Common Security Attacks on a TCP/IP Environment

S. A. Paschos
Department of Computer Science
University of Ioannina
Greece

F. N. Afrati
Department of Electrical and Computer Engineering
National Technical University of Athens
Greece

Abstract

The TCP/IP protocol suite are the most widely used communication protocols. Although their implementation begun at late 1960s, the tradeoffs between security and performance are not yet well understood. It is evident from the known severe attacks that has undergone, that encryption might be necessary on special messages (DNS messages, routing tables exchange) or in the IP level. In this paper, we focus on the attacks that become feasible because of the nature of the TCP/IP protocols. There is not yet a consent in the Internet community on how to make TCP/IP more secure without a major downrating on its performance. Finally we investigate, in detail, a recent attack, namely, *IP spoofing*, which takes advantage of the finer mechanisms of TCP/IP.

1 Introduction

These days, more and more computers—personal and workstations—get connected via the Internet. Since the Internet was born to fulfill the need of scientists to share information and resources, at the beginning it did not take into severe consideration any security issues. But its explosive growth during the last decade, left back the years of innocence.

Today, the use of Internet has changed. Although the new area of *electronic commerce* demands more and more secure transactions, there are still many problems in the lower layers that require special care ([Bellovin, 1996]). A secure exchange of information needs not only secure peers but also a secure communication channel.

In this paper we focus on the security issues raised by the nature and the implementation of the current version of TCP/IP protocols. In the next section, we describe briefly the TCP/IP protocol suite, for self-containment reasons. In section 3, we refer to the common security attacks known to be related to the TCP/IP functions; we point out that active attacks are the ones which need good knowledge of the TCP/IP finer structure. In section 4, we investigate, in detail, a recent active attack known as *IP spoofing*. This is the situation where a packet is transmitting in an IP network containing in its IP header an address other than that of the sending host.

We will describe how this attack can be successful and point out to weaknesses of the TCP/IP protocols that enable this attack.

2 TCP/IP in brief

In this section we give a brief presentation of the TCP/IP protocol suite, in order to help the reader understand the issues presented and discussed in this paper. More complete information can be found in [Stevens, 1994, Comer, 1995].

The aim of the TCP/IP protocol suite is to allow computers of different hardware, running different operating systems to communicate with each other. Networking protocols are developed in *layers*, with each layer responsible for a different facet of the communication. The TCP/IP protocol suite is a combination of different protocols at various layers. TCP/IP is considered a 4-layer system (figure 1).

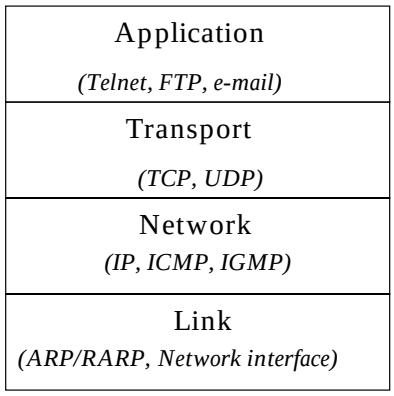

Figure 1: TCP/IP protocol suite

- The *link* layer, also called *host-to-network* or *network interface* layer. It includes the device driver in the operating system and, the network interface card (NIC) in the computer; it handles the details of the physical interfacing via the transmission medium.

- The *network* layer deals with the movement of packets around the network. For example, IP (Internet Protocol) [Postel, 1981a] deals with packet routing. IP provides an unreliable connectionless service known as *datagram delivery service*. The word *unreliable* implies that IP does not guarantee that a datagram arrives successfully to its destination, i.e., IP provides a best effort service. When a network failure occurs, IP discard the datagram and try to notify the source by sending an ICMP (Internet Control Message Protocol) [Postel, 1981b] error message. IP also specifies a uniform addressing scheme known as IP addresses, or Internet addresses. Every NIC on a TCP/IP network must have a unique IP address. The word *connectionless* implies that IP does not maintain state information about successive datagrams. That practically means that datagrams can get delivered out of order.

- The *transport* layer provides an end-to-end data flow. Two protocols—with significant differences—provide transport services in the TCP/IP protocol suite: TCP (Transmission Control Protocol) [Postel, 1981c] and UDP (User Datagram Protocol) [Postel, 1980].

