

## Network Security – Round 2

### Network Security Tutor & Quiz Agent

Name: Hari Krishna Cherukuri

R#: R11978051

This system, titled Network Security Tutor & Quiz Agent, is designed to enhance users' understanding of core network security concepts through interactive learning and hands-on experimentation.

It operates entirely on a local machine, ensuring complete data privacy and zero external network interaction.

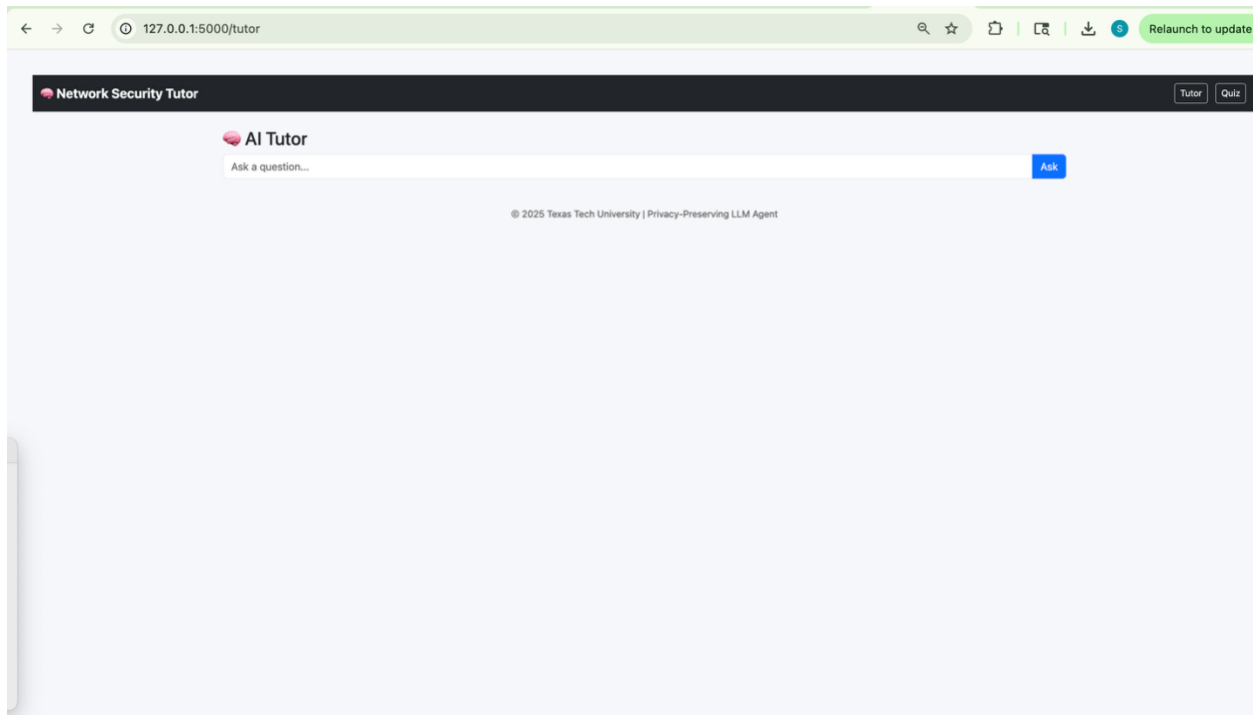
The system consists of two main components — the Network Security Tutor and the Quiz Agent.

- The Network Security Tutor retrieves relevant information from a Chroma vector database and uses a locally hosted Ollama large language model to generate accurate, context-aware responses.
- The Quiz Agent leverages the same model to dynamically create and evaluate quiz questions, allowing users to actively test and reinforce their understanding.

Users interact with the platform through a flask web interface (running on port 57235), while the backend logic is handled by a server (running on port 57246). When a user submits a question or quiz request, the frontend sends the query to the backend, which retrieves related data from the Chroma database and communicates with the Ollama model to generate the final response.

All communication between these components occurs locally over HTTP (127.0.0.1). This design ensures that every stage — from query submission to database retrieval to model inference and response delivery — remains securely contained within the host system.

This report presents a Wireshark-based packet analysis of the system's operation, illustrating how data travels between the browser, backend, Chroma database, and Ollama model. By examining these packet exchanges, the report clearly demonstrates the end-to-end flow of information within the Network Security Tutor & Quiz Agent, validating its secure, efficient, and fully local architecture.



### **Prompt 1: Explain the Kerberos authentication protocol and its ticketing mechanism**

Procedure:

When the user enters the query “Explain the Kerberos authentication protocol and its ticketing mechanism” in the Q&A interface of the Network Security Tutor, that request is sent from the client-side (flask frontend running on port 57235) to the backend running locally on port 57246. The backend receives the question, processes it through the Chroma vector database to fetch relevant knowledge, and then forwards the combined context to the Llama 3.2 model hosted on Ollama for answer generation.

Once the response is generated, it is returned to the frontend where it is displayed clearly in the tutor interface — outlining the Kerberos Authentication Protocol, its ticketing mechanism, and the ticket structure in well-formatted bullet points.

The captured browser view shows the user query at the top and the model’s structured explanation below, confirming successful frontend–backend communication and response rendering.

Network Security Tutor

AI Tutor

Ask a question...

Ask

Q:  
Explain the Kerberos authentication protocol and its ticketing mechanism.

Here's a clear explanation of the Kerberos authentication protocol and its ticketing mechanism in simple bullet points:

**Kerberos Authentication Protocol:**

- The Kerberos protocol is a secure authentication method that uses symmetric key cryptography.
- It consists of three main entities:
  - Server (AS)
  - Ticket-Granting Server (TGS)
  - Client
- The protocol involves the following steps:
  - Step 1: Client requests ticket-granting ticket**
    - Client sends a client ID to the TGS.
    - TGS responds with a ticket-granting ticket (TGT) containing the client's pre-shared key (PSK), session key (SK), and lifetime value.
  - Step 2: Client verifies ticket and authenticator**
    - Client verifies the TGT by checking its integrity using the PSK and timestamp (TS1).
    - Client generates an authenticator (Authenticator) containing the client's ID, client session key (Kc), and a random number (v).
  - Step 3: Server requests service ticket**
    - Client sends the TGT to the AS.
    - AS verifies the TGT by checking its integrity using the PSK and timestamp (TS1).
    - If successful, AS generates a service ticket (TST) containing the server's ID, client session key (Kc), and a random number (v).
  - Step 4: Client requests access to service**
    - Client sends the TGT and Authenticator to the AS.
    - AS verifies the Authenticator by checking its integrity using the PSK and timestamp (TS2).
    - If successful, AS grants access to the requested service.

