60, often providing defensive responses instead of offensive ones. The performance gap between Qwen1.5 72B and 7B suggests that model size scaling can affect the model's benign level. Based on our experiments, Qwen 7B tends to give defensive solutions when asked about the next steps.

Other open-source penetration testing cybersecurity models perform poorly. Lily 7B, based on Mistral Instruct fine-tuning, provides poor guidance at 36.23, followed by Whiterabbitneo 7B at 30.69. Whiterabbitneo's poor performance is due to its bias towards answering everything with code and often producing incoherent responses.

In summary, most penetration testing models struggle to handle the large volume of information and often suggest inaccurate next steps. However, state-of-the-art models like Llama 3 (70B) achieve the highest scores in outcome prediction. Despite this, our CIPHER DPO (7B) model outperforms even the largest models in terms of suggestion accuracy. This means that CIPHER guides users more effectively, enabling them to penetrate vulnerable machines faster, achieve a foothold, and gain the highest privilege accounts more efficiently than models ten times its size.

4.3.2. Performance by Machine Difficulties

To evaluate the model's performance across different complexity levels, we clustered the scores based on machine difficulties. This allows us to identify the areas where the model performs better or worse depending on the complexity. Table 10 shows that our model CIPHER performs better on average due to its distinguished high scores on easy and insane machines, with 56.25 and 50.96 points, respectively, creating the highest gaps of about 1.69 and 2.22 points from the second-place model. Llama 3 70B takes second place, with slightly better scores on medium and hard machines by 0.52 and 0.43 points. This demonstrates our model's strength in penetration testing reasoning guidance compared to other penetration testing models, even those with ten times the parameter size.

In the detailed results for easy-difficulty machines shown in Figure 13, the accuracy scores of various models on easy-difficulty Hack The Box vulnerable machines are compared. CIPHER dominates, outperforming Llama 3 70B with an accuracy of 56.25. This is attributed to CIPHER's extensive training with vulnerable machines in its dataset, which enhances its ability to generalize in these easier challenges, typically involving single-step problem-solving or a single vulnerability.

Table 10. Model performance FARR Flow reasoning test sorted by machine difficulty scores.

Model	Easy	Medium	Hard	Insane	Average
whiterabbitneo-7b [83]	33.75	30.01	28.47	28.75	30.25
lily-7b [81]	40.73	34.66	34.50	32.03	35.48
qwen1.5-7b [79]	44.68	41.31	42.02	41.88	42.47
colibri-0.1-8b [80]	49.74	47.38	45.82	48.75	47.92
hermes2pro-llama3-8b [84]	50.43	47.76	48.54	48.37	48.78
mistral-ins-0.2-7b [85]	51.21	49.34	47.27	47.89	48.93
qwen1.5-72b [87]	52.02	47.81	50.88	47.67	49.60
gpt-3.5-turbo [26]	53.41	48.18	49.67	47.72	49.74
openhermes-2.5-7b [46]	53.05	48.51	50.27	48.50	50.08
llama-3-70b-ins [82]	54.56	51.28	51.48	48.74	51.52
CIPHER DPO 7B (Ours)	56.25	50.80	51.05	50.96	52.26

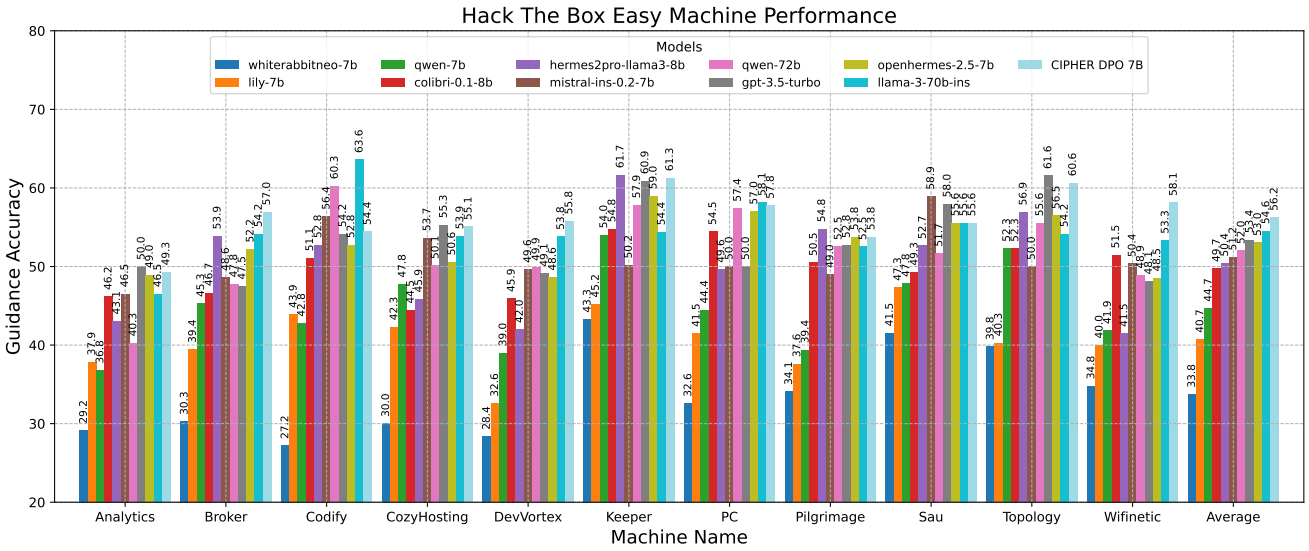


Figure 13. Easy machine performance results.

Following this scenario, in the medium machine challenge shown in Figure 14, CIPHER experienced a slight performance drop compared to Llama 3 70B. However, despite this small difference, CIPHER still achieved the highest score, creating a notable gap compared to models with 7B parameters. This suggests that CIPHER provides better guidance

than the smaller models. It is important to note that Llama 3 70B's high score primarily serves as a benchmark to gauge CIPHER's competitive ability against larger parameter models. Additionally, since Llama 3 70B is used as the evaluation model, its own score may be influenced by bias.

This phenomenon is also observed in the hard machine challenge in Figure 15, where CIPHER achieves the best score among the 7B models but still falls below the average performance of Llama 3 70B. We believe that, in addition to potential bias, there may be inherent limitations in the 7B model's ability to generalize reasoning and provide accurate guidance. These limitations could potentially be mitigated through dataset scaling and increasing the model size.

Moving on to the insane difficulty machine's case in Figure 16, multiple pieces of information must be combined and a chained exploitation plan is required. Surprisingly, CIPHER leads the benchmark with a 2-point gap over all other models. This demonstrates CIPHER's significant penetration testing guidance capabilities in extremely challenging scenarios that demand out-of-the-box exploitation, where even higher-parameter models reach their limits.

