OSF DCE SIG **J. Kotanchik (Security Dynamics)**
Request For Comments: 59.0 **March 1994**

KERBEROS AND TWO-FACTOR AUTHENTICATION

1. INTRODUCTION

The following quotation from the MIT Project Athena Kerberos V5Draft4 document identifies the primary strengths and weakness of the Kerberos environment:

> ''Kerberos provides a means of verifying the identities of principals (e.g., a workstation user or a network server) on an open (unprotected) network. This is accomplished without relying on authentication by the host operating system, without basing trust on host addresses, without requiring physical security of all the hosts on the network, and under the assumption that packets traveling along the network can be read, modified, and inserted at will.''

The ability of Kerberos to function in an open hardware and network environment is a unique strength. Kerberos suffers, however, from the same weakness that is characteristic of all traditional authentication paradigms: the reliable identification of the human component of the human/machine system.

Kerberos, like all other major machine authentication implementations, requires only a *single* factor for user identification: a privately owned *password* (''something you know'') which is used in conjunction with the user's public *name*. This password is quite susceptible to compromise and is the weakest link in an otherwise strong chain.

Two-factor authentication technology — ''something you know'' (a password) *and* ''something you possess'' (a *token*) — can be added to Kerberos Release 5 to provide a level of authentication for the human component as secure as is available from Kerberos for the machine component. Using optional fields in the initial client-to-authentication server (the ''AS Request'') exchange, adding a pre-authentication flag and linking with appropriate vendor-supplied authentication API, are all required.

Finally, note that although only ''raw'' Kerberos (as distributed from MIT) is explicitly mentioned, everything applies equally well to the version of Kerberos distributed with DCE.

2. DEFINITIONS

In the discussion below, several data items, hashing functions, and encryption steps are discussed. A consistent notation is used:

(a) A *password* is a reusable quantity, known only to an individual user. It usually consists of 4 or more alphanumeric characters. In the text below the password is referred to as `pwd`.

(b) A *PRN* (''pseudo-random number'') is a second authentication factor, generated by a handheld token device. There are a variety of such devices available from a number of vendors. The PRN typically consists of from 4 to 8 digits, and once used in an authentication sequence the PRN cannot be reused. Various kinds of tokens are available, some of which require special hardware for their use, such as swipe readers.

(c) When *Time* is referenced, it is the time of day (sampled at the authentication server) when the user initiated the login sequence.

(d) *Hashes* are one-way functions which transform an input quantity to a new quantity, usually with some data loss in the process. Hash functions are inherently one-way processes, in the sense that it is computationally infeasible to compute any input which produces a required hashed output. In the text below, use of a hashing function on some quantity `t` is indicated by the expression `h(t)`.

(e) *Encryption* of data is indicated by enclosing the data in braces $\{...\}_K$, where K is the encryption key. For example:

$\{\mathtt{PRN}\}_{\mathtt{h(pwd)}}$

indicates that the PRN is encrypted using a key which is the hashed value of the user password.

In the discussion of Kerberos request and authentication packets, only the fields relevant to the enhanced authentication sequence are shown and described.

3. HOW KERBEROS WORKS

The following describes the Kerberos authentication protocol[1] for the acquisition of the first ticket (a *ticket granting ticket*). It is a somewhat simplified description. It ignores realms and details such as the actual contents of tickets, which are themselves encrypted packets.

The user begins by providing the login name. The login name, the name of the ticket-granting service for which a ticket is desired, and the current time are put in a packet and sent to the Kerberos Key Distribution Center (KDC) in cleartext:

`{Login Name, TGS Name, Time}`

In this context the client workstation `Time` value is a convenient quantity for use in the ‘‘nonce’’ field of the `KRB_AS_REQ` transaction. When the transaction is received, the KDC looks up the login name and the ticket-granting service name and determines their private keys. In the case of the login name, the private key is the hashed user’s password. The KDC creates a temporary session key and a ticket for the ticket-granting service. The KDC returns a packet containing the temporary session key, the name of the ticket-granting service (same as in the cleartext request packet), the lifetime of the ticket, the nonce (copied from request packet), and the ticket to the ticket-granting service, encrypted under the already hashed user’s private key:

$\{\mathtt{TGS\ Key,\ TGS\ Name,\ Lifetime,\ Time,\ TGS\ Ticket}\}_{\mathtt{h(pwd)}}$

The principal requirements for Kerberos passwords (length, characters used, repeating fields, etc.) are derived from this use of the hashed password as an encryption key for the return packet. Use of short passwords, or ‘‘rational’’ passwords (e.g., user name, street address, Star Trek character names, ...) make the packet highly susceptible to password guessing games if the password hashing function is available to the trespasser. This, combined with the use of a static password, is the weakest point of the Kerberos authentication process.

