SECURE KERBEROS SYSTEM IN DISTRIBUTED ENVIRONMENT

Miss Deepali D.Rane[1] Mr. Gajanan Babhulkar[2]

[1,2]Assistant Professor, Dept of Information Technology, D. Y. Patil College of Engineering, Akurdi

Abstract - *Kerberos is a symmetric key authentication system that allows clients to securely access networked services. Time stamps in Kerberos messages restrict the window in which an attacker can retransmit messages. Proposed system is designed to use AES encryption algorithm to ensure security. In this system new sub-session key is used for communication, using this technique it prevent the system from any attack and also any encryption technique may be used for developing this system. In this system, tickets include an explicit start time & end time, allowing tickets with arbitrary lifetimes.*

***Key Words*: – AES, Kerberos, Authentication, Security**

1. INTRODUCTION

Kerberos is a symmetric key authentication system that allows clients to securely access networked services. It can be used as part of single sign-on (SSO) architecture.

The Kerberos authentication model is based on the Needham-Schroeder symmetric-key protocol, described in the 1978. Kerberos is designed to provide secure symmetric key exchange by leveraging trusted third parties and shared secrets, and it allows mutual authentication of principals (clients and services). Authentication is the verification of an identity claim. Keys are never passed in plaintext, making it secure when an attacker has access to all network traffic.

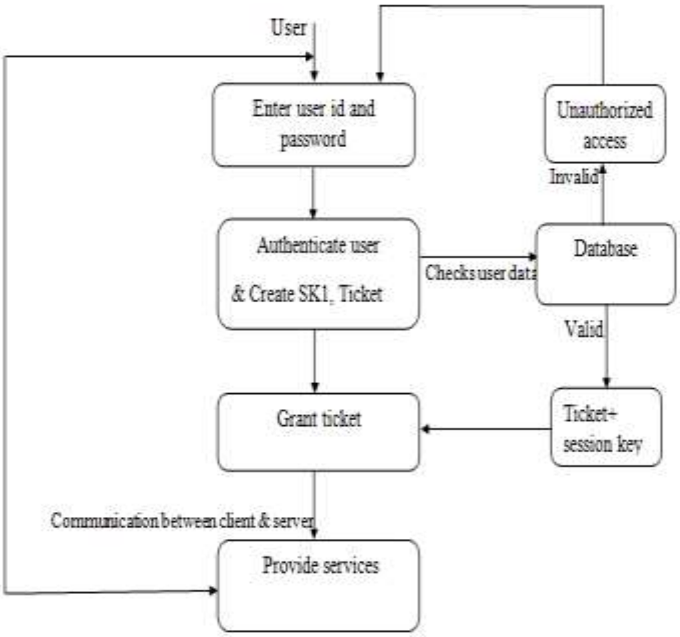
Much of Kerberos's complexity derives from the challenge of securely exchanging symmetric keys. Asymmetric or public key, cryptography addresses this issue.

The current version of Kerberos is version 5, released in 1993. Version 5 has addressed a number of security issues with Kerberos version 4, added new features, and introduced additional encryption methods beyond DES.

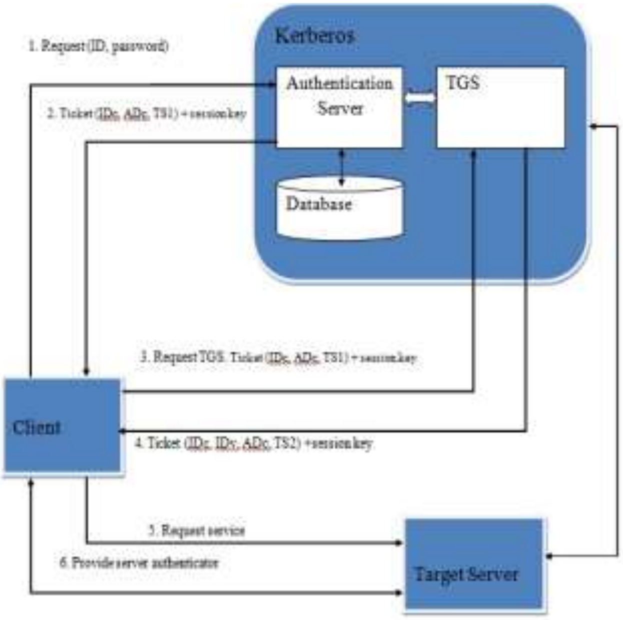
2. PROPOSED SYSTEM

This system is developed using Advanced Encryption Standard (AES) algorithm. In this system new sub-session key is used for communication, using this technique it prevent the system from any attack and also any encryption technique may be used for developing this system. In this system, tickets include an explicit start time & end time, allowing tickets with arbitrary lifetimes

2.1 System Architecture



2.2 Modules


Kerberos
1. Request (ID, password)
Authentication Server
TGS
2. Ticket (IDc, ADc, TS1) + session key
Database
3. Request TGS. Ticket (IDc, ADc, TS1) + session key
Client
4. Ticket (IDc, IDv, ADc, TS2) +session key
5. Request service
6. Provide server authenticator
Target Server

2.2.1 Authentication Server:

The client asks the authentication server for a ticket to the ticket-granting server. The authentication server looks up the client in its database, then generates a session key (SK1) for use between the Client and the TGS. Kerberos encrypts the SK1 using the client's secret key. The authentication server also uses the TGS's secret key to create and send the user a ticket-granting ticket (TGT).