  TCP is a very elegant protocol which, although it relies on the unreliable delivery service of IP, provides a reliable, connection-oriented, byte stream service to the application layer. TCP provides reliability by using a retransmission mechanism in order to get the chunks of data that lost by a hardware failure or discarded by IP. This mechanism relies on a positive acknowledging scheme, i.e., TCP acknowledges every portion of data that receives. The lost of data is detected by the expiration of a proper counter that TCP maintains. TCP also maintains a checksum on its control information (header) and data, which guarantees the integrity of the transmitting information. Since the IP datagrams sometimes arrive out of order, TCP provides a re-sequencing mechanism in order to pass the received data in the correct order to the application. The term connection-oriented denotes that TCP must establish a connection between the two peer computers before they can exchange data.

  UDP provides a primitive datagram-oriented transport service. It does not use acknowledgments, neither examines the integrity of data. Applications that use the services of UDP to transfer the data, must provide the proper reliability by their own.

- The *application* layer handles the details of the particular application. Most networking applications are written as two distinct programs, the *client* and the *server*.

3 Common Attacks

The attacks are usually divided into two categories:

- The *passive* attacks, where the attacker does not interact with the victim system, but rather collects information about it. For example, he/she monitors its transmissions trying to collect usernames and passwords.

- The *active* attacks, where the attacker interacts with the victim system either by modifying the data streams the system receives, or by submitting false data streams.

The final goal of an attack is, for the attacker to gain access to a system as an ordinary or a privileged user (superuser).

3.1 Passive attacks

The most used method for a passive attack is *sniffing*. In this case, the NIC captures data not addressed to the machine in which it resides; the NIC is then in *promiscuous* mode. This method is easily implemented if a lot of computers share the same communication channel (e.g. Ethernet).

Briefly, the intruder must take some actions in order to perform this task:

1. Gain access to a computer (even as a regular user), and then, monitor the traffic. This can be accomplished by guessing pairs of usernames and passwords. This technique is thoroughly examined (e.g. [Klein]) in the past years.

2. Become superuser. This can be achieved by exploiting either the security holes of the Operating System, or a weak security policy (e.g., easy to predict superuser password).

3. Execute a *packet sniffer*. At that stage, since in a normal networking environment the critical information (usernames and passwords) are transmitted in clear text, the intruder can also compromise all the machines on the local network.

We shall not focus on passive attacks here; they do not primarily make use of the deficiencies of the networking protocols they are rather well studied in the past ([Farmer and Vanema, 1993, de Vivo *et al.*, 1998]) and we already know of well behaving techniques to prevent them.

3.2 Active attacks

Although a possible active attack was already described in [Morris, 1985], the first well known attack that actually took place in the Internet was the notorious *Internet worm* [Spafford, 1988, Spafford, 1991]. Moreover, the Internet worm did not use the particular weakness of TCP/IP protocols pointed out by Morris.

The Internet worm was a program which, whenever detected, was executed on one or more machines in the Internet. The worm collected host and user information and then used that information to establish network connections and break into other machines, using flaws present in those machines' software (mainly the `sendmail` mail agent). After breaking in, the program would replicate itself and the replica would try to infect other systems in the same way. Although it seems that the weakness of the utilities which the worm exploited in order to perform its task have been corrected, the worm incident revealed an attack method and we do not know whether this method would not apply to newer versions of these or other system utilities.

Another attack method is presented in detail in [Bellovin, 1992] and [Cheswick, 1992]; this one is interactive (i.e., the cracker tries to break-in by giving commands in real time) as compared to the *Internet worm*; measures taken to prevent the cracker's success, are also presented. The defence against the cracker in [Bellovin, 1992] was (i) the use of extended logging of cracker's activity, and (ii) some software (packet sucker) which used counterintelligence strategies trying to find out the source of the attack. On the other hand, in [Cheswick, 1992] the method was to use a dedicated sacrificial machine in order to fool the cracker by giving him the impression that he gained access in his target.

In the rest of this text we will focus on a special active attack named *IP spoofing*. We will describe how this attack can be successful and point out to weaknesses of the TCP/IP protocol that enable this attack.

4 IP spoofing

IP spoofing is the situation where a packet is transmitting in an IP network containing in its IP header an address other than that of the sending host. IP spoofing try to make use of the concept of *trusted host*.

When a host trust another host, any user who has the same username on both machines can execute the r-commands (rsh, rlogin, rcp) from the trusted host to the trusting one, without giving a password. The above concept is realized in UNIX both by a global file (`hosts.equiv`)

and by files which reside in the home directory of a user (`.rhosts`). The second file can be a severe security hole, since any user on a machine can create in his home directory such a file, giving permissions to anybody he wants to connect to the specific machine without giving any credentials.[1]

IP spoofing is a severe attack which in general is the first step for a number of other attacks. Under IP spoofing attack, the attacking host masquerades as a trusted host, giving thus the attacker the possibility to execute remote commands on a machine. Moreover it makes almost impossible to track the real source of the attack. The false address used by the attacking host may be unassigned or may belong to another host. In the latter case, the attacker must perform some additional actions, which we will present in the following.

