Hiding Data in the OSI Network Model

Theodore G. Handel and Maxwell T. Sandford II

Weapon Design Technology Group
Los Alamos National Laboratory
Los Alamos, NM 87545
email: {thandel, mts}@lanl.gov

Abstract. Rather than searching for the holy grail of steganography, this paper presents the basis for development of a tool kit for creating and exploiting hidden channels within the standard design of network communications protocols. The Alice and Bob analogy, derived from cryptology, is used to present network protocols in a way that more clearly defines the problem. Descriptions of typical hidden channel design for each layer of the Open Systems Interconnect (OSI) network model are given. Methods of hiding and detection probabilities are summarized. Denying Bob and Alice the ability to communicate electronically may be the only absolute solution.

1 Introduction and Background

Rather than joining the quest for the holy grail of steganography, this paper focuses on the implementation of hidden writing techniques and covert channels within the OSI network model.

Our objective is to demonstrate the covert channels that exist throughout the network protocol architecture. It is important that network designers and security managers understand that their network systems may be effectively subverted in a wide variety of locations within the network. Inherent vulnerabilities included with network functionality cannot be ignored. The OSI model is the standard network model against which nearly all current network models are compared. However, the OSI model does not exist *per se* in functional systems. The most useful purpose of this model is to break complex networks into manageable pieces that can be easily understood.

We use the Bob and Alice analogy to describe the function of the OSI network protocol in a way that does not encumber the reader with the details of specific equipment. The examples given are not necessarily exceptional steganographic techniques, but are used to illustrate the principles of how data may be concealed within the OSI layers.

2 The Alice and Bob Protocol Analogy

This analogy is used to describe the details of any protocol in a palatable way (Schneier[1]). It describes the concept of the protocol without getting into numerous specific technical details.

2.1 The Plot

Bob and Alice are incarcerated in prison. They want to communicate overtly (which is permitted by Walter, the warden) and covertly (which is absolutely not permitted). Furthermore, if Walter discovers that Alice and Bob are using cryptography or communicating covertly, he will permanently disallow all subsequent communication between them. Alice and Bob can communicate using networked computers. Walter manages the network, and can monitor their message traffic. Bob and Alice wish to devise ways to communicate secretly. It is up to Walter to foil Alice and Bob's attempts at secret communication, and to devise a way to alter the contents of the secret communication.

We will attempt to answer these questions:

1.) Does technological improvement of the information system benefit Alice and Bob?

2.) Does the technological improvement of the information system benefit Walter?

3.) Does placing additional Walters or Eves in the system assure detection of hidden messages?

3 Potential impact of data hiding on information security

3.1 A tool kit of potential techniques

Functionality is present that can exploited in every layer of the OSI Model. One or more of these techniques to be discussed may be selected. Security

1 Applied Cryptography Chapter 2. See references

personnel should be aware that there is a broad spectrum of exploitation opportunities that should not be overlooked.

Not all techniques work in all systems. Therefore, several techniques should exist to cover a range of possibilities. Multiple techniques will assure that if one is discovered, others may remain secure. A litmus test exists for finding the system resources qualifying for exploitation, for covert communication, and for hiding data.

For covert channels, system elements that arbitrate resources are likely candidates. The quality of the covert channel can be expressed in terms of delectability (arbitration must be measurable by the recipient), indistinguishability (false arbitration versus real arbitration) and bandwidth (arbitrations per second).

For hiding data, the system elements having uncertainty, redundancy, or degeneracy are likely candidates as containers. The quality of the container can be expressed in terms of concealment (hidden data cannot be recognized as an uncorrelated component of cover data), indistinguishability (hidden data cannot be separated from the cover information), and bandwidth (ratio of hidden data to cover data).

We develop the toolkit from the basic source code for each layer of the OSI model. The modified source code is compiled and the new executable modules replace the existing modules. In some cases it is not necessary to replace existing modules, but merely to create an application that can access the internal network caches or tables already existing in the target system. A protocol analyzer, a C language compiler, an assembler, a general knowledge of network architecture, and some degree of programming skill fill out the remaining requirements. Note that protocol analyzers, complete with source code, now exist as software packages[2] that can be installed on a workstation utilizing existing network hardware. As it can be seen, the capital investment for such a venture is trivial, and well within the means of individuals so motivated. Hardware modification is not necessary. Except for the physical layer, hardware can be completely avoided.