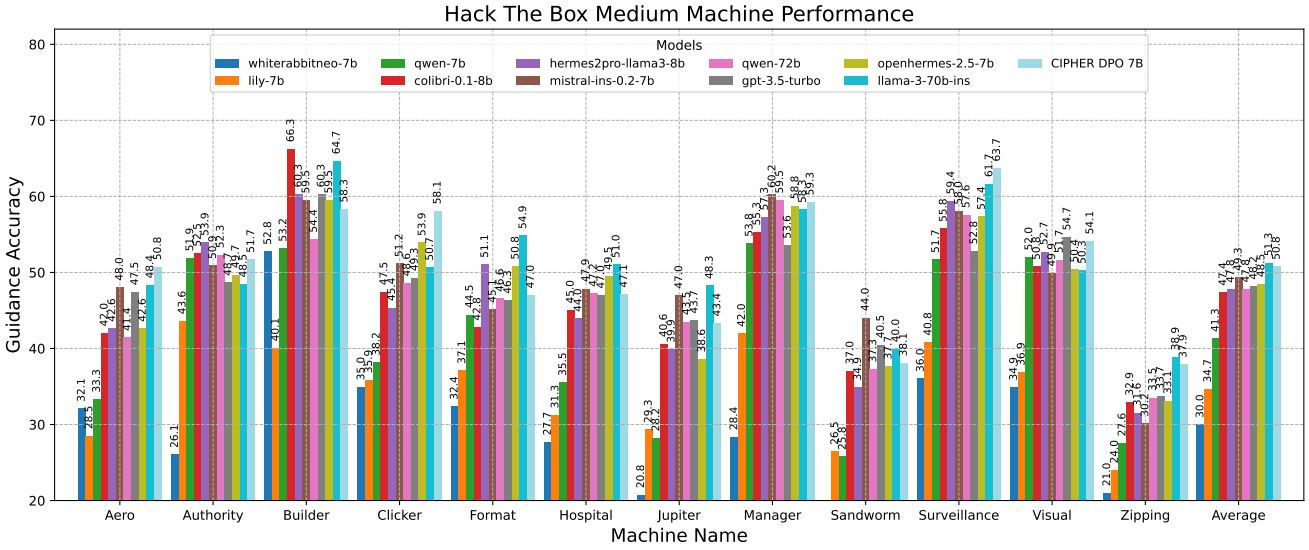


Figure 14. Medium machine performance results.

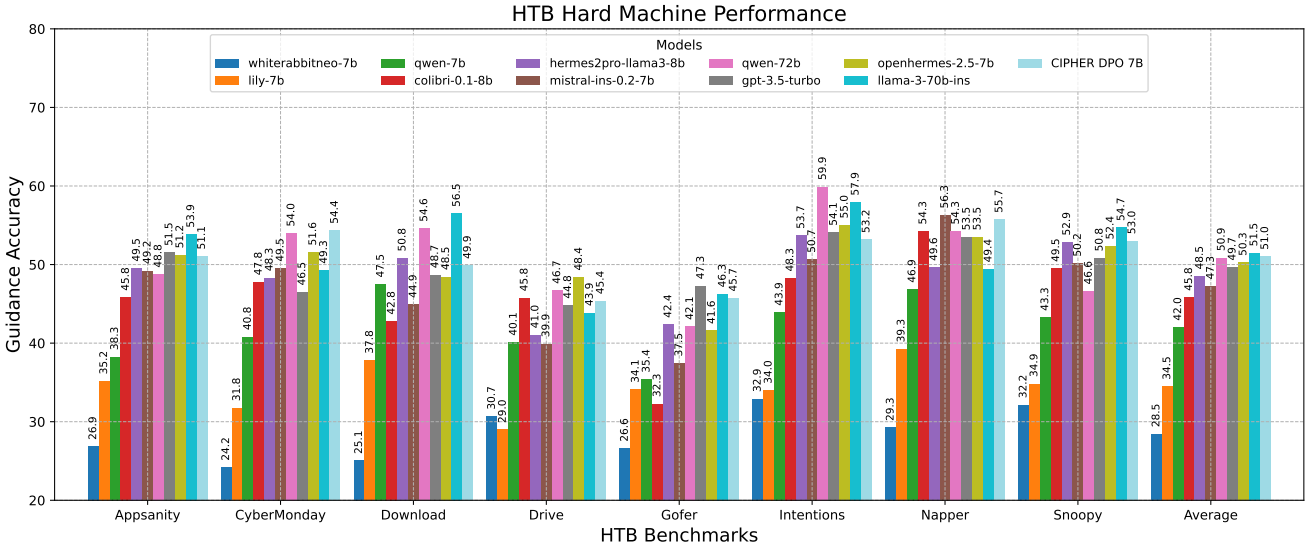


Figure 15. Hard machine performance results.

This experiment proves that penetration testing intuition remains a challenge that requires experience rather than just vast knowledge already present in large parameter LLMs. By training CIPHER with a penetration testing-focused guidance dataset, CIPHER can guide users more accurately on unseen vulnerable machines than other specific cybersecurity/penetration testing models and even larger parameter general models.

The results highlight CIPHER's effectiveness across different difficulty levels, particularly excelling in easy and insane categories. This balanced performance across varying complexities underscores the model's versatility and robustness in penetration testing scenarios. The consistent superiority over models with significantly larger parameter counts emphasizes the importance of specialized training and domain-specific knowledge in achieving high-quality results in targeted tasks like penetration testing guidance.

Overall, CIPHER achieved the highest total average score, demonstrating superior performance in both the easy and insane tasks. It also excelled in the medium and hard tasks compared to other pentesting models of its size, placing second only to Llama 3 70B. This may be due to the fact that Llama 3 70B was also the evaluator, and its larger parameter size could have contributed to its advantage. Additionally, we discuss potential improvements for CIPHER in Section 5.2.

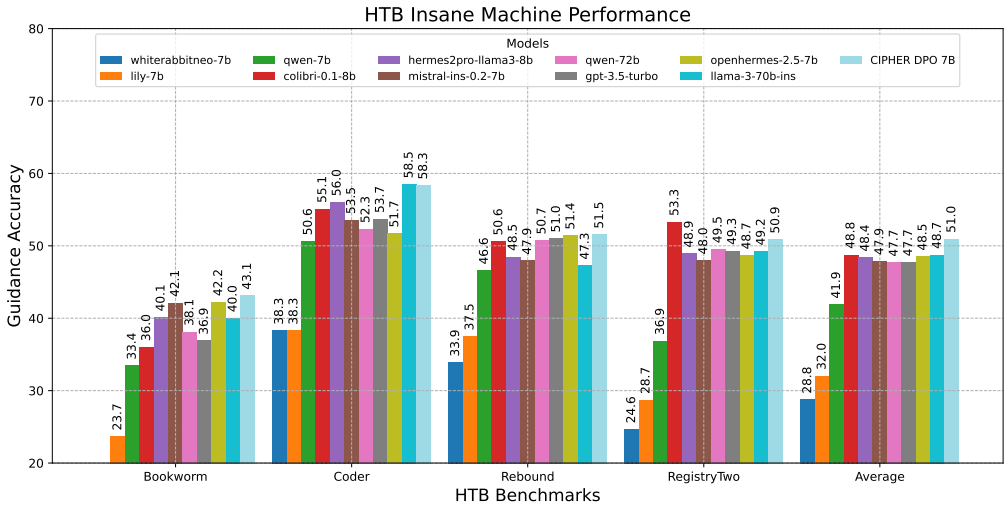


Figure 16. Insane machine performance results.

4.4. PurpleLlama CyberSecEval

We also evaluate our model using the most popular cyber security evaluation for LLMs, PurpleLlama CybersecEval2 [44]. This evaluation was initially designed to measure the safety of the LLM model before being released. Since we are developing a model that accurately suggests offensive exploitation techniques, we can also measure how well our model is capable of attacks listed in MITRE ATT&CK. The value that we show in Figure 17 is the malicious score, unlike the original paper that showcases the benign score.

CIPHER is based on the Mistral model and leverages the OpenHermes 2.5 dataset for enhanced general knowledge and conversational capabilities. Our synthetic dataset significantly bolsters CIPHER's proficiency in the MITRE ATT&CK framework, particularly in major categories such as command and control, collection, discovery, evasion, and execution. Given our meticulously curated privilege escalation data, it was anticipated that CIPHER would outperform in persistence and privilege escalation tasks, areas not extensively documented online, which accounts for the subpar performance of other general models. Additionally, the impressive scores of CIPHER are partly because other models incorporate safety measures that limit their ability to engage in offensive activities. In contrast, CIPHER includes these offensive capabilities to ensure its effectiveness in assisting with penetration testing processes, where such skills are critical.

