SECURE FILE TRANSFER OVER TCP/IP*

Lawrie BROWN
Martin Gilje II JAATUN

Technical Report CS2/92

Department of Computer Science,
UC UNSW, Australian Defence Force Academy,
Canberra ACT 2600 Australia

Abstract

With the spread of internetworks able to link LANs together, and the increasing availability of cheap network hardware which can be used to monitor traffic on a network, the need to add additional security to traditional network utilities is becoming imperative. This paper details the development of security extensions to the FTP utility in the TCP/IP communications suite, to provide authentication of the user and encryption of the files transferred. We describe several means of providing these extensions, detail our final choice, and describe the results of the initial trials of our implementation.

1 Introduction

Traditionally it has been assumed that computer networks are secure. In the days when networks were the province of large organisations, and computers were managed by operators in secure rooms, this fiction was acceptable most of the time. Consequently, many common network utilities for remote terminal sessions, file transfer, remote printing and others, were written based on this assumption. With the increasing growth of internetworking, which links LANs together and leads to traffic over a wide area, and the ease of obtaining cheap network computers which can be used to monitor all traffic on a network, there is an urgent need to extend these network utilities to incorporate greater security features.

*This research was supported by ATERB grant N031/169

In this paper we detail the development of security extensions to the FTP utility, used to transfer files in the TCP/IP communications suite. After overviewing file transfer, we discuss the issues that need to be addressed in providing security extensions to it. These include the authentication of the user and encryption of the files transferred. We then describe some alternatives for providing these security extensions, and explain our final choice, which is based on earlier work in enhancing the security of the telnet utility. Some initial trials of our implementation are presented, which show that on a single LAN link there is a performance penalty in providing encryption, but that over a wide area internetwork, this is masked by the network delays. We conclude with a critique of our approach, and note some areas needing further work.

2 File Transfer - The Issues

A file transfer utility provides a means of exchanging files between two computer systems sharing a network. This involves copying a file from a file system on one computer running some operating system to the file system on another computer running another operating system. At this stage we are not considering the case of a network file system. In general the user interacts with the file transfer client on one system, which interacts with a server on the other system to control the exchange of the files (see Fig ??).

Two issues that arise from this arrangement are validating the identity of the user to both of the computer systems and protecting the privacy of the information being exchanged. Traditionally, authentication of the user has been performed by supplying a username and password. Over a dedicated line this method has a reasonable level of security, but when it is exchanged over an insecure network (where it may be intercepted), an appropriate level of security is not achieved. Some alternative methods include:

- Challenge-Response — which requires the server to send a challenge to the client, who encrypts it using a secret key known to both, and returns it for verification to the server (for more details see [davies price security networks 1984], [1]).
- Trusted-Key Server — involves the client and server both participating in a trusted-key authentication system, such as Kerberos [2], with whom they each have a secret key. It is used to exchange messages validating the identity of the parties in a session.
- Public Notary — is similar to the above, but uses either public-key digital signatures or zero-knowledge proofs-of-correctness to validate the identity of the parties. May not have to be involved in every session, but has a performance penalty compared to private-key systems.

In implementing our security extensions, there was a requirement to minimize any changes to the systems involved. Since both trusted-key servers and a public notary require major changes to all systems involved to introduce the new authentication framework, we chose the challenge-response scheme. This requires only the utilities involved to be changed.

The second security issue, that of providing privacy, may be addressed by encrypting the contents of the file being transferred. At issue then is the choice of cipher:

- Public-key ciphers have the ability to publish keys, but exact a performance penalty (see [3] for a detailed description).
- Private-key ciphers run much faster, but require session keys.

We decided to use private key ciphers, including both the DES [4], and the locally developed LOKI cipher [5], [brown loki asiacrypt 1992], and derived the session key from the authentication phase of the protocol.

Other factors to be decided include the choice of mode of use of the cipher, and the selection of the session key to be used by the cipher. All of these choices need to be addressed and negotiated by the security extensions to the basic protocol. These extensions thus require the use of a fairly complex negotiation phase, which impacts on the means by which they can be implemented.

3 FTP - A File Transfer Utility

FTP is the file transfer utility and protocol used in the TCP/IP communications suite (see [6]). FTP is a client-server protocol, with the following architecture:

Figure 1: FTP Architecture

The user interacts with the FTP client's User Interface (UI) on one computer to specify the operations required. The clients Protocol Interpreter (PI) interacts with the server's PI to initiate the requested operations. The file contents are then moved over a separate connection by the Data Transfer Protocol (DTP). Thus, two network channels are used in an FTP session – the control channel carrying FTP protocol messages, and a data channel for the actual file contents. The data channel bit stream is passed through uninterpreted.

