Data Spread: A Novel Authentication and Security Technique

John Žic
School of Computer Science and Engineering
University of New South Wales
E-mail: `johnz@serg.cse.unsw.edu.au`

March 30, 1999

Abstract

This paper describes an authentication and security protocol called *data spread* for use on the Internet. The protocol applies address space diversity to outgoing messages, and when combined with reasonable (but not necessarily strong) encryption techniques, offers fast, secure and authentic-able information exchange between communicating entities.

1 Introduction

Message security and authentication has been of interests to humanity for as long as there have been secrets.

The development of broadcast radio-based communication meant that all communication was available to anyone who could receive the radio signal. As a consequence, radio communication in the early to mid 20th century relied on encryption techniques to provide message secrecy. However, there were two major problems with using plain encryption. First, messages could be intercepted, and subsequently analyzed, with the aim of breaking the encryption cipher (as was done with the Enigma Code). Once the cipher was broken, message security is naturally diminished. A second problem, and related to the latter, was that messages were susceptible to radio jamming as well as insertion and replay attacks if the cipher was broken. These problems, of message secrecy, security, robustness against interference lead to the development of *spread-spectrum* radio communication. The principles underlying spread-spectrum radio communication were patented by Lemarr and Antheil [4]. There are two distinct spread-spectrum techniques in use today in most mobile and secure radio communication systems, *frequency hopping spread spectrum* (developed by Lemarr and Antheil) and *direct sequence spread spectrum*:

Frequency Hopping Spread Spectrum or FHSS, is a scheme where the transmitter jumps from one frequency to another in a particular, but pseudo-random, sequence. The receiver "tracks" the transmitter in that it uses identical pseudo-random sequence of frequencies. Messages are thereby smeared across many frequencies, and if viewed in any particular channel, appear as random bursts of noise.

Direct Sequence Spread Spectrum or DHSS. In this scheme, the incoming bit stream is mixed with a Pseudo-random Noise Code (PN-code) which has many transitions possible during each bit time. This mixing effectively buries the real bit stream in noise. Extraction of the original bit stream by the receiver requires that it not only knows the characteristics of this noise and but that it can duplicate and track the the noise generated in the transmitter.

Both of these techniques cause the signalling power density to decrease inside a particular band, even though the total power transmitted is still the same whether or not spread-spectrum techniques are used. This in turn makes it more difficult to intercept or interfere with a message (the desired "signal") unless the spreading (effectively, the "noise") characteristics are known to the intruder. [8]

The Internet is an insecure broadcast medium with similar characteristics to that of broadcast radio. Anyone can, if they so desire, file contents, passwords, key files and so on. Although there have been many methods of securing and authenticating secret and sensitive information, we propose that an approach similar to the radio spread spectrum techniques be adopted which is transparently used on top of the Internet Protocol Suite i.e., without altering the underlying protocols. Other techniques which provided secure and authenticated built include Network Layer Security Protocol (NLSP), Transport Layer Security Protocol (TLSP) and swIPe [3].

The data spread protocols provide diversity based on using the address space (rather than frequency as done in spread spectrum radio systems). Sensitive information is hidden by scattering the data over a large set of addresses (for example, port numbers, IP addresses, sequence numbers or a combination of these) by the application of a time-varying *scattering function* whose characteristics are known only to the communication end points and whose execution at the end-points is synchronised. To make the system workable, the scattering function must be such that it is possible to recover the original message sequence. [1]

Some of these particular scattering functions can be used to provide address diversity to an application's port numbering space (provided under the Internet Protocol Suite's

[1]The formalisation of these functions will be the subject of a future paper

TCP/UDP). It is important to note (once again) that address diversity is not just restricted to using the port numbers, and could equally be well used at the data link layer, or the network layer, or a combination of all three. Furthermore, optimal selection of a particular scattering function (and its subsequent realisation in a data spread protocol) may be possible if sufficient information is known about the type of data being exchanged and the dynamic characteristics of the topology which underlies the address space.

This paper is divided into three parts. First, a brief discussion on the most common ways that network intrusion can occur on the Internet is presented. Following on from this, the paper presents a sequential data spread protocol, and give an algorithm which a pair of applications may use to provide secure and authenticated communication, albeit slowly and sequentially. The paper closes with a discussion of how this sequential port-based technique may be extended to include other levels of the Internet Protocol Suite in providing security and authentication.