If a spoofing attack is originating within a network, it is difficult to protect the other hosts of the same physical network. In this situation only good security policies (e.g., deactivation of r-commands) can prevent the success of this attack. When the attack comes from another network the activation of packet filter rules on the border router can prevent the attack, by paying probably a performance penalty.

In order to describe in detail the IP spoofing attack, we need to recall some features of the finer structure of the TCP protocol.

The establishment of a TCP connection requires the following steps:

1. The source machine (also called *client*) send to the destination machine (also called *server*) a packet without data, and:
   - fills the appropriate field in TCP header (fig. 2) with a 32-bit number called *initial sequence number* (ISN); this number is the starting point after which every byte sent across this connection will be numbered. Thus the first byte of data will have sequence number ISN + 1
   - turns on the SYN flag
2. The destination machine returns back a packet, also without data containing the following information in its header:
   - the SYN flag is turned on and the server's initial number is recorded
   - the ACK flag is turned on and a 32-bit *acknowledge number* is set to the client's ISN plus one (the serial number of the first byte expected to be received)
3. The client acknowledges the server's SYN packet by turning on the ACK flag and recording the server's ISN plus one to the acknowledge number

This sequence of messages (shown in fig. 3), which is mandatory in order to establish a TCP connection, is called the *three-way handshake*.

Other critical fields in the TCP header are the two 16-bit fields namely *source port number* and *destination port number*. The 4-tuple consisting of the source port number, source IP address, destination port number and destination IP address,[2] specify the two end points, that uniquely identify each TCP connection.

[1] The special pattern `+ +` in the `.rhosts` file means that everyone from everywhere can connect to that account without giving the proper password.