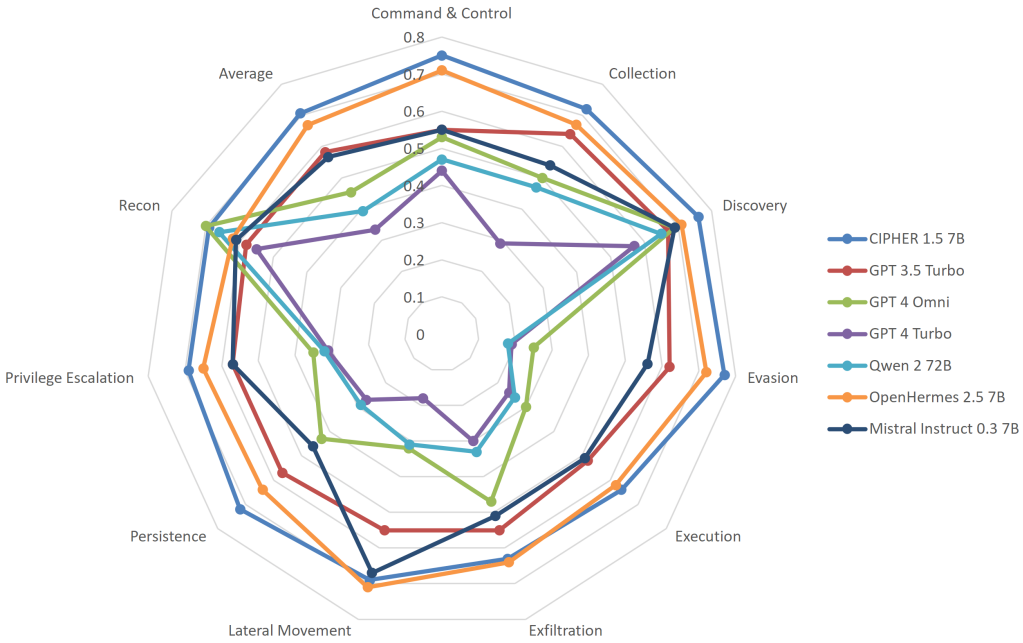


Figure 17. MITRE ATT&ACK capabilities results.

In contrast, we observe that the performance of general models tends to peak in reconnaissance and discovery techniques, even outperforming CIPHER in these areas. This is because reconnaissance and service discovery are common topics that are not considered dangerous. However, using only high-quality augmented write-ups and raw context-hacking techniques material, CIPHER enhances the MITRE ATT&CK capabilities of its base model and base conversation dataset. While this benchmark does not represent the model's full performance, it showcases its extensive knowledge coverage across different attack techniques.

4.5. Real-World Beginner Use Case Experiment

In this experiment, we explore and analyze how a beginner penetration tester interacts with CIPHER to carry out common tasks such as reconnaissance, vulnerability discovery, and privilege escalation. The goal of this experiment is to evaluate the effectiveness of CIPHER in guiding new penetration testers through the initial stages of a penetration test, while also assessing how well the LLM adapts to novice-level queries and provides appropriate responses. Even though CIPHER has demonstrated strong performance in FARR Flow, this experiment aims to further validate its practical usability in real-world penetration testing scenarios, ensuring that it can handle practical questions effectively.

4.5.1. Reconnaissance

We begin the experiment with a typical beginner's interaction with CIPHER during a reconnaissance phase. The user is prompted to ask for guidance on the first step in assessing the target system's open ports. Figure 18 demonstrates CIPHER's suggestion of using Nmap, a popular network scanning tool, to enumerate open ports on the target system. The model provides the appropriate command syntax `nmap -p- -min-rate` and also explains its relevance to the overall penetration testing process, emphasizing the need to identify all active services and open ports quickly for further exploitation without saving to any file.

The next scenario is involved with domain fuzzing, as shown in Figure 19. Providing an actionable guide for using `ffuf` in subdomain fuzzing, CIPHER effectively breaks down each component of the command. It explains common parameters like `-u` for the target URL, `-H` for adding custom headers, and `-w` for specifying wordlists, showcasing a solid

understanding of both the tool and its practical applications. The use of placeholders such as `FUZZ` in the Host header is a standard fuzzing technique, and the inclusion of `-fw 18` and `-mc all` shows a keen awareness of optimizing results and capturing a wide range of HTTP responses.

The response is highly effective for both novice and intermediate users, balancing simplicity with technical detail. CIPHER's guidance is structured well, leaving room for customization, and demonstrates proficiency in subdomain enumeration. It could be enhanced with tips on handling challenges like rate-limiting and optimizing wordlists, but overall, it successfully conveys core principles and best practices as a penetration testing assistant.

In the next scenario, the user reports discovering port 5985 open on the target system and asks for advice on utilizing a long list of usernames and passwords. Figure 20 shows CIPHER's response, providing a concise explanation of port 5985 being associated with Windows Remote Management (WinRM). CIPHER recommends using `crackmapexec` to automate testing combinations of usernames and passwords against the WinRM service, guiding the user with clear command syntax: `crackmapexec winrm <target-ip> -d <domain-name> -u <username-list> -p <password-list>` and explaining each element's role in the process.

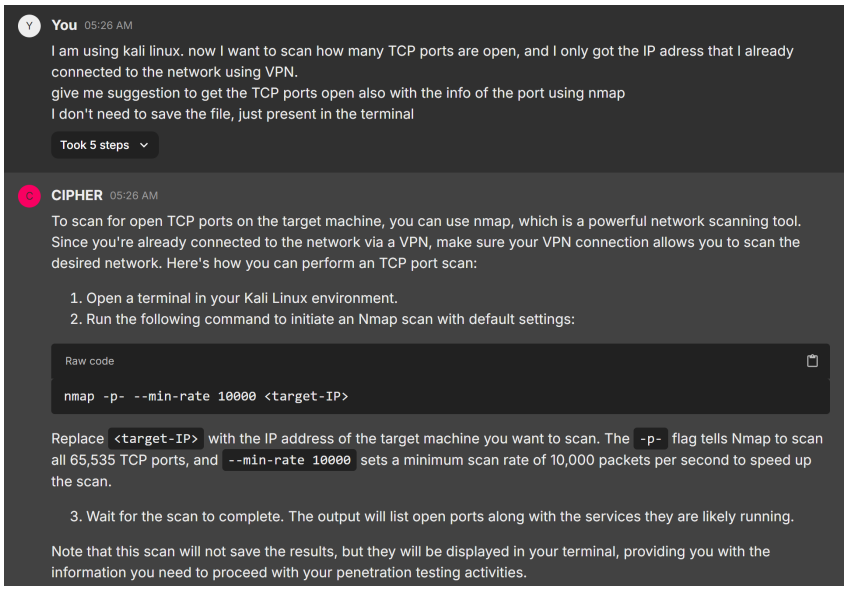


Figure 18. Nmap suggestion to scan all ports.



Figure 19. FFUF subdomain fuzzing.

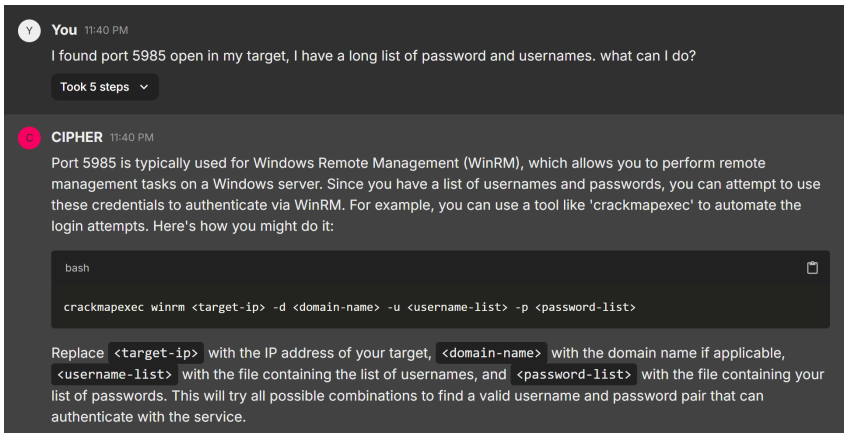


Figure 20. CrackMapExec bruteforcing WinRM service.

