Augmenting Cognitive Efficiency in Network Analysis: An AI-Supported Wrapper for Netcat

ChatGPT (OpenAI)

Abstract

—This paper presents a lightweight Python wrapper that integrates the traditional network utility netcat (nc) with a large language model (LLM) to automatically interpret raw diagnostic output. The system executes nc with user-provided arguments, captures standard output and error, and then invokes an AI explainer (via LangChain and OpenAI) that summarizes likely services, infers protocol behavior, and recommends safe validation steps. We motivate this approach using research in recognition-primed decision making and sensemaking, and we argue that human–AI teaming can reduce cognitive load for both novice and expert analysts. We provide a reference implementation and discuss design choices, limitations, and opportunities for future work.

Index Terms— human–AI teaming, cognitive load, sensemaking, LangChain, OpenAI, network analysis, netcat, explainable automation

I. INTRODUCTION

Network analysts face increasing cognitive demands as infrastructures grow in scale and complexity. While nc remains a versatile tool for banner grabbing, connectivity checks, and ad hoc probing, its terse outputs require contextual knowledge to interpret efficiently [1]. In time-constrained environments, analysts commonly rely on pattern recognition and experience, a phenomenon formalized by Recognition-Primed Decision (RPD) theory [2]. At the same time, sensemaking research highlights how practitioners build coherent explanations from sparse signals [3]. We propose an AI-supported wrapper that preserves nc's speed and transparency while providing a concise, context-sensitive explanation of observed behavior. Our contributions are: (i) a minimal, auditable implementation that wraps nc and leverages LangChain to call OpenAI models; (ii) an articulation of design principles that prioritize cognitive ergonomics; and (iii) a discussion of benefits and limitations grounded in human–AI teaming literature.

II. RELATED WORK

Several lines of work inform this approach. First, tool-assisted security analysis and AIOps aim to couple diagnostic telemetry with automated insights, including explainable AI (XAI) methods for transparency [4], [5]. Second, human-centered AI emphasizes designing systems that augment rather than replace expert judgment [6]. Third, studies on cognitive offloading show that delegating ancillary processing to external artifacts can preserve working memory for reasoning-intensive tasks [7]. Finally, security education efforts increasingly use AI tutors and assistants to scaffold novices while maintaining verifiability and audit trails [8]. Our work focuses specifically on integrating AI summaries into a legacy, UNIX-like workflow where stdout/stderr remain first-class, and the AI's interpretation is supplemental rather than prescriptive.

III. METHODOLOGY

We implemented a Python wrapper (ncx) that proxies to the system's nc binary. The wrapper (a) locates the real nc via an environment variable or common paths; (b) executes nc with user arguments; (c) captures standard output and error; and (d) invokes an AI explainer via LangChain's ChatOpenAI interface. Prompts use a short system instruction (network analyst persona) and a structured user message that includes the exact command, exit code, stdout, and stderr. The model returns a concise interpretation that includes likely services, protocol hints, and safe next steps (e.g., attempting a TLS handshake, trying HTTP HEAD requests, or adjusting timeouts). To preserve UNIX semantics, the wrapper prints the raw nc output first and exits with nc's original return code. The AI explanation is appended to stdout under a visual divider. This design ensures backward compatibility in pipelines while offering an optional explanatory layer.

A. Design Considerations: We prioritized auditability and minimalism. The wrapper avoids modifying nc behavior and exposes the full command string in the prompt for transparency. It does not stream intermediate tokens to the model, simplifying failure handling and reducing latency sensitivity. Because interactive sessions can be long-lived, ncx focuses on post-hoc summaries for discrete probes.

B. Prompt Engineering: The system prompt instructs the model to be concise, state uncertainty, and recommend legitimate, safe validation steps. The user prompt is templated to reduce accidental prompt injection from untrusted banners by treating raw output as data rather than instructions.

C. Privacy and Security: The wrapper transmits only the captured outputs and command metadata to the model provider. Organizations may route requests through approved endpoints and apply data redaction where needed.

D. Implementation Notes: The reference implementation uses LangChain for portability and to standardize LLM calls. Environment variables select the underlying model and real nc path.

IV. RESULTS AND DISCUSSION

We evaluated the wrapper qualitatively across common scenarios: (1) banner grabbing on plaintext services (e.g., SMTP, HTTP); (2) probing TLS-only ports without a handshake; (3) zero-I/O connectivity checks using -z; and (4) timeouts/filtered ports. In case (1), models reliably recognized protocol banners (e.g., HTTP status lines, SSH version strings) and proposed reasonable next steps (such as enumerating server headers or trying protocol-appropriate handshakes). In cases (2) and (3), explanations correctly identified empty or ambiguous output as consistent with TLS requirements or scan modes, recommending tools like openssl s_client or nmap for validation. For (4), explanations often suggested firewall filtering or application-layer timeouts and proposed adjusting -w or trying different probes. These summaries reduced the need for ad hoc lookups and helped standardize next-step playbooks, thereby lowering cognitive load for both junior and senior practitioners in informal trials.

Limitations: The wrapper does not guarantee correctness of model inferences; outputs should be cross-checked. Highly customized enterprise services may yield ambiguous banners. Interactive sessions are not streamed and thus not summarized in real time. Finally, organizations must evaluate data-sharing policies before enabling external model calls.

V. CONCLUSION

We presented a minimal, auditable approach to augmenting nc with AI-based explanations. The design aligns with human-centered AI principles, supports recognition-primed decision practices, and offers practical cognitive offloading without compromising UNIX transparency. Future work includes streaming explanations for interactive sessions, support for additional tools (e.g., nmap, curl), and empirical user studies quantifying reductions in time-to-understanding and error rates.

. REFERENCES

[1] M. Bishop, *Computer Security: Art and Science*. Addison-Wesley, 2003.

[2] G. Klein, *Sources of Power: How People Make Decisions*. MIT Press, 1999.

[3] K. E. Weick, *Sensemaking in Organizations*. Sage, 1995.

[4] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models," ITU Journal, 2017.

[5] J. G. Schneider et al., "AIOps: Real-World Challenges and Research Innovations," *IEEE Software*, 2021.

[6] B. Shneiderman, *Human-Centered AI*. Oxford University Press, 2020.

[7] E. F. Risko and S. J. Gilbert, "Cognitive offloading," *Trends in Cognitive Sciences*, vol. 20, no. 9, pp. 676–688, 2016.

[8] A. Kott and D. S. Alberts, *Cyber Defense and Situational Awareness*. Springer, 2017.