2 Network Intrusion: Principles and Practice

2.1 History

The Internet Protocol suite is based on a *best-effort, unreliable datagram* service. It was born of the needs of a military network system which could survive large and permanent failures during wars. The Internet developed with minimum trust and authentication of messages and no centralized control, once again to ensure network survivability under war situations. As a consequence of these design decisions, it is relatively easy to read, alter, insert and delete messages at any layer of the network's protocol stack. It is also straightforward to attack the network so that services and resources are no longer available to a user by using up all available services and resources.

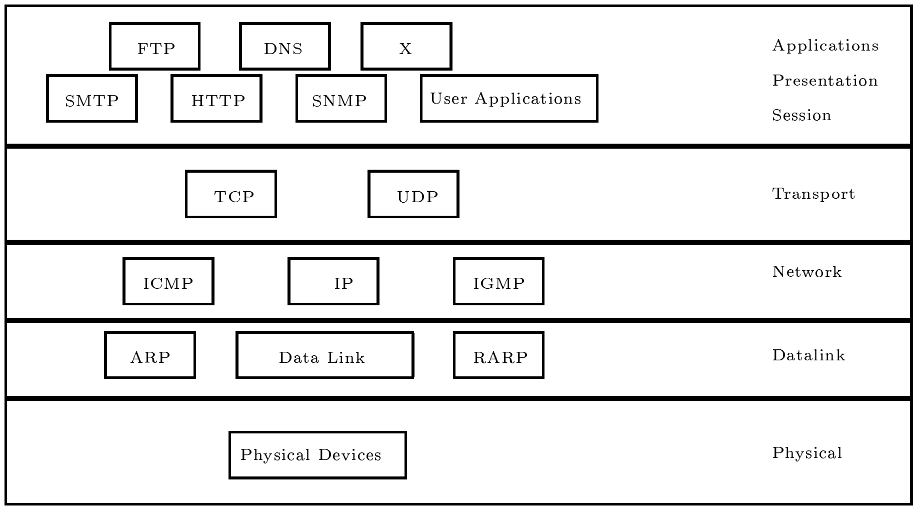

Figure 1: Internet Protocol Suite

The Internet Protocol Suite (popularly known as TCP/IP) consists of many components (see Figure 1) at all levels of the network, all of them susceptible to one form of attack or another. For example, the ARP protocol is used to find the MAC level address from a given IP address. A malicious user could easily generate fake ARP requests and divert all traffic to the user's machine. TCP is susceptible to in some circumstances to *sequence number attacks* [5, 1]. SMTP can allow fake e-mail addresses to be generated—it is a trivial exercise for mail to apparently come directly from `Bill.Door@MacroSaft.com`, for example. Recently, the "smurf attack" [9] has become somewhat fashionable and disrupted some of our networked facilities. A "smurf attack" leads to ICMP echo reply messages flooding a network, allowing very little capacity for other traffic. Many other examples of attack are well-documented (eg [2, 9]) and make interesting and enlightening reading, particularly to system administrators and others interested in security.

These security and authentication issues were only minor concerns in the early days of the Internet (when it was the ARPAnet) since it was an experimental, research network which consisted of a small number of trusted nodes. Any problems were easily traceable, and was seldom malicious. However, when the ARPAnet became the Internet, the network size grew exponentially, which in turn lead to more cases of "security", "authentication" and other breaches. This in turn lead to extensive use of filtering (including IP address filtering) and firewalling[2] techniques. Other techniques used to make malicious attacks difficult rely on encryption and authentication protocols between hosts on a network. Some of these measures, and their inherent weaknesses, are discussed by Ptacek and Newsham [6].

These prevention techniques rely on masking or hiding sensitive information and networks either by adding noise to the data stream, or by restricting the accessibility of information (by only allowing specific traffic). The proposed data spread technique hides information and data by *scattering* it throughout some address space. This scattering leads makes it difficult for an intruder to predict whether the data is part of one message, or part of some other group of messages. Reconstruction of a message is made more difficult without having access to the scattering sequences that is only known to the communicating parties.

