VulnScanAI

*Abstract*— Cybersecurity scanning tools generate detailed reports essential for identifying vulnerabilities across domains such as web applications, cloud infrastructure, mobile platforms, databases, and SSL configurations. However, these reports are often highly technical and difficult for non-expert stakeholders to interpret, limiting their practical utility. This paper presents a unified, web-based cybersecurity assessment platform integrated with an AI-powered chatbot to enhance accessibility and understanding of scan results. The system architecture includes report parsing, semantic chunking, and vector embedding using a fine-tuned all-mpnet sentence transformer model, with cosine similarity-based retrieval to match user queries. Depending on the query type, the system either searches within a temporary report-specific vector store or consults a curated cybersecurity knowledge base. A locally hosted GGUF-format Mistral model generates natural language responses and summaries, ensuring that no user data or reports are sent to the open web. Additionally, we are working on introducing a downloadable executable agent capable of scanning the user’s device locally to identify potential vulnerabilities, such as exposed services or misconfigurations, and allowing the user to take informed action. This agent extends the platform’s functionality to endpoint-level monitoring and automated remediation. Experimental evaluation demonstrates the system’s effectiveness in simplifying complex security data, protecting user privacy, and supporting both technical and non-technical users in maintaining a strong security posture.

*Keywords*—Cybersecurity, Vulnerability Scanning, Report Summarization, Privacy Preserving AI, Local Language model, Natural Language interface, Secure AI systems, Endpoint Scanning.

# **INTRODUCTION**

T

HE increasing complexity and surface area of modern digital systems have made comprehensive cybersecurity assessments more critical than ever. Organizations routinely employ a wide range of specialized tools to identify vulnerabilities in web applications, cloud infrastructures, databases, mobile platforms, and cryptographic implementations. While these tools are effective in identifying technical weaknesses, the reports they generate are often dense, highly specialized, and difficult for non-technical stakeholders to interpret. As a result, critical findings are frequently misunderstood, overlooked, or left unaddressed—creating a disconnect between security professionals and decision-makers, and ultimately compromising timely remediation efforts.

Moreover, most cybersecurity solutions focus solely on detection, leaving interpretation and action as manual, fragmented tasks. Existing tools lack unified workflows, human-readable outputs, and user interactivity, all of which are essential for scalable and inclusive security operations. In addition, reliance on cloud-based AI services raises privacy concerns when sensitive reports or system data are uploaded for

processing. To address these limitations, we present a unified cybersecurity assessment platform that integrates multiple scanning tools within a single web-based interface and augments them with a locally hosted, AI-driven chatbot system.

The platform is designed not only to automate vulnerability scanning but also to make the results accessible and actionable for both technical and non-technical users. Central to the system is a large language model (LLM) in GGUF format, hosted entirely on-premises, that parses complex scan reports, generates human-readable summaries, and supports interactive question-answering through natural language queries.Queries are intelligently classified as either report-specific or general cybersecurity-related. Report-specific queries are matched against semantically embedded content using a fine-tuned transformer model and cosine similarity search within a temporary vector store, enabling precise retrieval of contextually relevant report segments. General queries are answered using a curated knowledge base covering widely referenced topics such as OWASP vulnerabilities, SSL/TLS configurations, and port security. All language-based responses and summaries are generated locally by the GGUF-format Mistral model, ensuring that no user data is transmitted externally or exposed to the open web.

In addition to the web-based scanning and conversational interface, we introduce a lightweight, downloadable executable agent designed for endpoint-level monitoring and remediation. This agent performs local vulnerability detection—such as identifying insecure services or configurations—and enables users to take guided actions, including whitelisting or disabling specific components. By running entirely on the user’s system, the agent maintains privacy while extending protection beyond infrastructure-level scans.

Together, these components form a cohesive platform that not only automates multi-domain vulnerability assessments but also enhances usability, interactivity, and privacy. Our approach aims to bridge the gap between technical output and human understanding, empowering a broader range of users to engage with cybersecurity data effectively and securely.

