Authentication Applications

Session 4

LEARNING OBJECTIVES

On completion of this session you should:

- Appreciate the importance of authentication tools
- Understand Kerberos authentication protocol
- Be familiar with X.509 directory authentication service

Contents

Reading

Prescribed readings

Reading 1: Read W. Stallings, Network Security Essentials - Application and Standard, 4th edition, Prentice Hall, 2010, pp. 112 to 113.

Reading 2: Read W. Stallings, Network Security Essentials - Application and Standard, 4th edition, Prentice Hall, 2010, pp. 113 to 117.

Reading 3: Read W. Stallings, Network Security Essentials - Application and Standard, 4th edition, Prentice Hall, 2010, pp. 115 to 122.

Reading 4: Read W. Stallings, Network Security Essentials - Application and Standard, 4th edition, Prentice Hall, 2010, Section 4.4, pp. 130 to 138.

Skim through:
http://publib16.boulder.ibm.com/pseries/en_US/aixbman/security/cas_pki.htm

4.0 Introduction

In today's organization, we mostly have an open distributed architecture consisting of workstations (clients) and distributed/centralized servers. Therefore, enforcement of access restriction to the services on servers is crucial for security. In such an environment, a workstation can not be trusted to identify its user correctly to network services for a number of reasons:

- a user may pretend to be another user,
- a user may alter the network address of a workstation,

- a user may eavesdrop on message exchanges and use a replay attack.

In this environment, the following three approaches to security can be considered [1]:

- rely on client workstations to assure user(s) identity and rely on server to enforce a security policy based on user id.
- require that client systems authenticate themselves to servers, but trust the client systems to assure user(s) identity.
- require the user to prove identity for each service on the servers and also the servers to prove their identity to clients.

4.1 Authentication Requirements

The first or second strategy may be suitable for a small closed environment but is inadequate for larger open environment. In this case, the third approach is needed for protection of information and resources. Kerberos, an authentication protocol, supports the last approach. As per specification Kerberos satisfies the following requirements [1]:

Security: Kerberos should be secure enough to prevent an eavesdropper to obtain necessary information to impersonate a user.

Reliability: Kerberos should be highly reliable to ensure the availability of the service that relies on Kerberos for access control.

Transparency: The user should not be aware of authentication taking place except entering password.

Scalability: The system should be capable of supporting large number of clients and servers.

Reading 1:
Read W. Stallings, Network Security Essentials - Application and Standard, 4th edition, Prentice Hall, 2010, pp. 112 to 113.

4.2 Simple Authentication Dialogue

When a request is made by a client for a network service, the server must be able to confirm the identity of the clients. This places a substantial burden on the server where each client/server interaction requires authentication. An alternative is to use an Authentication Server (AS). AS stores the password of all users in a centralized database and share a unique key with each server. A simple dialogue between client (C), Authentication Server (AS) and Server (V) could follow the steps below [1]:

1. C -> AS: ID_c || P_c || ID_v
2. AS -> C: *Ticket*
3. C ->V: ID_c || *Ticket*

   $Ticket = Ek_v [ID_c || AD_c || ID_v]$

Here ID_c =identifier of user on C, ID_v =identifier of server V, P_c =password of user on C, AD_c =network address of C, K_v=secret encryption key shared by AS and V, and || indicates a concatenation.

In this scenario, the following things take place:

1. C sends a message to AS that includes user's ID, server's ID and user's password.
2. AS checks the database for user ID and password match, and whether the user has access permission to V. If passed, it takes the user as authentic. To convince the server V, AS creates a ticket. The ticket contains user ID, server ID and network address, all encrypted by a secret key shared by AS and V.
3. C sends a message with C's ID and the ticket. V decrypts the ticket and verifies whether the user ID in the ticket is the same as the unencrypted user ID. If those match, the server grants requested service.

4.3 More Secure Authentication Dialogue

The above simple authentication protocol is too simple to use practically and has the following problems:

1. The user needs to enter password every time he/she wants to access a service. Each attempt for the same service requires reentering the password.

2. It needs a new ticket for every different service.

3. It transmits the password in plaintext format (step 1) which makes it easy for an eavesdropper to capture the password.