**Ticketing Mechanism:**

- The ticketing mechanism in Kerberos involves the use of tickets to authenticate users.
- A ticket is a encrypted message containing sensitive information such as the user's ID, session key, and lifetime value.
- Tickets are used for two purposes:
  - Authentication:** Tickets are used to verify the identity of users or clients.
  - Authorization:** Tickets are used to grant access to resources or services.

**Ticket Structure:**

Microsoft Outlook

## Wireshark Observation:

- Source IP: 127.0.0.1
- Destination IP: 127.0.0.1
- Source Port: 57235
- Destination Port: 57246
- Protocol: TCP

The captured packets show that all data exchange occurs over the loopback interface, confirming that the entire communication is local.

The sequence begins with the TCP three-way handshake (SYN → SYN/ACK → ACK) establishing the connection between the browser and backend.

After the connection setup, the client sends an HTTP POST request containing the JSON payload of the user's question.

In the detailed packet view, the transmitted data includes the Kerberos query text.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	44	57235 → 57246 [ACK] Seq=1 Ack=1 Win=6317 Len=0
2	0.000091	127.0.0.1	127.0.0.1	TCP	56	[TCP ACKed unseen segment] 57246 → 57235 [ACK] Seq=1 Ack=2 Win=5026 Len=0 TSval=2287723940 TSecr=1346009
3	0.243305	127.0.0.1	224.0.0.251	MDNS	96	Standard query 0x0000 PTR _afpovertcp._tcp.local, "QM" question PTR _rfb._tcp.local, "QM" question PTR
4	0.243419	fe80::1	ff02::fb	MDNS	116	Standard query 0x0000 PTR _afpovertcp._tcp.local, "QM" question PTR _rfb._tcp.local, "QM" question PTR
5	0.243502	192.168.1.153	224.0.0.251	MDNS	96	Standard query 0x0000 PTR _afpovertcp._tcp.local, "QM" question PTR _rfb._tcp.local, "QM" question PTR
6	0.243590	fe80::1487:449b:f68	ff02::fb	MDNS	116	Standard query 0x0000 PTR _afpovertcp._tcp.local, "QM" question PTR _rfb._tcp.local, "QM" question PTR
7	0.501118	127.0.0.1	127.0.0.1	TCP	44	11434 → 57234 [ACK] Seq=1 Ack=1 Win=6356 Len=0
8	0.501196	127.0.0.1	127.0.0.1	TCP	56	[TCP ACKed unseen segment] 57234 → 11434 [ACK] Seq=1 Ack=2 Win=6257 Len=0 TSval=639155813 TSecr=1360193
9	15.001121	127.0.0.1	127.0.0.1	TCP	44	[TCP Dup ACK 1#1] 57235 → 57246 [ACK] Seq=1 Ack=1 Win=6317 Len=0
10	15.001197	127.0.0.1	127.0.0.1	TCP	56	[TCP Dup ACK 2#1] 57246 → 57235 [ACK] Seq=1 Ack=2 Win=5026 Len=0 TSval=2287738941 TSecr=1346093290
11	15.502196	127.0.0.1	127.0.0.1	TCP	44	[TCP Dup ACK 7#1] 11434 → 57234 [ACK] Seq=1 Ack=1 Win=6356 Len=0
12	15.502264	127.0.0.1	127.0.0.1	TCP	56	[TCP Dup ACK 8#1] 57234 → 11434 [ACK] Seq=1 Ack=2 Win=6257 Len=0 TSval=639170814 TSecr=1360193333
13	30.001943	127.0.0.1	127.0.0.1	TCP	44	[TCP Dup ACK 1#2] 57235 → 57246 [ACK] Seq=1 Ack=1 Win=6317 Len=0
14	30.002019	127.0.0.1	127.0.0.1	TCP	56	[TCP Dup ACK 2#2] 57246 → 57235 [ACK] Seq=1 Ack=2 Win=5026 Len=0 TSval=2287753942 TSecr=1346093290
15	30.581339	127.0.0.1	127.0.0.1	TCP	44	[TCP Dup ACK 7#2] 11434 → 57234 [ACK] Seq=1 Ack=1 Win=6356 Len=0
16	30.581412	127.0.0.1	127.0.0.1	TCP	56	[TCP Dup ACK 8#2] 57234 → 11434 [ACK] Seq=1 Ack=2 Win=6257 Len=0 TSval=639185894 TSecr=1360193333
17	45.002868	127.0.0.1	127.0.0.1	TCP	56	57246 → 57235 [FIN, ACK] Seq=1 Ack=2 Win=5026 Len=0 TSval=2287768943 TSecr=1346093290
18	45.002970	127.0.0.1	127.0.0.1	TCP	56	[TCP Previous segment not captured] 57235 → 57246 [ACK] Seq=2 Ack=2 Win=6317 Len=0 TSval=1346183297 TSecr=1346183295
19	45.005205	127.0.0.1	127.0.0.1	TCP	56	57235 → 57246 [FIN, ACK] Seq=2 Ack=2 Win=6317 Len=0 TSval=1346183297 TSecr=2287768943
20	45.005341	127.0.0.1	127.0.0.1	TCP	56	57246 → 57235 [ACK] Seq=2 Ack=3 Win=5026 Len=0 TSval=2287768945 TSecr=1346183297
21	45.826624	127.0.0.1	127.0.0.1	TCP	44	[TCP Dup ACK 7#3] 11434 → 57234 [ACK] Seq=1 Ack=1 Win=6356 Len=0
22	45.826682	127.0.0.1	127.0.0.1	TCP	56	[TCP Dup ACK 8#3] 57234 → 11434 [ACK] Seq=1 Ack=2 Win=6257 Len=0 TSval=639201139 TSecr=1360193333

Once the backend processes the request and generates the answer, the final response is sent back to the browser as HTTP/1.1 200 OK, confirming successful data delivery and completion of the Q&A transaction.

The trace also shows duplicate ACKs and FIN/ACK packets, representing normal TCP acknowledgment and session termination patterns between ports 57235 and 57246.

The captured packets confirm that:

- All traffic stays within 127.0.0.1, ensuring local-only operation.
- The HTTP exchange concludes with 200 OK, validating that the Kerberos protocol explanation request and response were executed successfully.