# **TECHNICAL DESIGN: VULNSCAN AI**

The proposed system consists of three major components working together to provide explainable and security focused assessments:

* 1. Web-based Scanning Platform.
  2. AI powered Report Interpretation and Chat Interface.
  3. Executable security agent for endpoint scanning.

**2.1. Web-based Scanning Platform:**

The core of the system is a unified web-based platform that integrates a diverse suite of open-source security tools to automate comprehensive vulnerability assessments across multiple asset types. This component serves as the user’s central interface for initiating scans, managing scan configurations, and reviewing structured results. All tools operate in isolated, containerized environments to ensure consistent and secure execution. The scanning platform supports the following domains and tools:

**i. Network Scanning:**

Assesses network infrastructure for exposed entry points and potential misconfigurations by detecting open ports, running services, and underlying OS fingerprints.

* Nmap: A versatile network scanner used for port scanning, service detection, operating system identification, and network inventory enumeration.

**ii. Web Application Scanning:**

Identifies common web-based vulnerabilities, including those outlined in the OWASP Top 10, such as cross-site scripting (XSS), SQL injection, and security misconfigurations.

* OWASP ZAP: A feature-rich web application scanner that performs active and passive vulnerability discovery through crawling, fuzzing, and request interception.
* Nikto: A fast web server scanner that detects outdated software, default files, insecure configurations, and known vulnerabilities.

**iii. Mobile Application Scanning:**

Analyzes mobile applications (APK and IPA files) for security flaws in both static code and dynamic behaviour, enabling early-stage mobile threat detection.

* MobSF (Mobile Security Framework): A comprehensive tool supporting static and dynamic analysis of Android and iOS applications, including permissions misuse, hardcoded secrets, and insecure API calls.

**iv. Database Scanning:**

Focuses on identifying injection vulnerabilities and potential data exposure risks at the backend of web applications.

* SQLMap: An open-source penetration testing tool that automates the detection and exploitation of SQL injection vulnerabilities and database fingerprinting.

**v. SSL/TLS Scanning:**

Evaluates the strength and correctness of SSL/TLS implementations in servers to prevent cryptographic attacks and misconfigurations.

* SSLScan: A tool that probes supported cipher suites, protocol versions, and certificate details to highlight weak or deprecated configurations.

**vi. Cloud Security Auditing:**

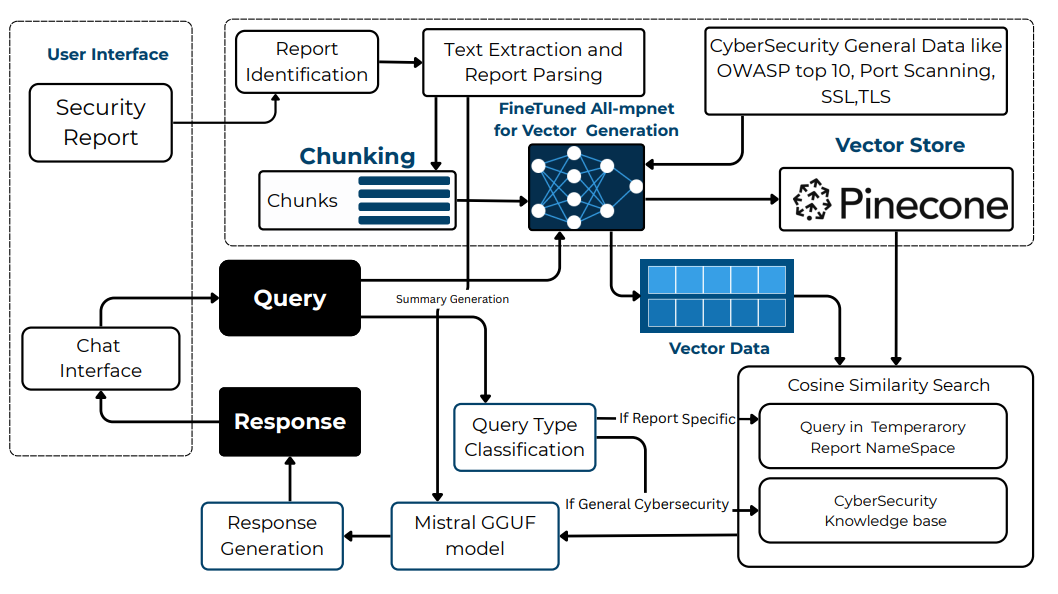
Performs security and compliance audits for cloud environments, initially focusing on AWS, with plans to support Azure and GCP in future iterations.