The above scheme can be improved by avoiding plaintext password and using another server, called ticket-granting server (TGS). The new scheme could incorporate following steps [1].

Once per user logon session:

1. C -> AS: ID_c || ID_{tgs}

2. AS -> C: Ek_c $[Ticket_{tgs}]$

Once per type of service:

3. C -> TGS: ID_c || ID_v || $Ticket_{tgs}$

4. TGS -> C: $Ticket_v$

Once per service session:

5. C -> V: ID_c || $Ticket_v$

   $Ticket_{tgs} = Ek_{tgs} [ID_c || AD_c || ID_{tgs} || TS_1 || Lifetime_1]$

   $Ticket_v = Ek_v [ID_c || AD_c || ID_v || TS_2 || Lifetime_2]$

The scheme operates as below:

1. The client requests a ticket-grant ticket to AS by sending user's ID and TGS ID.

2. AS sends a ticket encrypted with a key K_c ***derived from the user's password***. When C receives this encrypted ticket, it decrypts the ticket only if the password is correct. Any user with an incorrect password won't be able to recover the ticket. If an opponent manages to capture the ticket and waits until the user logs off, he/she can use the captured ticket to gain access to that workstation or configure his networks with the same network address. To avoid this, the ticket includes a timestamp and a lifetime. The client gets a reusable ticket and need not re-enter the password for each new

service until the ticket expires. The ticket expires when the user logs off.

3. The client requests a service granting ticket.

4. The TGS decrypts the ticket by using the secret shared key k_v. It checks the lifetime, compare the user ID and network address of the incoming message with those of the decrypted ticket. If the user has access permission for V, the TGS issues a ticket. This ticket also contains timestamp and lifetime, and is reusable.

5. The client request access to the service V using the ticket.

The scheme requires ***only one password query per user session*** and protection of the user password.

Reading 2:
Read W. Stallings, Network Security Essentials - Application and Standard, 4th edition, Prentice Hall, 2010, pp. 113 to 117.

4.4 Kerberos V.4 Authenticate Dialogue

The above scheme offers more security but still two problems need to be addressed. They are:

1. The lifetime associated with the ticket-granting ticket- If it is too short, the user will be repeatedly asked for password; if it is too long, the opponent will have greater opportunity to replay.

2. There may be a requirement for servers to authenticate themselves to users.

The Kerberos authentication dialogue addresses these problems. The authentication scheme has the following steps [1]:

Authentication Service Exchange: to obtain ticket-granting ticket

1. C -> AS: $ID_c \,||\, ID_{tgs} \,||\, TS_1$

2. AS -> C: $Ek_c\,[\, K_{c,tgs} \,||\, ID_{tgs} \,||\, TS_2 \,||\, Lifetime_2 \,||\, Ticket_{tgs}\,]$

   $Ticket_{tgs} = Ek_{tgs}\,[\, K_{c,tgs} \,||\, ID_c \,||\, AD_c \,||\, ID_{tgs} \,||\, TS_2 \,||\, Lifetime_2]$

Ticket-Granting Service Exchange: to obtain service-granting ticket

3. C -> TGS: ID_v || $Ticket_{tgs}$ || $Authenticator_c$

4. TGS -> C: $Ek_{c,tgs}$ [$K_{c,v}$ || ID_v || TS_4 || $Ticket_v$]

$Ticket_{tgs} = Ek_{tgs}$ [$K_{c,tgs}$ || ID_c || AD_c || ID_{tgs} || TS_2 || $Lifetime_2$]

$Ticket_v = Ek_v$ [$K_{c,v}$ || ID_c || AD_c || ID_v || TS_4 || $Lifetime_4$]

$Authenticator_c = Ek_{c,tgs}$ [ID_c || AD_c || TS_3]

Client/Server Authenticaion Exchange: to obtain service

5. C -> V: $Ticket_v$ || $Authenticator_c$

6. V -> C: $Ek_{c,v}$ [TS_5+ 1] (for manual authentication)

   $Ticket_v = Ek_v$ [$K_{c,v}$ || ID_c || AD_c || ID_v || TS_4 || $Lifetime_4$]

   $Authenticator_c = Ek_{c,v}$ [ID_c || AD_c || TS_5]

The Kerberos authentication operates as below [1]:

1. The user logs on to workstation and requests service on host.

2. AS verifies user's access right in database, creates a ticket-granting ticket and a session key. Results are encrypted using a key derived from user's password.

