Network Security Via Reverse Engineering of TCP Code: Vulnerability Analysis and Proposed Solutions*

Biswaroop Guha and Biswanath Mukherjee
Department of Computer Science
University of California
Davis, CA 95616, U.S.A.

{*guha, mukherje*}*@cs.ucdavis.edu*
Tel: +1-916-{785-5863, 752-4826}
Fax: +1-916-752-4767
Corresponding Author: Biswanath Mukherjee

November 7, 1995

Abstract

The Transmission Control Protocol/Internet Protocol (TCP/IP) [1] suite is a very widely used technique that is employed to interconnect computing facilities in modern network environments. However, there exist several security vulnerabilities in the TCP specification and additional weaknesses in a number of widely-available implementations of TCP. These vulnerabilities may enable an intruder to "attack" TCP-based systems, enabling him/her to "hijack" a TCP connection or cause denial of service to legitimate users. We analyze TCP code via a "reverse engineering" technique called "slicing" to identify several of these vulnerabilities, especially those that are related to the TCP state-transition diagram. We discuss many of the flaws present in the TCP implementation of many widely used operating systems, such as SUNOS 4.1.3, SVR4, and ULTRIX 4.3. We describe the corresponding TCP attack "signatures" (including the well-known 1994 Christmas Day Mitnick Attack) and provide recommendations to improve the security state of a TCP-based system, e.g., incorporation of a "timer escape route" from every TCP state.

Keywords and Phrases: Network Security, TCP, IP, Reverse Engineering, Slicing, Vulnerability Analysis, State Transitions, Timer Escape Route.

*This work has been supported by the Advanced Research Projects Agency (ARPA) under Contract No. DOD/DABT63-93-C-0045. A short, summarized version of this paper will appear in the Proceedings of the IEEE Infocom '96 Conference.

1 Introduction

Internetworking is an approach that allows dissimilar computers on dissimilar networks to communicate with one another in a seamless fashion by hiding the details of the underlying network hardware. The most widely used form of internetworking is provided by the **Transmission Control Protocol/Internet Protocol** (TCP/IP) suite.

There are some inherent security problems in the TCP/IP suite [7] which makes the situation conducive to intruders. TCP sequence number prediction, IP address spoofing [9], misuse of IP's source routing principle, use of Internet Control Message Protocol (ICMP) messages for denial of service, etc. are some methods to exploit the network's vulnerabilities. Considering the fact that most important application programs such as Simple Mail Transfer Protocol (SMTP), telnet, r-commands (rlogin, rsh, etc), File Transfer Protocol (FTP), etc. have TCP as their transport layer, security flaws in TCP can prove to be very hazardous for the network.

The objectives of this paper are to identify and analyze the vulnerabilities of TCP/IP and to develop security enhancements to overcome these flaws. Our work is based on analyzing the state-transition diagram of TCP and determining the security relevance of some of the *"improperly-defined"* transitions between different states in the state-transition diagram of many widely used TCP code implementations. Also, we determine the importance of timers in different states and security problems associated with them if a state does not have the necessary *timer-backup* or *escape route*.

We analyze the TCP state-transition diagram using a "reverse engineering" technique called **slicing** [13]. Program slicing is an abstraction mechanism in which code that might influence the value of a given variable at a location is extracted from the full source code of the program. We employ slicing to filter out the relevant state-transition information from the TCP source code. In particular, using the slicing techniques discussed later in the paper, a 1700 line C file implementation of state-transitions in TCP along with all the header file definitions has been reduced to a very small and manageable size of approximately 180 lines of sliced code, which contains only the relevant state-transition information. This process aids in abandoning unnecessary information and simplifying the code. In this process, we have been able to locate extraneous state-transitions present in some implementations of TCP. In other words, using the method of slicing, we have determined the

presence of several spurious state-transitions in a number of TCP implementations such as SUNOS 4.1.3, SVR4, and ULTRIX 4.3; these transitions are not defined in the TCP protocol specification. Using our approach, we can identify various sequences of packets in the network which can be potentially hazardous to the security state of the system. These "attack signatures" represent TCP vulnerabilities, which can possibly be exploited by an intruder. Any "network-sniffer" will be able to determine these signatures and inform the system's security administrator of the intrusion. We also provide several recommendations to enhance the security state of a TCP-based system.

