**CS5342 Network Security**

**Group-based Project**

**Network Security Tutor & Quiz Agent**

**Rishitha Sana Vanapalli**

**R11967143**

This project demonstrates a Network Security Tutor & Quiz Agent developed to help users learn core concepts of network security. The system operates entirely on a local machine, consisting of a interface (running on port 3000) and a backend (running on port 8000). When the user enters a question, the backend retrieves relevant content from a Chroma vector database, forwards it to the Ollama-hosted Llama 3.2:3b model, and returns the generated response to the frontend. All communication takes place through 127.0.0.1, ensuring that every data transaction query submission, database retrieval, and model inference remains fully local with no external network interaction. The Wireshark packet captures collected for each prompt clearly illustrate how data travels from the browser to the backend, how the Django server interacts with the Chroma DB and the Llama 3.2:3b model, and how the final HTTP response is sent back to the user interface.

**Prompt 1: What is a Man-in-the-Middle (MITM) attack and how can it be prevented?**

A screenshot of a computer

AI-generated content may be incorrect.

**Step 1 – User Prompt**

The user entered the following question in the locally hosted AI Tutor web interface:

“What is a Man-in-the-Middle (MITM) attack and how can it be prevented?”

This request was sent through the local endpoint http://127.0.0.1:5000/tutor.

The backend Flask server processed the query via the local vector database (ChromaDB) that stored embeddings of lecture slides and textbook chapters related to network security.

**Step 2 – Processing / Local Model Interaction**

After the “Ask” button was clicked, the Flask application invoked the Sentence-Transformer model (all-MiniLM-L6-v2) to create an embedding vector for the user prompt.

This vector was compared against the “network\_security” collection in ChromaDB, retrieving the most relevant document chunks from:

* Katz\_Lindell\_IntroModernCrypto.txt
* Stallings\_Cryptography\_and\_Network\_Security.txt
* Network-Security-Essentials-6th-Edition-William-Stallings.pdf

The retrieved context was then passed to the locally hosted Llama 3.2 model running through Ollama, which generated a comprehensive textual response.

All processing occurred offline; no external API calls or internet access were required.

**Step 3 – Model Output**

The model generated a structured explanation:

What is an MITM attack? An active attack where the adversary intercepts and relays messages between two legitimate users, making each believe they are communicating directly.

How it works: The attacker spoofs one or both IP addresses to insert themselves in the communication path.

Prevention methods: Use HTTPS (SSL/TLS), verify sender identities, enable two-factor authentication, keep systems updated, and use VPNs on public networks.

Sources identified automatically by the system:

Katz\_Lindell\_IntroModernCrypto.txt, Stallings\_Cryptography\_and\_Network\_Security.txt, and Network-Security-Essentials-6th-Edition-William-Stallings.pdf.

This verifies the response was fully grounded in locally stored course materials.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 4 – Trace Data Analysis (Wireshark Observation)**

Wireshark was run simultaneously on interface lo0 (loopback) to monitor the data flow.

Captured packets show:

Source / Destination: 127.0.0.1 ↔ 127.0.0.1

Protocols: TCP / HTTP / JSON

Source Port: 60819

Destination Port: 60830

Representative Frame: HTTP 2019 HTTP/1.1 200 OK (text/html)

Additional Info: ACK and FIN packets confirming session completion

This proves that:

All communication stayed within the local environment (frontend ↔ Flask backend).

No external IPs, DNS queries, or internet transmissions occurred.

The packets correspond to local HTTP POST requests carrying the prompt and the JSON-formatted model response.

**Prompt 2: Explain how Denial of Service (DoS) and Distributed DoS (DDoS) attacks function.**

A screenshot of a computer

AI-generated content may be incorrect.

**Step 1 – User Prompt**

The user entered the following query in the local AI Tutor web interface:

“Explain how Denial of Service (DoS) and Distributed DoS attacks function.”

This prompt was issued to the locally hosted server at 127.0.0.1:5000/tutor.

The Flask backend accepted the HTTP POST request and routed it to the internal processing module connected to ChromaDB, which stored embedded lecture and textbook content related to network attack mechanisms.

**Step 2 – Processing / Local Model Interaction**

After receiving the query, the Flask server used the all-MiniLM-L6-v2 sentence-embedding model to transform the question into a vector representation.

The application then performed a semantic similarity search inside the “network\_security” vector database, retrieving top-ranked content segments from:

* Stallings\_Cryptography\_and\_Network\_Security.txt
* Network-Security-Essentials-6th-Edition-William-Stallings.pdf
* NIST\_SP800-53.txt

The context retrieved was passed to the local LLaMA 3.2 model hosted via Ollama for offline inference.

No external API calls or internet connections were established; the full pipeline—from vector retrieval to text generation—remained on the local machine.

**Step 3 – Model Output**

The model produced a comprehensive explanation distinguishing DoS and DDoS attacks:

Denial of Service (DoS) – An attacker floods a target system or network with excessive malicious traffic (packets, requests, or data), exhausting its resources and making it unavailable to legitimate users.

