Volume 3, Issue 11, November 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Kerberos: Simplified Ticketing

Randhir Bhandari
Computer Science Department
Shoolini University, Solan, (H.P), India

Sachin Sharma
Computer Science Department,
Shoolini University, Solan, (H.P), India

Abstract— The Kerberos Authentication Service designed & developed by Massachusetts Institute of Technology (MIT) it provides authentication by encrypting essential information it is widely adopted by organizations and comes in different versions latest available versions are version 4 & version 5. Kerberos is a standard based strong authentication environment. It provides single sign-on capability for clients. Generally Kerberos ticket generation and exchange is very complex process. In this we have generalized the ticket exchange process of version 5.

Index Terms- Kerberos, cryptography, encryption, decryption, ticket.

I. INTRODUCTION

The Kerberos in Greek Mythology means a three headed dog which stands at the gates of house of hades which let the dead's passes by to enter and eat anyone try to go back to the land of living. Here the three heads denotes the AAA known as Authentication, Authorization and Accounting. These three are the basic requirements for designing any secure environment. The Kerberos protocol is designed & developed to provide authentication services. The MIT designed Kerberos for providing authentication across unsecure networks by using the private key cryptography. The authentication in Kerberos is done through a trusted third party who is denoted as an authentication server. Till now MIT developed five versions of Kerberos. Version 1 through 3 was used internally by MIT version 4 is accepted beyond MIT. Models for administration and use of computer services differ from site to site and some environments require support that isn't present in Version 4. Version 5 of the Kerberos protocol incorporates new features suggested by experience with Version 4, making it useful in more situations. Version 5 was based in part upon input from many contributors familiar with Version 4.

"**Kerberos is an authentication protocol for trusted hosts on untrusted networks**". Kerberos is not beneficial if the host is not trustworthy. Otherwise, the intruder can use host as a key to get authentic. Intruder can impersonate by obtaining IP address for that server.

II. THE KERBEROS TICKET EXCHANGE MODEL

Kerberos came in existence to provide identity authentication between client and server. To fulfill this purpose, the message exchange is done between client, authentication server and application server. Initially, client presents its identity to the server using ticket, which contains principal, authenticator, service principal. Issuing of ticket is catered by Key Distribution Center (KDC). The KDC Stores the secret keys of clients and servers in the Database. These keys serve the client and server for the authenticity of the tickets they received. A ticket is limited in its lifetime, which decides its expiration.

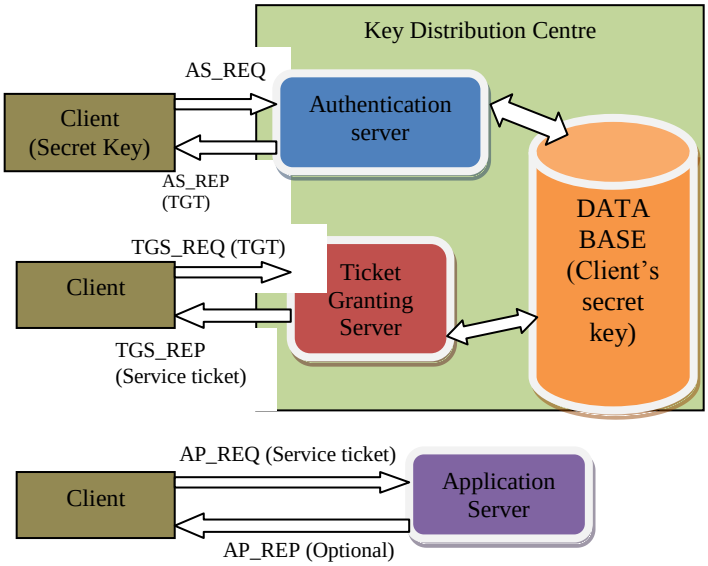

Figure 1: Kerberos Ticket exchange model

The Kerberos ticket exchange model is shown in the figure 1. It consists of KDC which contains:

1. *Authentication Server (AS):* It serves as authentication part of the Kerberos environment.
2. *Ticket granting Server (TGS):* It serves as service ticket provider for the Kerberos environment.
3. *Database (DB):* It serves as storage of secret keys of clients and servers for the Kerberos environment.

In Kerberos model client communicates with Authentication Server (AS_REQ, AS_REP), Ticket Granting Server (TGS_REQ, TGS_REP) to get authenticated and to get service ticket ($T_{c,s}$) respectively, in the end with the application server (AP_REQ, AP_REP) to get access of required service.

III. KERBEROS WORKING

Client to KDC (AS_REQ): The client sends its credentials to get authenticated by authentication server (AS), the part of Key Distribution Center (KDC) environment.

KDC to Client (AS_REP): AS replies to the request of the client with TGT and session key. TGT is encrypted using the TGS secret key and session key encrypted using user's secret key.