This guidance is technically sound and user-friendly, offering both a practical tool for automating brute force attempts and explaining the relevance of port 5985. The response includes all the essential details, such as why this method is suitable for WinRM, making it a valuable resource to learn penetration testing. The combination of clear instructions and technical insight highlights CIPHER's effectiveness as an educational tool in this context.

4.5.2. Tools Suggestion

In this scenario, the user inquires about the appropriate tool for connecting to WinRM on a target system. Figure 21 shows CIPHER's response, recommending the use of Evil-WinRM, a tool specifically designed for penetration testing. CIPHER explains that Evil-WinRM provides a reliable shell for remote command execution and offers clear command syntax: `evil-winrm -i <target-ip> -u <username> -p <password>`, where the user inputs the target IP, username, and password.

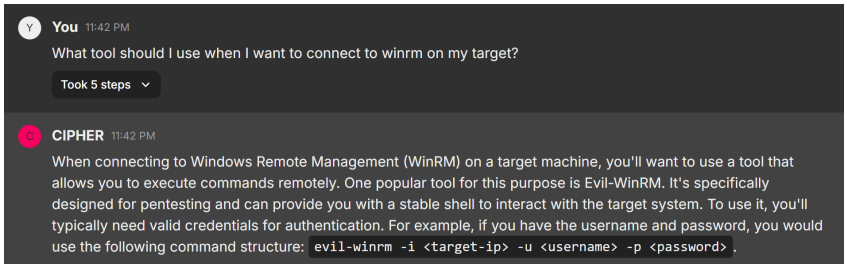


Figure 21. Evil-WinRM as tool suggestion.

This interaction demonstrates CIPHER's deep understanding of penetration testing techniques. The model not only suggests the correct tool but also explains its relevance, emphasizing the need for valid credentials when working with WinRM-enabled systems. By providing clear instructions, CIPHER ensures that even novice users can confidently execute the task, making it a valuable educational resource for penetration testers.

In this scenario, the user discovers that both FTP and HTTP ports are open on the target, with the ability to upload files anonymously to the FTP server. Figure 22 shows CIPHER's response, which acknowledges the progress and suggests a common strategy: uploading a webshell. This is a script that allows command execution on the server via a web interface. CIPHER advises the user to first check the server's configuration to determine which file extensions are allowed for uploads, as this will dictate the type of webshell that can be used. For example, if the server supports PHP, the user could upload a PHP webshell to gain control over the system.

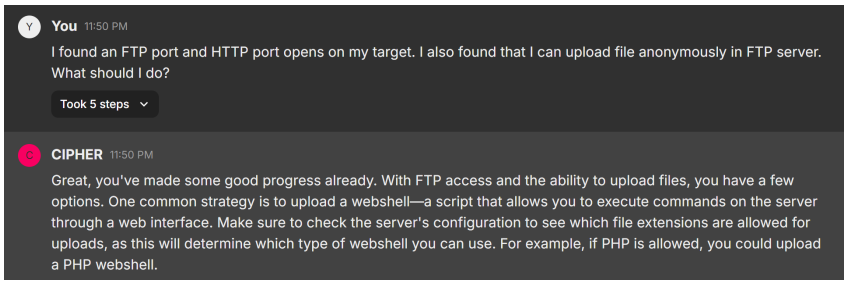


Figure 22. FTP webshell recommendation.

CIPHER's guidance shows an understanding of the penetration testing process, particularly when dealing with misconfigured FTP and HTTP services. The suggestion to check file extension permissions reflects the nuanced approach needed when attempting to escalate access, ensuring that the webshell method is tailored to the target's configuration.

In the next case study, the user asks how to generate a reverse shell for a 64-bit Windows target in a DLL format. Figure 23 illustrates CIPHER's detailed response, recommending the use of msfvenom from the Metasploit framework. CIPHER begins by advising the user to check the system's architecture using the systeminfo command, ensuring the correct architecture set is used. Once confirmed, CIPHER provides a command syntax to generate the DLL reverse shell. The user is instructed to replace placeholders with appropriate values to create the reverse shell, which can then be used in the penetration testing process.

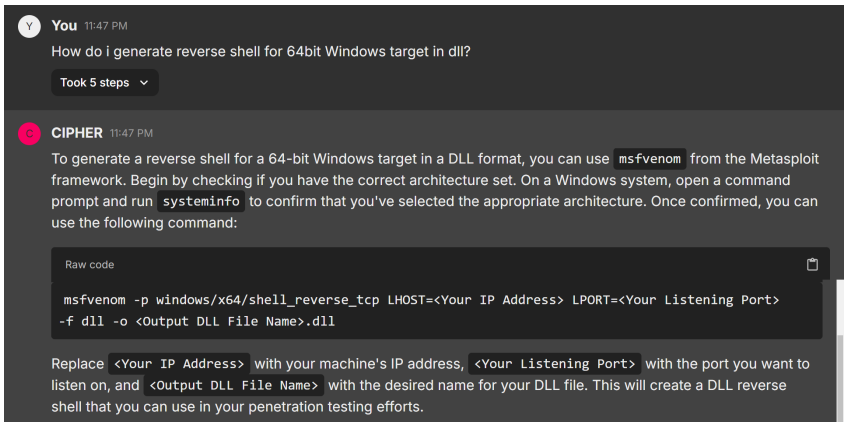


Figure 23. Exploit creation suggestion.

This response reflects a solid understanding of reverse shell generation, covering both technical accuracy and practical guidance. CIPHER explains not only how to generate the reverse shell but also emphasizes verifying the target's architecture, demonstrating a nuanced approach to penetration testing and ensuring that the process runs smoothly.

4.5.3. Attack Vector Suggestion

In this scenario, the user asks about potential attack vectors for a website that accepts URLs and converts the content into PDF format. Figure 24 displays CIPHER's response, which suggests several potential attack vectors, starting with Server-Side Request Forgery (SSRF). This attack allows the attacker to manipulate the server into fetching resources from unintended locations, possibly exposing internal services or sensitive data. CIPHER also points out the security risks associated with the PDF generation process, which could be exploited via Cross-Site Scripting (XSS) or SQL Injection if not properly secured. The model emphasizes the importance of gathering as much information as possible about the target system through enumeration before launching any attacks, demonstrating a nuanced understanding of web-based vulnerabilities.

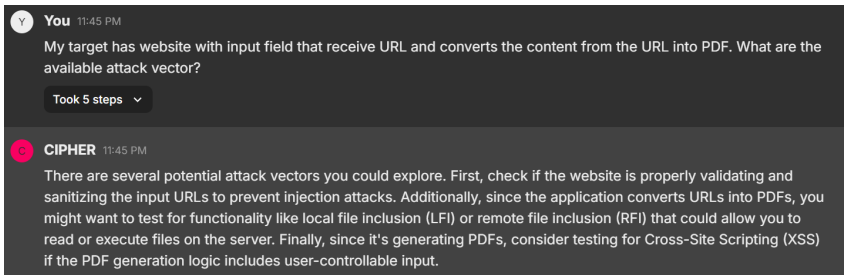


Figure 24. Attack vector suggestion.

CIPHER's response in Figure 25 provides a clear and practical guide to gaining code execution via WordPress, offering step-by-step instructions on file manipulation, such as editing the '404.php' file and uploading a theme. The attention to detail ensures that users of all experience levels can follow along easily, with the model demonstrating a solid understanding of WordPress-specific vulnerabilities and how themes can be exploited for remote code execution.