3. Workstation prompts the user for password and then uses the password to decrypt the incoming message. It then sends ticket and authenticator that contain user's name, network address, and time to TGS.

4. TGS decrypts ticket and authenticator, verifies request, and then creates ticket for the requested server.

5. Workstation sends ticket and authenticator to server.

6. Server verifies the ticket and authenticator match and then grants access to service. If mutual authentication is required, server returns an authenticator.

Please see Fig. 4.1, page 106 in the textbook for a broad overview of Kerberos operation.

Reading 3:
Read W. Stallings, Network Security Essentials - Application and Standard, 4th edition, Prentice Hall, 2010, pp. 115 to 122.

4.5 X.509 Authentication Service

X.509 defines a framework of authentication services by X.500 directory to its user. The directory is a distributed set of servers maintaining information of users and may serve as a repository of public key certificates. Every user account participating in PKI authentication has a unique PKI certificate. Certificate authentication service uses digital-signature to authenticate a user during login. It locates the user's certificate and keystore based on the user's account name, obtains the certificate's matching private key from the user's keystore using the user's password, signs a data item with the user's private key, and checks the signature using the user's public key from the certificate. The user certificate is created by some trusted certification authority (CA) (e.g., Verisign) and placed in the directory by the user or CA. After the user authenticates, the system stores the user's certificate in protected memory, associating the certificate with every process created by the user. X.509 certificate format is used in S/MIME, IP security, SSL/TLS and SET.

The X.509 certificate has the following format [1]:

1. Version: version number of X.509 (1, 2 and 3).
2. Serial number: a certificate number which is unique within the issuing CA.
3. Signature algorithm identifier: the algorithm used to sign the certificate and associated parameters.
4. Issuer name: name of the issuing CA who has created and signed the certificate.
5. Period of validity: first and last date of the validity of the certificate.
6. Subject name: name of the owner of the certificate.
7. Subject's public key: public key of the subject
8. Issuer unique identifier: an optional bit string to identify the issuing CA uniquely.

9. Subject unique identifier: an optional bit string to identify the user uniquely.

10. Extensions: a set of extension fields used in version 3.

11. Signature: contain hash code of other fields encrypted with CA's private key.

A new certificate is issued to the user before the current one expires. However, a CA may revoke a certificate before it expires due to one of the following reasons:

- Compromise of the certificate's private key.
- Certificate owner left the company.
- Compromise of the CA's certificate.

The issuing CA handles the revocation of certificates by maintaining a certificate revocation lists (CRLs). Each entry consists of the serial number of the certificate and the revocation date.

X.509 includes three authentication procedures: one-way authentication, two-way authentication and three-way authentication. One-way authentication involves a single transfer of information from one user to another. Two-way authentication involves transfer of information in both ways. In three-way authentication, the user that initiates the authentication sends another message containing signed copy of nonce.

Reading 4:
Read W. Stallings, Network Security Essentials - Application and Standard, 3rd edition, Prentice Hall, 2007, Section 4.4, pp. 130 to 138.

Skim through:
http://publib16.boulder.ibm.com/pseries/en_US/aixbman/security/cas_pki.htm

4.6 Conclusion

In this study guide, we discuss the importance of authentication in ensuring security and the requirements of successful authentication tools. In a distributed client-server environment, besides the user (client) and the

service (server), other entities like authentication server and ticket-granting server are required to prevent a proponent from stealing/replaying a ticket. Examples in Section 4.2-4.4 show how all these entities engage into a dialogue to authenticate the user of a service. Kerboros is one of the most widely used authentication protocol. X.509 certificate based on PKI also provides a technique for authenticating users.

4.7 References

[1] W. Stallings, Network Security Essentials - Application and Standard, 4th edition, Prentice Hall, 2010.