* Prowler: An open-source tool that evaluates AWS environments against security best practices, misconfiguration risks, and incident response readiness.

All scan results are standardized and stored in a structured format, making them directly compatible with the report parsing and chatbot components. Users can initiate and monitor multiple scan types through the interface and view their outputs in real time or download them for offline analysis. The platform ensures all scans are performed securely, with no external data exposure, and is designed to be extendable to support future tools and scanning domains.

**2.2. AI powered Report Interpretation and Chat Interface.**

To address the challenge of making complex cybersecurity reports accessible and actionable for a broad range of users, the platform features a locally hosted, AI-powered chatbot interface. This component enables natural language interaction with security findings while ensuring that no sensitive data is exposed to the open internet. The chatbot system operates through a multi-stage architecture that combines report parsing, semantic retrieval, context-aware prompting, and local language model inference.



## Architecture Diagram: AI Analyzer

**i. Report Upload and Identification:**

When a scan report is uploaded, the system first performs report type identification to determine its source (e.g., ZAP, Nmap, MobSF, etc.). Based on this identification, the appropriate parser is selected to extract structured information from the report and transform it into a unified JSON schema. This schema includes vulnerabilities, severity levels, affected components, and remediation suggestions.

**ii. Summarization and Embedding Generation:**

The structured JSON is used in two parallel operations:

A prompt template is constructed and fed to a locally hosted GGUF-format LLM (Mistral) to generate a human-readable summary of the report. This summary is stored in a runtime context instance for future interaction.

Simultaneously, the parsed report is chunked and passed through a fine-tuned all-mpnet-base sentence transformer, which generates dense vector embeddings for each chunk. These embeddings are stored in a temporary, report-specific namespace in Pinecone, a secure, cloud-based vector database. This vector store is ephemeral and is deleted after the session to ensure data privacy.

**iii. User Interaction and Query Classification:**

When a user submits a query through the chatbot interface, the system first performs query type classification to determine whether the question pertains to the uploaded report or is a general cybersecurity query. The query is then embedded using the same fine-tuned all-mpnet model, and a cosine similarity search is performed:

* For report-specific queries, the system searches within the report’s temporary Pinecone namespace.
* For general queries, it searches a persistent cybersecurity knowledge base (KB) namespace, populated with pre-embedded QA pairs.

**iv. Knowledge Base and Finetuning:**

The knowledge base is constructed from manually curated QA pairs covering topics such as OWASP Top 10, network scanning, cloud security, SSL, mobile app security, and web application vulnerabilities. These QA pairs are categorized into eight categories: Basic Understanding, Technical Explanation, Vulnerability Identification, Prevention Methods, Example Scenarios, References, Statistics, Proactive Suggestions

These QA pairs not only serve as the basis for the general cybersecurity KB but were also used to fine-tune the all-mpnet sentence transformer, ensuring optimized semantic understanding for both user queries and content retrieval.

**v. Prompt Construction and Response Generation:**

Once relevant content is retrieved, the system dynamically constructs a contextual prompt by combining:

* Retrieved chunks from Pinecone
* The stored summary and structured context
* The user's original question

This prompt is then fed into the locally hosted GGUF-based Mistral model, which generates a precise, human-readable response. Whether the question is about a vulnerability’s impact, recommended mitigation, or a general security concept, the model provides accurate and contextualized answers without relying on any third-party APIs or exposing user data.

