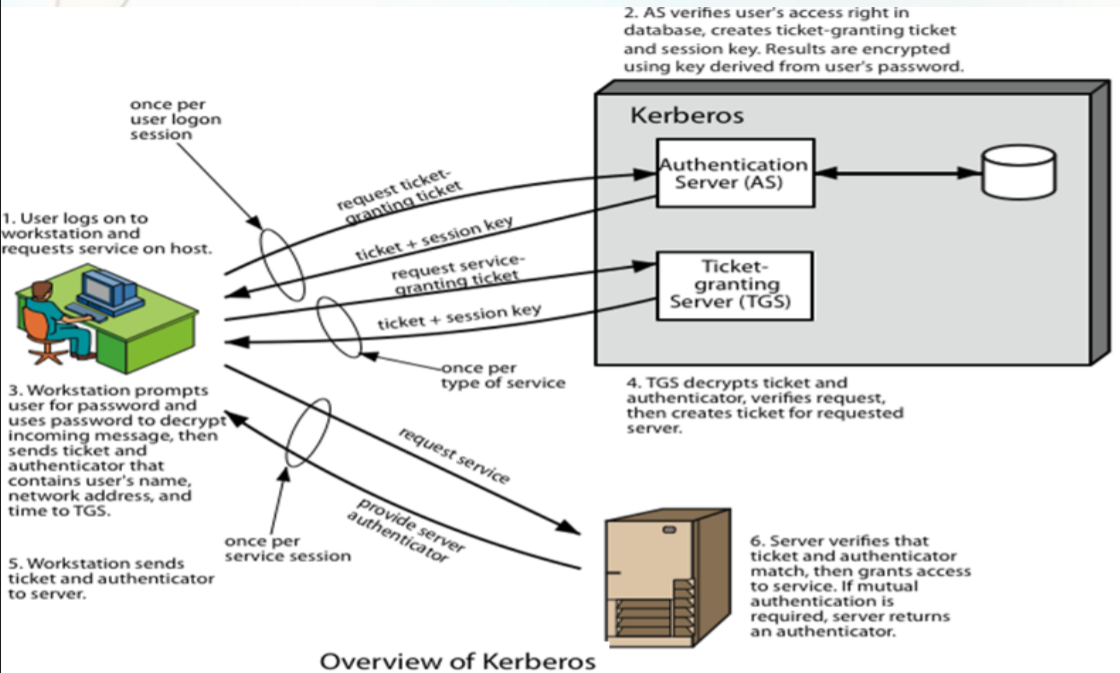
**KERBEROS**

• Kerberos is a computer network authentication protocol that works based on 'tickets' to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner.

• The protocol was named after the character Kerberos (or Cerberus) from Greek mythology, the ferocious three-headed guard dog of Hades (hellhound).

• Its designers aimed it primarily at a client–server model and it provides mutual authentication—both the user and the server verify each other's identity.

• Kerberos protocol messages are protected against eavesdropping and replay attacks.



