Lab 5: Implementation and Application of Kerberos

Kerberos is a key distribution and user authentication service. It provides a centralized server to authenticate users to servers and servers to users. There are different versions of Kerberos, and in this lab, we are going to implement Version 4, which makes use of DES

Task

It is an individual project. Firstly, you need to create a chat program (socket) to establish the communication between client and servers. Basically, you will need a client **C**, and two servers, one of them serves as **V**, another serves as both **AS** and **TGS**. After establishing the communication, you can send messages step by step following the order shown in the figure below. In step 3 and 5, you should check if the received tickets are still valid (do not expire), which will be explained below. Note that "||" means concatenation.

Summary of Kerberos Version 4 Message Exchanges

(1) C → AS $ID_c \| ID_{tgs} \| TS_1$

(2) AS → C $\mathrm{E}(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}])$

$$Ticket_{tgs} = \mathrm{E}(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| \mathrm{Lifetime}_2])$$

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) C → TGS $ID_v \| Ticket_{tgs} \| Authenticator_c$

(4) TGS → C $\mathrm{E}(K_{c,tgs}, [K_{c,v} \| ID_v \| TS_4 \| Ticket_v])$

$$Ticket_{tgs} = \mathrm{E}(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| \mathrm{Lifetime}_2])$$

$$Ticket_v = \mathrm{E}(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| \mathrm{Lifetime}_4])$$

$$Authenticator_c = \mathrm{E}(K_{c,tgs}, [ID_C \| AD_C \| TS_3])$$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(5) C → V $Ticket_v \| Authenticator_c$

(6) V → C $\mathrm{E}(K_{c,v}, [TS_5 + 1])$(for mutual authentication)

$$Ticket_v = \mathrm{E}(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| \mathrm{Lifetime}_4])$$

$$Authenticator_c = \mathrm{E}(K_{c,v}, [ID_C \| AD_C \| TS_5])$$

(c) Client/Server Authentication Exchange to obtain service

Some initial parameter settings:

1. Use these fixed ID in your codes:

ID_c = "CIS3319USERID"

ID_v= "CIS3319SERVERID"

ID_{tgs}= "CIS3319TGSID"

2. Network address of client: AD_c= "127.0.0.1:{port}"

{port} here is the port number you use.

3. TS: Use Unix time (aka. Epoch time) in seconds as timestamp.

4. $Lifetime_2$= 60 and $Lifetime_4$= 86400 (in seconds)

5. Keys: K_c, K_{tgs}, K_v (default using 8 bytes) are pre-defined keys between C and AS, AS and TGS, TGS and V, respectively. Similar to lab 1 and 2, generate the keys in advance and load them in the proper process. The other keys $K_{c,tgs}$, $K_{c,v}$ (default using 16 bytes)

Check the validity of tickets:

Timestamps Checking: In step 3 and 5, TGS and V need to check if the $Ticket_{tgs}$and $Ticket_v$ are expired. For example, if (current Unix time – TS_2) $< Lifetime_2$, then $Ticket_{tgs}$ is still valid. Otherwise, it's expired and the conversation would be terminated.

IDs Checking: In step 1, 3, and 5, AS, TGS, and V need to check whether each notation such as ID_c, ID_v, AD_c is correct. Once there is a mismatch, the authentication should be terminated. For example, if the ID_c sent by Client ***C*** is different from that stored at TGS in step 4. The conversation should be terminated. Please test at least 3 situation such as ID_c, ID_v or AD_c mismatch respectively.

Printout:

Your codes are supposed to show the following information on screen for each step when executed.

step (1): print out the received message on AS side.

step (2): print out the plaintext of the received ciphertext, as well as $Ticket_{tgs}$on C side.

step (3): print out the received message and validity (valid or not) of $Ticket_{tgs}$on TGS side.

step (4): print out the plaintext of the received ciphertext, as well as $Ticket_{v}$on C side.

step (5): print out the received message and validity (valid or not) of $Ticket_{v}$on V side.

step (6): print out the plaintext of the received ciphertext, which should be TS_5+1 on C side.

Submission

- Submit a lab.zip to Canvas including two parts:

a) All the code you have for completing the task.

b) Documentation

  - Briefly summary how you design your program, problems you encountered and how you solved them.
  - Tests under different situations (Check the previous slides). First, you should test your program under initial settings. Then change the value of some parameters and record the results. Screen captures are recommended.

Rubric

Authentication Service Exchange (15)

Ticket-Granting Service Exchange (15)

Client/Server Authentication Exchange (15)

Documentation (25)

Program design (5)

Your testing results under different situations (15)

Demo (30)

No comments (-5)

Missing the deliverable (missing each of them would deduct 7.5)