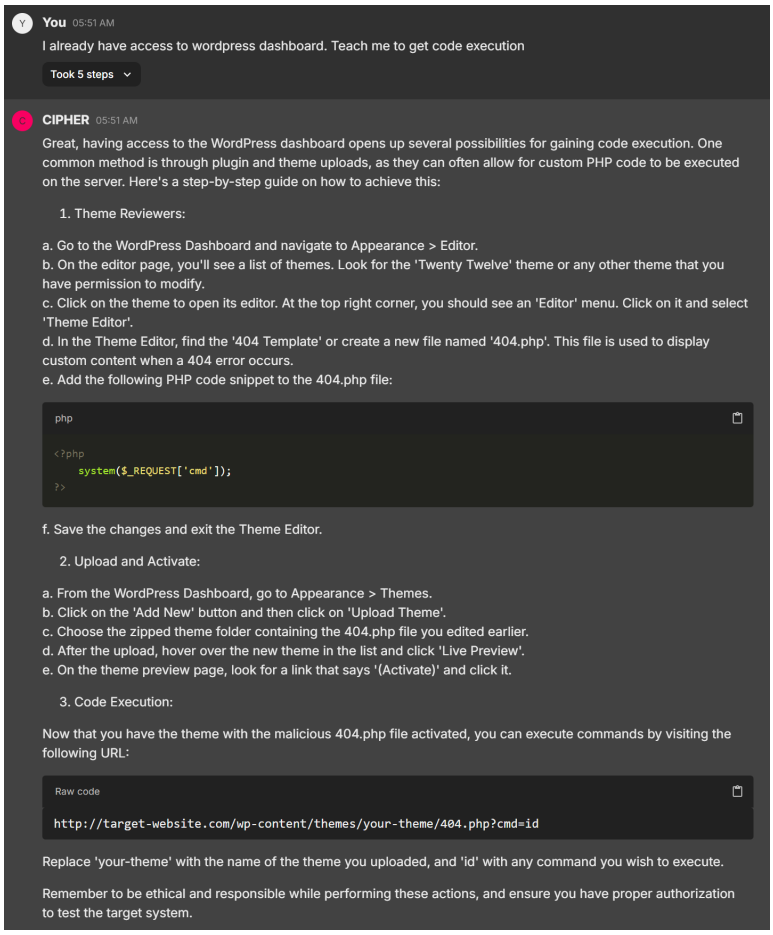


Figure 25. Wordpress RCE instruction.

The choice to use a `404.php` file and inject the `system($_REQUEST['cmd']);` payload shows smart tactics, as this file is often overlooked. While the response could enhance stealth advice, such as clearing logs or minimizing site disruption, it effectively balances clarity and technical depth, making it suitable for both beginners and more advanced users. CIPHER's output showcases a well-rounded understanding of penetration testing within WordPress environments.

4.5.4. Privilege Escalation Suggestion

Shifting to the next example response in Figure 26, CIPHER offers a comprehensive, systematic approach to privilege escalation after gaining a foothold as a local user, aligning with established penetration testing methodologies. Starting with enumeration using tools like 'ps', 'netstat', 'id', and 'sudo -l', CIPHER ensures the user collects key system information, laying a strong foundation for identifying privilege escalation opportunities. The explanation of SUID/SGID files and how to locate them with 'find' is particularly valuable, providing actionable advice for both novice and experienced users.

CIPHER further demonstrates its depth by covering misconfigurations, kernel exploits, and shell escape techniques, which are crucial for exploiting system weaknesses. The model's attention to advanced scenarios, such as credential reuse and inspecting insecure cron jobs, along with its focus on maintaining stealth throughout the process, shows a strong understanding of real-world penetration testing tactics. This balance of clarity and technical depth makes CIPHER's guidance highly effective for beginners.

The next study in Figure 27 showcases CIPHER's solid understanding of privilege escalation by helping a user exploiting Git's 'PAGER' environment variable to execute arbitrary shell commands. The command `sudo PAGER='sh -c ''exec sh 0<&1''' git -p help` takes advantage of Git's behavior to spawn a shell, effectively bypassing restricted environments. CIPHER explains each component of the command clearly, ensuring that even users unfamiliar with environment variables can follow along, while demonstrating technical depth and practicality.



Figure 26. Privilege escalation advice after foothold first gained.

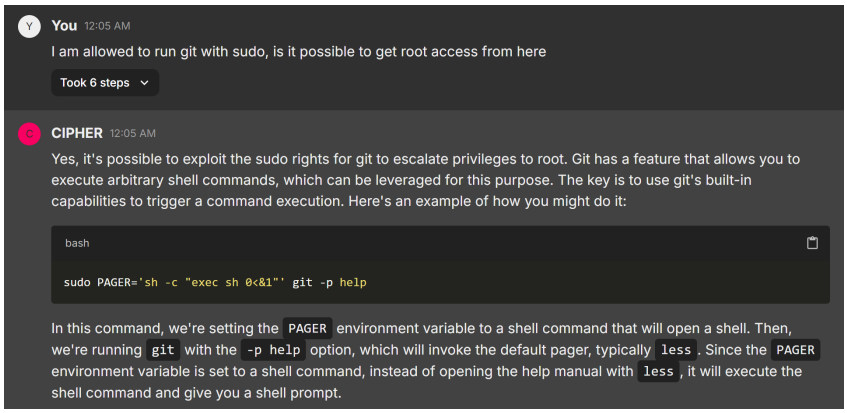


Figure 27. Privilige escalation using misconfigured Git.

The response balances usability and insight, making it highly accessible to beginners. CIPHER's guidance is clear, actionable, and well-aligned with established privilege escalation techniques. A minor enhancement could include addressing potential limitations in more restrictive environments, but overall, the quality of CIPHER's output is excellent, effectively supporting penetration testers to gain highest privilege.

Based on this experiment, we have proven that CIPHER can be effectively used as a practical penetration testing assistant, guiding users through the utilization of multiple tools from reconnaissance to post-exploitation. We observed that while CIPHER performs well overall, there are areas for improvement, such as providing more detailed explanations, example cases, or alternative commands. These improvements can be achieved by scaling up the dataset to support longer and more detailed responses. There is also available room for improvement in model size; despite this, CIPHER is able to respond accurately and assist users effectively.

5. Discussion and Future Works

5.1. Limitations

While CIPHER demonstrates superior average performance compared to other models, including the higher parameter Llama 3 70B, in the FARR Flow reasoning evaluation, this does not imply that CIPHER excels at reasoning during problem debugging. Our observations indicate that CIPHER is proficient at predicting the accurate next step given limited information. However, acting often requires more detailed information, which beginners might need to ask for. CIPHER is not capable of troubleshooting the commands it provides. It sometimes makes mistakes, such as hallucinating command arguments. It tends to hallucinate even more when further questions about the correct command are asked.

We believe this issue arises from a bias in the penetration testing data. When a user reports a problem, CIPHER relates it to penetration testing. For instance, if a provided penetration testing command does not work, instead of correcting the command or asking for the arguments used, CIPHER might assume the target is unreachable or suggest alternative tools for the same purpose. This focus degrades its ability to handle general command errors or Linux command troubleshooting despite incorporating raw datasets that include command line documentation.

Since CIPHER is designed to understand the current situation and suggest subsequent actions, we did not heavily emphasize coding datasets. CIPHER's coding performance can be enhanced by changing the base model to a coding model or using a higher-parameter base model. From a data-centric perspective, improving coding troubleshooting performance by creating synthetic datasets is possible. This can be achieved similarly to the CIPHER method, involving experts to fix code and provide explanations, reasoning, and examples.

Our FARR Flow augmentation is imperfect; some generated augmentations still mention "the author" when they are supposed to be just like finding notes. We have explicitly forbidden this behavior, but it is still in some samples. This could be caused by the limitation of the model used for generation. However, this does not affect the model response when augmented into a question, and even if it exists, it will only cause minor issues in perspective sentences.

