Design of a Local, GPU-Accelerated Conversational Agent for Defensive Scambaiting Research

Group Members (Group 14):

Martin Jajarmi

Sukhraj Singh

Yahya Fazal

Department of Computer Science and Technology, Algoma University

Dr. Yazan Otoum

February 5, 2026

Abstract

Telephone-based social-engineering scams remain a significant vector for cybercrime, relying on real-time persuasion to extract credentials, payments, or system access from technologically naive victims. Defensive scambaiting, the controlled engagement of scammers to delay operations and collect behavioral data, has emerged as a potential research and mitigation strategy. Our project proposes the design and implementation of a fully local, GPU-accelerated conversational agent capable of engaging scam callers in real time while enforcing strict safety constraints. The system integrates automatic speech recognition (ASR), large language model (LLM) inference, dialogue policy enforcement, and text-to-speech (TTS) synthesis into a modular, cross-platform architecture. Our proposed solution emphasizes low-latency inference, sandboxed deployment, and extensibility through a core engine (a statically linked library) with platform-specific wrappers, alongside a standalone application frontend, written entirely in C++20.

I. Introduction

Telephone fraud and voice-based phishing (vishing) continue to scale globally, exploiting human trust rather than software vulnerabilities. Recent public reporting indicates that reported fraud losses have increased substantially in the United States, with billions of dollars in reported losses in 2024 and a notable share of losses attributed to contact methods such as calls and texts [1]. Telecom-side mitigations such as caller ID authentication under the STIR/SHAKEN framework aim to reduce spoofing and improve trust signals at the network layer [2], [3]. However, these mitigations do not fully address human-in-the-loop persuasion once a call is connected.

Scambaiting, as in, deliberately engaging scammers to occupy their resources, has historically been performed manually. While human-operated scambaiting can be effective, it is not easily reproducible or scalable for academic-driven experimentation. This project proposes an automated, local-first conversational agent designed for controlled, inbound-only research scenarios. The system prioritizes low-latency interaction while enforcing hard safety constraints to prevent disclosure of sensitive information or execution of harmful actions.

II. Problem Statement

How can a real-time, fully local conversational agent be designed to safely engage scam callers for extended periods while preventing harmful actions and maintaining low-latency interaction? The solution we propose involves having to integrate systems for speech processing and LLM inference, enforcement of safety constraints in code, while also remaining portable across platforms.

III. Case Study Context

This work is conducted as part of a course project for Information Technology Security & Privacy, with a heavier focus on defensive techniques against social-engineering attacks. The agent is treated as a research tool rather than an operational, production-ready mitigation system. Evaluation is restricted to controlled inbound calls and synthetic test callers, with appropriate legal practices for any recording or data retention.

IV. Review of Related Work

A. Vishing and Social Engineering. Vishing has been studied as a form of social engineering where attackers apply persuasion and compliance techniques during live voice interactions. Jones et al. analyzed how persuasion principles are applied during vishing attacks, providing insight into the conversational tactics that enable escalation and victim compliance [4]. More recent work highlights vishing and voice-based social engineering as an increasing challenge and motivates automated methods for understanding and mitigating spoken deception [5]. Broader surveys covering smishing and vishing detection discuss feature engineering and machine learning approaches for identifying suspicious communications [6].

B. Network-Layer and Call-Screening Mitigations. Telecom-focused interventions, including STIR/SHAKEN caller ID authentication, attempt to reduce caller ID spoofing and enable downstream blocking and labeling strategies [2], [3]. Despite these advances, scammers adapt by using legitimate numbers, call forwarding, or social pretexting, leaving a gap for defenses that operate at the conversational layer. Proposals for suspicious call detection using conversational AI have emerged in applied contexts, including approaches that analyze call content and interaction patterns to flag fraud [7]. Recent research has also explored the use of large language models to detect phone scams by analyzing conversational dynamics, suggesting that LLMs can complement traditional call-screening techniques [8].

