© 2017 IJSRSET | Volume 3 | Issue 6 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099
Themed Section: Engineering and Technology

Protect Data from Attacking by Authenticate Clients Using Kerberos Protocol

Jabbar Al-Gburi

Faculty of Informatics, University of Debrecen, Hungary

ABSTRACT

Kerberos it is an authentication protocol, achieves all security needed for a system like authentication, confidentiality, integrity Kerberos helps us to protect our data from attacker from losing information while sending data Through the internet. Kerberos provides a distributed authentication services that allow a client running on behalf of a user to prove its identity to an application server. All this process is done without sending data across the network that makes difficult for the attacker or the verifier to impersonate the user. This paper explains Kerberos, authentication model and explains how Kerberos based on secret key encryption technology uses data Encryption algorithm to extend security for the system. Before Kerberos, if any user wants to access the network service they required to enter password each time, Which is very time-consuming process and insecure methods when user access services on a remote machine When the user logged on to a remote machine, the password travels in plain text though the network. Kerberos fixes these problems when it provides single-sign-on, which lets a user log into a system and access multiple system or applications without the need to enter the username and password multiple time. In addition, Kerberos is designed so that entities have to authenticate them.

Keywords : Authentication Protocol Cryptography, Kerberos Protocol, Authenticate Clients, Attacker, Data Encryption Algorithm, KDC, TGT, DCE

I. INTRODUCTION

Kerberos was developed to enable network applications to securely identify their peers. To achieve this, the client (initiating party) conducts a three-party message exchange to prove its identity to the server (the contacted party). The client proves its identity by presenting to the server a ticket (shown in figures as Tc,s) which identifies a principal and establishes a temporary encryption key that may be used to communicate with that principal, and an authenticator (shown in figures as Ac,s) which proves that the client is in possession of the temporary encryption key that was assigned to the principal identified by the ticket. The authenticator prevents an intruder from replaying the same ticket to the server in a future session. Tickets are issued by a trusted third party Key Distribution Center (KDC). The KDC, proposed by Needham and Schroeder [Nee78], is trusted to hold in confidence secret keys known by each client and server on the network (the secret keys are established out-of-band or through an encrypted channel). The key shared with the KDC forms the basis upon which a client or server believes the authenticity of the tickets it receives. A Kerberos ticket is valid for a finite interval called its lifetime. When the interval ends, the ticket expires; any later authentication exchanges require a new ticket from the KDC. Each installation comprises an autonomously administered realm and establishes its own KDC. Most currently-operating sites have chosen realm names that parallel their names under the Internet domain name system (e.g. Project Athena's realm is ATHENA.MIT.EDU). Clients in separate realms can authenticate to each other if the administrators of those realms have previously arranged a shared secret.

The Initial Ticket Exchange

Figure 1 shows the messages† required for a client to prove its identity to a server. The basic messages are the same for Versions 4 and 5 of Kerberos though the details of the encoding differ. A typical application uses this exchange when it first establishes a connection to a server. Subsequent connections to the same server require only the final message in the exchange (client caching eliminates the need for the first two messages until the ticket expires). In the first message the client contacts the KDC, identifies itself, presents a nonce (a timestamp or other non-repeating identifier for the request), and requests credentials for use with a particular server. Upon receipt of the message the KDC selects a

random encryption key Kc,s , called the session key, and generates the requested ticket. The ticket identifies the client, specifies the session key Kc,s , lists the start and expiration times, and is encrypted in the key Ks shared by the KDC and the server. Because the ticket is encrypted in a key known only by the KDC and the server, nobody else can read it or change the identity of the client specified within it. The KDC next assembles a response, the second message, which it sends to the client. The response includes the session key, the nonce, and the ticket. The session key and nonce are encrypted with the client's secret key Kc (in Version 4 all fields are encrypted in Kc). Upon receiving the response the client decrypts it using its secret key (usually derived from a password). After checking the nonce, the client caches the ticket and associated session key for future use. In the third message the client presents the ticket and a freshly-generated authenticator to the server. The authenticator contains a timestamp and is encrypted in the session key Kc,s . Upon receipt the server decrypts the ticket using the key it shares with the KDC (this key is kept in secure storage on the server's host) and extracts the identity of the client and the session key Kc,s . To verify the identity of the client, the sever decrypts the authenticator (using the session key Kc,s from the ticket) and verifies that the timestamp is current.

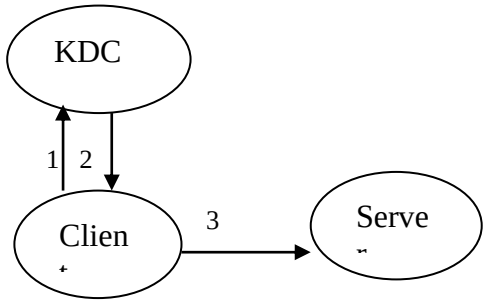

Figure 1.