3.2 Detection and countermeasures

Designers must realize that covert channels and data hiding locations exist, can be exploited, and are *inherent* components of the information system. It must be understood clearly that there is unknown functionality in modern information system components. Security can deal effectively with known vulnerabilities, but not with unknown functionality. The effectiveness of security is limited by cost-performance tradeoffs, and penalties for additional overhead of logging and analysis. Unknown functionality exists for a number of reasons. Design errors, project deadlines, production shortcuts, after-market

2 The "Snooper", from General Software, Redmond, WA. ($350 US)

modifications, upgrades, system integration oversights and the like, all contribute to creating a rich exploitation environment. Critical tasks once assigned to humans are now given to information systems without the inherent oversight capability to detect anomalous behavior that can be detected by humans.

4 Overview of the OSI Model

4.1 Background

The Open Systems Interconnection (OSI) model was designed to be just that; an open system. The design features are not proprietary, meaning that complete details of the system are available and can be used freely. The OSI model is documented fully and provides hooks that can be used to develop programs, or to "enhance" the system. Source code for the various layers is available from a number of locations or vendors, and freely across the Internet. The OSI model is derived from a model set forth by the International Standards Organization, ISO[3].

4.2 Why is the OSI Model is used?

The OSI model represents an idealized network. The model uses layers to organize the network into well defined, documented, functional modules. In the layered network, each layer provides specific functionality or services to the adjacent layer. In reality, the actual design of the network may differ from the idealized model, and from other networks designed using the same model.

4.3 Characteristics of each layer

The OSI model contains seven functional layers (figure 1). The OSI model is used because it gives logical structure to the network function. Each layer provides a communication service to the layer above it. The implementation details of each layer is hidden, by design, from the other layers. The objective of the OSI network model is to make the operation of the information system independent from the operation of the network. Some network designs do not clearly define all of the OSI layers. Such designs may integrate the functionality of two or more layers, or ignore some layers completely. Regardless of the network design, specific functionality of the network can be

3 Complete reference material on TCP/IP is available on-line. See References.

correlated to the OSI model. The network model for Windows NT is shown in figure 2.

We will discuss each layer of the OSI model beginning with the physical layer, and present one or more examples of hidden communication possibilities for each layer. Let us again stress that these are only examples. Many more proprietary examples are known to exist, and we hope that your imagination will be further stimulated to produce an even larger set of examples that can be included in the toolkit.

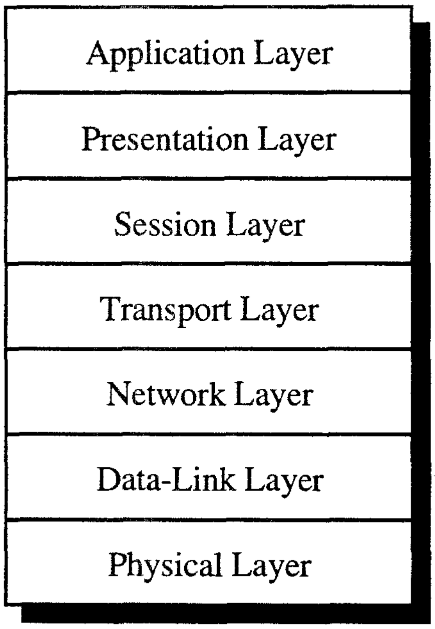

Figure 1: The OSI idealized network model

The modularized design of the OSI model makes discovery and exploitation of hidden communication channels an easier task than would be possible by attempting to analyze or exploit the network design as a whole. For example, hidden data may pass from one layer to an adjacent layer, but not to layers much higher or lower in the stack.

Although the physical communication between hosts occurs only at the lowest, physical layer, virtual connections exist at each layer between hosts. This gives the appearance of connection between corresponding layers between hosts. When properly implemented, the virtual connections allow hidden communications to exist between hosts.