Client to KDC (TGS_REQ): Client sends request to TGS for service ticket. It contains TGT from the previous message and authenticator encrypted with the help of session key.

KDC to Client (TGS_REP): The TGS replies to the request of the client. It contains the service ticket encrypted using secret key of the service and session key generated by TGS and encrypted using session key generated in previous message by AS.

Client to Application Server (AP_REQ): In this client sends a request to access the service it requires. The request contains the service ticket, service name and the authenticator generated by client encrypted by service session key generated by TGS.

Application Server to Client (AP_REP): It's an optional reply to the client to prove authenticity of application server. It is generated when mutual authentication is required.

IV. TICKET GENERATION PROCESS

TGT

1. $Client \rightarrow KDC: C, S, n$
2. $KDC \rightarrow \{T_{c,s}\}K_s : \{C, lifetime, K_{c,s}\}K_s$
3. $KDC \rightarrow Client: \{K_{c,s}, n\}K_c, \{T_{c,s}\}K_s$
4. $Authenticator(A_c) = \{C, timestamp\}K_{c,s}$
5. $Client \rightarrow Server: \{A_c\}K_{c,s}, \{T_{c,s}\}K_s$

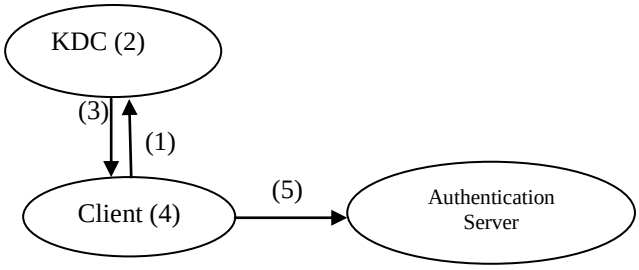

Figure 2: Process of getting TGT

V. TICKET GRANTING TICKET

Initial Ticket Exchange

Step1: The client sends the request to the KDC, containing credentials for identifying itself to the authentication server. It presents name of the client, service and a non-repeating identifier.

$$Client \rightarrow KDC: C, S, n$$

*Step 2***:** In this KDC generates a ticket containing random encryption key called session key ($K_{c,s}$).

$$KDC \rightarrow \{T_{c,s}\}K_s : \ \{C, lifetime, K_{c,s}\}K_s$$

Step 3: KDC uses the shared secret key (which is shared between the KDC and the Application Server) to encrypt session key ($K_{c,s}$) and the lifetime of TGT.

$$KDC \rightarrow Client: \{K_{c,s}, n\}K_c, \{T_{c,s}\}K_s$$

Step 4: In this client generates the authenticator (A_c) which is encrypted using session key ($K_{c,s}$) and contains timestamp and user (C) and present TGT and the authenticator (A_c) to the server.

$$Authenticator(A_c) = \{C, timestamp\}K_{c,s}$$

Step 5: Server extracts the credentials of client and session key ($K_{c,s}$) by decrypting ticket using secret key (K_s) shared between KDC and server.

$$Client \rightarrow Server: \{A_c\}K_{c,s}, \{T_{c,s}\}K_s$$

*Step 6***:** The Authentication Server uses session key ($K_{c,s}$) to decrypt the authenticator, which verifies the timestamp which proves that the client possesses the session key ($K_{c,s}$).

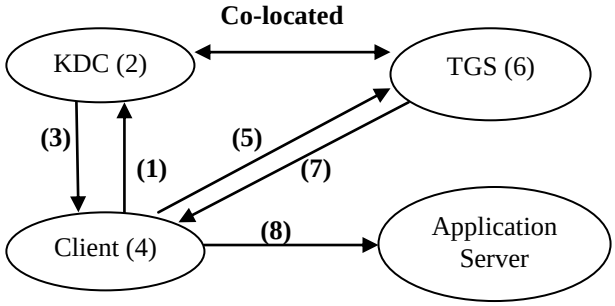

Figure 3: Process of getting ticket from TGS

TGS

1. $Client \rightarrow KDC: C, tgs, n$
2. $TGT = T_{c,tgs} = C, S, n, lifetime, K_{c,tgs}$
3. $KDC \rightarrow Client: \{K_{c,tgs}, n\}K_c, \{T_{c,tgs}\}K_{tgs}$
4. $Authenticator = A_c = \{C, timestamp\}K_{c,tgs}$
5. $Client \rightarrow TGS: A_c, \{T_{c,tgs}\}K_{tgs}, S, n$
6. $T_{c,s} = T_{service} = C, S, n, IPlist, lifetime, K_{c,s}$
7. $TGS \rightarrow Client: \{K_{c,s}, n\}K_{c,tgs}, \{T_{c,s}\}K_s$
8. $Client \rightarrow Server: \{A_c\}K_{c,s}, \{T_{c,s}\}K_s$

VI. TICKET GRANTING SERVER

The service ticket generation

In the initial ticket exchange the client obtain the ticket to start communication with the TGS. TGS is a special server which decreases the risk of exposure of the client's secret key K_c. TGS also make the KDC environment transparent to the client (user). As soon as the client received the TGT to communicate with the TGS it deletes its private (secret) key.