**vi. Secure, Interactive Chat Interface:**

All interactions occur via a browser-based chat interface. Users can ask follow-up questions, clarify previous responses, or explore detailed aspects of the findings. The interface maintains a lightweight session memory to ensure coherent, state-aware conversations.

This multi-layered chatbot architecture combines advanced natural language processing with secure data handling to offer an explainable and privacy-respecting cybersecurity assistant. By merging local LLM inference, semantic search, report understanding, and fine-tuned embeddings, the system transforms technical scan data into a fully interactive and intelligible experience for both technical and non-technical users.

**2.3. Executable Agent for Endpoint Protection (Proposed Work)**

As an extension of the centralized scanning platform, we propose a lightweight, standalone executable agent designed to perform localized vulnerability assessment and remediation directly on the user’s device. The agent is intended to operate independently, without the need for an active internet connection or external data transmission, ensuring that all processing and remediation remain confined to the local environment.

The agent will be capable of scanning for common host-level exposures such as open ports, insecure configurations, and unnecessary services. Based on these findings, it will guide the user through remediation actions such as whitelisting essential services and blocking unwanted or potentially vulnerable ports via system-level firewall integration. The interface will be designed to support both technical and non-technical users, providing clear explanations of risks and recommended actions.

A working prototype has been developed with initial functionality focused on local port scanning and interactive firewall rule enforcement on Windows systems. The current version identifies the device’s IP address, detects open TCP ports, presents this information to the user, and allows them to selectively block or ignore specific ports.

Future development plans include adding continuous monitoring for port and service state changes, detecting insecure local configurations, and syncing endpoint data with the central dashboard. The agent may also incorporate automated risk scoring and context-aware remediation recommendations. Although still in early stages, this component is envisioned as a critical part of the system’s move toward active, host-level cybersecurity management. Beyond initial scanning and remediation, the executable agent is also envisioned to support lightweight, continuous monitoring functionalities.

This would include periodic checks for changes in port status, newly exposed services, or unauthorized configuration modifications. By operating in the background with minimal system overhead, the agent can alert users to potential risks as they emerge, enabling timely intervention. These monitoring features will be designed to run securely and efficiently without compromising user privacy. Over time, the agent is expected to evolve into a proactive endpoint defense mechanism that complements external scanning by providing persistent, device-level threat awareness and response.

# **IMPLEMENTATION**

**3.1 Web-Based Scanning Platform:**

The web-based component of the system serves as the primary user interface for initiating scans, navigating between scanning modules, and accessing the integrated chatbot assistant. The backend is implemented using the Flask web framework in Python, providing RESTful routing for scan execution, file handling, and UI navigation. The front-end interface is constructed using standard HTML templates served through Flask, offering a straightforward and user-friendly experience.

The web application includes support for all major scanning modules detailed in Previous section, including web, network, cloud, mobile, database, and SSL scanning. Each tool is exposed through its own route, and scan results are captured and stored for downstream processing. In addition, the platform features a dedicated route that links directly to the chatbot interface, allowing users to transition from scan initiation to AI-assisted interpretation seamlessly.

Upon completion of each scan, the system generates two versions of the report. The first is a full version intended for the end-user, containing all discovered vulnerabilities, technical details, and metadata. The second version is a privacy-aware report in which sensitive information—such as internal IP addresses, system identifiers, or credentials—is masked or redacted. This masked version is passed to the chatbot for parsing and summarization, ensuring that the AI module never processes identifiable or sensitive data, thereby maintaining strict data confidentiality.

While complete system-level details are under internal development, the current deployment leverages daemon-based processes to handle scanning tools available on the server. These daemons ensure asynchronous, non-blocking execution of resource-intensive scans while maintaining responsiveness in the user interface. The platform is currently in the delivery stage, with core components built and integrated, and ongoing refinement of the UI and backend structure to improve usability and stability.