When this packet is received, the user is prompted for his or her password, which is hashed to produce the user’s private key. The password is destroyed at this point. The packet above is decrypted using this key. The current time and name of the ticket-granting service are checked for validity. At this point the user has a temporary session key and ticket to the ticket-granting service. The user’s private key is destroyed at this point since it will not be required again until the next login.

4. THE THREAT

The concern is that someone snooping on the network can read the cleartext initial request, and could capture the reply. Because the format and contents of the reply would to some extent be known, someone trying different passwords would know when the packet had been deciphered. Success has been achieved when the name of the ticket-granting service and the current time in the decrypted reply packet equal those in the original cleartext request packet. It would not matter how long the deciphering would take; the lifetime of tickets would be of no concern, because once the decryption key (the hashed user’s password) was known, the initial cleartext request could be replayed. In this way the first ticket could be obtained through normal, legal channels, but by someone not authorized to do so! This is the network-sniffer analog of a compromised password.

1. Project Athena Technical Plan, Section E.2.1, ‘‘Kerberos Authentication and Authorization System’’.

5. THE RESPONSE: TWO-FACTOR AUTHENTICATION

In general there are three ways that an individual can identify him/herself. By ‘‘something secret we know’’ (such as a password), by ‘‘something unique that we possess’’ (a token of some sort), and/or by ‘‘something we are’’ (retinal pattern, voice print, etc.). The present Kerberos implementation utilizes only one of these components, the ‘‘something secret we know’’. Adding a second component would greatly enhance the reliability of the identification.

The current technology for using the ‘‘something we are’’ factor is still in its infancy, and most implementations suffer from either (and usually both) high cost and low authentication reliability. The use of the ‘‘something we possess’’[2] reliable authentication tool.

For nearly all vendors, the ‘‘something we posses’’ is a token, frequently of credit card or small calculator form factor, or of the size of a key fob, but having a display (and possibly a keypad) for the presentation of a 4-to-8 digit number. These tokens utilize one of two fundamentally different technologies:

(a) *Time-Varying* tokens present a number on the token LCD display for the user. This number changes at regular intervals. The current value of the number is supplied to the authentication software along with the login name and password.

(b) *Challenge/Response* tokens always have a numeric keypad. As part of the authentication sequence, the authentication server will provide a number to the user (the challenge) who must then enter it and a PIN into his or her token. The token will generate a new number (the response) on the LCD which the user will then key into the computer.

It can be shown that there is virtually no difference in the level of security provided by each of these competing technologies. The differences occur primarily in ease of use and token reliability.

From the user’s point of view the authentication process for each technology is quite different. A typical interaction using a challenge/response token is:

(a) The user is prompted for his or her login name and password.

(b) The login name and password (or value derived from the password) are subjected to validation by the Kerberos authentication server.

(c) If the test above succeeds, the token support software is invoked with the login name and returns some piece of information considered the *challenge*. Typically this is a numeric value.

(d) The user takes the number (the challenge) and enters it into his or her token. In most cases the user must also enter some other piece of information known only to the user (a PIN) and used to validate the user to the token. The token produces a value which is the *response* to the challenge.

(e) The response is returned to the token support software and the second stage of validation is performed.

And for the time-varying token:

(a) The user is prompted for his or her login name and password.

(b) The login name and password (or value derived from the password) are subjected to validation by the Kerberos authentication server.

(c) If the test above succeeds, the user is prompted for his or her *passcode* (the current value displayed on the token LCD).

(d) The response is returned to the token support software and the second stage of validation is performed.