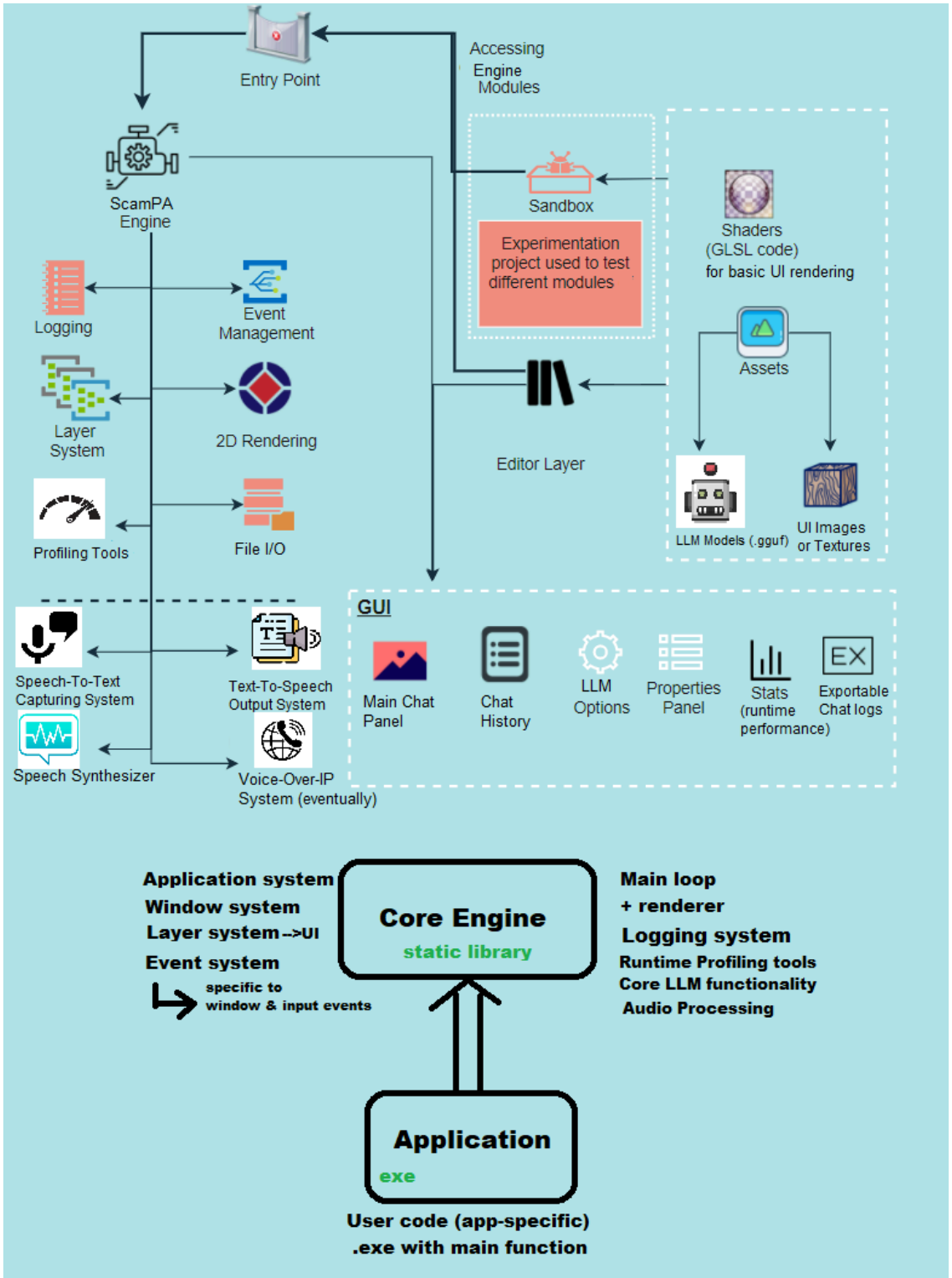
C. LLM-Driven Agents and Dialogue Control. While general-purpose LLMs can generate fluent dialogue, robust interactive systems typically require an explicit control layer to manage objectives and safety. Agent paradigms such as ReAct interleave reasoning and action selection, improving interpretability and reducing error propagation by structuring how models decide what to do next [9]. In the scambaiting/call-screening setting, a comparable approach is to separate policy decisions (e.g., stall, clarify, refuse) from surface realization (the spoken response), and to enforce hard constraints outside the model.