Distributed Denial of Service (DDoS) – Multiple compromised hosts (botnets) simultaneously send traffic toward a target, masking the attacker’s identity and amplifying the scale.

Key Characteristics:

Single point of failure for DoS.

Network-wide impact for DDoS.

Legitimate user access blocked in both cases.

Sources Retrieved Automatically:

NIST\_SP800-53.txt, Stallings\_Cryptography\_and\_Network\_Security.txt, Network-Security-Essentials-6th-Edition-William-Stallings.pdf, and Security-Private-Communication-in-a-Public-World-2nd-Ed.pdf.

This confirms the answer was generated exclusively from local course resources.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 4 – Trace Data Analysis (Wireshark Observation)**

Wireshark monitoring on interface lo0 (Loopback) captured packet activity corresponding to this query (see 2b.png).

Observed details include:

Source/Destination: 127.0.0.1 ↔ 127.0.0.1

Source Port: 60819

Destination Port: 60830

Protocols: TCP / HTTP / JSON

Representative Frame: HTTP 200 OK (text/html) and HTTP application/json responses

Additional Frames: ACK and FIN packets confirming session closure

Interpretation:

The data exchange occurred entirely on the loopback interface; no external IP addresses or DNS queries were present.

Packets correspond to local client-server communication between the Streamlit/Flask frontend and backend.

The JSON payload observed in Wireshark contained only the user prompt and model response metadata.

**Prompt 3: What is IP spoofing and how can it be detected?**

A screenshot of a computer

AI-generated content may be incorrect.

**Step 1 – User Prompt**

The user entered the following query in the local Network Security Tutor interface:

“What is IP spoofing and how can it be detected?”

This request was sent to the locally hosted web service running at 127.0.0.1:5000/tutor.

The Flask backend received the HTTP POST request and routed it to the application’s internal logic that interacts with the local knowledge base of embedded network-security documents.

**Step 2 – Processing / Local Model Interaction**

After submission, the backend used the Sentence-Transformer model (all-MiniLM-L6-v2) to convert the user’s prompt into an embedding vector.

The system then queried the ChromaDB vector database within the “network\_security” collection to locate relevant chunks from:

* Stallings\_Cryptography\_and\_Network\_Security.txt
* Network-Security-Essentials-6th-Edition-William-Stallings.pdf
* Firewalls\_and\_IP\_Filtering.txt

The retrieved context was passed to the locally hosted LLaMA 3.2 model running under Ollama for response generation.

All computations—including vector search and inference—occurred locally without any API calls or external connections.

**Step 3 – Model Output**

The local LLM produced a clear, multi-section answer covering both the concept and the countermeasures:

What is IP Spoofing: An attack in which a malicious actor forges the source IP address in packets to masquerade as a trusted host, bypassing access controls or filters.

How it Works: The attacker sends packets with a spoofed source address to a target host or router, which then processes them as though they came from an authorized system.

Detection Methods:

Discard packets with internal source IPs arriving from external interfaces.

Validate source addresses and implement IP verification algorithms (e.g., time-of-flight tests, hash-based identification).

Employ firewall rules and packet timestamp analysis.

Best Practices: Regularly monitor firewall logs and deploy source validation filters to detect spoofed traffic.

Sources Used Automatically:

Stallings\_Cryptography\_and\_Network\_Security.txt, Network-Security-Essentials-6th-Edition-William-Stallings.pdf, and Firewalls\_and\_IP\_Filtering.txt.

The citation trail confirms the response was derived solely from local documents, maintaining data privacy.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 4 – Trace Data Analysis (Wireshark Observation)**

Wireshark monitoring on interface lo0 (Loopback) captured traffic corresponding to this query.

Captured details include:

Source / Destination: 127.0.0.1 ↔ 127.0.0.1

Source Port: 60819

Destination Port: 60830

Protocols: TCP / HTTP / JSON

Representative Frame: HTTP 200 OK (text/html) and HTTP application/json responses

Supporting Packets: Typical ACK and FIN handshakes indicating local session completion

Interpretation:

All communication occurred entirely on the loopback interface without external IP traffic or DNS queries.

The JSON payload contained the prompt and model response metadata, confirming closed-loop local operation.

No outbound connections were made to external servers or cloud APIs.

**Prompt 4: Describe the concept of port scanning and its preventive measures.**

A screenshot of a computer

AI-generated content may be incorrect.

**Step 1 – User Prompt**

The user typed the following question into the locally hosted AI Tutor interface:

“Describe the concept of port scanning and its preventive measures.”

This request was sent to the private local web server at 127.0.0.1:5000/tutor.

The Flask backend received the HTTP POST request and directed it to the local vector database for semantic retrieval from stored security materials.

**Step 2 – Processing / Local Model Interaction**

Upon submission, the backend’s Sentence-Transformer model (all-MiniLM-L6-v2) embedded the question into a dense numerical vector.

The system queried the ChromaDB vector database (collection “network\_security”) and retrieved relevant context from:

* Stallings\_Cryptography\_and\_Network\_Security.txt
* Network-Security-Essentials-6th-Edition-William-Stallings.pdf
* Scanning\_and\_Network\_Defense\_Techniques.txt

These text chunks were then supplied to the locally hosted LLaMA 3.2 model (via Ollama) for inference.