Other differentiating factors in the two technologies could include token cost, reliability and warrenty of the token from the vendor, and availability of authentication software for different platforms and network protocols.

---

2. Obviously the ‘‘thing’’ possessed must be extremely difficult or extremely expensive to counterfeit.

6. ASSUMPTIONS

The remainder of this document proposes a set of modifications to the initial authentication sequence for a Kerberos environment. A number of assumptions have been made which have an impact on the selection of a strategy for supporting 3rd party authentication mechanisms.

(a) For a given Kerberos environment, there is likely to be a mixed community of users, some of whom are not subject to secondary authentication. For those who are require secondary authentication, it is possible that a variety of tokens are in use representing both time varying and challenge/response technologies.

(b) Modifications to the Kerberos protocol should be kept to a minimum, and for the default case of no secondary authentication required, the changes should be negligible.

(c) The changes necessary to support secondary authentication should be independent of vendor and technology.

(d) Changes to the content and architecture of the Kerberos user database should be limited to the addition of a small amount of new data. This will minimize the impact on the Kerberos administrative software.

(e) Changes in client code (`kinit`) should be generic in nature so that the same client code can be freely distributed regardless of the presence or absence of secondary authentication.

The following section presents a model for authentication which satisfies all of the above requirements.

7. GENERAL KERBEROS AUTHENTICATION SEQUENCE

This section proposes a modification to the Kerberos initial authentication protocol which supports proprietary tokens of the two predominant types. For users who are not required to provide second factor authentication, the protocol defaults to the standard Kerberos message sequence. For users with authentication tokens, a second interaction with the KDC is required which uses the `KRB_AS_REQ` and `KRB_AS_REP` message types.

The architecture hinges on three essential elements:

(a) Each token vendor can provide an API package for essential authentication functions as required by the Kerberos environment.

(b) The Kerberos database contains a new data item that identifies which, if any, secondary authentication token is assigned to a user. An unsigned binary byte field would allow up to 255 different token types to be supported. The identifier is by token type, but the protocol itself is vendor independent. Note that some vendors supply multiple token types. The identifier could be used as in index into an array of entry point addresses for the vendor supplied programming interface.

(c) The introduction of an initial *identification exchange* between the client and the Kerberos authentication server. This exchange allows the server to determine the type of authentication sequence to be invoked. The identification exchange uses two new application transactions `KRB_AS_IREQ` and `KRB_AS_IREP`. Definition of their contents is given below where these record types are introduced.

In general the proposal relies heavily on the use of the `padata` field which is present in both the `KRB_AS_REQ` and `KRB_AS_REP` messages.[3]

Briefly described, the steps are as follows:

(a) The user is prompted for his or her login name.

(b) The initial identification transaction is constructed as a message of type `KRB_AS_IREQ`.[4] The format of this transaction is:

---

3. See MIT Project Athena Kerberos V5Draft4-RFC, for complete definitions of these and the other message types used in this proposal.

4. All items in `KRB_AS_IREQ` are defined in Section 5.3.1 of Project Athena Kerberos V5Draft4-RFC.

```
KRB_AS_IREQ ::= [APPLICATION ?] SEQUENCE {
pvno[1]               INTEGER,
msg-type[2]           INTEGER,
req-body[3]           KDC-REQ-BODY
}
```

The transaction is sent to the Kerberos authentication server (AS) and the login name is used to identify the user in the Kerberos database. If the login name is not found in the Kerberos data base, a KRB_ERROR message is returned.

(c) If secondary authentication is required the AS invokes the appropriate vendor supplied interface routine to obtain the challenge data. For the case of time-varying tokens and users not requiring secondary authentication a meaningless random number (time of day is probably a good choice) should be used in order to avoid constructing an encrypted field with a well known cleartext value. For challenge/response tokens, the challenge value is returned by the vendor supplied interface.