D. Speech Processing Components for Local Agents. Modern local ASR has been accelerated by large-scale weakly supervised training. The Whisper model family demonstrates robust speech recognition across many conditions and languages, providing a foundation for local transcription pipelines [10]. For deployment on consumer hardware, open implementations and optimized runtimes are commonly used. For example, whisper.cpp provides a C/C++ implementation

enabling local inference for Whisper-derived models [11]. Voice activity detection (VAD) is a standard component of real-time speech systems to segment speech and reduce unnecessary computation; practical WebRTC-based VAD implementations are widely used in voice applications today [12].

E. Local Inference Runtimes and Cross-Platform I/O. Local inference runtimes such as llama.cpp target efficient LLM inference in C/C++ across a wide range of hardware configurations and deployment constraints [13]. Wrappers for llama.cpp, such as LlamaLib, enable rapid deployment of LLMs and easier usage of llama.cpp's API overall [19]. To build a cross-platform real-time voice agent, portable audio I/O is required. RtAudio provides a cross-platform C++ interface for realtime audio streaming on major desktop operating systems [14]. For GPU acceleration across vendors, Vulkan offers a cross-platform API for high-performance computation and graphics; its specification and registry are maintained by the Khronos Group [15], [16]. Additional considerations for file I/O, multithreading, and input handling, will be custom tailored for Linux, MacOS and Windows platforms at compile time, based on the differences in operating system service standards that each platform follows (i.e., POSIX vs. Windows API). For cross-platform windowing, GLFW will be used [17], alongside Dear ImGui for its immediate-mode user interface rendering capabilities [18], since they both come with backend support for Vulkan.

V. Proposed System Architecture



The system is organized around a platform-agnostic engine core implemented as a static library. The core engine will contain LLM-specific modules for dialogue state management, policy enforcement, prompt construction, inference orchestration, and metrics logging. Platform-specific dependencies, such as audio I/O, GPU backends, and speech synthesis, are accessed via narrow adapter interfaces or wrappers. A standalone console application links against the engine core and provides a UI frontend via the engine's layering system (i.e., user interface & overlay layers), which in-turn would, for example, provide flexible configuration of the LLM itself at runtime. Overall, this design will allow for substituting ASR/TTS vendors, swapping LLM models, while maintaining cross-platform portability. Future work may explore direct VOIP integration; however, initial evaluation will rely on external calling software and virtual audio routing tools for controlled testing. Products such as Virtual Audio Cable will be used for end-to-end audio I/O capture and output [20], alongside third-party calling software such as Google Voice [21].

VI. Proposed Evaluation Metrics

Future phases will measure end-to-end latency (utterance end to TTS start), ASR accuracy, LLM response time, safety-policy trigger rates, and sustained engagement time under controlled scenarios. Metrics will be logged using custom instrumentation tools embedded within the library itself, which would output JSON files to be reviewed via Google Chrome's Trace Format Viewer on a per-function basis, in order to identify bottlenecks and validate real-time performance.

VII. Conclusion and Phase 2 Plan

This Phase 1 report defined the cybersecurity motivation for automated defensive scambaiting, reviewed related work in vishing, call-screening, LLM agents, and speech processing, and proposed a modular, cross-platform architecture for a local voice agent. Phase 2 will implement the engine core, integrate local ASR and LLM inference with Vulkan acceleration where available, and validate safety enforcement and latency targets in controlled experiments.