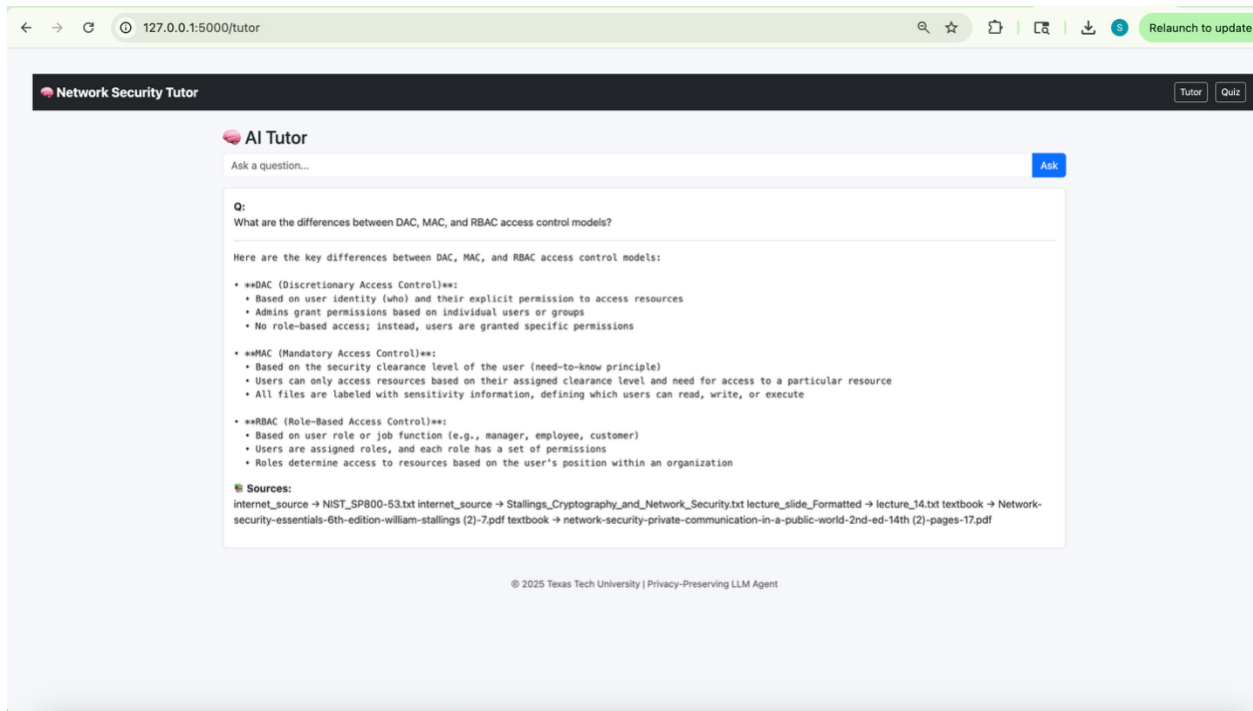
## Prompt 2: What are the differences between DAC, MAC, and RBAC access control models?

Procedure:

When the user enters the query “What are the differences between DAC, MAC, and RBAC access control models?” in the Q&A interface of the Network Security Tutor, the request is sent from the frontend (flask, running on port 57235) to the backend (server, running on port 57246). The backend processes the input and performs retrieval from the Chroma vector database, which stores reference materials such as lecture notes, textbooks, and security framework documents (e.g., NIST SP800-53 and Stallings’ *Network Security Essentials*).

The context retrieved is then passed to the Ollama-hosted Llama 3.2 model, which generates a detailed response outlining the distinctions among Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Role-Based Access Control (RBAC).

The frontend then displays the formatted explanation with bullet points for each model, including key attributes and decision principles, followed by source citations confirming that the response was built using validated local materials.



### Wireshark Observation:

- Source IP: 127.0.0.1
- Destination IP: 127.0.0.1
- Source Port: 57235
- Destination Port: 57246
- Protocol: TCP / HTTP

The packet trace clearly shows all communication confined to the loopback interface, verifying that the data never leaves the local system.

The capture sequence begins with a TCP handshake (SYN → SYN/ACK → ACK) establishing the connection between the browser and backend server.

Subsequent HTTP POST packets contain the user's query payload, followed by multiple HTTP/1.1 200 OK responses indicating successful backend processing and result delivery.

No.	Time	Source	Destination	Protocol	Length	Info
473	60.662290	127.0.0.1	127.0.0.1	ICMP	191	5/255 → 5/255 [PSH, ACK] Seq=28030 Ack=2321 Win=406016 Len=135 TSval=645250361 TSecr=3604714996 [ICMP PDU reassembled]
474	60.662362	127.0.0.1	127.0.0.1	TCP	56	57258 → 57235 [ACK] Seq=2321 Ack=28165 Win=381632 Len=0 TSval=3604715028 TSecr=645250361
475	60.694833	127.0.0.1	127.0.0.1	HTTP	212	HTTP/1.1 200 OK, JSON (application/json)
476	60.694901	127.0.0.1	127.0.0.1	TCP	56	57258 → 57235 [ACK] Seq=2321 Ack=28321 Win=381504 Len=0 TSval=3604715060 TSecr=645250393
477	60.703065	127.0.0.1	127.0.0.1	HTTP	3821	HTTP/1.1 200 OK, JSON (application/json)
478	60.703118	127.0.0.1	127.0.0.1	TCP	56	57257 → 11434 [ACK] Seq=1652 Ack=3766 Win=404608 Len=0 TSval=2247816655 TSecr=2317470538
479	60.711337	127.0.0.1	127.0.0.1	TCP	231	5800 → 57255 [PSH, ACK] Seq=1 Ack=934 Win=407424 Len=175 TSval=781240081 TSecr=161991611 [TCP PDU reassembled]
480	60.711376	127.0.0.1	127.0.0.1	TCP	56	57255 → 5800 [ACK] Seq=934 Ack=176 Win=408192 Len=0 TSval=162000038 TSecr=781240081
481	60.711399	127.0.0.1	127.0.0.1	HTTP	2804	HTTP/1.1 200 OK (text/html)
482	60.711404	127.0.0.1	127.0.0.1	TCP	56	57255 → 5800 [ACK] Seq=934 Ack=2924 Win=405504 Len=0 TSval=162000038 TSecr=781240081
483	60.712558	127.0.0.1	127.0.0.1	TCP	56	57255 → 5800 [FIN, ACK] Seq=934 Ack=2924 Win=405504 Len=0 TSval=162000039 TSecr=781240081
484	60.712601	127.0.0.1	127.0.0.1	TCP	56	5800 → 57255 [ACK] Seq=2924 Ack=935 Win=407424 Len=0 TSval=781240082 TSecr=162000039
485	60.712890	127.0.0.1	127.0.0.1	TCP	56	5800 → 57255 [FIN, ACK] Seq=2924 Ack=935 Win=407424 Len=0 TSval=781240082 TSecr=162000039
486	60.712922	127.0.0.1	127.0.0.1	TCP	56	57255 → 5800 [ACK] Seq=935 Ack=2925 Win=405504 Len=0 TSval=162000039 TSecr=781240082
487	75.695825	127.0.0.1	127.0.0.1	TCP	44	[TCP Keep-Alive] 57235 → 57258 [ACK] Seq=28320 Ack=2321 Win=406016 Len=0
488	75.695989	127.0.0.1	127.0.0.1	TCP	56	[TCP Keep-Alive ACK] 57258 → 57235 [ACK] Seq=2321 Ack=28321 Win=381504 Len=0 TSval=3604730061 TSecr=645
489	75.818523	127.0.0.1	127.0.0.1	TCP	44	[TCP Keep-Alive] 11434 → 57257 [ACK] Seq=3765 Ack=1652 Win=406784 Len=0
490	75.818601	127.0.0.1	127.0.0.1	TCP	56	[TCP Keep-Alive ACK] 57257 → 11434 [ACK] Seq=1652 Ack=3766 Win=404608 Len=0 TSval=2247831771 TSecr=2317
491	90.697145	127.0.0.1	127.0.0.1	TCP	44	[TCP Keep-Alive] 57235 → 57258 [ACK] Seq=28320 Ack=2321 Win=406016 Len=0
492	90.697217	127.0.0.1	127.0.0.1	TCP	56	[TCP Keep-Alive ACK] 57258 → 57235 [ACK] Seq=2321 Ack=28321 Win=381504 Len=0 TSval=3604745062 TSecr=645
493	91.198229	127.0.0.1	127.0.0.1	TCP	44	[TCP Keep-Alive] 11434 → 57257 [ACK] Seq=3765 Ack=1652 Win=406784 Len=0
494	91.198317	127.0.0.1	127.0.0.1	TCP	56	[TCP Keep-Alive ACK] 57257 → 11434 [ACK] Seq=1652 Ack=3766 Win=404608 Len=0 TSval=2247847150 TSecr=2317