2.2.2 Ticket Granting Server:

The client decrypts the message and recovers the session key, then uses it to create an authenticator containing the user's name, IP address and a time stamp. The client sends this authenticator, along with the TGT, to the TGS, requesting access to the target server. The TGS decrypts the TGT, and then uses the SK1 inside the TGT to decrypt the authenticator. It verifies information in the authenticator, the ticket, the client's network address and the time stamp. If everything matches, it lets the request proceed. Then the TGS creates a new session key (SK2) for the client and target server to use, encrypts it using SK1 and sends it to the client. The TGS also sends a new ticket containing the client's name, network address, a time stamp and an expiration time for the ticket all encrypted with the target server's secret key and the name of the server.

2.2.3 Application Server:

The client decrypts the message and gets the SK2. Finally ready to approach the target server, the client creates a new authenticator encrypted with SK2. The client sends the session ticket (already encrypted with the target server's secret key) and the encrypted authenticator. Because the authenticator contains plaintext encrypted with SK2, it proves that the client knows the key. The encrypted time stamp prevents an eavesdropper from recording both the ticket and authenticator and replaying them later. The target server decrypts and checks the ticket, authenticator, client address and time stamp. For applications that require two-way authentication, the target server returns a message consisting of the time stamp plus 1, encrypted with SK2. This proves to the client that the server actually knew its own secret key and thus could decrypt the ticket and the authenticator.

3. FRAMEWORK & METHODOLOGY

- Authentication exchange

The client asks the authentication server for a ticket to the ticket-granting server. The authentication server looks up the client in its database, then generates a session key (SK1) for use between the client and the TGS. Kerberos encrypts the SK1 using the client's secret key. The authentication server also uses the TGS's secret key to create and send the user a ticket-granting ticket (TGT).

Steps:

1. C=>AS: ID(c)||ID(tgs)||TS1
2. AS=>C: E(K(c)[K(c,tgs)||ID(tgs)||TS2||Lifetime2||Ticket(tgs)])

Ticket(tgs)=E(K(tgs),[K(c,tgs)||ID(c)||AD(c)||ID(tgs)||TS2||Lifetime2])

- Ticket-granting service exchange

TGS creates a new session key (SK2) for the client and target server to use, encrypts it using SK1 and sends it to the client. The TGS also sends a new ticket containing the client's name, network address, a time stamp and an expiration time for the ticket all encrypted with the target server's secret key and the name of the server.

Steps:

1. C=>TGS: ID(v)||Ticket(tgs)||authenticator(c)
2. TGS=>C: E(K (c,tgs),[K(c,v)||ID(v)||TS4||Ticket(v)])

Ticket(tgs)=E(K(tgs),[K(c,tgs)||ID(c)||AD(c)||ID(tgs)||TS2||Lifetime2])

Ticket(v)=E(K(v),[K(c,v)||ID(c)||AD(c)||ID(v)||TS4||Lifetime4])

Authenticator (c) =E(K(c,tgs),[ID(c)||AD(c) ||TS3])

- Client/server exchange

The client decrypts the message and gets the SK2. Finally ready to approach the target server, the client creates a new authenticator encrypted with SK2. The client sends the session ticket (already encrypted with the target server's secret key) and the encrypted authenticator. Because the authenticator contains plaintext encrypted with SK2, it proves that the client knows the key. The encrypted time stamp prevents an eavesdropper from recording both the ticket and authenticator and replaying them later. The target server decrypts and checks the ticket, authenticator, client address and time stamp. For applications that require two-way authentication, the target server returns a message consisting of the time stamp plus 1, encrypted with SK2. This proves to the client that the server actually knew its own secret key and thus could decrypt the ticket and the authenticator.

Steps:

1. C=>V: Ticket(v)||authenticator(c)
2. V=>C: E(K(c,v),[TS5+1])

Ticket(v)=E(K(v),[K(c,v)||ID(c)||AD(c)||ID(v)||TS4||Lifetime4])

Authenticator (c) =E(K(c,v),[ID(c)||AD(c) ||TS5])

CONCLUSION

The Kerberos protocol is designed to be secure even when performed over an insecure network. Since each transmission is encrypted using an appropriate secret key, an attacker cannot forge a valid ticket to gain unauthorized access to a service without compromising an encryption key or breaking the underlying encryption algorithm, which is assumed to be secure. Kerberos is also designed to protect against replay attacks, where an attacker eavesdrops legitimate Kerberos communications and retransmits messages from an authenticated party to perform unauthorized actions.

REFERENCES

[1] S.M. Bellovin and M. Merritt, " Encrypted key exchange: Password-based protocols secure against dictionary attacks", Published by IEEE Computer Society Conference on Research in Security and Privacy, year: 1992.

[2] Steven M. Bellovinand Michael Merritt, " Limitations of the kerberosauthetication system", published by Winter USENIX Conference,year:1991.

[3] Thomas Wu, "The secure remote password protocol", Published by Internet Society Network and Distributed System Security Symposium.

[4] William Stallings, "Cryptography and Network Security Principles and Practices", Published by Prentice Hall, Pages: 592, Year: 2005