References

[1] Federal Trade Commission, “Consumer Sentinel Network Data Book 2024,” 2025. [Online]. Available: https://www.ftc.gov/system/files/ftc_gov/pdf/csn-annual-data-book-2024.pdf

[2] Federal Communications Commission, “Combating Spoofed Robocalls with Caller ID Authentication (STIR/SHAKEN),” 2020–2025. [Online]. Available: https://www.fcc.gov/call-authentication

[3] Federal Communications Commission, “Call Authentication Trust Anchor,” Federal Register, Aug. 19, 2025. [Online]. Available: https://www.federalregister.gov/documents/2025/08/19/2025-15809/call-authentication-trust-anchor

[4] K. S. Jones, M. E. Armstrong, M. K. Tornblad, and A. S. Namin, “How social engineers use persuasion principles during vishing attacks,” 2020. [Online]. Available: https://par.nsf.gov/servlets/purl/10281805

[5] A. Triantafyllopoulos et al., “Vishing: Detecting social engineering in spoken communication,” 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0885230825000270

[6] A. E. Chichwadia et al., “Detecting Smishing and Vishing Attacks using Machine Learning,” 2024. [Online]. Available: https://infonomics-society.org/wp-content/uploads/Detecting-Smishing-and-Vishing-Attacks-using-Machine-Learning.pdf

[7] K. M. Rao, “Suspicious Call Detection and Mitigation Using Conversational AI,” 2023. [Online]. Available: https://www.tdcommons.org/cgi/viewcontent.cgi?article=7596&context=dpubs_series

[8] “Combating Phone Scams with LLM-based Detection,” arXiv, 2024. [Online]. Available: https://arxiv.org/html/2409.11643v2

[9] S. Yao et al., “ReAct: Synergizing Reasoning and Acting in Language Models,” arXiv:2210.03629, 2022. [Online]. Available: https://arxiv.org/abs/2210.03629

[10] A. Radford et al., “Robust Speech Recognition via Large-Scale Weak Supervision,” Proc. ICML, 2023. [Online]. Available: https://proceedings.mlr.press/v202/radford23a.html

[11] ggml-org, “whisper.cpp,” GitHub repository, accessed 2026. [Online]. Available: https://github.com/ggml-org/whisper.cpp

[12] serenadeai, “webrtcvad,” GitHub repository documentation, accessed 2026. [Online]. Available: https://github.com/serenadeai/webrtcvad/blob/master/README.md

[13] ggml-org, “llama.cpp: LLM inference in C/C++,” GitHub repository, accessed 2026. [Online]. Available: https://github.com/ggml-org/llama.cpp

[14] G. P. Scavone, “RtAudio: A Cross-Platform C++ Class for Realtime Audio Input/Output,” ICMC, 2002. [Online]. Available: https://caml.music.mcgill.ca/~gary/papers/rtaudio2002.pdf

[15] Khronos Group, “Khronos Vulkan Registry,” accessed 2026. [Online]. Available: https://registry.khronos.org/vulkan/

[16] Khronos Group, “Vulkan® API Specification,” accessed 2026. [Online]. Available: https://registry.khronos.org/vulkan/specs/latest/html/vkspec.html

[17] M. Magnusson and C. Lövdahl, “GLFW – An OpenGL and Vulkan Windowing Library,” accessed 2026. [Online]. Available: https://www.glfw.org

[18] O. Cornut, “Dear ImGui: Immediate Mode GUI,” GitHub repository, accessed 2026. [Online]. Available: https://github.com/ocornut/imgui

[19] undreamai, “LlamaLib: High-Level LLM Runtime for C++ and C#,” GitHub repository, accessed 2026. [Online]. Available: https://github.com/undreamai/LlamaLib

[20] Eugene Muzychenko, “Virtual Audio Cable,” accessed 2026. [Online]. Available: https://vac.muzychenko.net

[21] Google LLC, “Google Voice,” accessed 2026. [Online]. Available: https://voice.google.com

[22] Google, “Trace Event Format,” Chromium Project Documentation, accessed 2026. [Online]. Available: https://chromium.googlesource.com/catapult/+/HEAD/tracing/README.md