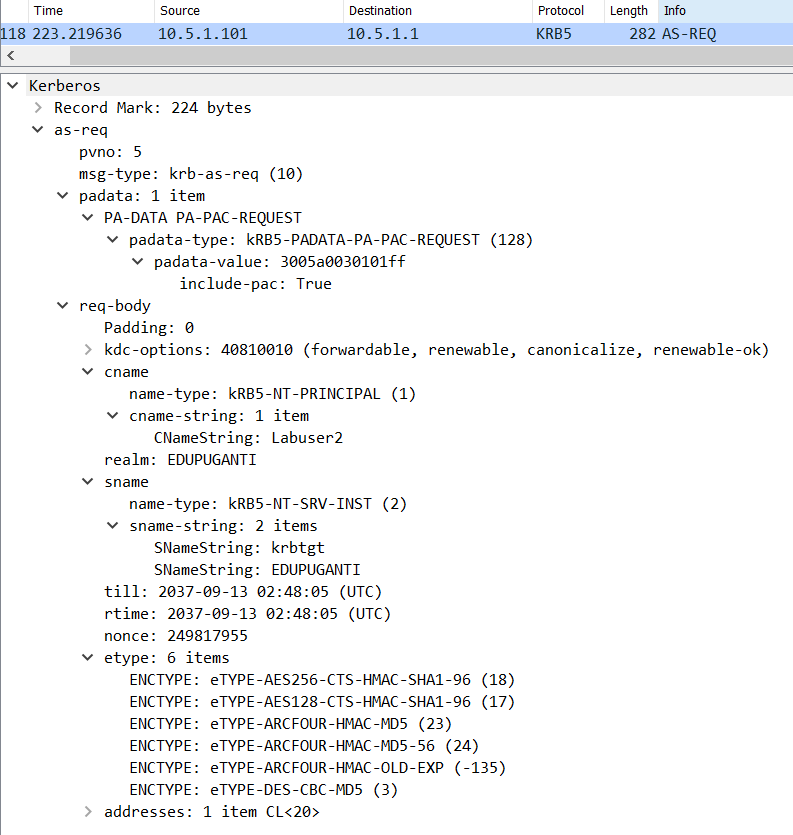
**Need for Kerberos:**

In an open distributed environment, users at workstations wish to access services on servers distributed throughout the network. In such environments, servers should be able to restrict access to unauthorized users and be able to authenticate requests for service. In this environment, a workstation cannot be trusted to identify its users correctly to network services. Three kinds of threats exist: User pretends to be another user, User alters the network address of a workstation and User eavesdrop on exchanges and uses a replay attack.

**Client log on working:**

1. Firstly, the client logs in to the system using his long-term symmetric key, then client sends a AS\_REQ to Domain controller which consists of Privilege Account Certificate (PAC) which contains the pre-authentication information such as security identifiers, password privileges and group membership.
2. The Domain controller then receives it and sends an KRB\_PRE\_AUTH\_ERROR to the client requesting for an encrypted timestamp. This is the default behavior of the Kerberos, client sends a request for Ticket Granting Ticket (TGT) and then the domain controller sends the pre-authentication challenge which is DC asks the client to send the current time encrypted with the client key. The main aim of this step is to prevent the man in the middle attack, where attacker steals the Authentication Request which is successful and try to reuse it. By providing the time stamp the Authentication Server can Authenticate the correct client.
3. After that the client again sends the AS\_REQ to DC requesting for TGT with an encrypted timestamp in it.
4. AS receives the AS\_REQ sent by the client and it will decrypt the encrypted timestamp with the client long term symmetric key (AS will have the copy of all the client’s and server’s long terms symmetric key which are in the same domain) and checks the current timestamp and if both are same or nearly same the AS will send an AS\_REP message which consists of TGT (TGT contains the session key, user id of client, timestamp and expiry time of the ticket) which is encrypted by the TGS symmetric key and session key encrypted by the client’s symmetric key.
5. Client the receives the TGT but cannot read it as it encrypted with the TGS symmetric key client uses TGT for establishing session with the TGS. Client stores the TGT in its cache memory. TGT provides a way for communicating the Session key from AS to TGS. So, the client now decrypts the session key with his symmetric key.
6. The Client now sends a TGS\_REQ to TGS for requesting a ticket for specific application server or file server. The TGS\_REQ consists of the TGT (which is acquired in previous step) and request containing user id, timestamp and File server request encrypted with the Session key.
7. TGS will receive this request and decrypts the TGT with its symmetric key and extracts the session key from it. Using that session key, it decrypts the request and compares the user id and timestamp. If both user id and timestamp match it will send a TGS\_REP message. The TGS\_REP message contains a ticket which contains session key (between client and app server requested) user id of the client, time stamp and expiry of the ticket encrypted with the symmetric key of the Application server and other package which contains the session key and requested Application server name and address encrypted with TGS session key.
8. Client receives this message and decrypts the package containing the Client-Application Server session key and extracts address from it and tries to connect with it.
9. The Client then sends an AP\_REQ message which contains the ticket received from the TGS and another package containing the user id of client and time stamp encrypted using the Client-Application server session key.
10. The Application receives the message and decrypts the ticket and extracts the session key from it. The Server then uses the session key to decrypt the other package and compares the user id and time stamp to ensure that the request came from the correct user. If both matches the Server sends a package with its name and timestamp encrypted with session key in AP\_REP message.
11. The Client receives the message and decrypts it using session key and checks the server name and validates the timestamp.
12. The Client stores the TGT and Server ticket in its cache memory so that they can be used for future communication.
13. **Authorization Request:**