**3.2. AI powered Report Interpretation and Chat Interface:**

The AI component of the system is designed to interpret technical vulnerability reports and allow users to interact with them using natural language. This module is fully functional and integrated into the web platform, providing report summarization, semantic search, and chatbot-based question answering. The primary goal is to help both technical and non-technical users understand scan results clearly and securely.

At the core of the system is a locally hosted large language model, based on Mistral-7B in GGUF format, running through the efficient llama.cpp inference framework. This local setup ensures that all report data and user queries are processed securely without depending on external APIs or cloud-based AI services.

When a scan report is generated, it goes through a multi-step processing pipeline. First, the raw report is parsed and key information is extracted to produce a structured JSON format. A summary prompt is then constructed from this structured data and passed to the LLM to generate a concise, readable overview of the report. In parallel, the report is broken into chunks and processed using a fine-tuned all-mpnet-base-v2 sentence transformer to generate embeddings for each section of the report. These embeddings are stored in a temporary namespace in Pinecone, a secure cloud-based vector database. The vector store is session-specific and is deleted after use to maintain privacy.

When a user asks a question through the chatbot, the system first classifies the query using a semantic classification mechanism. This determines whether the question relates to the uploaded report or is a general cybersecurity question. The query is then embedded using the same fine-tuned transformer model. If the question is related to the report, it is compared against the report’s vector store using cosine similarity, retrieving the most relevant chunks. For general questions, the system queries a persistent knowledge base, which contains pre-embedded QA pairs across major cybersecurity topics.

The retrieved information, along with the stored summary, is used to build a context-specific prompt. This prompt is passed to the local LLM, which then generates a natural language response that is shown to the user in the chat interface.

The sentence transformer model was fine-tuned using a curated set of QA pairs covering topics like OWASP Top 10, cloud security, SSL/TLS, port scanning, and mobile app vulnerabilities. These QA pairs were grouped into categories such as basic understanding, technical explanation, vulnerability identification, prevention methods, example scenarios, references, statistics, and proactive suggestions. This dataset not only improved the quality of semantic search but also formed the core of the system’s general-purpose cybersecurity knowledge base.

To protect user privacy, a second “masked” version of each scan report is created by redacting sensitive data (such as internal IPs and system details) before it is used by the AI pipeline. This ensures that while the chatbot provides meaningful, context-aware responses, it never processes confidential or identifying information.

Overall, this module combines local inference, fine-tuned semantic search, and structured prompting to provide accurate, secure, and user-friendly interpretations of technical cybersecurity reports.

**3.3 Executable Agent for Endpoint Protection (Proposed Work)**

The executable agent extends the platform's capabilities to the endpoint level, enabling localized vulnerability detection and basic remediation from within the user’s device. Unlike traditional scanners, this component is designed to interact directly with the host operating system, providing real-time visibility and user-guided control. It is built to work in close integration with the web platform, enabling future bidirectional communication and centralized management.

Developed in Python, the agent supports both Windows and Linux environments through a shared core logic, with system-specific command sets implemented for compatibility. Its current functionality includes automatic IP detection, TCP and UDP port scanning, and firewall rule application. Upon execution, the agent scans for open ports and presents a list of detected services to the user. The interface allows users to whitelist necessary ports and block others. Blocking operations are performed using native system commands—PowerShell on Windows and tools like iptables or ufw on Linux—followed by socket-level verification to confirm whether the selected ports have been successfully closed.

Future enhancements will include continuous monitoring of service state changes, detection of insecure system configurations, and secure synchronization with the central dashboard to provide unified visibility into endpoint security.

While still under development, the agent already demonstrates the feasibility of a lightweight, modular, and privacy-conscious endpoint protection layer, capable of empowering users to take direct control of local vulnerabilities within a broader cybersecurity platform.

