NETWORK AUTHORISATION PROTOCOL

George D.M. Ross 21/11/85

The Computer Science Department's LAN currently has on it upward of 70 machines, many of which are capable of acting as servers in some capacity or other while most are capable of acting as clients of these servers. A user wishing to access the facilities of any of these servers must first identify him/herself to that server, usually by quoting a username and corresponding password, even though the same username and password might already have been used to gain access to some other resource. There are two reasons for this: until recently the inconvenience this causes was not so pronounced since there were only one or two servers on the network; and because some clients are inherently untrustworthy the protocols were designed to be defensive against all. The increasing sophistication of clients in recent years, however, has meant that it is now practicable to introduce more complex protocols in order to provide for the increased functionality required by users. One of these increases in functionality would be for a user only to be required to "log on" once, with all the facilities of the network servers then becoming available. Note, however, that the protocol must still be designed with untrustworthy clients in mind.

It is likely that there will be systems on the network for some considerable time which will insist on performing some authorisation functions themselves: for example, users logging on to time-sharing systems via a directly-connected terminal line will, in many instances, be required by that system to quote a username and password to it before it will allow them to access its resources (network logins might enjoy more flexibility here, as the necessary code at the host end would probably have to be developed locally in the first instance). In order to accommodate such systems, and in order to enhance the reliability of network services, the protocols must be designed in such a way that there can be an arbitary number of authorisation points on a network (of course, the servers are not obliged to trust all of these).

The mechanism we adopt is that a user is identified to an authorisation point by quoting a username and password. In exchange for this, the user receives a token, selected by the server in some hard-to-predict way from a very large domain of tokens. When the user wishes to make use of some network service this token is presented to the server along with the identity of the authorisation point which issued the token: the server then queries the authorising agent as to the identity of the user.

Given long enough it would be possible to determine by exhautive search the value of some other user's token, thereby gaining access to all that user's resources. Alternatively, given time it might prove possible to frustrate attempts to render a broadcast medium secure by such means as encryption. Burglary of this kind can be made more difficult by restricting the valid lifetime of tokens. Each time a

token is requested one of the authorisation points the user must specify the desired maximum lifetime of the token; the authorisation point will issue a token with a lifetime no longer than this. An apparently longer lifetime must be synthesised by the user by exchanging the token for a new one before it expires. Lifetimes are expressed in seconds to expiry time.

The length or format of the tokens is entirely up to the authorisation point to decide: the protocol imposes no structure on them. In order to accommodate varying lengths of tokens consisting of arbitrary bit-patterns, tokens are passed as counted strings, viz. a length byte followed immediately by the data bytes.

An authorisation point should implement the following network requests (requests from a coresident user of a multi-tasking system might be handled differently):

Authorise: the username and password are checked, and if found to be valid a token of no more than the specified lifetime is issued.

Validate: the specified token is examined, and if found to be valid the identity of the user to whom it was issued is returned.

Exchange: the specified token is examined, and if found to be valid a new token of no more than the specified lifetime is issued. The new token will identify the same user as did the old one. If an additional username and password is specified, the new token identifies that username as well as the original one (this is a means of gaining additional privilege). The old token remains valid.

Cancel: the specified token is rendered invalid, and can no longer be quoted to the authorisation point.

Password: the username, old password and new password are specified. If the username and old password are found to be valid the username's password is set to be the new password. An arbitrary non-null response indicates success, in order to distinguish from the failure indication provided by a null response.

These requests and the corresponding responses would be packaged up as follows: for requests, there would be a request code followed by any fixed-length data (such as the specified lifetime) followed by variable-length data (usernames, passwords, tokens); for successful responses there would be any fixed-length data (lifetimes), followed by variable-length data (tokens, usernames). Failure would be indicated solely by a null response, with no indication as to the reason, in order to discourage attempts to compromise the authorisation point.