The data spread technique comes in two forms: a simple, sequential version, and a more complex parallel version. Work on the parallel version is still in progress, and only the preliminary ideas are discussed in this paper.

We now present the sequential version of the data spread protocol.

3 A Sequential Data Spread protocol

Underpinning both the FHSS and DHSS spread-spectrum techniques is the introduction of a "noise" signal whose characteristics are known only to the sender and receiver. In this version of the data spread technique noise is introduced whose characteristics are known only to a client and its server.

In this example, we restrict our address space to that of TCP/UDP IP addresses ip and ports p. The pair (ip, p) defines a communication end points for particular applications which are executing (potentially) different hosts. A pair of end points defined by $sap = ((ip_k, p_k), (ip_l, p_l))$ is the specification of a unique communication channel between these applications, one which is executing on host k (with corresponding IP address ip_k and port p_k) and the other on another host l, with IP address ip_l and port p_l. As an example, an application executing on a host with IP address `129.94.214.32` and using port `6000` communicates via the communication end point defined by `(129.94.214.32, 6000)` to another application executing on host whose IP address is `129.94.242.40` and using port `60001` whose communication is always directed via its end-point `(129.94.242.40, 60001)`.

The scattering function we select in this example produces sequences of end point time pairs of the form $\langle (sap_0, t_0), (sap_1, t_1), \ldots (sap_i, t_i), \ldots \rangle$ where each $t_i \in \mathcal{T}$, and any

[2]A *firewall* is defined [2] is a collection of components placed between two networks such that

- all traffic between the networks must pass via the firewall,
- a local security policy defines which traffic is authorized to pass through the firewall and
- the firewall is immune to penetration.

Practical implementation meeting these requirements, is, of course, difficult.

sequence of t_i extracted from any sequence is non-decreasing. [3] For this reason it is referred to as the sequential scattering function.

This scattering function was the basis of the sequential data spread protocol which was implemented by Zaidi [10].

3.1 The implemented sequential data spread protocol

Here are the steps taken in establishing and exchanging messages during a session between a single client and server:

1. The server creates a socket on a well-known port and listens for client requests. This well-known port is an initial TCP control connection endpoint.
2. A client, upon successful connection to the server on this well-known port, authenticates itself, possibly by sending an encrypted string of the user's ID and password.
3. The client then sends a request to the server and waits for the server to send back some state information along the control connection.
4. The server uses a pseudo-random number to generate a pair of key indices which are used to indicate the line in the key file and the offset within the key. This information is used to calculate the number of new connections to use, the key used to change connections, and an encryption key.
5. This pair of numbers is sent back to the client which does the same calculation as the server, and since it has a copy of the key file, calculates identical values for each of these parameters.
6. The client and server each perform the following iterative computations to establish a new connection (modulo the number of channels to use selected previously) which is to be used for the next message.

   The server repeats the following in attempting to create a new connection endpoint.

   (a) If it can create a new endpoint, it signals *success* back to the client along the previously (possibly initial) established connection, and then waits for a connection to be established from the client using this as an endpoint.

   (b) If it cannot create a new server socket, the server signals a *failure* back to the client. The server then iteratively calculates the next server port number to use.

   The client waits for a signal from the server along the previously established connection.

   (a) If the client receives a failure signal, it proceeds to calculate the next server endpoint to use and goes back to wait for a signal from the server.

   (b) On receiving a success signal, it computes a client port number (using a separate pseudo-random number) and proceeds to complete the new TCP data connection between the client and server.

   (c) If the computed client port number cannot be used, it also skips ahead until one can be found and thereby successfully establishes a new TCP connection.

[3]Notice in particular that there is the possibility of having more than one end point associated with a particular time by the parallel "invocation" of that end point. Similarly, there is also the possibility of having more than one time associated with each end point in cases where there are multiple sequential "invocations" of the end point.

3.2 Discussion

The implemented sequential protocol has been shown to offer protection against eavesdropping attacks (either passive or active) ([7]). It also provides protection against sequence number attacks (outlined in [1]) by making it difficult for an intruder to predict the next sequence number to use, since only the trusted client and its server can predict the correct order.