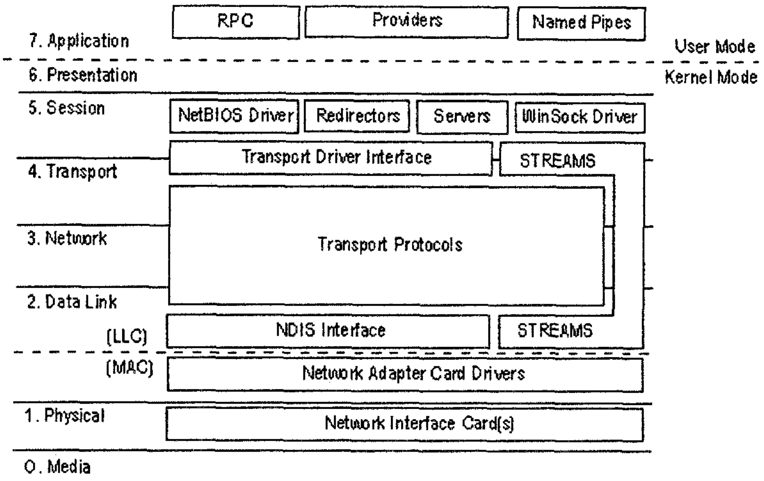

Figure 2: Typical Network Model (Windows NT)

5 Data Hiding in the Physical Layer

5.1 Function of the physical layer

The physical layer transmits data through the network's communications channels. The physical layer contains all the hardware necessary to accomplish this task. Network interface cards are the major component of this hardware. Depending on the network, several variants of hardware can exist. The communications method, including control signals and timing are included in the physical layer.

5.2 Serial Communications Port Manipulation

There is a difference between channel capacity and throughput. The baud rate of the serial communications port can be adjusted over a wide range, and its setting generally defines the channel capacity. The throughput is adjusted by a handshaking mechanism that controls data flow. This mechanism is necessary in the event that data cannot be processed as quickly as it is received.

Throughput can be biased using the Clear to Send/Ready to Send (CTS/RTS) signals, and can be adjusted independently from the data rate. It should be noted that even though Alice and Bob may not "own" the serial data being transmitted, they may exercise some latitude of control over the throughput by manipulating the control lines.

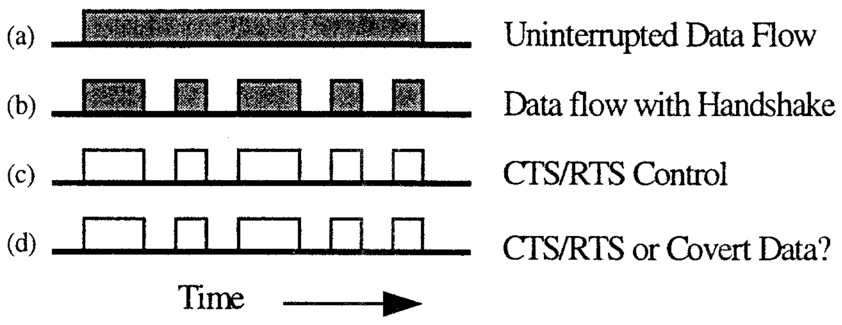

Figure 3: Waveform timing in the serial link.

Analysis of the channel by Walter (figure 3) will reveal the normal data flow with handshaking as illustrated in (a) or (b). Walter must decide if the signal variations in (b) are legitimate handshakes or perhaps a hidden data stream at a lower data rate. On 9600 BPS channels it is quite feasible to toggle the CTS/RTS at 300 BPS thus establishing a second communication channel for sending hidden messages. Interestingly, the CTS/RTS data (c and d) does not need to be ASCII, and thus can be any signaling technique that one would care to devise (including Morse Code). Careful design of control software and use of encryption on the hidden channel (d) make discovery and identification of this hidden channel difficult.

This method is of limited use because it exploits point-to-point connections that are not universal to networks. The greatest utility would be applications where network connections are made over dial-up circuits or dedicated links.

5.3 CSMA/CD Manipulation

The collision detection system in the Ethernet physical layer can also be modified to transmit hidden data. On packet collision, a jamming signal is issued and both parties "back off" a random amount of time and retry transmission. Collision control and random number generation are controlled by software operating in conjunction with the hardware.