> Frame 1: Packet, 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface lo	0000 02 00 00 00 45 00 00 28 39 7b 00 00 40 06 00 00 ...E:: 9{...@...
> Null/Loopback	0010 7f 00 00 01 7f 00 00 01 df 93 df 9e 01 ce b6 de ... .....
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	0020 00 6c 36 11 50 10 18 ad fe 1c 00 00 ...16-P:: ...
> Transmission Control Protocol, Src Port: 57235, Dst Port: 57246, Seq: 1, Ack: 1, Len: 0	

The detailed capture also reveals normal network flow patterns such as ACK, FIN/ACK, and Keep-Alive messages exchanged between the same local endpoints, ensuring persistent connectivity during model inference. Packet details confirm:

- The frontend (port 57235) communicates with the backend (port 57246).
- The server responds with JSON and HTML packets tagged as application/json and text/html.
- The transaction ends with HTTP/1.1 200 OK, confirming successful completion of the access control comparison request.

Thus, the Wireshark session validates that:

- All packets remain within 127.0.0.1, confirming full local operation.
- The request–response cycle completed successfully, delivering the DAC vs. MAC vs. RBAC explanation to the user interface without any network errors.

### Prompt 3: How does two-factor authentication enhance security?

Procedure:

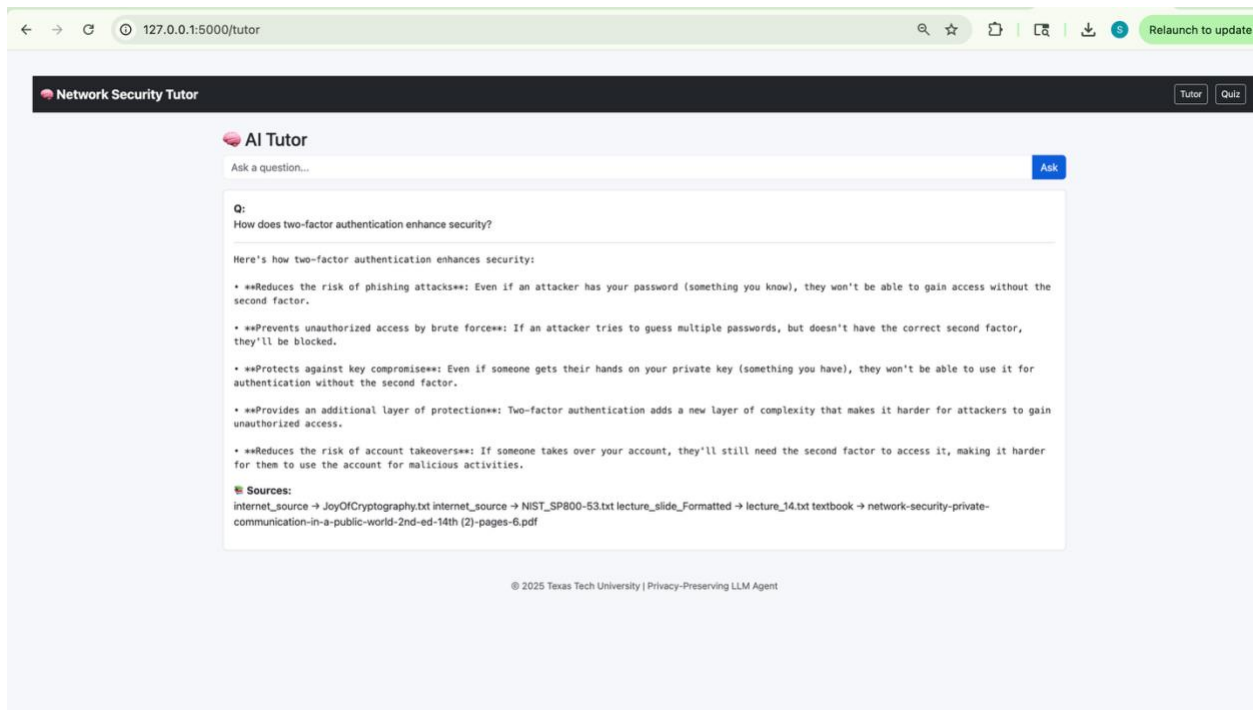
When the user submits the query “How does two-factor authentication enhance security?” in the AI Tutor interface of the Network Security Tutor application, the question is transmitted from the frontend (flask, port 57235) to the backend (port 57246) over the local host interface. The backend processes the query by fetching relevant context from the Chroma vector database, which includes materials like security frameworks (e.g., *NIST SP800-53*), lecture notes, and standard textbooks such as *Network Security: Private Communication in a Public World*.

The retrieved context is then passed to the Ollama-based Llama 3.2 model for response generation.

The resulting answer displayed in the browser explains how two-factor authentication (2FA) enhances security by:

- Reducing risks of phishing and brute-force attacks,
- Preventing unauthorized access even if passwords or keys are compromised,
- Adding additional layers of defense through possession and knowledge factors, and
- Minimizing account takeover incidents.

