Poster: Automatizing Cybersecurity via LLMs

Burak Hasırcıoğlu, Vasilios Mavroudis, Chris Hicks
The Alan Turing Institute, London, UK
{bhasircioglu, vmavroudis, c.hicks}@turing.ac.uk

***Abstract*—We propose an agentic framework for Large Language Models (LLMs) to automate cybersecurity tasks, helping to measure the capabilities of LLMs in this domain. We introduce the action-diversity method which ensures that LLM-based agents consider a wide range of alternatives before deciding on an action, allowing them to select the best option based on their knowledge. Thanks to the action-diversity method, our framework successfully solves all the Capture the Flag (CTF) challenges previous agents have addressed and several challenges that earlier works could not solve.**

I. Introduction

LLMs have gained significant attention for their impressive performance in natural language tasks, including question answering, summarization, and text generation. Trained on extensive internet text, they acquire vast knowledge across many topics, enhancing their applications in various domains that require both linguistic skills and automation.

An important field where this potential is particularly high impact is cybersecurity. While LLMs can automate certain cybersecurity operations, like penetration testing, and aid in discovering vulnerabilities in software and systems—potentially reducing cybersecurity costs—they also pose risks. The ability to automatically identify vulnerabilities can empower malicious actors, both lowering the barrier to entry for beginners and saving time and resources for experienced hackers.

Therefore, it is crucial to evaluate the capabilities of LLMs in automating cybersecurity operations to fully understand their benefits and risks. In this paper, we propose an agentic framework designed to address cybersecurity tasks. Our framework consists of multiple LLM agents, each assigned different responsibilities. Some of these agents have code-executing capabilities, allowing them to implement proposed solutions and refine their approaches as needed.

Additionally, we introduce an action-diversity method where each agent considers various alternative actions and evaluates them before deciding on the next step for a given subtask. We have found that our approach considerably enhances the success rate of LLMs in cybersecurity tasks compared to previous work.

To assess the performance of our framework, we utilized challenges from various CTF competitions as a benchmark. The results demonstrate that our framework outperforms previous approaches by successfully solving challenges where other methods failed.

II. Methodology

We provide an overview of our framework in Fig. 1. It consists of three hierarchical layers: planning, substep execution, and action execution.

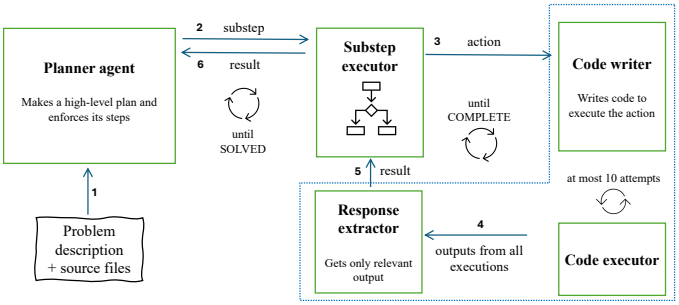


Fig. 1. Overview of the proposed framework

A. Planning Layer

The planning layer features a single agent known as the planner agent. This agent is based on an LLM and serves as the starting point for the problem-solving process. Users can query this agent by providing a problem description and, if applicable, the names of relevant source files. The primary responsibility of the planner agent is to create a high-level plan and ensure its proper execution.

After generating the plan, the planner agent sequentially creates a general, high-level description for each substep of the plan, which is then executed in the substep execution layer. Based on the outcomes of each substep, the planner agent either proceeds to the next substep or requests a refinement of the results from the previous substep. Additionally, the planner agent has the flexibility to modify its plan if necessary, depending on the results obtained.

B. Substep Execution

The substep execution layer consists of a single LLM-based agent called the substep executor. When the planner agent assigns a substep, the substep executor is queried with this substep along with a general description of the challenge being addressed. To tackle the assigned substep, the substep executor generates T different potential actions.

For each proposed action, it provides a detailed analysis of both the value and the likelihood of success. Based on this analysis, the substep executor selects the best action to proceed with. This chosen action leads to the generation of a new subtask for the code action execution layer.

Once the action execution layer either solves the subtask, or reports a failure, the substep executor reacts accordingly. If successful, it may refine the subtask description and request further results from the action execution layer. If there is a

failure, the executor proposes a new action, repeating the same process of generating T alternative proposals and evaluations.

This iterative process continues until the substep executor determines the substep assigned by the planner agent is either complete or cannot be resolved. Finally, the substep executor reports the outcome of the substep back to the planner agent, enabling it to proceed with the next step.

C. Action Execution

In the action execution layer there are two LLM-based agents called the code writer and the response extractor, and one software-based agent known as the code executor. Once the substep executor decides on an action and generates a corresponding query, the query is relayed to the code writer which generates the executable code to fulfill the action.

After the code is generated, it is forwarded to the code executor, which executes the code after conducting several safety checks. At this stage, we also allow the human user to intercept and block the execution of the code if there is a possibility of harmful actions.

