Authentication applications – Kerberos

Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users. Kerberos relies exclusively on conventional encryption, making no use of public-key encryption.

Requirements for Kerberos:

Secure: A network eavesdropper should not be able to obtain the necessary information to impersonate a user. More generally, Kerberos should be strong enough that a potential opponent does not find it to be the weak link.

Reliable: Kerberos should be highly reliable and should employ a distributed server architecture, with one system able to back up another.

Transparent: Ideally, the user should not be aware that authentication is taking place, beyond the requirement to enter a password.

Scalable: The system should be capable of supporting large numbers of clients and servers.

Kerberos Version-4

A simple authentication dialogue

In an unprotected network environment, any client can apply to any server for service. The security risk is that of impersonation. To counter this threat, servers must be able to confirm the identities of clients who request service. But in an open environment, this makes burden on each server.

- An alternative is to use an authentication server (AS) that knows the passwords of all users and stores these in a centralized database. In addition, AS shares a unique secret key with each server.

The simple authentication dialogue is as follows:

1. C ->AS: IDc||Pc||IDv

2. AS -> C: Ticket

3. C ->V: IDc||Ticket

Ticket= E(Kv,[IDc||ADc||IDv])

Where, C: Client, AS: Authentication Server, V: Server, IDc : ID of the client,

Pc:Password of the client, ADc: Address of client, IDv : ID of the server,

Kv: secret key shared by AS and V, ||: concatenation

AS issues the ticket only if the client is authentic.

A more secure authentication dialogue

There are two major problems associated with the previous approach:

– Plaintext transmission of the password.

– Each time a user has to enter the password.

To solve these problems, we introduce a scheme for avoiding plaintext passwords, and a new server, known as ticket granting server (TGS).

Once per user logon session:

1. C ->AS: IDc||IDtgs

2. AS -> C: E(Kc,Tickettgs)

Once per type of service:

3. C ->TGS: IDc||IDv||Tickettgs

4. TGS -> C: ticketv

Once per service session:

5. C ->V: IDc||ticketv

Tickettgs=E(Ktgs,[IDc||ADc||IDtgs||TS1||Lifetime1])

Ticketv= E(Kv,[IDc||ADc||IDv||TS2||Lifetime2])

1. The client requests a ticket-granting ticket on behalf of the user by sending its user's ID to the AS, together with the TGS ID.

2. The AS responds with a ticket that is encrypted with a key that is derived from the user's password.

   Because only the correct user should know the password, only the correct user can recover the ticket. Password is not transmitted in plain text.

3. The client requests a service-granting ticket on behalf of the user. For this purpose, the client transmits a message to the TGS containing the user's ID, the ID of the desired service, and the ticket-granting ticket.

4. The TGS decrypts the incoming ticket and verifies. It checks to make sure that the lifetime has not expired. If the user is permitted to access to the server V, the TGS issues a ticket to grant access to the requested service.

If the user wants access to the same service at a later time, the client can simply use the previously acquired service-granting ticket and need not bother the user for a password.

5. The client requests access to a service on behalf of the user. For this purpose, the client transmits a message to the server containing the user's ID and the service-granting ticket. The server authenticates by using the contents of the ticket.

Kerberos V4 Authentication Dialogue Message Exchange

- Two additional problems remain in the more secure authentication dialogue:

1. Lifetime associated with the ticket granting ticket. If the lifetime is very short, then the user will be repeatedly asked for a password. If the lifetime is long, then the opponent has the greater opportunity for replay.

2. Requirement for the servers to authenticate themselves to users.

Kerberos Realms and Multiple Kerberi

A full-service Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers requires the following:

1. The Kerberos server must have the user ID and hashed passwords of all participating users in its database. All users are registered with the Kerberos server.

2. The Kerberos server must share a secret key with each server. All servers are registered with the Kerberos server.

Such an environment is referred to as a Kerberos realm.

- A Kerberos realm is a set of managed nodes that share the same Kerberos database. The Kerberos database resides on the Kerberos master computer system, which should be kept in a physically secure room
- A read-only copy of the Kerberos database might also reside on other Kerberos computer systems.
- A related concept is that of a Kerberos principal, which is a service or user that is known to the Kerberos system.
- Each Kerberos principal is identified by its principal name.
- Principal names consist of three parts: a service or user name, an instance name, and a realm name
- Networks of clients and servers under different administrative organizations typically constitute different realms.

Kerberos provides a mechanism for supporting interrealm authentication. For two realms to support interrealm authentication, a third requirement is added:

3. The Kerberos server in each interoperating realm shares a secret key with the server in the other realm. The two Kerberos servers are registered with each other.

- The scheme requires that the Kerberos server in one realm trust the Kerberos server in the other realm to authenticate its users. Furthermore, the participating servers in the second realm must also be willing to trust the Kerberos server in the first realm.

Kerberos version 5

Version 5 of Kerberos provides a number of improvements over version 4.

provides improvements over v4

– addresses environmental shortcomings

– and technical deficiencies

Differences between version 4 and 5 Version 5 is intended to address the limitations of version 4 in two areas:

Environmental shortcomings

- Encryption system dependence
- Internet protocol dependence
- Message byte ordering
- Ticket lifetime
- Authentication forwarding
- Inter-realm authentication

Technical deficiencies

- Double encryption
- PCBC encryption
- Session keys
- Password attacks

The version 5 authentication dialogue

First, consider the authentication service exchange. Message (1) is a client request for a ticket-granting ticket. As before, it includes the ID of the user and the TGS. The following new elements are added:

Realm: Indicates realm of user

Options: Used to request that certain flags be set in the returned ticket

Times: Used by the client to request the following time settings in the ticket:

from: the desired start time for the requested ticket

till: the requested expiration time for the requested ticket

rtime: requested renew-till time

Nonce: A random value to be repeated in message (2) to assure that the response is fresh and has not been replayed by an opponent

The authenticator includes several new fields as follows:

Subkey: The client's choice for an encryption key to be used to protect this specific application session. If this field is omitted, the session key from the ticket (Kc,v) is used.

Sequence number: An optional field that specifies the starting sequence number to be used by the server for messages sent to the client during this session. Messages may be sequence numbered to detect replays.