Another difficulty in FARR Flow augmentation is providing challenging findings. Generated findings often contain hints, such as "found a zip file, need to crack the password". While this challenges the model to determine which tools to use for cracking, it already suggests a line of reasoning. As a result, the model's focus will be on cracking the password instead of first checking the metadata or trying known target user passwords. This nuance needs to be understood by the model generating the FARR Flow. Although this is not a direct limitation of FARR Flow augmentation, it opens up future work to refine the pipeline and eliminate such results. Rather than waiting for AGI-level LLM, a feasible approach is to use agents or more advanced chain algorithms.

In the FARR Flow evaluation, we use Llama 3 70B as the judge LLM. Even though it is a state-of-the-art model, it has limitations and biases. Different judges can introduce different biases. Using multiple judge LLMs in future work is a possible solution to reduce this bias.

5.2. Scaling

CIPHER's current performance in penetration testing is influenced by the size of its dataset and model. There are several strategies to enhance the model's capabilities further.

The first strategy involves scaling the dataset. This can be improved by incorporating more diverse and specific augmentations for the same dataset, using techniques like Orca [21] and Evolve instruct [88], allowing the model to generalize better. Also, more contextual details can be added into the dataset for the model to understand details of the current chunk to be generated such as date, scope of the context, or even source metadata, preventing the model hallucinating on unknown context. In addition, wide coverage of penetration testing knowledge can be obtained by adding CVE, CWE, and CAPEC as resources for CIPHER dataset augmentation. Currently, the cost of CIPHER dataset augmentation ranges from USD 800 to USD 1000, excluding experimental costs, which can reach around USD 10,000 for full augmentation with the current dataset coverage.

The second strategy is to increase the model size. The current 7B model is insufficient for handling complex tasks like real penetration testing. Larger models with more parameters can significantly enhance generalization ability [89]. Training CIPHER to its best performance at 7 epochs took approximately 4 days on 8xH100 GPUs with a 1M dataset. Training a larger model with 70B parameters, such as Llama 3 or Qwen1.5, is estimated to take about one month or more. There is considerable potential to improve performance, and with sufficient research, we are confident that LLMs can achieve expert capabilities in assisting with penetration testing tasks.

5.3. Measuring Other Penetration Testing Capabilities

Our FARR Flow augmentation method extracts a raw list of critical information from penetration testing write-ups. While our research primarily focuses on measuring the accuracy of next-step prediction guidance, it is possible to craft different question structures from the four items provided in FARR Flow steps. This approach can be used to assess various capabilities, such as guessing the findings when given an action or predicting the outcome when shown the reasoning.

An exciting way to evaluate the model's reasoning capabilities is by not providing the next findings if the model's action is incorrect. In this scenario, at the initial step, the model is given only the target's IP address. We then continuously prompt the model to generate the following action. If the action likely results in the same outcome—verified by targeting

the same service and vulnerability, even with different tools—the evaluator provides the results as the next piece of information.

Using this method, we can measure how quickly the model helps the user penetrate and take over the target machine. A model that requires fewer actions to produce results and can gain privileged access more efficiently will be deemed superior.

For future work, we plan to deeply explore how CIPHER can assist in uncontrolled environments, where system behaviors are unpredictable or information is incomplete. This exploration will allow us to assess CIPHER's ability to adapt to irregular scenarios, such as misconfigured services or unexpected vulnerabilities. By comparing its performance in both controlled and uncontrolled environments, we aim to evaluate how well CIPHER can identify alternative actions and adapt its predictions when faced with less predictable conditions. This investigation will provide valuable insights into the robustness and potential of CIPHER's integration into FARR Flow's next-step prediction accuracy testing.

5.4. Ethical Consideration

We have decided not to release the CIPHER model due to ethical considerations. However, to encourage research and improvement in the domain of penetration testing with LLMs, we will publicly release our novel FARR Flow augmentation and evaluation frameworks on GitHub after our paper is published. We hope that the community will develop more sophisticated and standardized automatic pentesting LLM evaluation methods.

6. Conclusions

Our research focused on developing CIPHER, an AI chatbot designed to guide beginner penetration testers through the pentesting process. We utilized synthetic novice and expert conversations, high-quality documentation, and open-source task datasets. Synthetic questions were crafted to assess the model's understanding and ability to suggest accurate actions.

Our novel FARR Flow evaluation revealed that CIPHER slightly decreases general capabilities but significantly enhances pentesting skills, providing more accurate suggestions than other models on average, particularly in the easy and insane machine categories. However, it shows slight underperformance in the medium and hard categories, but still leads among the 7B parameter models.

In practical testing, CIPHER demonstrated it can assist effectively with its current parameter size, handling a wide range of tasks from reconnaissance to post-exploitation. There is room for improvement, particularly in providing more detailed explanations and alternative actions.

Future enhancements could involve full dataset augmentation, additional security knowledge base, and larger parameter models, as well as refining the FARR Flow method by crafting more complex questions or developing a virtual vulnerable machine LLM evaluation based on FARR Flow steps.