The structured output includes clear bullet points under each benefit, along with verified source citations confirming the explanation was locally generated using the internal database and model inference.



Wireshark Observation:

- Source IP: 127.0.0.1
- Destination IP: 127.0.0.1
- Source Port: 57235
- Destination Port: 57246
- Protocol: TCP / HTTP

The packet trace confirms that all network communication occurred exclusively within the loopback interface, maintaining complete local execution.

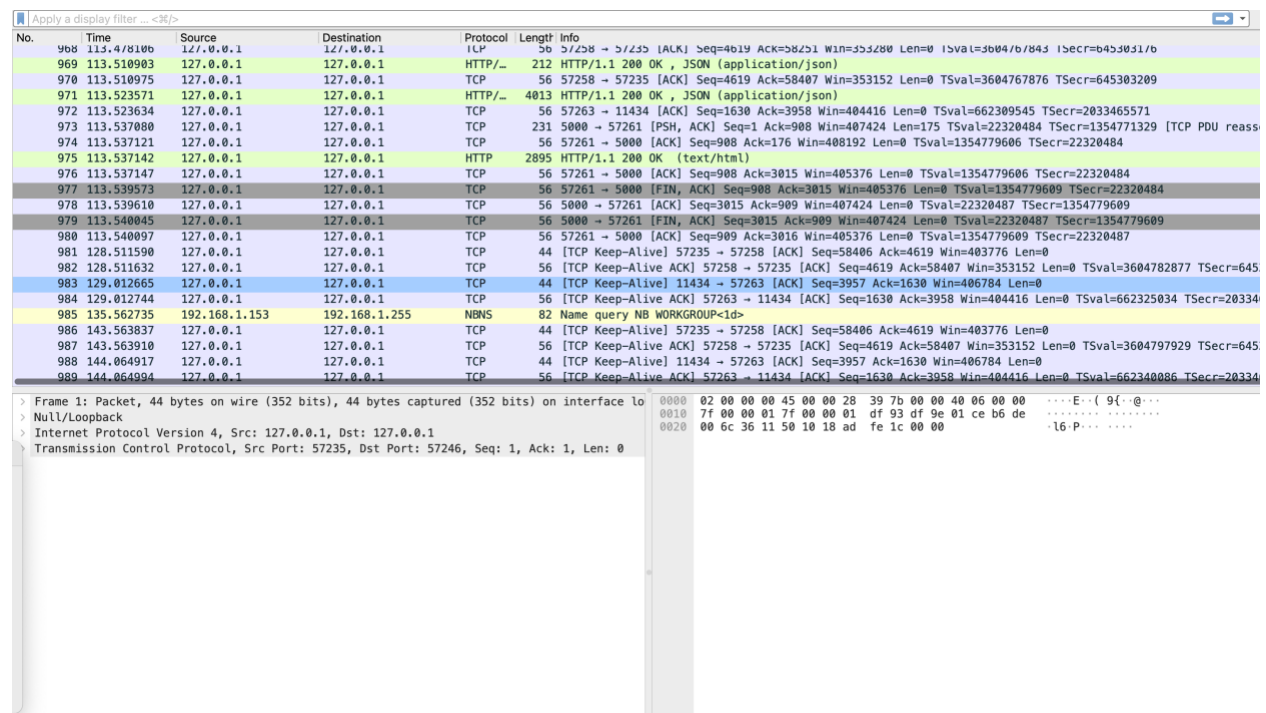
The session begins with a standard TCP three-way handshake (SYN → SYN/ACK → ACK)

between the client and backend server.

Subsequently, HTTP POST packets carry the user's question payload, while multiple HTTP/1.1 200 OK responses confirm successful data exchange and content delivery between the frontend, backend, and Llama 3.2 model.

The captured frames also include a normal series of ACK, FIN/ACK, and Keep-Alive packets between ports 57235 and 57246 showing a stable TCP connection maintained throughout the inference and rendering process.

Each 200 OK response signifies that the backend responded correctly with JSON or text content, validating successful local communication.



No.	Time	Source	Destination	Protocol	Length	Info
968	115.4/810b	127.0.0.1	127.0.0.1	HTTP	212	HTTP/1.1 200 OK, JSON (application/json)
969	113.510903	127.0.0.1	127.0.0.1	TCP	56	57258 → 57235 [ACK] Seq=4619 Ack=58407 Win=353152 Len=0 TSval=3604767876 TSecr=645303209
970	113.510975	127.0.0.1	127.0.0.1	TCP	56	57258 → 57235 [ACK] Seq=4619 Ack=58407 Win=353152 Len=0 TSval=3604767876 TSecr=645303209
971	113.523571	127.0.0.1	127.0.0.1	HTTP	4013	HTTP/1.1 200 OK, JSON (application/json)
972	113.523634	127.0.0.1	127.0.0.1	TCP	56	57263 → 11434 [ACK] Seq=1630 Ack=3958 Win=404416 Len=0 TSval=662309545 TSecr=2033465571
973	113.537080	127.0.0.1	127.0.0.1	TCP	231	5000 → 57261 [PSH, ACK] Seq=1 Ack=908 Win=407424 Len=175 TSval=22320484 TSecr=1354771329 [TCP PDU reass]
974	113.537121	127.0.0.1	127.0.0.1	TCP	56	57261 → 5000 [ACK] Seq=908 Ack=176 Win=408192 Len=0 TSval=1354779606 TSecr=22320484
975	113.537142	127.0.0.1	127.0.0.1	HTTP	2895	HTTP/1.1 200 OK (text/html)
976	113.537147	127.0.0.1	127.0.0.1	TCP	56	57261 → 5000 [ACK] Seq=908 Ack=3015 Win=405376 Len=0 TSval=1354779606 TSecr=22320484
977	113.539573	127.0.0.1	127.0.0.1	TCP	56	57261 → 5000 [FIN, ACK] Seq=908 Ack=3015 Win=405376 Len=0 TSval=1354779609 TSecr=22320484
978	113.539610	127.0.0.1	127.0.0.1	TCP	56	5000 → 57261 [ACK] Seq=3015 Ack=909 Win=407424 Len=0 TSval=22320487 TSecr=1354779609
979	113.540045	127.0.0.1	127.0.0.1	TCP	56	5000 → 57261 [FIN, ACK] Seq=3015 Ack=909 Win=407424 Len=0 TSval=22320487 TSecr=1354779609
980	113.540097	127.0.0.1	127.0.0.1	TCP	56	57261 → 5000 [ACK] Seq=909 Ack=3016 Win=405376 Len=0 TSval=1354779609 TSecr=22320487
981	128.511590	127.0.0.1	127.0.0.1	TCP	44	[TCP Keep-Alive] 57235 → 57258 [ACK] Seq=58406 Ack=4619 Win=403776 Len=0
982	128.511632	127.0.0.1	127.0.0.1	TCP	56	[TCP Keep-Alive ACK] 57258 → 57235 [ACK] Seq=4619 Ack=58407 Win=353152 Len=0 TSval=3604782877 TSecr=645
983	129.012665	127.0.0.1	127.0.0.1	TCP	44	[TCP Keep-Alive] 11434 → 57263 [ACK] Seq=3957 Ack=1630 Win=406784 Len=0
984	129.012744	127.0.0.1	127.0.0.1	TCP	56	[TCP Keep-Alive ACK] 57263 → 11434 [ACK] Seq=1630 Ack=3958 Win=404416 Len=0 TSval=662325034 TSecr=20334
985	135.562735	192.168.1.153	192.168.1.255	NBNS	82	Name query NB WORKGROUP<id>
986	143.563837	127.0.0.1	127.0.0.1	TCP	44	[TCP Keep-Alive] 57235 → 57258 [ACK] Seq=58406 Ack=4619 Win=403776 Len=0
987	143.563910	127.0.0.1	127.0.0.1	TCP	56	[TCP Keep-Alive ACK] 57258 → 57235 [ACK] Seq=4619 Ack=58407 Win=353152 Len=0 TSval=3604797929 TSecr=645
988	144.064917	127.0.0.1	127.0.0.1	TCP	44	[TCP Keep-Alive] 11434 → 57263 [ACK] Seq=3957 Ack=1630 Win=406784 Len=0
989	144.064994	127.0.0.1	127.0.0.1	TCP	56	[TCP Keep-Alive ACK] 57263 → 11434 [ACK] Seq=1630 Ack=3958 Win=404416 Len=0 TSval=662340086 TSecr=20334