(d) The KDC constructs a KRB_AS_IREP message whose format is:

```
KRB_AS_IREP ::= [APPLICATION ?] SEQUENCE {
pvno[1]               INTEGER,
msg-type[2]           INTEGER,
enc-part[3]           IREQEncpart
}
```

where IREQEncpart is defined as:

```
IREQEncpart ::= SEQUENCE {
ath-type[1]             INTEGER,
padata[2]               PA-DATA
}
```

The enc-part field is encrypted using the hashed user password obtained from the Kerberos user data database. The pvno, msg-type, and padata fields are as defined in various locations in the Kerberos specification. The values for ath-type are: 0 = Default Kerberos authentication; 1 = challenge/response authentication; and 2 = time-varying authentication. The padata field contains either the ‘‘challenge’’ or the time as noted above.

(e) When the KRB_AS_IREP transaction is received by the client, the user is prompted for his or password which is immediately hashed using the Kerberos hash function. The cleartext user password is removed from the client machine. The enc-part of the KRB_AS_IREP is then decrypted in the client using the hashed password as the key. Failure to decrypt the padata field results in a failed authentication.

(f) If no secondary authentication is required (indicated by ath-type set to zero) a standard Kerberos authentication sequence is initiated using KRB_AS_REQ/REP exchanges.[5]

(g) If second factor authentication is required, as indicated by the ath-type field, the user is prompted for additional authentication data. If the value of ath-type is 2 the client would issue a generic prompt of the form Enter PASSCODE:. If ath-type is 1, and after decryption of the padata field, a generic prompt similar to Enter response to <challenge>: would be issued. In either case, the user will supply a response which is a numeric string which will be termed a PRN for the following discussion.

(h) The client then constructs two strings to be used as encryption and decryption keys. They are (where || denotes concatenation):

---

5. Note that in the case of no secondary authentication, the Kerberos authentication server could without loss of generality have returned a KRB_AS_REP message (rather than KRB_AS_IREP) thus avoiding a second exchange of messages.

```
key1 = h(PRN||h(pwd))
key2 = h(h(pwd)||PRN)
```

(i) The value of the PRN (the user response) may then be discarded. The PRN itself is present in cleartext in the client machine only for the time required to construct the two keys above. The first key is used to construct the `padata` field which will be used as described in the next step.

(j) The client now constructs and delivers a `KRB_AS_REQ` transaction where the `padata` field contains the time encrypted under `key1`.

(k) The KDC can interpret this message by examining the `padata` field in the message, and from the vendor identifier in the Kerberos database invoke the vendor supplied entry point providing the `padata` field, the cleartext time value from the `KRB_AS_REQ` message and the hashed user password from the Kerberos database.

(l) The vendor software will validate the `padata` field and if successful will construct a copy of `key2` and return it to the KDC.

(m) If authentication fails, the KDC returns a `KRB_ERROR` message. If successful, the KDC prepares a `KRB_AS_REP` message with the TGT ticket encrypted using `key2` and dispatches it to the client with the standard Kerberos session key.

(n) The client is able to decrypt the ticket using its saved value of `key2`. The client then constructs its messages to the TGS using the session key. Note that the process of encrypting the return TGT ticket with a key that is a function of the PRN is a mutual authentication of the server to the client workstation. Since a server spoofing as a Kerberos server would not be able to generate the PRN required for the encryption of the return packet, the client workstation is assured that the authentication service is really a valid Kerberos server.

8. SUMMARY

The extended authentication sequence above requires only minimal changes for support of an extremely wide variety of tokens of the two basic types. It utilizes two new message types and two existing message types and stays within the confines of the well-defined Kerberos protocol. The most significant change to Kerberos is the maintenance of the token vendor data element in the Kerberos database.

At a more detailed level, the modifications to Kerberos fall primarily into two areas: changes to the existing authentication process in `kinit`, and changes within the Kerberos authentication server. Some modifications will also be necessary in the `kadmin` and `kpasswd` routines for the administration of the authentication type field in the Kerberos database. These changes do not change the user interaction with Kerberos except to request the second factor for user authentication or to process the system challenge.

9. ACKNOWLEDGEMENTS

The author is grateful to Bob Blakley (IBM), without whose encouragement and sponsorship this document would not have been published in the DCE-RFC series.

AUTHOR'S ADDRESS

Jim Kotanchik
Security Dynamics Inc.
One Alewife Center
Cambridge, MA 02140
USA

(Internet email not available)
Telephone: +1-617-547-7820