This control mechanism can be adjusted (figure 4) so that any packets generated by Carol (another user) will be jammed by Alice. By setting Alice's delay to either zero or maximum, packets retried by Alice will either lead or lag packets regenerated by Carol, thus creating a bit-per-packet hidden channel.

Bob recovers the hidden data by detecting the collision and analyzing the order of arrival of packets from Alice and Carol over the network. In this example, Carol serves merely as a pivot point for data transmitted by Alice. Note that if Carol randomly selects to re-transmit at zero or maximum time, and Alice selects the same delay time, collisions will result until Carol delays a different amount than zero or maximum, thus maintaining the correct sequence. Alice and Bob need not pick on Carol, but may use the next available packet. Bob knows that if he senses a collision and Alice is involved, the next packet he receives will have hidden data for him.

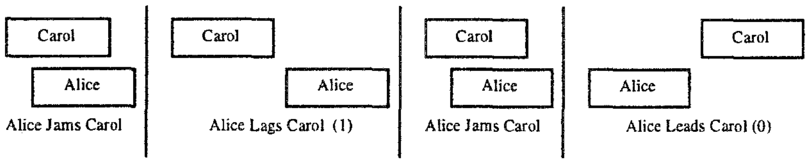

Figure 4: CSMA Collision manipulation

Walter must determine if Alice is deliberately jamming Carol's packets in an effort to transmit hidden data to Bob. Note that this technique does not require a recipient. Alice need not establish a connection with Bob. Bob must be able to monitor packets from both Alice and Carol in order to recover the hidden message. Likewise, Alice need only to be able to monitor packets from Carol and Bob in order to recover the hidden message.

6 Hiding in the Data Link Layer

6.1 Function of the Data Link layer

The data link layer shapes the network data structure. Frames are created containing data to be transmitted over the physical circuits.. Frame headers contain to/from information. Trailers contain error control information (usually a cyclic redundancy check -- CRC). The primary function of the data link layer is to prevent data corruption within the physical layer. Implementation of the data link layer is usually little more than a buffer for storing data received from the network layer and forwarding it to the physical layer.

6.2 Data Frame Manipulation

Unused (slack) portions of the frame can be used to store covert data. A similar technique has been used successfully to store information in the slack space of allocated clusters in magnetic media. This technique is used to hide an auditing system that does not occupy any "real" disk space.

Hidden data is stored in the buffer, beginning at the end of the buffer and working toward the valid data. When the packet is transmitted, the entire buffer is exported, including the covert data. Some minor software modifications may be required. It is necessary to reduce the maximum number of legitimate bytes by two in order to provide at least one byte of hidden storage per packet. The second byte of the two is used as a data separator.

The process is reversed for data received from the physical layer. Data up to the delimiter is sent to the network layer. Data from the delimiter to the end of the frame buffer is sent to the covert process.

7 Hiding Data in the Network Layer

7.1 Function of the Network Layer

The network layer is the internal delivery system. Routing information and error control are added to the data as headers and trailers that define the source and destination for the packet, as well as error control for the data. Most of the variations in hardware and network topology become invisible at the network layer. This layer assures correct delivery and receipt of packets. The network layer may fragment packets at the source and reassemble them at the destination or at an intermediate location (such as a router).

7.2 Hiding data in the Internet Packet

Within the IP packet header there is an 8 bit type-of-service byte (figure 5), of which the two least significant bits are not used in current implementation. The bits can be used to store additional information.

Another subtle implementation of the IP packet can be made. Packets can be optionally time stamped. Thus, by using a coding sequence within the time stamp, additional data may be stored. Packets sent on even time increments represent a logical zero. Packets sent on odd increments represent a logical one. Time stamping is normally used for diagnostic testing and accounting purposes.

Precedence	D	T	R	Unused

Figure 5: IP Type of Service Field

Destruction of hidden will occur if intermediate handling devices (such as routers) are modified to strip out data from these locations.