The paper is organized as follows. Section 2 provides an overview of TCP with special emphasis on its state-transition diagram. Section 3 discusses different scenarios having security relevance to the TCP/IP suite. Section 4 discusses our slicing approach to identify extraneous state-transitions in TCP's state-transition diagram. Section 5 provides information on our analytic approach, and the results we have obtained after employing slicing techniques on the TCP source code. Section 6 discusses the attack "signatures", the test-bed which we have developed to test the vulnerabilities of TCP code in various implementations, and various recommendations to enhance the security state of a system. Section 7 concludes the paper including future research directions.

2 Basics of TCP/IP

2.1 Networking with TCP/IP

Network protocols employ a structured and layered approach, with each layer performing a separate function. This approach helps in developing individual layers without modifying other adjacent layers. Networking using the TCP/IP suite can be viewed as a combination of four layers as shown in Fig. 1. Note the correspondence between these layers and those of the International Standard Organization (ISO) seven-layer model: Layers 1 and 2 of the ISO model are combined into the lowest layer of the model in Fig. 1, while ISO Layers 5-7 are merged into the top-most layer in Fig. 1.

The responsibilities of the different layers of the model in Fig. 1 are well-defined.

- The lowest layer, the **data-link layer**,contains the network interface layer, connecting the system with the physical media. It includes device drivers in the operating system and network

interface cards connected to the cable. Data-link layers of different systems exchange *data packets*.

- The next layer is the **internet layer** or the **network layer**. It assists with the movement (routing) of packets in the network. **Internet Protocol** (IP) provides a best-effort, connectionless, and unreliable packet delivery service for the higher layer.
- User processes interact with the network layer through the **transport layer**. The **Transmission Control Protocol** (TCP) is the most common transport layer used in modern networking environments. TCP provides reliable data transfer between different application processes over the network. TCP provides flow control and congestion control as well.
- The **Application layer** handles the details of a particular application. This layer interacts with the user, gets data from the user, and sends the buffered data to the transport layer. At the same time, this layer gets data from transport layer and conveys it to the corresponding application. The application layer shields the user from the details of the transport layer.

When data is sent from the application layer down to the machine hardware, it moves through the different layers and each lower layer adds a header to the data it receives from the previous upper layer. This process of **encapsulation** enables a layer to easily interpret and parse the data that it receives from a lower layer and it has to pass on to the upper layer. Fig. 2 [3] illustrates the encapsulation process that occurs in the TCP/IP suite, assuming an Ethernet physical network.

2.2 Transport Layer

Among all of the transport layers, TCP is the most popular. Below, we examine the details of the header format of TCP along with the TCP state-transition diagram and TCP timers.

2.2.1 TCP Header

The size of the TCP header is 20 bytes, without counting its options, as we observe in Fig. 3 [3]. Each TCP segment contains the *source* and *destination port number* to identify the sending and receiving application programs, respectively. The *sequence number* is essential to maintain the bytes

of data from the sender to the receiver in proper order. By communicating the sequence number and the corresponding *acknowledgment number*, the sender and the receiver can determine lost or retransmitted data in the connection. There are six *flag* bits in the TCP header, namely URG, ACK, PSH, RST, SYN and FIN. At any given time, one or more of these flag bits can be set.

TCP provides flow control by advertising the *window size*. The *checksum* covers TCP header and TCP data and assists in determining any error in transmission of TCP header or data. TCP's urgent mode is a method for the sender to transmit emergency/urgent data. The *urgent pointer* is valid only if the URG flag is set in the header. It helps to locate the sequence number of the last byte of urgent data. There is an optional *options* field as well, taking care of vendor specific information.

2.2.2 TCP State-Transition Diagram

Initiation, establishment, and termination of a connection is governed by the TCP state-transition diagram, which consists of well-defined states and transition arcs between these states (see Fig. 4) [4].

2.2.3 TCP Timers

The TCP state-transition diagram is very closely associated with **timers**. There are various timers associated with connection establishment or termination, flow control, and retransmission of data.

- A *connection-establishment timer* is set when the SYN packet is sent during the connection-establishment phase. If a response is not received within 75 seconds (in most TCP implementations), the connection establishment is aborted.
- A *FIN_WAIT_2 timer* is set to 10 minutes when a connection moves from the FIN_WAIT_1 state to the FIN_WAIT_2[1] state [4]. If the connection does not receive a TCP packet with the FIN bit set within the stipulated time, the timer expires and is set to 75 seconds. If no FIN packet arrives within this time, the connection is dropped.

[1]These are some of the states that TCP uses to successfully terminate a connection.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.