This is the first Authorization server request sent by the client (10.5.1.101) to Domain controller (10.5.1.1) which contains Authorization Server. The contents of this request are as follows

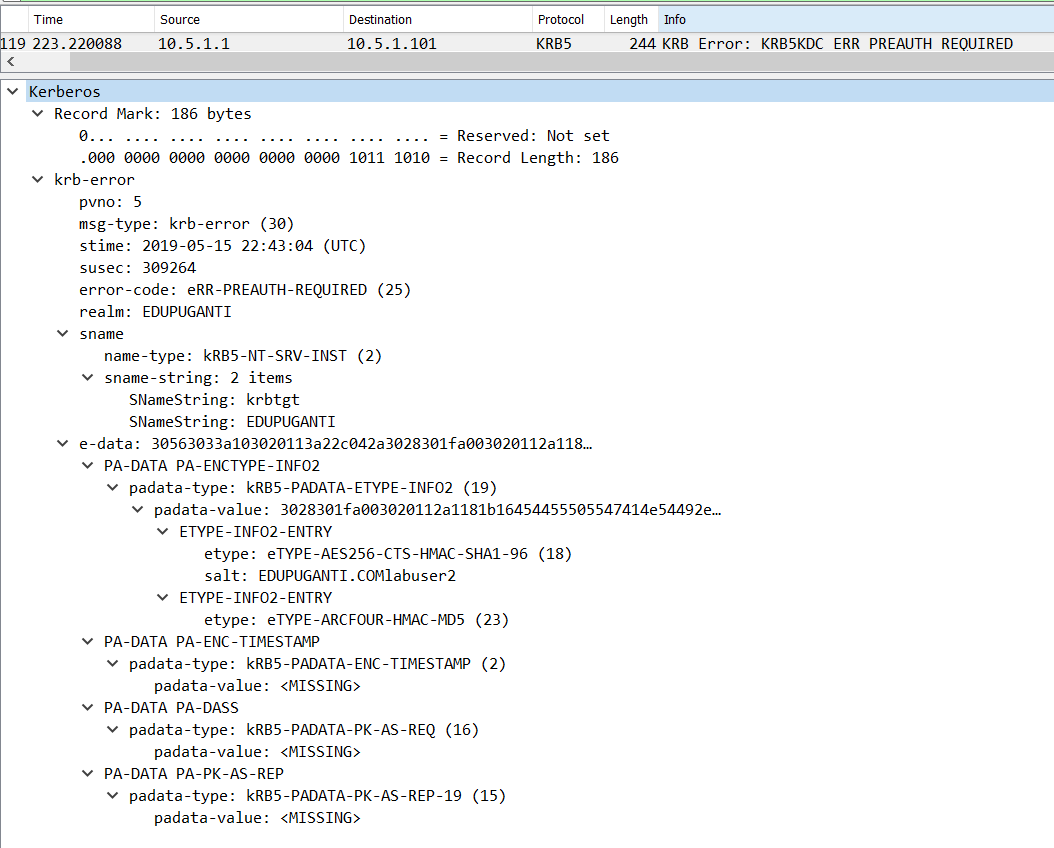


The body of the AS-req consists of

1. Pvno indicates the Kerberos version protocol number that is 5 and msg-type indicates the message type which in this case is Kerberos Authentication Server request.
2. The Padata consists of the pre-authentication data such as security identifiers, group membership and password credentials that is sent in form of Privilege Account Certificate (PAC) request.
3. CNameString indicates the username of the client and realm indicates the Domain name. We can see that the username and domain name are sent to DC in clear text.
4. The Sname indicates the Service name that is being requested, which in this case is the Kerberos TGT (Ticket Granting Ticket), it is a ticket issued by the Authentication Server (AS) to client if the authentication is successful. The client uses that ticket to communicate with the Ticket Granting Server (TGS).
5. Till is the request expiry time and rtime is the expiry time for the renewed ticket if generated.
6. The message Nonce is the unique randomly create string of numbers which is used to prevent replay attack.
7. The etype is the encryption types provided by the system. For example, ENCTYPE: eTYPE-AES256-CTS-HMAC-SHA1-96 (18) is the highest encryption type used for Active Directory domain controllers. The first part AES256-CTS indicates that it uses Advanced Encryption Standard with 256 bit key in cipher text stealing mode. The second part HMAC-SHA1-96 indicates that it is using High-bashed Message Authentication Code using SHA1 hash truncated to 96 bits.