Author Contributions: Conceptualization, D.P.; methodology, D.P.; software, D.P., N.S. and A.A.A.; validation, D.P., N.S., A.A.A., T.-T.-H.L., M.I. and A.Y.K.; formal analysis, D.P., N.S., A.A.A., T.-T.-H.L. and A.Y.K.; investigation, D.P.; resources, D.P.; data curation, D.P.; writing—original draft preparation, D.P.; writing—review and editing, D.P., T.-T.-H.L., A.A.A., N.S. and A.Y.K.; visualization, D.P., T.-T.-H.L. and A.A.A.; supervision, T.-T.-H.L. and H.K.; project administration, H.K.; funding acquisition, H.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the Convergence Security Core Talent Training Business (Pusan National University) support program (No. RS-2022-II221201) and was supported by the ITRC (Information Technology Research Center) support program (IITP-2024-RS-2020-II201797), both supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data and benchmark presented in this study are available publicly at the our repository (https://github.com/ibndias/CIPHER accessed on 21 October 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Denis, M.S.; Zena, C.; Hayajneh, T. Penetration testing: Concepts, attack methods, and defense strategies. In Proceedings of the 2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT), Farmingdale, NY, USA, 29 April 2016; pp. 1–6.
2. Jayasuryapal, G.; Pranay, P.; Kaur, H.; Swati. A Survey on Network Penetration Testing. In Proceedings of the 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM), London, UK, 28–30 April 2021; pp. 373–378.
3. Altulaihan, E.; Alismail, A.; Frikha, M. A Survey on Web Application Penetration Testing. *Electronics* **2023**, *12*, 1229. [CrossRef]
4. Clintswood; Lie, D.G.; Kuswandana, L.; Nadia; Achmad, S.; Suhartono, D. The Usage of Machine Learning on Penetration Testing Automation. In Proceedings of the 2023 3rd International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS), Yogyakarta, Indonesia, 9–10 August 2023; pp. 322–326. [CrossRef]
5. Kennedy, D.; O'Gorman, J.; Kearns, D.; Aharoni, M. *Metasploit: The Penetration Tester's Guide*, 1st ed.; No Starch Press: San Francisco, CA, USA, 2011.
6. GmbH, G.N. OpenVAS—Open Vulnerability Assessment System. 2024. Available online: https://www.openvas.org/ (accessed on 4 July 2024).
7. Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F.L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. Gpt-4 technical report. *arXiv* **2023**, arXiv:2303.08774.
8. Happe, A.; Kaplan, A.; Cito, J. LLMs as Hackers: Autonomous Linux Privilege Escalation Attacks. *arXiv* **2024**, arXiv:2310.11409.
9. Deng, G.; Liu, Y.; Mayoral-Vilches, V.; Liu, P.; Li, Y.; Xu, Y.; Zhang, T.; Liu, Y.; Pinzger, M.; Rass, S. PentestGPT: An LLM-empowered Automatic Penetration Testing Tool. *arXiv* **2023**, arXiv:2308.06782.
10. Xu, J.; Stokes, J.W.; McDonald, G.; Bai, X.; Marshall, D.; Wang, S.; Swaminathan, A.; Li, Z. Autoattacker: A large language model guided system to implement automatic cyber-attacks. *arXiv* **2024**, arXiv:2403.01038.
11. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6000–6010.
12. Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. Llama: Open and efficient foundation language models. *arXiv* **2023**, arXiv:2302.13971.
13. Jiang, A.Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D.S.; Casas, D.d.l.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; et al. Mistral 7B. *arXiv* **2023**, arXiv:2310.06825.
14. Young, A.; Chen, B.; Li, C.; Huang, C.; Zhang, G.; Zhang, G.; Li, H.; Zhu, J.; Chen, J.; Chang, J.; et al. Yi: Open Foundation Models by 01.AI. *arXiv* **2024**, arXiv:2403.04652.
15. Bai, J.; Bai, S.; Chu, Y.; Cui, Z.; Dang, K.; Deng, X.; Fan, Y.; Ge, W.; Han, Y.; Huang, F.; et al. Qwen Technical Report. *arXiv* **2023**, arXiv:2309.16609.
16. Jiang, A.Q.; Sablayrolles, A.; Roux, A.; Mensch, A.; Savary, B.; Bamford, C.; Chaplot, D.S.; Casas, D.d.l.; Hanna, E.B.; Bressand, F.; et al. Mixtral of Experts. *arXiv* **2024**, arXiv:2401.04088.
17. Qwen Team. Qwen-Agent. 2024. Available online: https://github.com/QwenLM/Qwen-Agent (accessed on 25 July 2024).
18. Qwen1.5-7B-Chat. Available online: https://huggingface.co/Qwen/Qwen1.5-7B-Chat (accessed on 25 July 2024).
19. Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; et al. Language models are few-shot learners. *arXiv* **2020**, arXiv:2005.14165.
20. Wei, J.; Bosma, M.; Zhao, V.Y.; Guu, K.; Yu, A.W.; Lester, B.; Du, N.; Dai, A.M.; Le, Q.V. Finetuned Language Models Are Zero-Shot Learners. *arXiv* **2022**, arXiv:2109.01652. .
21. Mukherjee, S.; Mitra, A.; Jawahar, G.; Agarwal, S.; Palangi, H.; Awadallah, A. Orca: Progressive Learning from Complex Explanation Traces of GPT-4. *arXiv* **2023**, arXiv:2306.02707.
22. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q.V.; Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 24824–24837.
23. Hassanin, M.; Moustafa, N. A Comprehensive Overview of Large Language Models (LLMs) for Cyber Defences: Opportunities and Directions. *arXiv* **2024**, arXiv:2405.14487.
24. Gururangan, S.; Marasović, A.; Swayamdipta, S.; Lo, K.; Beltagy, I.; Downey, D.; Smith, N.A. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv* **2020**, arXiv:2004.10964.
25. Christiano, P.F.; Leike, J.; Brown, T.; Martic, M.; Legg, S.; Amodei, D. Deep reinforcement learning from human preferences. *arXiv* **2017**, arXiv:1706.03741.
26. Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. Training language models to follow instructions with human feedback. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 27730–27744.
27. Chen, Z.; Cano, A.H.; Romanou, A.; Bonnet, A.; Matoba, K.; Salvi, F.; Pagliardini, M.; Fan, S.; Köpf, A.; Mohtashami, A.; et al. MEDITRON-70B: Scaling Medical Pretraining for Large Language Models. *arXiv* **2023**, arXiv:2311.16079.
28. Quan, S. Automatically Generating Numerous Context-Driven SFT Data for LLMs across Diverse Granularity. *arXiv* **2024**, arXiv:2405.16579.

29. Happe, A.; Cito, J. Getting pwn'd by AI: Penetration Testing with Large Language Models. In Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE '23 ACM, San Francisco CA USA, 30 November 2023. [CrossRef]
30. Metta, S.; Chang, I.; Parker, J.; Roman, M.P.; Ehuan, A.F. Generative AI in Cybersecurity. *arXiv* **2024**, arXiv:2405.01674.
31. Holik, F.; Horalek, J.; Marik, O.; Neradova, S.; Zitta, S. Effective penetration testing with Metasploit framework and methodologies. In Proceedings of the 2014 IEEE 15th International Symposium on Computational Intelligence and Informatics (CINTI), Budapest, Hungary, 19–21 November 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 237–242.
32. Aksu, M.U.; Altuncu, E.; Bicakci, K. A first look at the usability of openvas vulnerability scanner. In Proceedings of the Workshop on usable security (USEC), San Diego, CA, USA, 24 February 2019.
33. Kumar, H. *Learning Nessus for Penetration Testing*; Packt Publishing: Birmingham, UK, 2014.
34. Valencia, L.J. Artificial Intelligence as the New Hacker: Developing Agents for Offensive Security. *arXiv* **2024**, arXiv:2406.07561.
35. Huang, J.; Zhu, Q. PenHeal: A Two-Stage LLM Framework for Automated Pentesting and Optimal Remediation. *arXiv* **2024**, arXiv:2407.17788.
36. Goyal, D.; Subramanian, S.; Peela, A. Hacking, The Lazy Way: LLM Augmented Pentesting. *arXiv* **2024**, arXiv:2409.09493.
37. Alshehri, I.; Alshehri, A.; Almalki, A.; Bamardouf, M.; Akbar, A. BreachSeek: A Multi-Agent Automated Penetration Tester. *arXiv* **2024**, arXiv:2409.03789.
38. Fang, R.; Bindu, R.; Gupta, A.; Zhan, Q.; Kang, D. Llm agents can autonomously hack websites. *arXiv* **2024**, arXiv:2402.06664.
39. Fang, R.; Bindu, R.; Gupta, A.; Kang, D. Llm agents can autonomously exploit one-day vulnerabilities. *arXiv* **2024**, arXiv:2404.08144.
40. Kumar, A.; Singh, S.; Murty, S.V.; Ragupathy, S. The ethics of interaction: Mitigating security threats in llms. *arXiv* **2024**, arXiv:2401.12273.
41. Lee, J.; Yoon, W.; Kim, S.; Kim, D.; Kim, S.; So, C.H.; Kang, J. BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* **2020**, *36*, 1234–1240. [CrossRef]
42. Cheng, D.; Huang, S.; Wei, F. Adapting large language models via reading comprehension. *arXiv* **2023**, arXiv:2309.09530.
43. Jiang, T.; Huang, S.; Luo, S.; Zhang, Z.; Huang, H.; Wei, F.; Deng, W.; Sun, F.; Zhang, Q.; Wang, D.; et al. Improving domain adaptation through extended-text reading comprehension. *arXiv* **2024**, arXiv:2401.07284.
44. Bhatt, M.; Chennabasappa, S.; Li, Y.; Nikolaidis, C.; Song, D.; Wan, S.; Ahmad, F.; Aschermann, C.; Chen, Y.; Kapil, D.; et al. CyberSecEval 2: A Wide-Ranging Cybersecurity Evaluation Suite for Large Language Models. *arXiv* **2024**, arXiv:2404.13161.
45. Strom, B.E.; Applebaum, A.; Miller, D.P.; Nickels, K.C.; Pennington, A.G.; Thomas, C.B. Mitre att&ck: Design and philosophy. In *Technical Report*; The MITRE Corporation: McLean, VA, USA, 2018.
46. Teknium. OpenHermes 2.5: An Open Dataset of Synthetic Data for Generalist LLM Assistants. 2023. Available online: https://huggingface.co/datasets/teknium/OpenHermes-2.5 (accessed on 25 July 2024).
47. GitHub—Axolotl-ai-Cloud/Axolotl: Go Ahead and Axolotl Questions—github.com. Available online: https://github.com/axolotl-ai-cloud/axolotl (accessed on 25 July 2024).
48. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.T.; Rocktäschel, T.; et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20, Vancouver, BC, Canada, 6 December 2020; Curran Associates Inc.: Red Hook, NY, USA, 2020.
49. Lee, S.; Shakir, A.; Koenig, D.; Lipp, J. Open Source Strikes Bread—New Fluffy Embeddings Model. 2024. Available online: https://www.mixedbread.ai/blog/mxbai-embed-large-v1 (accessed on 25 July 2024).
50. Li, X.; Li, J. AnglE-optimized Text Embeddings. *arXiv* **2023**, arXiv:2309.12871.
51. Beijing Academy of Artificial Intelligence. BAAI/bge-Reranker-Large. 2024. Available online: https://huggingface.co/BAAI/bge-reranker-large (accessed on 25 July 2024).
52. Xiao, S.; Liu, Z.; Zhang, P.; Muennighoff, N.; Lian, D.; Nie, J.Y. C-Pack: Packed Resources For General Chinese Embeddings. In Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, Washington, DC, USA, 14–18 July 2024; pp. 641–649. [CrossRef]
53. Dong, Q.; Li, L.; Dai, D.; Zheng, C.; Ma, J.; Li, R.; Xia, H.; Xu, J.; Wu, Z.; Chang, B.; et al. A Survey on In-context Learning. *arXiv* **2024**, arXiv:2301.00234.
54. Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; Steinhardt, J. Measuring Massive Multitask Language Understanding. In Proceedings of the International Conference on Learning Representations (ICLR), Vienna, Austria, 4 May 2021.
55. Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; Pinto, H.P.d.O.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; et al. Evaluating Large Language Models Trained on Code. *arXiv* **2021**, arXiv:2107.03374.
56. Lin, S.; Hilton, J.; Evans, O. TruthfulQA: Measuring How Models Mimic Human Falsehoods. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Dublin, Ireland, 22–27 May 2022; pp. 3214–3252. [CrossRef]
57. Lane, C.A.; Luciolebrillante; Jazzabeanie; Huber, S.; Wyssmann, A.; Tcpekin; Clifford, P.; Script, T.; Jishnu; Kalinke, S.; et al. Community-Sourced Cheatsheets. Available online: https://github.com/cheat/cheatsheets (accessed on 21 June 2024).
58. GitHub—tldr-pages/tldr: Collaborative Cheatsheets for Console Commands—github.com. Available online: https://github.com/tldr-pages/tldr (accessed on 25 July 2024).
59. Kali Tools | Kali Linux Tools—kali.org. Available online: https://www.kali.org/tools/ (accessed on 25 July 2024).