7.3 Alternative Method

A more subtle method of data hiding exists. The Internet control message protocol (ICMP) works with IP to provide error control. The source-quench command adjusts the sending device data rate when the destination or intermediate node cannot keep up with the data rate. Flow control in this layer is similar to the CTS/RTS implementation in hardware, where the hidden data is in the flow control and the cover data is irrelevant. Note that the hidden channel flow is reversed. Hidden data flow is toward the source, not toward the destination.

8 Hiding Data in the Transport Layer

8.1 Function of the Transport Layer

The transport layer is responsible for delivering data from the network to the correct process within the host computer. The transport layer must interact with multiple programs running on the host, and has system level access to processes. Transmission Control Protocol (TCP) is the Internet implementation of this layer. The transport layer can receive data strings of unlimited length from the upper processes and provides duplex acknowledged, connection oriented, flow controlled transport to remote network stations.

8.2 TCP Packet Manipulation

There are unused data bits in the TCP header (figure 6) similar to those found in the IP header. Six bits are available between the data offset byte and the urgent pointer. These bits are not used in the current implementation. These six bits, combined with the two bits in the IP header provide one byte of hidden data per packet transmitted.

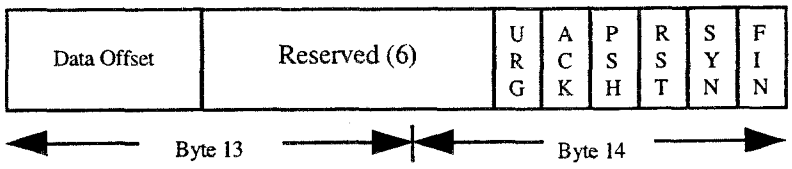

Figure 6: Reserved bytes in TCP Packet Header

Walter can discover the use of this reserved space if he has packet monitoring in place to detect the usage of reserved areas. Some routers may strip out this information, depending on how the router software is implemented. However, this data is hidden from normal analysis.

9 Hiding Data in the Session Layer

9.1 Function of the Session Layer

The session layer is the user's access point to the network. The session layer establishes connections between processes on different hosts, thus the name "session." The session layer is used by Windows® (NETBEUI) network. Users retain full control over their workstation, but accessing the network requires user ID and password. Once on the network, users may have access to restricted resources of other users on the network. including servers that may require further authentication. Functionality at the session layer is achieved through the use of a redirector. The redirector determines if the requested function call can be processed by the local operating system, or processed on the remote system using remote procedure calls (RPC).

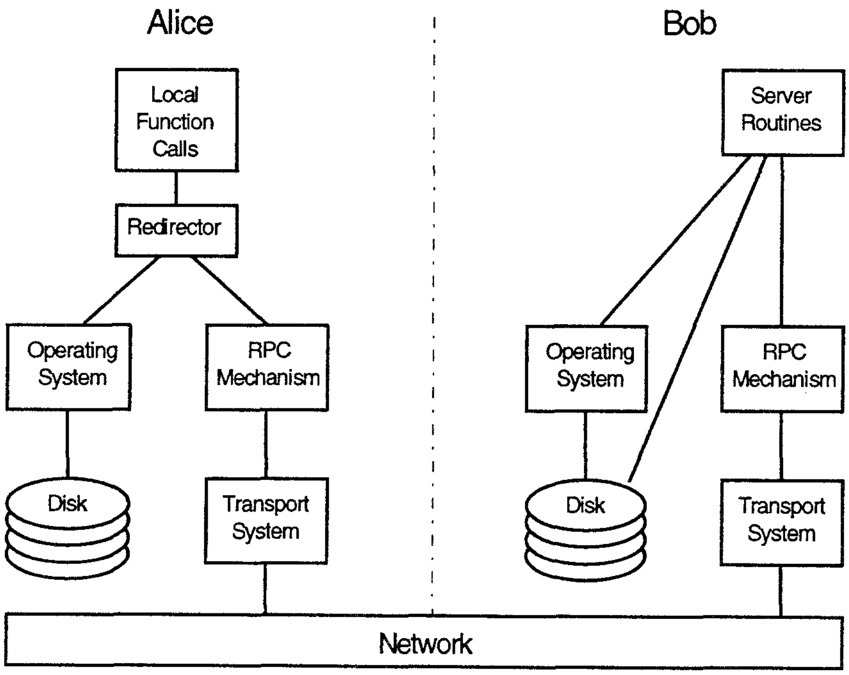

Figure 7: Redirector in Peer-to-Peer LAN.