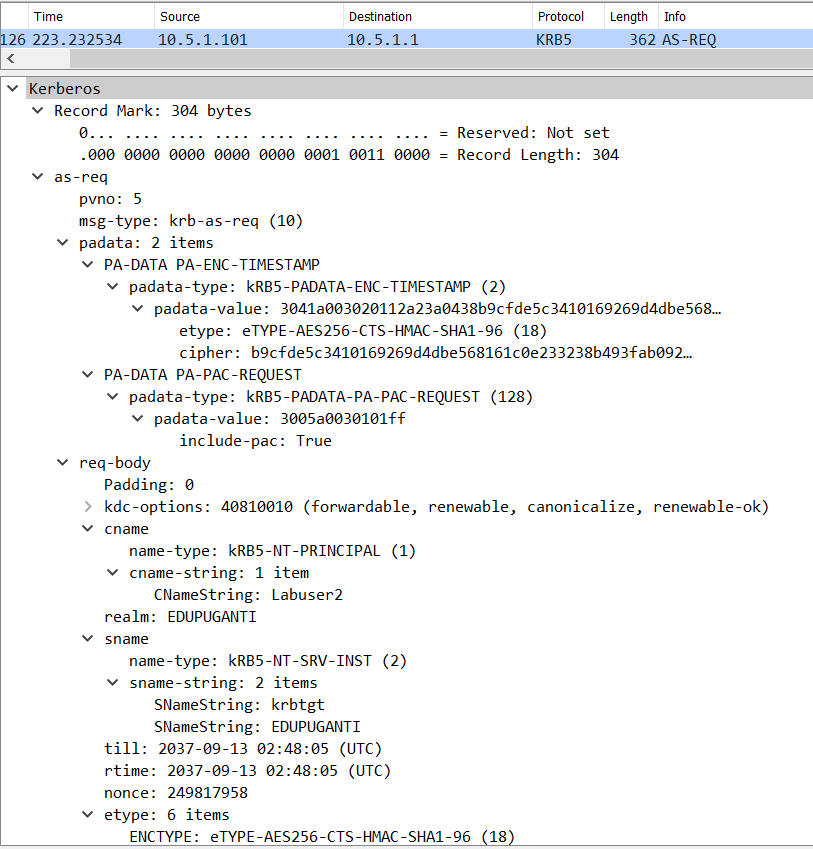
**2. The Kerberos error message:**

This is the default behavior of the Kerberos



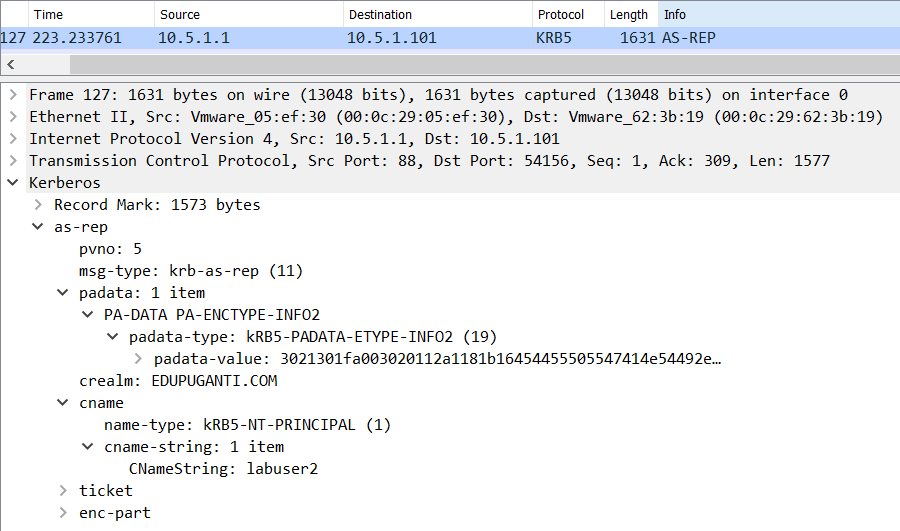
1. In this case the Message type is Kerberos error and the error code is pre auth required. This is the step where DC gives pre-authentication challenge to client.
2. E-data is the additional data about the error for use by the application to help it to recover.

**3**. **AS\_REQ message after challenge:**

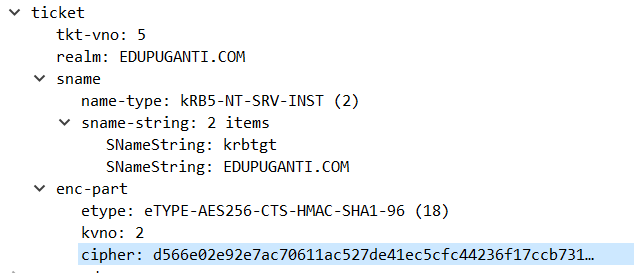


1. This is AS\_REQ message from client to Domain Controller. This message is same as the first message except there is an encrypted timestamp.
2. The timestamp is encrypted with the long-term symmetric key.

4. **AS\_REP Message:**

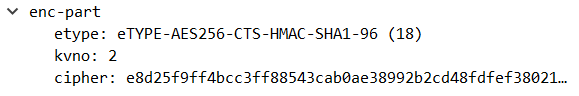


1. The AS\_REP is the reply message to the request message sent by the client to Domain Controller. The contents of the AS\_REP body are
2. PVNO is the Kerberos protocol version number and Message type is Kerberos Authentication Server Reply message.
3. The PA data consists of information to client such as which salt is used for the session to decrypt the message from AS request.
4. The Domain Name (CRealm) and User name (CNameString) are unencrypted.
5. The AS reply message contains two packages:
6. **Ticket:**



1. Tkt-vno – ticket format version number which in this case is 5.
2. Realm – The realm this ticket issued for, which in this case is EDUPUGANTI.COM
3. Sname indicates the name of the service ticket was issued for which is krbtgt in this case and the domain name.
4. Enc-part: Ticket Granting Ticket encrypted using the TGS secret key.