However, two issues that need to be addressed can be identified in this implementation. First, this protocol, like many others, relies on having secure key files, as well as providing for their secure distribution. Like other protocols which rely on key files, the sequential protocol given above fails if the intruder obtains access to the key file (by whatever means, including bribery or direct physical theft). This difficulty may be eliminated by eliminating the static, globally applicable key file and using of key which is calculated and exchanged using public-key cryptography during session establishment.

Second, this implementation of a data spread protocol is extremely inefficient, since it is generates only sequential connections. This is expected to be remedied by moving to a parallel version of the protocol, where the sent and received data uses different parallel channels. Parallel transfers across several network connections would also speed up the transfer of information across the Internet. An unfortunate, but predictable, side effect of using this sequential form of data spread is that it is eminently suitable to timing analysis as well as allowing the intruder to use their port scanning and packet sniffing tools with confidence to break the protocol, as was recently demonstrated [7]. Once again, by adopting a *parallel* data spread protocol, the intruder's task is made more difficult in using these tools to work out proper message sequencing, and hence, the reconstruction of the (now unsecure) secure data.

The parallel data spread protocol is expected to replace a the sequential protocol's single (forward) channel with a *set* of channels whose port numbers can be selected using a time-varying scattering function whose characteristics are known only to the communicating parties. As in the sequential form, the sender and receiver must be well synchronised in order for the protocol to work. Suitable recovery protocols are being investigated that will assure synchronisation is maintained over a session.

4 Conclusions

The data spread protocols are believed to be an interesting development in the context of the Internet Protocol Suite, since they mirror the use and development of the spread spectrum techniques developed for the preceding insecure, broadcast medium – radio.

The question may be asked as to what sort of advantage there is in using the data spread protocols over the established security and authentication protocols. The answer lies in the fact that none of the existing protocols can be proven to be unbreakable (with the exception of the one-time pad), and so the use of data spread protocols in conjunction with these makes the intruder's workload higher in trying to get at the sensitive information. By scattering messages (in a controlled way known only to the participating entities) throughout an address space seems to provide another way of *hindering* the progress of an intruder accessing and reconstructing message sequence information (and thereby possibly decoding the sensitive information) by hiding the information in the noise of the remainder of the Internet traffic.

Another advantage in using these data spread protocols is that there is the possibility of using scattering functions which are amenable to parallel implementation. These parallel implementations may offer substantial increases in transfer rates by allowing the data to be spread over several parallel network connections.

Future work will investigate particular parallelisable data scattering functions, as well as their formalisation and analysis. Finally, these data spread protocols will be implemented and tested on realistic networked systems to gain experience in their use.

5 Acknowledgements

The idea of applying spread spectrum techniques to data authentication was the result of the author searching for a short single semester project for a Master's student. Ahsen Zaidi [10] implemented the initial sequential protocol, and Johannes Reiner [7] has just completed an investigation into the reliability and security aspects of this particular implementation.

References

[1] Steven M. Bellovin. Security problems in the TCP/IP protocol suite. *Computer Communications Journal*, 19(2):32–48, April 1989.

[2] William R. Cheswick and Steven M. Bellovin. *Firewalls and Internet Security: repelling the wily hacker*. Professional Computing Series. Addison-Wesley, 1994.

[3] John Ioannidis and Matt Blaze. The architecture and implementation of network-layer security under Unix. In *Fourth Usenix Security Workshop*, October 1993.

[4] Heddy Lemarr and George Antheil. U.S patent, 1941. Number 2,292,387.

[5] Robert T. Morris. A weakness in the 4.2BSD Unix TCP/IP software. Technical Report 117, AT&T Bell Laboratories, Murray Hill, NJ, February 1985. Computing Science Technical Report.

[6] Thomas Ptacek and Timothy N. Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical report, Secure Networks, Inc, January 1998.

[7] Johannes Reiner. The data spread technique: an intruder's guide. Master's thesis, School of Computer Science and Engineering, University of New South Wales, November 1998.

[8] Claude Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27(3,4):379–423, 623–656, July, October 1948.

[9] TFreak. papasmurf.c. Available from `http://www.rootshell.com`.

[10] Ahsen Zaidi. Data authentication using the data spread technique. Submitted as part of 45 credit project for course work Masters program in the School of Computer Science and Engineering, UNSW, November 1997.