Once the code is executed, the output is sent back to the code writer, which determines whether the action has been completed or if the code needs to be modified to address any errors. The code writer is allowed a maximum of 10 attempts to correct such errors, and if it cannot successfully fix the code within that limit, it reports a failure.

Whether the action executed successfully or resulted in failure after 10 attempts, the response extractor reviews the entire code execution loop. It retrieves only the relevant outputs and generates a report for the substep executor.

III. Experiments and Discussion

To evaluate our framework, we conducted experiments using the same benchmarks as previous related work [1], [2], specifically challenges from the HackTheBox (HTB), GlacierCTF (G-CTF), SekaiCTF (S-2X) and picoCTF (p-CTF) competitions. In [1], the earliest times at which human participants solved the challenges are provided. During our evaluation, we focused on challenges completed by human participants in under 2 hours. This decision was made based on our current time constraints and the estimated success rate of the current version of the framework. Similarly, for p-CTF, we selected challenges that the framework in [2] could not solve to determine if our approach performs better. Additionally, we included some arbitrary medium-difficulty challenges. We plan to test all challenges from these competitions in the complete version of our work.

For all LLM-based agents, we utilize Anthropic's *claude-3-5-sonnet-20241022* as the underlying model. To evaluate whether a challenge has been unsuccessful, we allow the framework to operate for up to half an hour. If the challenge remains unsolved after this period, we terminate the run and conclude that the task has failed. We set the action-diversity parameter in the substep executor to $T = 10$. A comparison of our framework with related work is presented in Table I, where each experiment is conducted only once.

TABLE I
Experiment Results

Challenge	Previous work	Ours
Loot Stash, HTB, Reversing	✓	✓
Packed Away, HTB, Reversing	✓	✓
Crushing, HTB, Reversing	X	✓
Permuted, HTB, Crypto	X	X
Labyrinth Linguist, HTB, Web	X	X
Partial Tenacity, HTB, Crypto	X	X
Delulu, HTB, pwn	X	✓
Skilift, G-CTF, Crypto	✓	✓
Glacier Exchange, G-CTF, Web	X	X
SOP, G-CTF, Reversing	X	X
Noisy CRC, S-23, Crypto	X	X
Network Tools, S-23, pwn	X	X
Chunky, S-23, Web	X	X
Failproof, S-22, Crypto	X	X
Urgent, HTB, Forensics	X	✓
Flag Command, HTB, Web	✓	✓
It Has Begun, HTB, Forensics	✓	✓
Dynastic, HTB, Crypto	✓	✓
Primary Knowledge, HTB, Crypto	✓	✓
Data Siege, HTB, Forensics	X	X
Missing Bits, G-CTF, Crypto	X	X
SLCG, G-CTF, Crypto	X	X
RPGO, G-CTF, Reversing	X	X
SOP, G-CTF, Reversing	X	X
Eval Me, S-23, Forensics	✓	✓
Noisier CRC, S-23, Crypto	X	X
White Pages, p-CTF, Forensics	✓	✓
MacroHard WeakEdge, p-CTF, Forensics	X	✓
Mind your Ps and Qs, p-CTF, Crypto	✓	✓
Pitter, Patter, Platters, p-CTF, Forensics	X	✓
Powershelly, p-CTF, Reversing	X	X
New Vignere, p-CTF, Crypto	X	X

As shown in Table I, our framework successfully addresses all the challenges that Cybench [1] can solve and additionally tackles three challenges—*Crushing*, *Delulu*, and *Urgent*—that Cybench was unable to resolve. These challenges are among those reported to be completed in under two hours by the top human performers. Furthermore, we have demonstrated that our proposed framework can solve *MacroHard WeakEdge* and *Pitter, Patter, Platters*, which are unsolved challenges from [2]. During the development phase, we noticed that the solutions to these additional challenges became achievable after we introduced the action-diversity prompt. Therefore, we believe that these preliminary results suggest a promising direction for enhancing the performance of LLM-based frameworks in cybersecurity tasks. Additionally, these new techniques may imply that the capabilities of LLMs in the cybersecurity domain are currently underestimated due to the limitations of existing methodologies. It is possible that LLMs have the potential to achieve significantly more than what is recognized in the current state of the field.

References

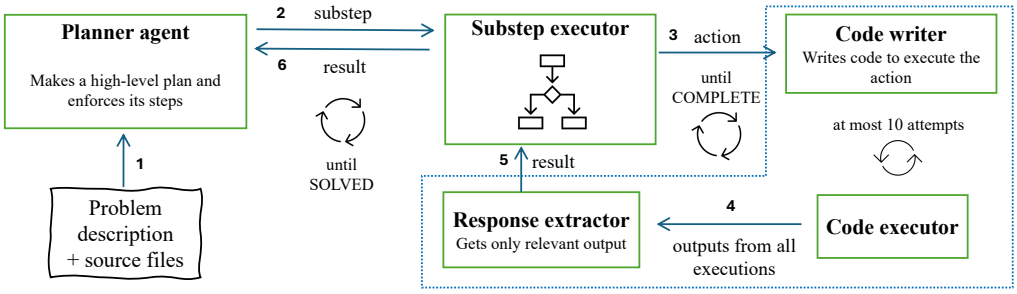
[1] A. K. Zhang, N. Perry, R. Dulepet, J. Ji, J. W. Lin, E. Jones, C. Menders, G. Hussein, S. Liu, D. Jasper *et al.*, "Cybench: A framework for evaluating cybersecurity capabilities and risks of language models," *arXiv preprint arXiv:2408.08926*, 2024.