**2.ENC\_PART:**



1. This consists of the Client-TGS Session key encrypted using the Users Secret key.

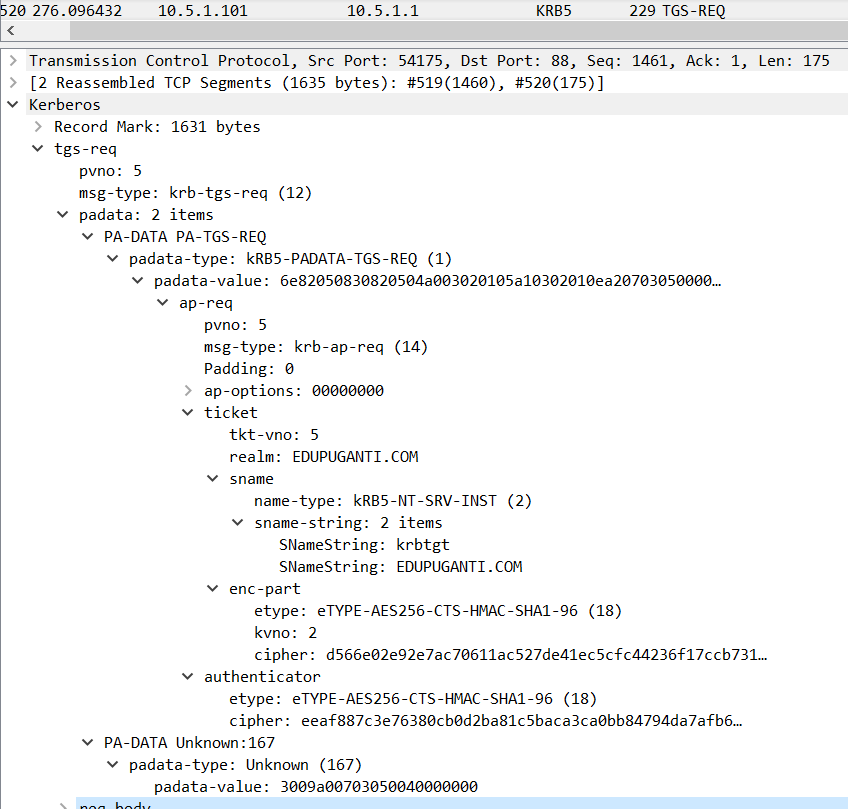
6. The client now contains session key (decrypted from ENC\_PART using users secret key). In TGT there will be the same session key, but client cannot decrypt that. In next step client uses TGT to create a session.

1. **TGS\_REQ:**

Next step, the client establishes the session and sends TGS\_REQ message.

The request consists of two parts:

1.TGS\_REQ: In this part the client uses the TGT to request file server’s access.

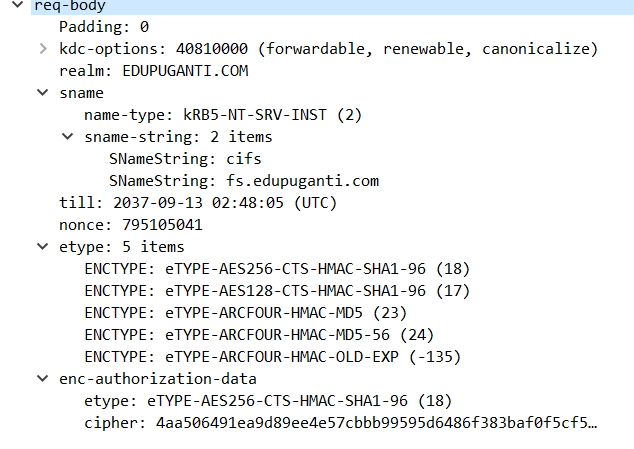


The pa data consists of application server request using the TGT received in previous step.

1.The contents of tickets are same as above message we can see that in encrypted part in ticket the cipher is same as the cipher in the ticket received in previous step (AS\_REP message ticket).

2.The Client also sends an authenticator which contains timestamp encrypted with the Client-TGS session key.

