A FEW THOUGHTS ON USER VERIFICATION WITHIN A PARTY-LINE NETWORK

Paul Newland, ad7i
Post Office Box 205
Holmdel, New Jersey 07733

Abstract

This paper presents an idea for verifying that a user within a party-line network is who he or she claims to be. The idea assumes that the channel is a party-line and that potential intruders will monitor authorized communications and may attempt to masquerade as authorized users. No attempt is made to encrypt the authorized user's data for transmission over the party-line.

Introduction

Verification of a users identity to a host, historically, has been a major problem. At the lowest level of protection, many hosts require the user to "login" before he or she can make use of any of the system's protected resources. Login procedures, making use of passwords, are useful on most systems because it is assumed that a potential intruder will not be able to monitor easily the login process and determine someone else's password.

However, in an open system interconnected by unencrypted data communications radios (i.e., a party-line), the login process using passwords is ineffective because any casual observer can, by monitoring the radio channel, determine any other user's password. Also, once the user is logged in, there is nothing to prevent someone else from masquerading as the authorized user.

Offered below is an alternative approach to passwords for open, unencrypted, communications with a host. The method assumes that the user wants only to give commands to the host, not to do file transfers. However, with a few embellishments, the system could be expanded to include file transfers. The system would be as follows.

Proposal

When the host gives a prompt to the user, telling the user that she or he may send it the next command, the prompt will include a random number (RN) as part of the prompt. Each time a prompt is transmitted, a new random number is included. The host then encrypts the last RN transmitted using the key-for-the-day (KEY) previously agreed to by both the authorized user and the host; I will call this "Encrypted RN" the "ERN". The host would then place the ERN in storage for safe keeping and quick access. The user, on receipt of the new RN, would also encrypt it using the KEY (obtaining the same ERN if the KEYs are the same) and would save it, too, in a handy place. When the user wanted to issue a command to the host, it would give the command, as the host requires, but it would include the latest calculated ERN. When the host receives the command line, it compares the ERN received from the user with the ERN it calculated from the last RN transmitted as part of the prompt. If they match, the host assumes that the user knows the correct key. Thus, by returning the expected ERN, the user has proved to the host that he or she is authorized to access the system.

Vulnerabilities

This scheme is not without its problems. Because the key is changed only once a day, the range of numbers used for RNs must be large to ensure that RNs are not repeated during the 24-hour period. Also, the range of RNs must be large enough to make the probability of either determining the KEY or "guessing" the ERN sufficiently small. A RN of 64 bits should be large enough to overcome this problem. Another important consideration is the quality of the encryption technique. For Amateur Radio applications the DEA (that's Data Encryption Algorithm -- the software approach to DES) should be suitably strong.

This system is vulnerable to the sophisticated intruder who monitors the channel to determine the user's ERN and, before the user's packet transmission to the host is complete, the intruder jams the channel to block the user's packet from reaching the host. Next, the intruder sends his or her own packet with an illicit command and the ERN received

from the user. This scenario can be avoided by having the user transmit the ERN as the last item contained in the data segment of the packet.

Implementation

How might a system such as this be implemented? One way would be to include the host's 64 bit RN in the command prompt using hex notation and prefacing the RN with a special character sequence, perhaps composed of an unlikely sequence of punctuation characters so that a computer could identify it. Here's an example where the normal host system prompt is the word "yes?" and is prefaced by the RN.

```
://:0FEB23178ED83A9A yes?
```

The user% terminal emulation program (or, possibly, packet controller (TNC)) would assume that a new RN follows the escape sequence ://: and that it is in hex notation. It would, using the key-for-the-day, encrypt the RN to create an ERN to be saved for future use. When the user gives a command to the host that requires the transmission of the latest ERN, the operator would tell the terminal program (or, again, the TNC) to substitute the ERN in place of an escape sequence. When the terminal program sees the escape sequence (such as ##), it would substitute the ERN (with a different preface than the RN) before transmission. An example of a command line with the escape sequence might be:

```
COFFEEPOT ON ##
```

The terminal program would substitute the escape sequence (##) with the last calculated ERN. An example:

```
COFFEEPOT ON /::/07EAF832B1A9E9A2
```

A method such as this would add about 20 characters 'of overhead to each packet that contained a command for a "protected" host. However, if the protection is adequate for the environment, that appears to be a small price to pay.

Extensions

Again, although the idea presented here is simple, it should prove adequate protection against the more-than-average hostile user who might try to disrupt authorized communications. Other extensions to improve its protection might include calculation of a checksum or a more complex data reduction technique across all data within the frame and use that as another dimension for encryption to create the ERN[1]. Certainly what has been presented here is only a first step. I encourage others to pursue this and additional methods of providing authentication within an open, party-line, network.

Notes

[1] Private communication with Mr. Philip Karn, ka9q, Bell Communications Research.