The entire pipeline from embedding lookup to natural-language generation—operated offline, ensuring no internet connectivity or data transmission beyond localhost.

**Step 3 – Model Output**

The model returned a clear explanation structured into conceptual and practical sections:

What is Port Scanning?

Port scanning is an active reconnaissance technique used to identify open or closed network ports on a host, revealing running services or potential vulnerabilities.

How it Works:

The scanner sends a SYN (initialization) packet to each target port.

If the host replies with SYN-ACK, the port is open; if it returns RST, the port is closed.

Types of Scans:

Full Scan: tests all ports in a specified range.

Half Scan: probes selected odd-numbered ports.

SYN Scan: sends only the SYN packet to detect open ports stealthily.

Preventive Measures:

Configure firewalls to restrict unnecessary inbound traffic.

Use Access Control Lists (ACLs) to limit host-to-host communication.

Apply port redirection / NAT to mask internal open ports.

Employ network segmentation to isolate subnetworks and reduce the attack surface.

Local Sources Referenced:

Stallings\_Cryptography\_and\_Network\_Security.txt,

Network-Security-Essentials-6th-Edition-William-Stallings.pdf, and

Scanning\_and\_Network\_Defense\_Techniques.txt.

This confirms the explanation was derived entirely from offline materials.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 4 – Trace Data Analysis (Wireshark Observation)**

Wireshark captured local loopback activity for this query on interface lo0 (see 4b.png).

Captured Details:

Source / Destination: 127.0.0.1 ↔ 127.0.0.1

Source Port: 60819

Destination Port: 60830

Protocols: TCP / HTTP / JSON

Representative Frame: HTTP 1.1 200 OK (text/html) with JSON payload

Additional Frames: TCP ACK and FIN packets completing the local session

Interpretation:

Traffic remained strictly within the localhost environment.

The captured frames correspond to internal communication between the Streamlit/Flask frontend and the backend LLM service.

No external IPs, DNS queries, or third-party API requests were detected.

**Prompt 5: Generate and attempt a quiz on network attacks and defense mechanisms**

A screenshot of a computer

AI-generated content may be incorrect.

**Step 1 – User Prompt**

The user selected the Quiz Agent option from the locally hosted Network Security Tutor interface and entered the following configuration:

Quiz Type: Specific Topic

Topic: Network attacks and defense mechanisms

Number of Questions: 4

This command was processed through the web application running on the local endpoint 127.0.0.1:5000/quiz.

The frontend transmitted the request to the Flask backend using an HTTP POST operation.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**Step 2 – Processing / Local Model Interaction**

The backend initiated the Quiz Generation Agent workflow:

The Sentence Transformer model (all-MiniLM-L6-v2) embedded the topic phrase (“network attacks and defense mechanisms”) for semantic retrieval.

The system queried the ChromaDB vector database (collection network\_security) to extract context material from:

* lecture\_slide\_Formatted → lecture\_23.txt
* textbook → Network-Security-Essentials-6th-Edition-William-Stallings.pdf
* textbook → Computer-Networking-A-Top-Down-Approach-8th-Edition.pdf

Retrieved segments were then formatted into multiple-choice, true/false, and short-answer question types.

The local LLaMA 3.2 model (via Ollama) generated final question text, answer options, and answer-key validation—all executed offline.

No data was transmitted externally, satisfying the privacy-preserving requirement.

A screenshot of a quiz

AI-generated content may be incorrect.

A screenshot of a computer error

AI-generated content may be incorrect.

**Step 3 – Model Output**

The generated quiz contained four questions:

Q1: What is a common defense mechanism to prevent attackers from intercepting communication?

Answer: A) Encryption (TLS)

Q2: A Man-in-the-Middle attack can be defended against using only TLS encryption.

Answer: False

Q3: Explain how network security addresses the threat of attacks on computer networks.

Answer: The system evaluates conceptual understanding, focusing on threat prevention and mitigation.

Q4: What type of attack occurs when an attacker bombards a system with traffic to consume resources?

Answer: B) Denial of Service (DoS)

User Score: 3/4 (as shown in the quiz result screen).

Feedback and explanations for each question were automatically generated and referenced back to the appropriate local documents.

This confirms the quiz agent used only embedded and local sources.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 4 – Trace Data Analysis (Wireshark Observation)**

The Wireshark capture (5b.jpeg) recorded the communication activity during quiz generation and submission using the loopback interface (lo0).

Captured Data Summary:

Source / Destination: 127.0.0.1 ↔ 127.0.0.1

Protocols: TCP / HTTP / JSON

Representative Frame: HTTP 1.1 200 OK (application/json) and HTTP 200 OK (text/html)

Observation: JSON payload contained quiz structure, user answers, and evaluation metrics.

Additional Frames: ACK and FIN packets confirmed proper session closure.

Interpretation:

All packets indicate local communication only between the frontend (browser) and backend (Flask app).

No DNS requests, external IPs, or third-party services were accessed.

The captured session validates the offline processing and secure local data flow for quiz generation and grading.