**2.TGS\_REQ\_BODY:**



1. Kdc-options describes the type of service request.

2. Realm indicates the domain for which ticket is issued for.

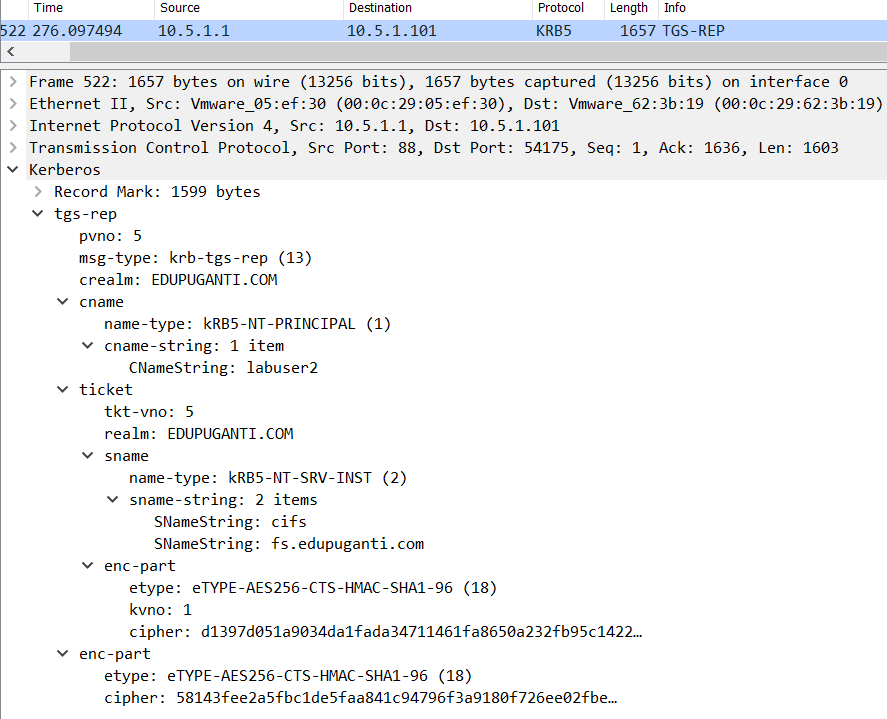
3. sname is the service name for which ticket is requested for, which in this case is File server we can see from above picture sname is fs.edupuganti.com.

4. Till is the expiry time for this request. Nonce is for this request.

5. The etype is the desired encryption type for this request.

1. **TGS\_REP:**

The TGS reply is sent from the Domain controller to Client machine with ticket and Client-FS session key.



The contents of TGS reply are same as above. Only message type is changed to TGS\_REP.

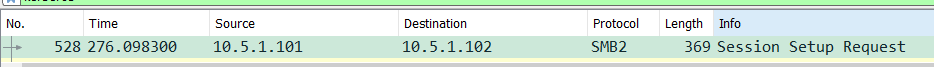
The reply message contains ticket which is encrypted using the File Server Secret key which cannot be decrypted by the client but still client stores the ticket in its cache memory. The contents of the ticket are:

1. Ticket version Number is 5.
2. Realm is the domain name.
3. Sname is the service name for which ticket is issued for.
4. Enc\_part is the encrypted part of the ticket with Servers secret key.

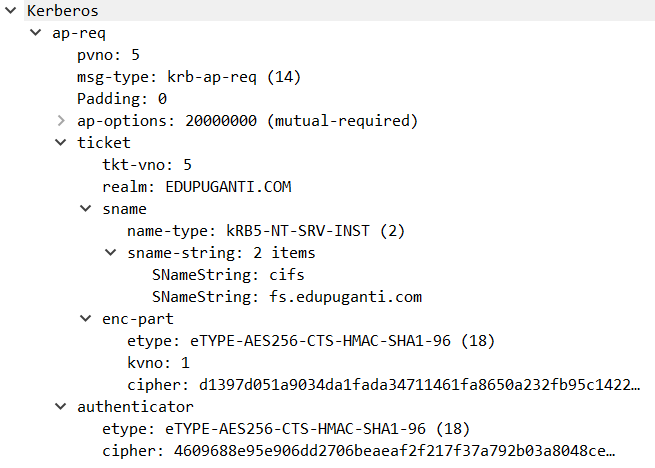
The enc-part contains the client-server session key encrypted using the client-TGS session key. In next step the client uses this session key to establish session with the File server.

7. **AP\_REQ:**

The client now uses the client-server session key to establish a session between client and File server. The client now requests for application’s service by providing the ticket stored in its cache memory and authenticator encrypted using the client-server session key.



The client sends the AP\_REQ message to the file server (10.5.1.102). The AP\_REQ message contains Client to File Server ticket (which is received from Domain Controller) and Authenticator which is timestamp encrypted with the Client-File Server Session key.