[2] Both source and destination IP addresses are located in the IP header

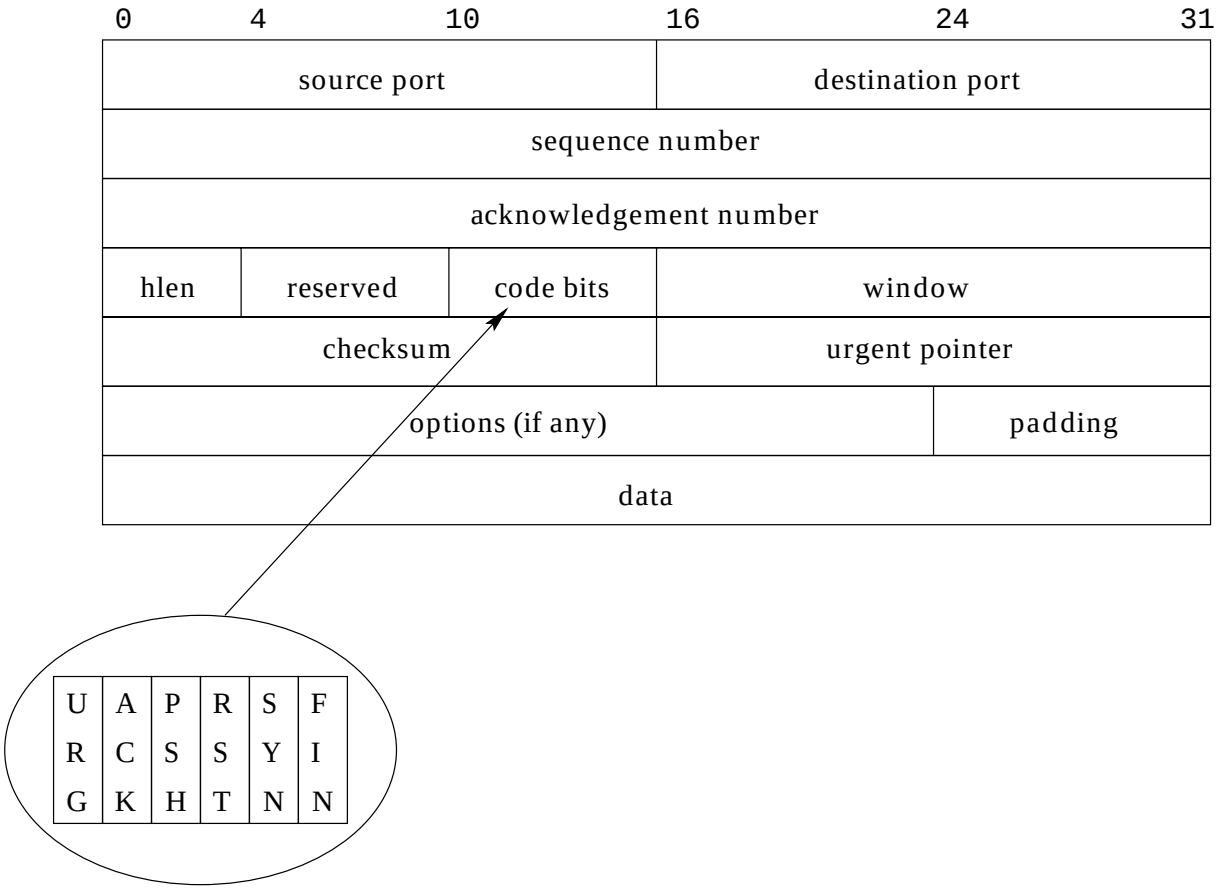

Figure 2: The TCP segment

Now, we can go in detail into the description of the IP spoofing attack: In order a host to masquerade as an existing host (victim), it must somehow interfere to the three-way handshake. A technique to achieve that, was presented by R. Morris in [Morris, 1985], but it received little attention. The procedure consists of two steps: first, the intruder must predict the server's ISN and second, he must prevent the communication between the server and the client (thus, it will be easy for the intruder to masquerade his machine as a legitimate client).

4.1 ISN prediction

This step was first presented in [Morris, 1985] and generalized in [Bellovin, 1989]. The application of this step achieved an actual break-in in 1995, which led to the arrest and imprisonment of Kevin Mitnick.

Supposing for a moment that there is a way for an intruder to predict the server's ISN. This lead to the following scenario:

1. The intruder sends a SYN message and its ISN number to the server, pretending it is a legitimate client (performing IP spoofing).

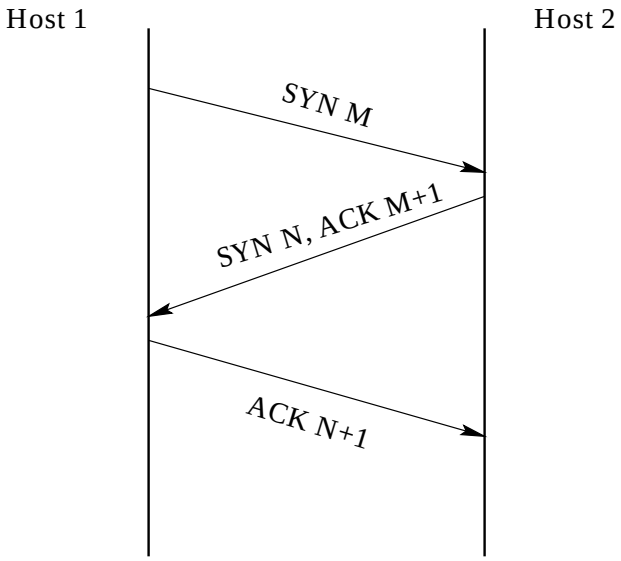

Figure 3: The establishment of a TCP connection

2. The server responds by acknowledging intruder's SYN and ISN, sending in addition a SYN and its ISN to the IP address extracted by the previous message, i.e., to the client, not the intruder.

3. Although the intruder probably does not receive the previous message (if he is not on the same transmission medium, he cannot use a sniffer), having predicted the server's ISN, sends an acknowledgment, performing, thus, the third step of the three-way handshake.

After these three steps, the server establishes the connection and the intruder may execute some malicious instructions.

The prediction of server's ISN that we supposed above can be achieved either by sniffing the traffic, or by calculating the number, presuming that the routine used for this calculation by the specific Operating System is known. The last statement is considered, because not all the implementations follow the TCP specification. Bellovin showed that the formal specification of TCP does not prevent the prediction. For this reason he suggested either to randomize the increment (taking of course into account the size in bits of the ISN), or to use a cryptographic algorithm for ISN generation.

The weak link in the scenario above, is the second step. As Morris pointed out, when the client receives the server's ACK, it finds out that the ISN which the server acknowledges is not valid, so it sends a reset packet trying to reset the connection. If the server will receive this reset message it will close the connection initiated by the intruder. That's why the communication between the client and the server must be prevented.

