Chat2Pot: Chatting a Honeypot into Existence

Tillmann Angeli*, Karina Elzer†, Devon Perryman* and Hans D. Schotten*‡
*German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany
†Technical University of Denmark (DTU), Kongens Lyngby, Denmark
‡Department of Electrical and Computer Engineering, RPTU Kaiserslautern-Landau, Kaiserslautern, Germany
Email: {tillmann.angeli, devon.perryman, hans.schotten}@dfki.de, kaelz@dtu.dk

***Abstract*—The creation and deployment of honeypots traditionally require significant manual effort and specialized expertise, limiting their adaptability and diversity. Recent advances in Large Language Models (LLMs) enable new forms of automation through natural language interfaces. This work introduces *Chat2Pot*, a system that leverages LLMs to dynamically generate diverse, Docker-based honeypots from natural-language descriptions. By automating the generation of technical artifacts, *Chat2Pot* significantly reduces manual effort while enabling scenario-specific customization. The system integrates validation, orchestration, and containerized deployment to ensure robustness and safe automation. This live demonstration showcases the on-demand generation and deployment of honeypots based on user-provided descriptions of target environments.**

***Index Terms*—Cyber Deception, Network Security, Honeypot, Large Language Models**

I. Introduction

Honeypots, intentionally vulnerable and valueless systems, have risen in popularity as an active deception defense mechanism for detecting, monitoring, and studying malicious behavior [1]. However, the practical deployment remains limited by the diverse and rapidly evolving threat landscape. Current honeypot development is a complex and time-consuming task that requires technical expertise and extensive manual effort. As a result, honeypots are often selected from a small existing set. These widely reused honeypots become outdated, lack maintenance and versatility in terms of emulated services. This limits flexibility in usage and the creation of deception defenses tailored to specific scenarios or environments, and increases the likelihood of honeypot identification through fingerprinting. This motivates the need for a more dynamic and customizable honeypot generation.

Recent research employs Large Language Models (LLMs) as honeypots to create more dynamic and realistic responses [2], [3]. While these approaches significantly enhance the behavior of deployed honeypots, the underlying deployment process is typically still manually designed and statically defined. Furthermore, the use of LLMs for generating technical artifacts such as `Dockerfiles` and configurations, shifting lower level implementation details from humans to machines, has been demonstrated [4]. However, their work focuses on reliably generating up-to-date and executable environments, while honeypot creation intentionally requires the deployment of outdated or vulnerable services. These lines of work highlight an unexplored opportunity: leveraging LLM-based technical artifact generation to enable dynamic honeypot generation. LLMs are particularly suited for this task due to their ability to map natural language to structured artifacts. Their non-deterministic nature becomes an advantage, as one prompt can generate different honeypots, increasing diversity and making fingerprinting more difficult. This demo of *Chat2Pot* demonstrates how LLMs can be leveraged not as honeypots themselves, but as an enabling technology for simplified, automated honeypot creation. By integrating natural language interfaces with validation, orchestration, and containerized deployment, *Chat2Pot* allows users to generate scenario-specific honeypots with minimal manual effort. Our approach demonstrates how LLM-driven artifact generation can significantly increase honeypot diversity and adaptability, thereby enhancing deception-based defenses in the face of a rapidly evolving threat landscape.

This work was supported by the German Federal Ministry of Research, Technology and Space through CASTLE (Grant No. 16KIS1906K).

II. System Overview

Chat2Pot consists of three main layers. At the **presentation layer**, the user interacts with the system via a graphical user interface (GUI) or a command line interface (CLI). This layer is responsible solely for user interaction and the visualization of system states and results.

The **processing layer** comprises two tightly coupled components: the LLM backend and an orchestration component. The LLM backend generates honeypot artifacts based on a scenario described by the user. These artifacts are represented in a Docker-based format and serve as the foundation for the subsequent deployment process. The orchestration component coordinates the overall workflow and interoperability of all layers and components. It validates user input, ensures that prompts conform to predefined requirements, and performs syntactic and semantic validation of the artifacts. Only verified and well-formed results are retained, thereby ensuring robustness and automation safety. A carefully designed system prompt primes the LLM backend to generate exactly three artifacts required for honeypot creation: a `Dockerfile` (environment definition), a `supervisord.conf` (process control), and a `setup_services.sh` script (service initialization).

At the **infrastructure layer**, a Docker controller manages the life cycle of the honeypot containers. Based on the validated artifacts, it enables building the Docker image as well as managing the honeypot environment.