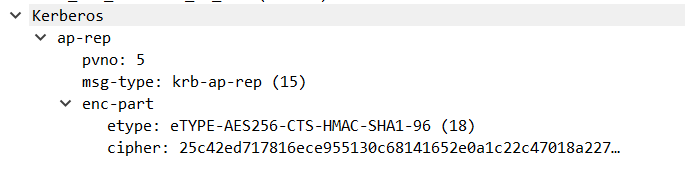


The enc-part of the ticket contains cipher which is same which is received in precious step.

1. **AP\_REP:**

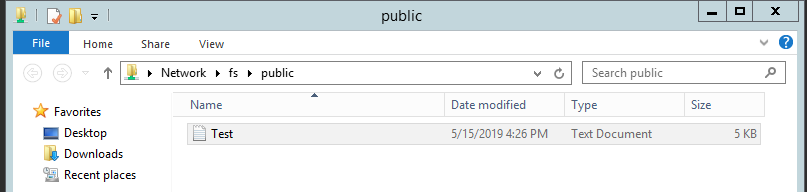
The File server receives the packets and decrypts the ticket and extracts the session key, using that session key it decrypts the authenticator and checks the time stamp. If timestamp is same or nearly same, the File server then sends the time stamp which is encrypted using client-server session key.

****



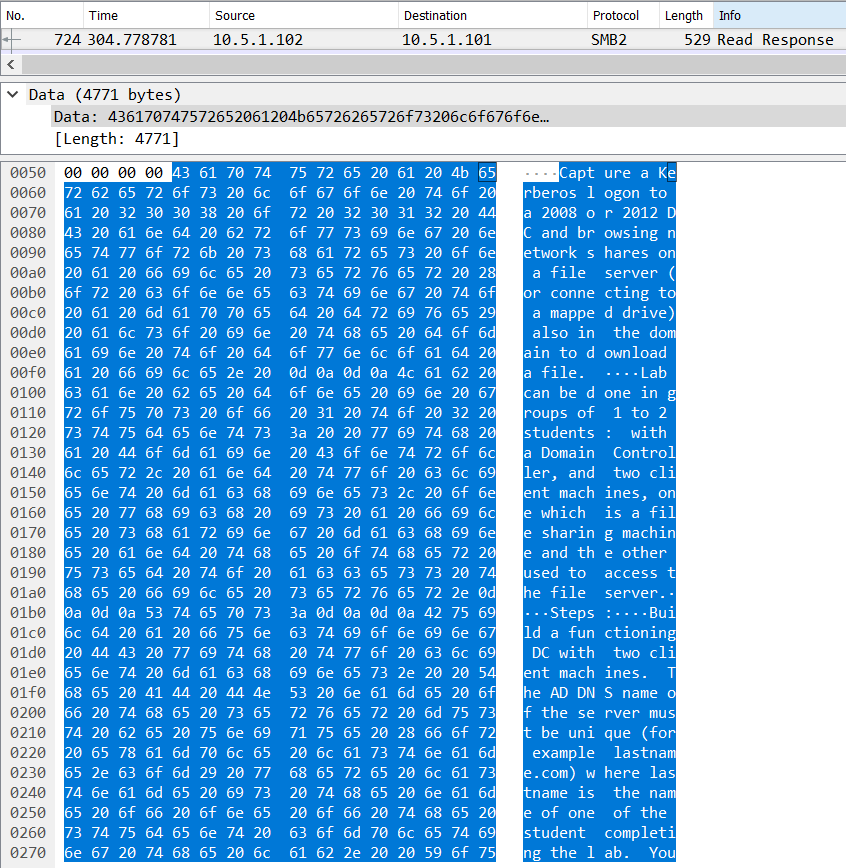
The client receives the encrypted timestamp and decrypts that and compares the timestamp to verify that the message came from the File Server.

9.Now the session is established between Client and File Server. The client can now access the files in file server. The client can find the files in path: //fs/public.



The file is sent over in clear text. The file is received by the client from file server.

**OUPUT:**



**RESULT:** We can see the contents of file in clear text.