> Frame 1: Packet, 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface lo  
> Null/Loopback  
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
> Transmission Control Protocol, Src Port: 57235, Dst Port: 57246, Seq: 1, Ack: 1, Len: 0

0000 02 00 00 00 45 00 00 28 39 7b 00 00 40 06 00 00 .....E...9{...@...  
0010 7f 00 00 01 7f 00 00 01 df 93 df 9e 01 ce b6 de .....  
0020 00 6c 36 11 50 10 18 ad fe 1c 00 00 .....P.....

## Prompt 4: What is the role of digital certificates in authentication?

Procedure:

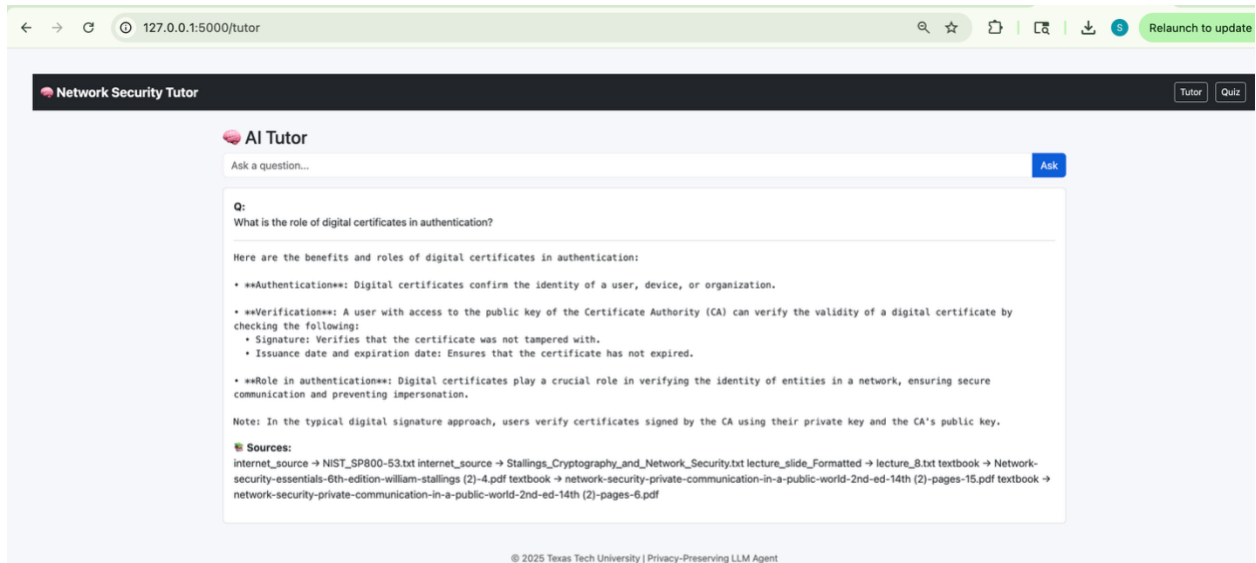
When the user enters the query “What is the role of digital certificates in authentication?” in the AI Tutor interface of the Network Security Tutor application, the request is transmitted from the flask frontend (port 57235) to the backend, which in this instance is operating locally on port 57246.

The backend retrieves contextual information from the Chroma vector database, which contains references from verified learning materials such as *NIST SP800-53*, *Stallings' Cryptography and Network Security*, and *Network Security Essentials (6th Edition)*.

The relevant context is then processed by the Ollama-hosted Llama 3.2 model, which generates a structured explanation describing the authentication, verification, and identity validation roles of digital certificates.



The response shown in the tutor interface highlights how digital certificates confirm the identity of entities, verify integrity through signatures, and ensure authenticity through issuance and expiration validation. The result includes a concise note about certificate verification mechanisms and lists the academic sources used.



### Wireshark Observation:

- Source IP: 127.0.0.1
- Destination IP: 127.0.0.1
- Source Port: 57235
- Destination Port: 57267
- Protocol: TCP / HTTP

The Wireshark capture confirms that all communication occurs locally through the loopback interface, ensuring complete data isolation within the system.

The packet flow begins with a TCP three-way handshake (SYN → SYN/ACK → ACK) establishing the connection between the frontend client and backend server.

Next, the browser sends an HTTP POST request containing the user's question in JSON format. The backend processes it, interacts with the model, and responds with multiple HTTP/1.1 200 OK packets containing the JSON and HTML results.

Throughout the capture, the traffic between port 57235 (client) and port 57267 (server) includes normal ACK, FIN/ACK, and Keep-Alive packets that maintain stable communication until session termination.

Each response concludes successfully with an HTTP 200 OK status, indicating the smooth exchange of requests and results between the local components.