9.2 Covert Channel Using Session Level Redirectors

The most common use of a redirector (figure 2) is to "mount" remote disks on a local machine. If a user can read a remote disk, then a covert channel can be established. For example, Bob may place two files on his disk that Alice can read remotely. Bob can monitor his local disk activity. Whenever Alice reads the first file, Bob records a logical zero, and whenever she reads the second file, Bob records a logical one. The file contents are irrelevant. Thus, Walter's suspicions are not raised. Figure 7 shows half of the affected hardware; the client portion of Alice, and the server portion of Bob.

A subtle variant of this scheme was demonstrated at Los Alamos in 1982 using a Wang word processor. A Silent 700™ terminal was connected to the seek driver of the Wang's hard disk. Data was successfully transmitted over the covert channel to the Silent 700 at 300 BPS. The technique is similar to the seek timing techniques used to make the older disk drives "walk" across the floor.

10 Hiding Data in the Presentation Layer

10.1 Function of the Presentation Layer

The presentation layer handles the network's interface to devices, such as printers, video displays, and the file system. The presentation layer is what begins to make differences in operating systems transparent. The Windows® API is implemented at this layer. The presentation layer is the proper location for encryption and compression processes, etc.

10.2 Data Embedding

The multimedia components of the presentation layer can act as hidden data transport mechanisms. The statistical properties of audio and video information contain stochastic noise that can be characterized and modified. The stochastic noise can be replaced with pseudo-random noise containing hidden data. Multimedia files are traditionally large data sets, making them attractive hosts for storing hidden data. An example of a data embedding method is given by Sandford, Handel, and Bradley (1995)[4].

Other hiding locations exist. The messaging system in Windows® permits passing data between processes through user-defined messages. Much of the presentation layer facilitates transporting data in the system. Data can be hidden within the fields of system messages that are passed to all active processes.

4 Sandford, et. al., The Data Embedding Method, see references.

11 Hiding Data in the Application Layer

11.1 Function of the Application Layer

The application layer is nearest the user. Users create applications utilizing system resources, including the network. Functional examples include browsers (Mosaic), terminal emulators (telnet), file transfer programs (ftp), e-mail, word processing, and distributed databases.

11.2 Language Manipulation

Many of the classical steganographic approaches can be used at the application level. What was originally done with pencil and paper can be done here. For example, a covert messaging system can be devised (figure 8) using word substitution. Alice selects words for the cover message from two dictionaries that contain words having synonymous equivalents. Words selected from dictionary A will define one hidden message state, while words from dictionary B will define the other hidden message state. Words not found in either dictionary have no covert message content.

When Bob receives the message, he filters the cover message through a decoding macro that searches the two dictionaries using a technique similar to a spelling checker. The decoded output is then checked for errors, and the resulting index is used to lookup the corresponding message.

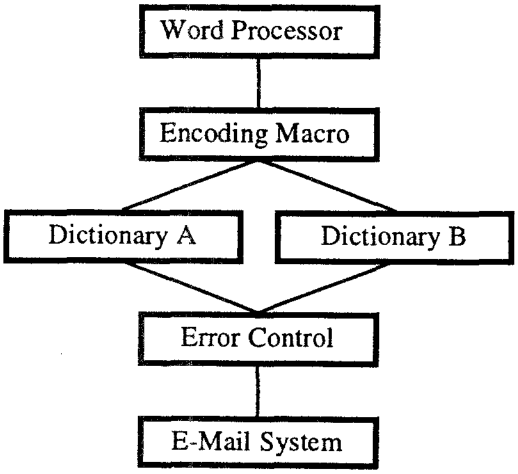

Figure 8: Language substitution steganography.

Assuming that Walter suspects this technique and alters any word in the message, an error detection and correction scheme is implemented to find and correct the word that was changed. Several algorithms are available[5] to detect and correct impulsive (single bit) errors. Walter would need access to a significant amount of information in order to decode the hidden message, including both dictionaries and the abbreviated message table.