4.2 Prevention of the communication between client and server

This step can be performed by leading the client to a situation known as *denial of service*. The most known ways to achieve that, are SYN flooding and "smurf"/"fraggle" attacks:

4.2.1 SYN flooding

Before the intruder perform the scenario presented in page 7, attacks the client as follows: It sends SYN messages requesting connection to that port of the client which it will use later to contact the server. Before a SYN flooding attack be observed, the kernel kept a buffer of limited size where it put the incoming connections. The client, when it receives the connection request from the intruder (first message of the three-way handshake), performs the ACK and SYN (second message). The intruder stays silent and does not send to the third message in order to accomplish the initiation of connection. The problem is that in the original TCP implementation the host that accepts the first message (in this case the client) does not use a timeout, so it waits for ever to establish the connection. The intruder can use this feature by sending as many connection requests to overfull the buffer. Now the client cannot accept any message destined to that port, so the message sent by the server during the second step of the spoofing scenario is discarded, and no reset message is initiated. The client is completely deaf in this port.

After a large scale attack using the SYN flooding mechanism, many vendors patched their code to avoid this attack.

4.2.2 The "smurf" and "fraggle" attacks

The "smurf" attack is a recent one and is named after its exploit program. The intruder sends a number of ICMP [Postel, 1981b] echo requests (i.e., it uses the same method used py the `ping` program), filling the source address field of these packets by the IP address of client (thus performing IP spoofing). In addition, it address these requests to a broadcast address, which means that every host on the same physical network will reply by sending back the packet. If the number of hosts connected on the same subnet is large enough, this technique can generate a storm of packets, thus consuming a lot of resources of the spoofed host (client).

The "fraggle" attack is similar to the "smurf"; the difference is that it uses UDP echo packets instead of ICMP echo packets.

In [Huegen, 1998] the enormous amount of traffic generated after a "smurf" attack is calculated for a network consisting of 100 hosts.

5 Conclusions

We considered here problems on a category of security attacks in a TCP/IP environment. It is evident that relying on the sender's IP address for authentication, does not prevent some serious dangers. Although many of these attacks will be impossible under the forthcoming IPv6, until then, there is a real need of a secure transport protocol, permitting signed or encrypted data.

Until then, we can apply good security policies, filter the incoming and outgoing traffic and, if all these fail, analyse the attack we have experienced.

References

[Bellovin, 1989] S. M. Bellovin. Security problems in the TCP/IP protocol suite. *Computer Communication Review*, 19(2):32–48, April 1989.

[Bellovin, 1992] Steven M. Bellovin. There be dragons. In *Proc. UNIX Security Symposium III*, pages 1–16, 1992.

[Bellovin, 1996] Steven M. Bellovin. Problem areas for the IP security protocols. In *Proc. of the Sixth Usenix UNIX Security Symposium*, 1996.

[Cheswick, 1992] Bill Cheswick. An evening with Berferd, in which a cracker is lured, endured, and studied. In *Proc. Winter USENIX Conference*, 1992.

[Comer, 1995] Douglas E. Comer. *Internetworking With TCP/IP. Principles, Protocols, and Architecture*, volume 1. Prentice Hall, Englewood Cliffs, New Jersey, third edition, 1995.

[de Vivo *et al.*, 1998] Marco de Vivo, Gabriela O. de Vivo, and Germinal Isern. Internet security attacks at the basic level. *Operating Systems Reviews*, pages 4–15, April 1998.

[Farmer and Vanema, 1993] Dan Farmer and Wietse Vanema. Improving the security of your site by breaking into it. Posted to USENET, 1993.

[Huegen, 1998] Graig A. Huegen. The latest in denial of service attacks: "smurfing" description and information to minimize effects. `http://www.quadrunner.com/~c-huegen/smurf.cgi`, December 1998.

[Klein] Daniel V. Klein. "Foiling the Cracker": A survey of, and improvements to, password security.

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

[Postel, 1980] J. Postel. RFC 768: User datagram protocol, August 1980.

[Postel, 1981a] J. Postel. RFC 791: Internet Protocol, September 1981.

[Postel, 1981b] J. Postel. RFC 792: Internet Control Message Protocol, September 1981.

[Postel, 1981c] J. Postel. RFC 793: Transmission control protocol, September 1981.

[Spafford, 1988] Eugene H. Spafford. The Internet worm program: An analysis. Technical Report CSD-TR-823, Department of Computer Sciences, Purdue University, 1988.

[Spafford, 1991] Eugene H. Spafford. The Internet worm incident. Technical Report CSD-TR-933, Department of Computer Sciences, Purdue University, 1991.

[Stevens, 1994] W. Richard Stevens. *TCP/IP Illustrated. Vol 1: The Protocols.* Addison-Wesley, 1994.