In more detail, the client sends FTP commands over the control channel to the server. These consist of a 4-character command, and textual arguments:

```
AAAA <parameter list>
```

The server processes these commands and returns a response which consists of a 3-digit status code followed by a human-readable diagnostic message:

```
nnn <diagnostic message>
```

The control channel is a telnet compliant channel, but the telnet option negotiation mechanism is not normally supported.

4 Securing FTP

We identified two alternative approaches to incorporating security extensions into the FTP protocol. We could define some new FTP commands to conduct the negotiation. This would seem to be the obvious approach. However as shown above, the required negotiation requires passing parameters in both directions, some of which include binary information. This conflicts with the other FTP commands which only pass parameters from the client to the server. The alternative approach involves using the telnet option negotiation mechanism over the control channel.

The telnet remote terminal protocol includes an extensible option mechanism. We have previously used it to provide authentication and encryption extensions to telnet [7]. Two new telnet options have been defined. The authentication option is responsible for validating the identity of the user using either a challenge-response scheme (developed at ADFA), or a Kerberos scheme (developed by the Telnet Working group of the IETF). The encryption option enables the encryption of data exchanged (in one or both directions), using one of several possible encryption algorithms (the ADFA variants use either DES or LOKI in OFB mode with keys derived from the authentication phase, the telnet working group version uses DES in either CFB or OFB modes, with session keys exchanged in the Kerberos tickets during authentication). These new options are currently being standardised by the IETF in the US.

By utilising the specification of the FTP control channel as a telnet compliant channel, we could directly incorporate these features without needing other additional commands. The disadvantage is that this extension is not normally used, and involves a second protocol in the FTP PI. However we felt that the ability to reuse an existing protocol already being standardised, and its support for passing binary data in both directions, overcame the disadvantages. Hence we finally chose to use telnet options.

We have implemented our security extensions in the BSD Tahoe FTP client and daemon. Initial trials of this implementation were conducted between Sun 3 workstations on a local ethernet LAN, and between Australia and Norway. These indicated that the implementation is compute bound on a local LAN (with about a factor of 5 reduction in transfer rate), but is communications bound over the wide area link (encryption time is swamped by network delays). If encryption is not performed, very little difference is seen.

5 Limitations, Alternatives and Further Work

The major limitation in the current implementation is that the control channel is not encrypted. Consequently a passive attacker can monitor the names and permissions of files being exchanged, and an active attacker could turn off encryption. Theoretically, there is no reason why the control channel should not also be encrypted, but in practise there are problems with the current FTP code.

Another limitation is that third-party transfers are not supported. The current version of the telnet negotiations being used has no means of providing this feature for FTP transfers. Further work is needed to identify a suitable approach.

A more general criticism of our approach is that we are adding security to utilities bit by bit, rather than addressing the provision of a secure transport service which could be used by all network services. Whilst we recognise this is the best long-term approach, in the short-term we are addressing a requirement to provide security now, with minimal changes to the underlying system. Providing a secure transport service, which is being developed elsewhere [8] would require major changes to the host systems.

In conclusion, we have addressed a requirement for providing user authentication and encryption extensions to the FTP protocol, based on earlier work providing these extensions to telnet. We intend to continue and extend this research into means of providing security to network services.

6 References

[1] C. Mitchell, "Limitations of Challenge-Response Entity Authentication," *Electronics Letters*, 25, no. 17, pp. , 17th Aug. 1989.

[2] J. G. Steiner, C. Neuman and J. I. Schiller, "Kerberos: An Authentication Service for Open Network Systems," in *Proc. Usenix Winter Conf.*. USENIX Assoc., pp. , 1988.

[3] J. Seberry and J. Pieprzyk, *Cryptography: An Introduction to Computer Security*. Englewood Cliffs, NJ, Prentice Hall, 1989.

[4] ASA, "Electronics Funds Transfer - Requirements for Interfaces, Part 5, Data Encryption Algorithm," Standards Association of Australia, Sydney, Australia, AS2805.5-1985, 1985.

[5] L. Brown, J. Pieprzyk and J. Seberry, "LOKI - A Cryptographic Primitive for Authentication and Secrecy Applications," in *Advances in Cryptology: Auscrypt '90* (Lecture Notes in Computer Science), vol. 453. Berlin: Springer Verlag, pp. , 1990.

[6] D. E. Comer, *Internetworking With TCP/IP - Principles, Protocols, and Architecture*. Englewood Cliffs, NJ, Prentice Hall, 1991.

[7] L. Brown, "Secure Remote Login - the SECLOG option," in *AUUG 90 Conference Proceedings*. Sydney, NSW, Australia: Australian UNIX Systems Users Group, pp. , Sept. 1990.

[8] R. Nelson and J. Heimann, "SDNS Architecture and End-to-End Encryption," in *Advances in Cryptology - CRYPTO'89* (Lecture Notes in Computer Science), vol. 435, G. Brassard, Ed. Berlin: Springer Verlag, pp. , 1990.