# RESULTS AND DISCUSSION

To assess the system’s performance and validate the accuracy of its components, we conducted a combination of quantitative and functional evaluations. These included testing the AI-powered chatbot’s ability to summarize and answer questions from structured cybersecurity reports, as well as validating the executable agent and web platform's integration and functionality.

## **Evaluation Setup for Chatbot Summarization and Q&A:**

## The chatbot was evaluated using a set of 10 representative test cases derived from real-world scan reports. Each test case involved a report summary prompt and multiple user queries, covering various domains such as web application security, open ports, SSL/TLS misconfigurations, and cloud auditing. Human-written responses were used as references. To evaluate the quality of the chatbot's responses, we used standard NLP metrics: BLEU for lexical precision, and ROUGE-1, ROUGE-2, and ROUGE-L for recall-based semantic similarity and structural overlap.

## **B. Performance Results:**

The system was tested across 10 samples, and the average scores observed were BLEU: 0.299, ROUGE-1: 0.591, ROUGE-2: 0.345, and ROUGE-L: 0.548, as summarized below:

|  |  |  |  |
| --- | --- | --- | --- |
| **BLEU** | **R1** | **R2** | **R\_L** |
| 0.290368714 | 0.610169487 | 0.375999995 | 0.598870051 |
| 0.294257954 | 0.590476185 | 0.316831678 | 0.523809519 |
| 0.274158942 | 0.587570616 | 0.330645156 | 0.576271181 |
| 0.398859179 | 0.680851059 | 0.484536077 | 0.666666662 |
|  | 0.666666662 | 0.526666662 | 0.666666662 |
| 0.517717643 | 0.493827156 | 0.214876028 | 0.444444439 |
| 0.296972816 | 0.596858634 | 0.293906805 | 0.523560204 |
| 0.213087342 | 0.56790123 | 0.267441856 | 0.518518514 |
| 0.205137061 | 0.511904757 | 0.252100835 | 0.488095233 |
| 0.239568625 | 0.623255809 | 0.349514558 | 0.55813953 |
| 0.295142234 | 0.592948159 | 0.341251965 | 0.5565042 |

These results indicate that the chatbot provides responses with strong semantic alignment to human references. The ROUGE-1 and ROUGE-L scores reflect a high degree of key phrase and structural overlap, suggesting that the system captures the essential findings and remediation strategies effectively. Although BLEU scores are moderate, this is expected in tasks where flexibility of language and concept-level understanding are prioritized over exact lexical matching. The variation across test cases reflects differences in query complexity and specificity, with higher scores observed in well-structured report-driven queries.

***C. Executable Agent Testing:***

The executable agent was evaluated for its functional reliability on both Windows and Linux systems. Testing confirmed successful detection of open TCP and UDP ports, correct interpretation of user input for port whitelisting or blocking, and accurate application of system-level firewall rules using platform-specific commands. Post-action verification using socket-level checks demonstrated that blocked ports were consistently enforced. The tool responded with minimal latency and executed as expected across multiple test runs.

***D. Web Platform Validation:***

The web interface was evaluated for usability, modular scanner integration, and transition to the chatbot system. End-to-end workflow testing validated that users can initiate scans, view results, and access the AI assistant for summarization and interactive Q&A. The dual-report design comprising a user-facing report and a masked version for AI processing was confirmed to work as intended, preserving sensitive information during chatbot inference.

***E. Discussion:***

The evaluation results indicate that the system performs effectively across multiple dimensions. The chatbot’s ability to extract, summarize, and explain findings is supported by strong ROUGE metrics, and its reliance on locally hosted inference ensures both performance and privacy. The executable agent has proven viable for real-time port-level remediation, with a clear path for expanded capabilities. The web platform’s smooth integration with scanning tools and the AI pipeline supports a cohesive and user-friendly experience.

While these results are promising, further evaluation particularly with real user feedback and broader datasets is planned to benchmark scalability, latency, and usability under production-like workloads. In future iterations, we also aim to test the full pipeline’s effectiveness in organizational settings where multiple scans and users are managed concurrently.