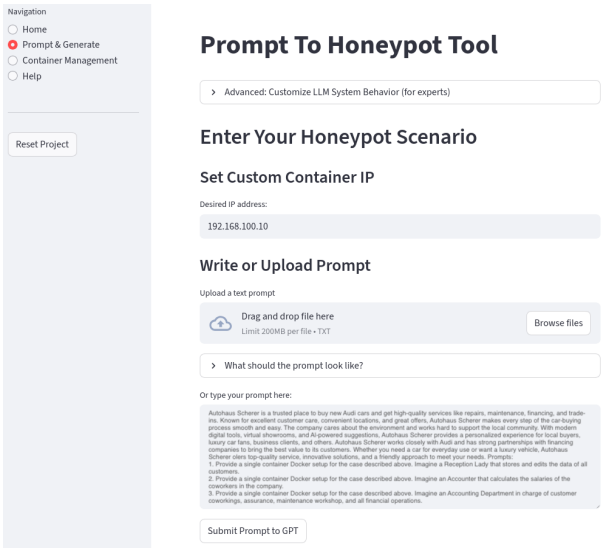

Fig. 1. Graphical User Interface in the browser with example prompt.

III. Demonstration

The demonstration of *Chat2Pot* showcases the process of creating and deploying honeypots based on user-provided input. The user begins by writing a natural-language prompt describing the environment in which the honeypot is intended to be deployed. This description can range from a general specification, such as the institution or company, to detailed information including departments, employees, and active subject areas. Based on this input, *Chat2Pot* generates a tailored honeypot configuration. Once the honeypot has been created, the user can orchestrate it using their preferred interface. The honeypot can be added to a network, started, stopped, or reset, allowing the demonstration of full operational control. Following deployment, the network can be scanned from another device to verify that the honeypot is active and exhibits the intended vulnerabilities. User interaction is facilitated through two complementary interfaces: a GUI and a CLI. The GUI, accessible via a locally hosted web server, provides a visually intuitive and user-friendly workflow.

As shown in Figure 1, the main window allows the user to input the textual prompt describing the desired honeypot environment. Such a prompt may describe a fictional organization in terms of its business domain, operational context, and internal roles, and may additionally specify concrete IT-related responsibilities to be modeled within the honeypot environment. In the example shown, the prompt describes an automotive dealership and requests a single-container Docker-based setup modeling different employee roles.

Additional functions related to Docker management, such as starting, stopping, or resetting containers, are accessible through a separate menu, visible on the left-hand side of the interface. The CLI offers the same functionality in a text-based format, providing direct and efficient access to all system operations. Integrated context-sensitive help is available in both interfaces to guide users through all operations. Moreover, the system allows modification of the initialization system prompt, giving users further control over the behavior of the LLM. All generated `Dockerfile` can also be inspected during the creation process, providing transparency and the possibility for manual adjustments if desired.

IV. Evaluation

In order to determine how well *Chat2Pot* fulfills its purpose, we evaluated the system's build robustness, which is measured as the relative proportion of successfully buildable (Docker build command yields exit code 0) Docker images among all generated `Dockerfile`. An approximation of the build success rate was determined using six unique descriptions of fictional organizations. These descriptions are four or five sentences long and include the organization's name, purpose, a description of employee roles and their tasks, as well as one or two additional sentences asking for a single container Docker setup for the previously described organization and a specific IT-related task, one of the employees might be in charge of. In total, 30 sets of LLM-generated configuration files were produced by executing each of the six descriptions five times, with each set consisting of previously mentioned artifacts. OpenAI's GPT4o model was used with a maximum token size of 5000 and a temperature of 0.5, allowing for variation in responses, despite intra-scenario prompts being identical. This simple setup resulted in a build success rate of 66.67%. Our analysis indicates that many build failures stem from outdated services whose dependencies are no longer distributed.

V. Future Work

Improving system robustness is a primary objective of future work. Initial efforts will focus on refining the system prompt. In addition, integrating a database for cross-referencing known vulnerabilities may help the LLM select appropriate and consistently available services, or compile them from source. We also aim to increase the diversity of generated honeypots with similar prompts, thereby further reducing the risk of fingerprinting. Future work will also explore enhancing realism by populating honeypots with synthetic, context-aware data generated by LLMs. Finally, we plan to evaluate alternative large language models to assess their suitability and performance for this use case.

References

[1] F. Cohen, "The use of deception techniques: Honeypots and decoys," *Handbook of Information Security*, vol. 3, no. 1, pp. 646–655, 2006.

[2] M. Sladić, V. Valeros, C. Catania, and S. Garcia, "VelLMes: A high-interaction AI-based deception framework," in *Proc. 2025 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 2025, pp. 671–679, doi: 10.1109/EuroSPW67616.2025.00082.

[3] A. Safargalieva, A. Rüffer, and E. Vasilomanolakis, "OHRA: dynamic multi-protocol LLM-based cyber deception," in *Proc. The 30th Nordic Conference on Secure IT Systems (NordSec)*, 2025.

[4] R. Hu, C. Peng, X. Wang, J. Xu, and C. Gao, "Repo2Run: Automated building executable environment for code repository at scale," in *Proc. The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. [Online]. Available: https://openreview.net/forum?id=fZsd3KLMje