[2] R. Turtayev, A. Petrov, D. Volkov, and D. Volk, "Hacking ctfs with plain agents," *arXiv preprint arXiv:2412.02776*, 2024.

Automatizing Cybersecurity via Large Language Models

Burak Hasırcıoğlu, Vasilios Mavroudis, Chris Hicks

The Alan Turing Institute, London UK, bhasircioglu@turing.ac.uk



Introduction

- Large Language Models (LLMs) have shown remarkable performance across various domains
- Cybersecurity presents a unique opportunity and challenge for LLM automation
- Our research explores an agentic framework to leverage LLMs for cybersecurity tasks

Motivation

- LLMs can automate penetration testing and vulnerability discovery
- Potential to reduce cybersecurity costs
- May also lower the entry barrier for beginners and save time & resources for experienced hackers
- Critical need to understand LLM capabilities and risks in cybersecurity

Framework Architecture

Three hierarchical layers:

1. **Planning Layer:** Planner agent creates high-level strategy
2. **Substep Execution:** Substep executor generates and evaluates actions
3. **Action Execution:** Code writer and executor implement solutions

Action-Diversity Method

Key innovation: Systematic action exploration

- Generate multiple potential actions
- Analyze value and success likelihood
- Select the most promising action
- Iterative refinement of approach

Experiment Setup

Benchmark: Capture The Flag (CTF) Challenges

- Evaluated across multiple competition challenges: HackTheBox (HTB), GlacierCTF (G-CTF), and picoCTF (p-CTF)
- Challenges are chosen the same as previous benchmarks [1, 2]
- Challenges spanning: Reversing, Web Exploitation, Cryptography, Forensics, Pwn
- Difficulty constraint: The best human performer can solve less than 2 hours per challenge

Experiment Results

Challenge	Previous work	Ours
Loot Stash, HTB, Reversing	✓	✓
Packed Away, HTB, Reversing	✓	✓
Crushing, HTB, Reversing	X	✓
Permuted, HTB, Crypto	X	X
Labyrinth Linguist, HTB, Web	X	X
Partial Tenacity, HTB, Crypto	X	X
Delulu, HTB, pwn	X	✓
Skilift, G-CTF, Crypto	✓	✓
Glacier Exchange, G-CTF, Web	X	X
SOP, G-CTF, Reversing	X	X
Noisy CRC, S-23, Crypto	X	X
Network Tools, S-23, pwn	X	X
Chunky, S-23, Web	X	X
Failproof, S-22, Crypto	X	X
Urgent, HTB, Forensics	X	✓
Flag Command, HTB, Web	✓	✓
It Has Begun, HTB, Forensics	✓	✓
Dynastic, HTB, Crypto	✓	✓
Primary Knowledge, HTB, Crypto	✓	✓
Data Siege, HTB, Forensics	X	X
Missing Bits, G-CTF, Crypto	X	X
SLCG, G-CTF, Crypto	X	X
RPGO, G-CTF, Reversing	X	X
SOP, G-CTF, Reversing	X	X
Eval Me, S-23, Forensics	✓	✓
Noisier CRC, S-23, Crypto	X	X
White Pages, p-CTF, Forensics	✓	✓
MacroHard WeakEdge, p-CTF, Forensics	X	✓
Mind your Ps and Qs, p-CTF, Crypto	✓	✓
Pitter, Patter, Platters, p-CTF, Forensics	X	✓
Powershelly, p-CTF, Reversing	X	X
New Vignere, p-CTF, Crypto	X	X

Key Achievements:

- Solved challenges previous frameworks could not
- Demonstrated LLM potential in cybersecurity

Implications

- LLMs may have underestimated capabilities in cybersecurity
- Action-diversity method shows promise
- Future work: Expand challenge coverage and refine methodology

References

[1] A. K. Zhang, N. Perry, R. Dulepet, J. Ji, J. W. Lin, E. Jones, C. Menders, G. Hussein, S. Liu, D. Jasper *et al.*, "Cybench: A framework for evaluating cybersecurity capabilities and risks of language models," *arXiv preprint arXiv:2408.08926*, 2024.

[2] R. Turtayev, A. Petrov, D. Volkov, and D. Volk, "Hacking ctfs with plain agents," *arXiv preprint arXiv:2412.02776*, 2024.

aicd The Alan Turing Institute