Creating a message with n substitutions would yield a reduced message set of 2^n combinations of possible messages. This technique would be most suitable for a table lookup message system. Being generous with error detection, one could set n=20, allowing 2^{20} possible messages, a message table that is well over one million.

Several examples of language steganography exist. The most common example would be to give secondary meaning to the first letter of every word in a sentence. The process requires certain skills in language arts that do not lend themselves readily to automation. The important thing to remember is that Alice and Bob now have access to highly sophisticated information systems. Word processors are now fully programmable, even allowing system API calls. Stenographic techniques once thought to be too complicated to attempt can now be integrated directly into word processing applications on networked platforms.

12 Summary

The OSI model reveals that a variety of locations exist for hiding data. The most common implementation of the OSI model is TCP/IP, a widely accepted and commonly used protocol for large networks. At least one credible example can be given for each layer of the protocol. The examples for each layer, differ in complexity and difficulty of implementation. Alice and Bob can freely select methods for communicating. Walter must monitor and analyze activity on all layers of the protocol stack. Discovery of one hiding place does not necessarily compromise other hiding places. Careful use of error correction and detection in the hidden channel alerts Alice and Bob to the possibility that Walter discovered the channel, and altered the message.

13 Conclusions

Modern information systems provide a very rich environment for hiding data. The flexibility intentionally designed into network communications using the layered model makes identifying and exploiting covert channels easier than for stand-alone systems.

Alice and Bob can communicate covertly, as long as they can communicate overtly. The greater the access Alice and Bob have to system resources, the

5 Digital Communications by Satellite, chapters. 11-15, (see references)

more opportunities exist to send hidden messages. In some circumstances, other network users such as Carol may be unwitting participants.

Walter's problem is complicated because he must design several measurements to analyze message content for hidden data. Transaction logging is meaningful for only as long as someone is willing to analyze and interpret the logs. Some automation is possible, but existing automated systems are based on heuristic detection of anomalous events. Hidden communication is possible because the transactions are ordinary, and raise no alarm for such detectors.

The vulnerability of an information system is more a function of Alice and Bob's skill to discover and implement unknown or ignored functionality within the system architecture. The security of the information system is more a function of the skills Walter can apply to detect subtle anomalies in communication between Alice and Bob.

Redesigning the system to foil Alice and Bob can and will, in many cases, create other opportunities for exploitation.

References

Schneier, Bruce. *Applied Cryptography* Second Ed., J Wiley & Sons, 1996

Jamsa, et al. *Internet Programming,* Jamsa Press, 1995

Novell, Inc. *Networking Technologies,* Novell Press, 1992

Connally, D.R., et. al. *Windows 3.1 Connectivity Secrets,* IDG Books, 1994

Bhargava, V.K. et. al. *Digital Communications by Satellite,* J Wiley & Sons, 1981

Sandford, M.T., Handel, T.G., and Bradley, J.N., *The Data Embedding Method, proc. SPIE Photonics East, 22-24 Oct., 1995.* Available electronically in PostScript format from:
http://info-server.lanl.gov:52271/usr/u078743/embed1.htm

ftp://nic.ddn.mil/rfc *Complete Internet Specification Archive*
rfc-index.txt - Index of documents (234 Kb)
rfc1780.txt - Official Internet Protocol (84 Kb)

Additional Reading

Tomassini, Marco. 1991, *"Programming with Sockets."* The C Users Journal Volume 9 (September 1991): 39-56.

Baker, M. Steven. 1992. *"Network Delivers." Windows Tech Journal*", Volume 1 (August 1992): 22-29.

Volkman, Victor R. 1992. *"Plug into TCP/IP with Windows Sockets."* Windows/DOS Developer's Journal Volume 3 (December
1992): 6-17.

Calbaum, Mike; Porcarro, Frank; Ruegsegger, Mark; Backman, Bruce. 1993. *"Untangling the Windows Sockets API."* Dr. Dobb's Journal Volume 18 (February 1993): 66-71.

Jablon, David P. 1994. *"Windows Sockets."* UNIX Review Volume 12 (October 1994): 37-44.