60. Introduction | OSCP Notes—gabb4r.gitbook.io. Available online: https://gabb4r.gitbook.io/oscp-notes (accessed on 25 July 2024).
61. Fauzi, F. OSCP Playbook. Available online: https://web.archive.org/web/20240129203805/https://fareedfauzi.gitbook.io/oscp-playbook/ (accessed on 26 July 2024).
62. GTFOBins—gtfobins.github.io. Available online: https://gtfobins.github.io/ (accessed on 25 July 2024).
63. LOLBAS—Lolbas-project.github.io. Available online: https://lolbas-project.github.io/ (accessed on 25 July 2024).
64. HackTricks | HackTricks—book.hacktricks.xyz. Available online: https://book.hacktricks.xyz/ (accessed on 25 July 2024).
65. GitHub—Swisskyrepo/PayloadsAllTheThings: A List of Useful Payloads and Bypass for Web Application Security and Pentest/CTF—github.com. Available online: https://github.com/swisskyrepo/PayloadsAllTheThings (accessed on 25 July 2024).
66. 0xdf hacks stuff—0xdf.gitlab.io. Available online: https://0xdf.gitlab.io/ (accessed on 25 July 2024).
67. Hack The Box: The #1 Cybersecurity Performance Center—hackthebox.com. Available online: http://hackthebox.com (accessed on 9 October 2024).
68. TryHackMe | Cyber Security Training—tryhackme.com. Available online: http://tryhackme.com (accessed on 9 October 2024).
69. Vulnerable by Design VulnHub—vulnhub.com. Available online: https://www.vulnhub.com/ (accessed on 9 October 2024).
70. argilla/dpo-mix-7k · Datasets at Hugging Face—huggingface.co. Available online: https://huggingface.co/datasets/argilla/dpo-mix-7k (accessed on 25 July 2024).
71. Sutawika, L.; Gao, L.; Schoelkopf, H.; Biderman, S.; Tow, J.; Abbasi, B.; Fattori, B.; Lovering, C.; Farzanehnakhaee, A.; Phang, J.; et al. A framework for few-shot evaluation of language models. Available online: https://github.com/EleutherAI/lm-evaluation-harness (accessed on 23 June 2024).
72. Clark, P.; Cowhey, I.; Etzioni, O.; Khot, T.; Sabharwal, A.; Schoenick, C.; Tafjord, O. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *arXiv* **2018**, arXiv:1803.05457.
73. Liu, J.; Cui, L.; Liu, H.; Huang, D.; Wang, Y.; Zhang, Y. LogiQA: A Challenge Dataset for Machine Reading Comprehension with Logical Reasoning. *arXiv* **2020**, arXiv:2007.08124.
74. Bisk, Y.; Zellers, R.; Bras, R.L.; Gao, J.; Choi, Y. PIQA: Reasoning about Physical Commonsense in Natural Language. *arXiv* **2019**, arXiv:1911.11641. [CrossRef]
75. Zellers, R.; Bisk, Y.; Schwartz, R.; Choi, Y. SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. *arXiv* **2018**, arXiv:1808.05326.
76. Zellers, R.; Holtzman, A.; Bisk, Y.; Farhadi, A.; Choi, Y. HellaSwag: Can a Machine Really Finish Your Sentence? *arXiv* **2019**, arXiv:1905.07830.
77. Nie, Y.; Williams, A.; Dinan, E.; Bansal, M.; Weston, J.; Kiela, D. Adversarial NLI: A New Benchmark for Natural Language Understanding. *arXiv* **2020**, arXiv:1910.14599.
78. Mihaylov, T.; Clark, P.; Khot, T.; Sabharwal, A. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. *arXiv* **2018**, arXiv:1809.02789.
79. Qwen1.5-7B. Available online: https://huggingface.co/Qwen/Qwen1.5-7B (accessed on 25 July 2024).
80. Colibri_8b_v0.1. Available online: https://huggingface.co/CyberNative-AI/Colibri_8b_v0.1 (accessed on 25 July 2024).
81. Lily-Cybersecurity-7B-V0.2. Available online: https://huggingface.co/segolilylabs/Lily-Cybersecurity-7B-v0.2 (accessed on 25 July 2024).
82. Meta-Llama-3-8B-Instruct. Available online: https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct (accessed on 25 July 2024).
83. Llama-3-WhiteRabbitNeo-8B-v2.0. Available online: https://huggingface.co/WhiteRabbitNeo/Llama-3-WhiteRabbitNeo-8B-v2.0 (accessed on 25 July 2024).
84. Hermes-2-Pro-Llama-3-8B. Available online: https://huggingface.co/NousResearch/Hermes-2-Pro-Llama-3-8B (accessed on 25 July 2024).
85. Mistral-7B-Instruct-v0.3. Available online: https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3 (accessed on 25 July 2024).
86. preemware. preemware/pentesting-eval. 2024. Available online: https://huggingface.co/datasets/preemware/pentesting-eval (accessed on 25 July 2024).
87. Qwen1.5-72B. Available online: https://huggingface.co/Qwen/Qwen1.5-72B (accessed on 25 July 2024).
88. Xu, C.; Sun, Q.; Zheng, K.; Geng, X.; Zhao, P.; Feng, J.; Tao, C.; Jiang, D. WizardLM: Empowering Large Language Models to Follow Complex Instructions. *arXiv* **2023**, arXiv:2304.12244.
89. Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; Vinyals, O. Understanding deep learning (still) requires rethinking generalization. *Commun. ACM* **2021**, *64*, 107–115. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.