Getting and using an Initial Ticket

Successful verification of the authenticator proves that the client possesses the session key Kc,s , which it only could have obtained if it were able to decrypt the response from the KDC. Since the response from the KDC was encrypted in Kc , the key of the user named in the ticket, the server may reasonably be assured that identity of the client is in fact the principal named in the ticket. If the client requests mutual authentication from the server, the server responds with a fresh message encrypted using the session key. This proves to the client that the server possesses the session key, which it could only have obtained if it was able to decrypt the ticket. Since the ticket is encrypted in a key known only by the KDC and the server, the response proves the identity of the server. For greater detail on the messages in Version 4 of Kerberos the reader is referred to [Ste88] and [Mil87]. Details about Version 5 can be found in [Koh92].

The Additional Ticket Exchange

To reduce the risk of exposure of the client's secret key Kc and to make the use of Kerberos more transparent to the user, the exchange above is used primarily to obtain a ticket for a special ticket-granting server (TGS). The client erases its copy of the client's secret key once this ticket-granting ticket (TGT) has been obtained, The TGS is logically distinct from the KDC which provides the initial ticket service, but the TGS runs on the same host and has access to the same database of clients and keys used by the KDC (see Figure 2). A client presents its TGT (along with other request data) to the TGS as it would present it to any other server (in an application request); the TGS verifies the ticket, authenticator, and accompanying request, and replies with a ticket for a new server. The protected part of the reply is encrypted with the session key from the TGT, so the client need not retain the original secret key Kc to decrypt and use this reply. The client then uses these new credentials as before to authenticate itself to the server, and perhaps to verify the identity of the server. Once the authentication is established, the client and server share a common session key Kc,s , which has never been transmitted over the network without being encrypted. They may use this key to protect subsequent messages from disclosure or modification. Kerberos provides message formats which an application may generate as needed to assure the integrity or both the integrity and privacy of a message.

Authorization Data

Kerberos is concerned primarily with authentication; it is not directly concerned with the related security functions of authorization and accounting. To support the implementation of these related functions by other services, Version 5 of Kerberos provides a mechanism for the tamper-proof transmission of authorization and accounting information as part of a ticket. This information takes the form of restrictions on the use of a ticket. The encoding of each restriction is not a concern of the Kerberos protocol, but is instead defined by the authorization or accounting mechanism in use. Restrictions are carried in the authorization data field of the ticket. When a ticket is requested, restrictions are sent to the KDC where they are inserted into the ticket, encrypted, and thus protected from tampering. In the protocol's most general form, a client may request that the KDC include or add such data to a new ticket. The KDC does not remove any authorization data from a ticket; the TGS always copies it from the TGT into the new ticket, and then adds any requested additional authorization data. Upon decryption of a ticket, the authorization data is available to the application server.

While Kerberos makes no interpretation of the data, the application server is expected to use the authorization data to appropriately restrict the client's access to its resources.

Among other uses, the authorization data field can be used in a proxy ticket to create a capability. The client requesting the proxy from the KDC specifies any authorization restrictions in the authorization data, then securely transmits the proxy and session key to another party, which uses the ticket to obtain limited service from an application server. Neuman [Neu91] discusses possible uses of the authorization data field in detail. The Open Software Foundation's Distributed Computing Environment uses the authorization data field for the generation of privilege attribute certificates (PACs). Privilege information is maintained by a privilege server. When a PAC is requested by a client the privilege server requests a Kerberos ticket identifying the privilege server itself, but restricting the groups to which the client belongs and specifying a DCE specific user ID. The ticket is then returned to the client which uses it to assert its DCE user ID and prove membership in the listed groups. In essence, the privilege server grants the client a proxy authorizing the client to act as the privilege server to assert the listed DCE user ID and membership in the listed groups. If the ticket did not include restrictions, it would indicate that the client was the privilege server, allowing the client to assert any user ID and membership in any group.

II. REFERENCES

[1]. F. Ricciardi, The Kerberos protocol and its implementations, November 2006

[2]. John T. Kohl, B.Clifford Neuman The Evolution of the Kerberos Authentication Service Digital Equipment Corporation

[3]. R. Rivest, "The MD4 Message Digest Algorithm," RFC 1320, MIT Laboratory for Computer Science (April 1992).

[4]. W. J. Bryant, Kerberos Programmer's Tutorial, M.I.T. Project Athena (In preparation)

[5]. J. Garman, Kerberos - The definitive guide, O'Reilly, August 2003 - ISBN: 0-596-00403-6

[6]. T. Ozha, An Authentication Protocol, Sushila Devi Bansal College of Engineering, Indore