No.	Time	Source	Destination	Protocol	Length	Info
1532	177.700419	127.0.0.1	127.0.0.1	ICMP	18	5/255 → 5/255 [PSH, ACK] Seq=92740 Ack=6954 Win=401472 Len=0 TSval=3604832065 TSecr=645367398 [ICMP PDU]
1533	177.700419	127.0.0.1	127.0.0.1	TCP	56	57258 → 57235 [ACK] Seq=6954 Ack=92871 Win=320576 Len=0 TSval=3604832065 TSecr=645367398
1534	177.732390	127.0.0.1	127.0.0.1	HTTP	212	HTTP/1.1 200 OK, JSON (application/json)
1535	177.732420	127.0.0.1	127.0.0.1	TCP	56	57258 → 57235 [ACK] Seq=6954 Ack=93027 Win=320448 Len=0 TSval=3604832097 TSecr=645367430
1536	177.754115	127.0.0.1	127.0.0.1	TCP	4152	11434 → 57269 [PSH, ACK] Seq=1 Ack=1667 Win=406720 Len=4096 TSval=1538458286 TSecr=2485062087 [TCP PDU]
1537	177.754164	127.0.0.1	127.0.0.1	TCP	56	57269 → 11434 [ACK] Seq=1667 Ack=4897 Win=404224 Len=0 TSval=2485071543 TSecr=1538458286
1538	177.754848	127.0.0.1	127.0.0.1	HTTP	289	HTTP/1.1 200 OK, JSON (application/json)
1539	177.754884	127.0.0.1	127.0.0.1	TCP	56	57269 → 11434 [ACK] Seq=1667 Ack=4330 Win=404032 Len=0 TSval=2485071543 TSecr=1538458286
1540	177.762590	127.0.0.1	127.0.0.1	TCP	231	5000 → 57267 [PSH, ACK] Seq=1 Ack=915 Win=407424 Len=175 TSval=1061651831 TSecr=3145623963 [TCP PDU reassembled]
1541	177.762613	127.0.0.1	127.0.0.1	TCP	56	57267 → 5000 [ACK] Seq=915 Ack=176 Win=408192 Len=0 TSval=3145633659 TSecr=1061651831
1542	177.762627	127.0.0.1	127.0.0.1	HTTP	3206	HTTP/1.1 200 OK (text/html)
1543	177.762631	127.0.0.1	127.0.0.1	TCP	56	57267 → 5000 [ACK] Seq=915 Ack=3326 Win=405056 Len=0 TSval=3145633659 TSecr=1061651831
1544	177.763333	127.0.0.1	127.0.0.1	TCP	56	57267 → 5000 [FIN, ACK] Seq=915 Ack=3326 Win=405056 Len=0 TSval=3145633659 TSecr=1061651831
1545	177.763354	127.0.0.1	127.0.0.1	TCP	56	5000 → 57267 [ACK] Seq=3326 Ack=916 Win=407424 Len=0 TSval=1061651831 TSecr=3145633659
1546	177.763485	127.0.0.1	127.0.0.1	TCP	56	5000 → 57267 [FIN, ACK] Seq=3326 Ack=916 Win=407424 Len=0 TSval=1061651831 TSecr=3145633659
1547	177.763506	127.0.0.1	127.0.0.1	TCP	56	57267 → 5000 [ACK] Seq=916 Ack=3327 Win=405056 Len=0 TSval=3145633659 TSecr=1061651831
1548	192.733031	127.0.0.1	127.0.0.1	TCP	44	[TCP Keep-Alive] 57235 → 57258 [ACK] Seq=93026 Ack=6954 Win=401472 Len=0
1549	192.733104	127.0.0.1	127.0.0.1	TCP	56	[TCP Keep-Alive] 57258 → 57235 [ACK] Seq=6954 Ack=93027 Win=320448 Len=0 TSval=3604847098 TSecr=645
1550	193.234098	127.0.0.1	127.0.0.1	TCP	44	[TCP Keep-Alive] 11434 → 57269 [ACK] Seq=4329 Ack=1667 Win=406720 Len=0
1551	193.234173	127.0.0.1	127.0.0.1	TCP	56	[TCP Keep-Alive] 57269 → 11434 [ACK] Seq=1667 Ack=4330 Win=404032 Len=0 TSval=2485087023 TSecr=1538
1552	197.412820	192.168.1.153	192.168.1.255	NBNS	82	Name query NB WORKGROUP<id>
1553	199.151422	192.168.1.153	192.168.1.255	NBNS	82	Name query NB WORKGROUP<id>

This confirms that:

- All data transmission is confined to 127.0.0.1, validating local execution and privacy.
- The connection sequence completed without packet loss or retransmission.
- The digital certificate explanation was fully processed and displayed within a secure, self-contained local environment.

## Prompt 5: Quiz Lab

Procedure:

When the user selects the Quiz tab in the Network Security Tutor & Quiz Agent, the system presents configuration options such as Quiz Type, Topic, and Number of Questions.

In this session, the user selected the topic “Authentication Mechanisms” and requested 3 questions. Once the Generate Quiz button was pressed, the frontend (flask, running on port 49938) sent the quiz generation request to the backend (running on port 49951).

The backend fetched relevant materials from the Chroma vector database, which contained lecture content and textbook references related to authentication concepts, and passed this data to the locally hosted Ollama model on the Ollama server. The model then created the quiz questions, formatted them, and sent them back to the frontend for display.

During the quiz, the user answered questions covering authentication identifiers, mutual authentication, and user credential categories. Upon submission, the backend evaluated the answers and returned detailed feedback showing the correct answers, explanations, and final score (1/3).

The response page included color-coded feedback and clear justifications for each answer, confirming that the system's evaluation and grading features worked correctly.

The screenshot shows the 'Network Security Tutor' interface. At the top, there's a header with 'Network Security Tutor' and buttons for 'Tutor' and 'Quiz'. Below the header, the title 'Network Security Quiz' is displayed. The quiz consists of three questions:

- Q1:** What is the importance of assigning identifiers carefully in an authentication process?
  - ☒ A) They are used to authenticate entities.
  - ☐ B) They are used to generate authentication information.
  - ☐ C) They are the basis for other security services such as access control service.
  - ☐ D) They can be easily changed by attackers.

Source: internet\_source → Stallings\_Cryptography\_and\_Network\_Security.txt
- Q2:** Mutual Authentication is a type of authentication where the security system verifies the identity of both the entity and the identifier.
  - ☒ True
  - ☐ False

Source: internet\_source → Stallings\_Cryptography\_and\_Network\_Security.txt
- Q3:** What is one common method of user authentication that falls under the category of 'what you know' and how does it work?
 