We can differentiate the TGS logically from KDC. But TGS take the help of the KDC database and run on the same machine on which the KDC runs.

Step 1: [Client to KDC (TGS)]. The client starts communicating with TGS by providing the KDC Host, its name (identity or Principal, C), timestamp (n) and the TGS address (TGSs).

$$Client \rightarrow KDC: C, tgs, n$$

*Step 2***:** to simplify the operation the TGT is defined as below. The TGT contains the username (principal, C), service name (S), timestamp (n), lifetime and the session key of the client and TGS ($K_{c,tgs}$).

$$TGT = T_{c,tgs} = C, S, n, lifetime, K_{c,tgs}$$

Step 3: [KDC (TGS) to Client]. The KDC (TGS) replies to the client the session key ($K_{c,tgs}$) and timestamp (n) encrypted with the secret key of the client (K_c). It also includes the TGT encrypted with the secret key of TGS.

$$KDC \rightarrow Client: \{K_{c,tgs}, n\}K_c, \{T_{c,tgs}\}K_{tgs}$$

Step 4: Now the client has to include the authenticator who can verify the identity of the client to TGS. The authenticator provides the information of the client i.e. the username, timestamp. It is encrypted with the session key of the client and the TGS.

$$Authenticator = A_c = \{C, timestamp\}K_{c,tgs}$$

Step 5: [Client to TGS]. The client sends the authenticator (A_c), TGT, name of service (S) and timestamp to the TGS.

$$Client \rightarrow TGS: A_c, \{T_{c,tgs}\}K_{tgs}, S, n$$

Step 6: Now as the TGS received the request from the client that what service it needs, the TGS has to verify the identity of the client with the help of the TGT, Authenticator, and the database of the KDC. TGS decrypts the authenticator and the TGT and check whether the client is the genuine user who can use the specified service or not. As the client seems to be authentic, the TGS generates the service ticket ($T_{c,s}$or ($T_{service}$). The ticket contains the username, service name, timestamp, IPlist, session key ($K_{c,s}$), lifetime of the ticket. The client cannot decrypt this ticket because this is encrypted with the secret key which is shared between the KDC environment and the application server.

$$T_{c,s} = T_{service} = C, S, n, IPlist, lifetime, K_{c,s}$$

Step 7: [TGS to Client]. As the ticket has been generated by TGS it sends this ticket to the client encrypted with the secret key shared between the application server and the TGS. It also sends the information of the client's session key with TGS.

$$TGS \rightarrow Client: \{K_{c,s}, n\}K_{c,tgs}, \{T_{c,s}\}K_s$$

Step 8: [Client to Application Server]. The client now starts communicating with the application server. The client sends the authenticator and the ticket to the application server. The client and the server now verify each other's identity to themselves.

$$Client \rightarrow Server: \{A_c\}K_{c,s}, \{T_{c,s}\}K_s$$

Step 9: Now the communication has been started between the client and the application server. To start exchanging messages the client and the server share a common session key ($K_{c,s}$)) to encrypt these messages which are sent over the network.

VII. CONCLUSION

The Kerberos provide only authentication by using tickets. Kerberos ticket generation and exchange is very complex process. It is very difficult for the beginners to understand the ticketing process. We have generalized this process. By going through this paper one can easily understand the exchange of tickets in Kerberos environment.

REFERENCES

[1] John T. Kohl and B. Clifford Neuman, ''the Kerberos Network Authentication Service,'' Version 5 Revision 5, Project Athena, Massachusetts Institute of Technology (April 1992).
[2] S. P. Miller, B. C. Neuman, J. I. Schiller, and J. H. Saltzer, *Section E.2.1: Kerberos Authentication and Authorization System*, M.I.T. Project Athena, Cambridge, Massachusetts (December 21,1987).
[3] gost.isi.edu/publications/kerberos-neuman-tso.html 11/11.
[4] http://kerberos.org/software/tutorial.html.
[5] C. Neuman, T. Yu, S. Hartman, K. Raeburn, *The Kerberos Network Authentication System* (RFC4120), July 2005.
[6] Roger M. Needham and Michael D. Schroeder, ''Using Encryption for Authentication in Large Networks of Computers,'' *Communications of the ACM* **21**(12), pp. 993-999 (December, 1978).