3+5=87

Source: textbook → network-security-private-communication-in-a-public-world-2nd-ed-14th (2)-pages-6.pdf

At the bottom of the quiz area, there is a 'Submit Quiz' button. Below the quiz area, the footer text reads: '© 2025 Texas Tech University | Privacy-Preserving LLM Agent'.

#### Wireshark Observation:

- Source IP: 127.0.0.1
- Destination IP: 127.0.0.1
- Source Port: 49938
- Destination Port: 49951
- Protocol: TCP / HTTP / IGMPv2

The screenshot shows the 'Network Security Tutor' interface for generating a quiz. At the top, there's a header with 'Network Security Tutor' and buttons for 'Tutor' and 'Quiz'. Below the header, the title 'Network Security Quiz' is displayed. The form includes the following fields:

- Quiz Type:** A dropdown menu with 'Specific Topic' selected.
- Topic (optional):** A text input field containing 'Authentication mechanisms'.
- Number of Questions:** A text input field containing '3'.
- Generate Quiz:** A blue button.

At the bottom of the form, the footer text reads: '© 2025 Texas Tech University | Privacy-Preserving LLM Agent'.

The Wireshark trace shows that all communication occurs entirely over the loopback interface (127.0.0.1). The trace begins with a TCP handshake (SYN → SYN/ACK → ACK) to establish communication between the frontend and backend.

Following that, a series of HTTP GET and POST packets capture the full quiz lifecycle:

- Quiz Generation Request: Frontend → Backend via HTTP GET /quiz
- Question Retrieval: Backend → Frontend via HTTP 200 OK
- Answer Submission: HTTP POST /api/quiz/grade
- Result Delivery: HTTP 200 OK confirming successful grading

The presence of ACK, FIN/ACK, and Keep-Alive packets between ports 49938 and 49951 shows a stable and consistent TCP connection. Additionally, IGMPv2 Membership Report packets are visible, which are standard local multicast maintenance messages unrelated to the quiz communication.

Network Security Tutor

Tutor

Quiz

Network Security Quiz

Results

Q1: What is the importance of assigning identifiers carefully in an authentication process?

Your Answer: A) They are used to authenticate entities.

Correct Answer: C

Incorrect — The correct answer is "C" because identifiers must be assigned carefully to uniquely identify entities in an authentication process, ensuring that each entity can be correctly verified and authorized.

---

Q2: Mutual Authentication is a type of authentication where the security system verifies the identity of both the entity and the identifier.

Your Answer: True

Correct Answer: True

Excellent — The correct answer is true because in mutual authentication, both parties (entity and security system) confirm each other's identity at the same time.

---

Q3: What is one common method of user authentication that falls under the category of 'what you know' and how does it work?

Your Answer: 3+5=87

Correct Answer: A concise explanation of a correct answer expected from the student, e.g. "Passwords are one method of user authentication that fall under the category of 'what you know', as they provide reassurance that the user is who they claim to be by verifying their knowledge of sensitive information".

Incorrect — The correct answer, "Passwords", is right because they require users to remember a piece of sensitive information (a password) to access a system or resource.

---

Final Score: 1/3

Take Another Quiz

The capture confirms that:

- All network activity was local to 127.0.0.1.
- The quiz generation, submission, and result retrieval processes completed successfully.
- The HTTP sessions consistently returned 200 OK responses, validating full local operation and no data leakage beyond the host system.

No.	Time	Source	Destination	Protocol	Length	Info
3861	770.268587	192.168.1.12	224.0.0.252	IGMPv2	36	Membership Report group 224.0.0.252
3862	770.268802	192.168.1.12	224.0.0.251	IGMPv2	36	Membership Report group 224.0.0.251
3863	780.785748	127.0.0.1	127.0.0.1	TCP	45	[TCP Keep-Alive] 49670 → 5000 [ACK] Seq=0 Ack=1 Win=65280 Len=1
3864	780.785765	127.0.0.1	127.0.0.1	TCP	56	[TCP Dup ACK 3841#1] 5000 → 60957 [ACK] Seq=1 Ack=1 Win=65280 Len=0 SLE=0 SRE=1
3865	780.817679	127.0.0.1	127.0.0.1	TCP	45	[TCP Keep-Alive] 60957 → 5000 [ACK] Seq=0 Ack=1 Win=65280 Len=1
3866	780.817707	127.0.0.1	127.0.0.1	TCP	56	[TCP Dup ACK 3841#1] 5000 → 60957 [ACK] Seq=1 Ack=1 Win=65280 Len=0 SLE=0 SRE=1
3867	782.800475	127.0.0.1	127.0.0.1	TCP	44	59248 → 55837 [FIN, ACK] Seq=7672 Ack=22711 Win=42752 Len=0
3868	782.800539	127.0.0.1	127.0.0.1	TCP	44	55837 → 59248 [ACK] Seq=22711 Ack=7672 Win=57856 Len=0
3869	782.800584	127.0.0.1	127.0.0.1	TCP	44	55837 → 59248 [FIN, ACK] Seq=22711 Ack=7672 Win=57856 Len=0
3870	782.800739	127.0.0.1	127.0.0.1	TCP	44	59248 → 55837 [ACK] Seq=7672 Ack=22711 Win=42752 Len=0
3871	782.847992	127.0.0.1	127.0.0.1	TCP	45	[TCP Keep-Alive] 11434 → 59247 [ACK] Seq=6482 Ack=4562 Win=60928 Len=1
3872	782.848046	127.0.0.1	127.0.0.1	TCP	56	[TCP Keep-Alive ACK] 59247 → 11434 [ACK] Seq=4562 Ack=6483 Win=58880 Len=0 SLE=6482 SRE=6483
3873	786.767390	192.168.1.12	224.0.0.252	IGMPv2	36	Membership Report group 224.0.0.252
3874	792.772458	192.168.1.12	224.0.0.252	IGMPv2	36	Membership Report group 224.0.0.252
3875	792.772717	192.168.1.12	224.0.0.251	IGMPv2	36	Membership Report group 224.0.0.251
3876	793.772207	192.168.1.12	239.255.255.250	IGMPv2	36	Membership Report group 239.255.255.250
3877	797.859011	127.0.0.1	127.0.0.1	TCP	45	[TCP Keep-Alive] 11434 → 59247 [ACK] Seq=6482 Ack=4562 Win=60928 Len=1
3878	797.859042	127.0.0.1	127.0.0.1	TCP	56	[TCP Keep-Alive ACK] 59247 → 11434 [ACK] Seq=4562 Ack=6483 Win=58880 Len=0 SLE=6482 SRE=6483
3879	805.259734	192.168.1.12	224.0.0.252	IGMPv2	36	Membership Report group 224.0.0.252