

Security weaknesses inherent in the design of TCP over IP

By Stephen Fewer

Contents

1 Introduction	2
2 TCP/IP Overview	2
2.1 The Internet Protocol	2
2.2 The Transmission Control Protocol	3
3 Analysis and Countermeasures	3
3.1 Source Routing	3
3.2 Sequence Number Prediction	4
3.3 Connection Spoofing	5
3.4 Connection Hijacking	7
4 Glossary	8
5 References	8

1 Introduction

When the DOD first designed TCP/IP the standard for packet switched communications was defined. The rules for the protocol (outlined in RFC793) are set into every TCP/IP stack whether it's UNIX, Windows or a firmware implementation (e.g. Router). However over the years cracks in its security have been appearing. Legends in the computer underground like Robert Morris (the 'Internet worm') and Kevin Mitnick have prized them out. The problem is that the weaknesses are inherent in the design itself and eradicating them is difficult. Also being inherent in the design means universal cross platform vulnerabilities are present in the very infrastructure of the Internet and most other networks (e.g. LAN's). The fact that TCP over IP is a low-level protocol means that all the higher level ones (e.g. HTTP, Telnet and SMTP) are vulnerable by inheritance (e.g. hijacking a Telnet connection). This paper is intended to introduce the technical concepts behind these vulnerabilities while outlining all if any defenses against such attacks. As an accompaniment to this paper I have designed two tools. The first is a tool to sample the initial sequence numbers of a machines TCP/IP stack. The second, Janus is a non-blind TCP/IP connection spoofer with ARP spoofing. These can be found online at the Harmony Security website.

2 TCP/IP Overview

2.1 The Internet Protocol

The IP is a connectionless, unreliable network protocol. Most protocols are encapsulated inside an IP packet (e.g. TCP, UDP). An IP datagram is made up from a header field and a data segment, usually an encapsulated TCP packet. IP provides the control information needed to route packets around a network. There in no mechanism

employed to ensure reliable delivery. Since it is connectionless every datagram is sent out without regard to the previous packet of the superceding one. It is trivial to modify the control information held in the IP header, thus being able to alter the source address.

2.2 The Transmission Control Protocol

TCP is a connection orientated reliable protocol used to establish a bi-directional data stream between two hosts. This means that for a connection to take place control information must be passed between client and server, implemented in the initial three-way handshake. Reliability is achieved in a number of methods. For the connection sequence numbers provides the reliability. To be able to recover from lost duplicated or out of order data TCP assigns a sequence number to every byte transferred and will expect an acknowledgement from the receiver. The receiver will then use the sequence number to ensure the correct ordering of data before passing it up the stack. For the data reliability is provided by check summing all data inside the TCP packet. TCP packets are encapsulated inside IP packets and have a header field containing control information and a data segment usually containing a high level protocol like HTTP or Telnet. The header field contains the source and destination port of the packet as well as other flags and control information. TCP uses flag to indicate what type of packet it is (e.g. a request for a connection or a

termination request). The six flags are SYN, ACK, PSH, URG, RST and FIN. The two main flags this paper is concerned with are SYN and ACK. The SYN flag is set to instruct the receiver to synchronize sequence numbers. The ACK flag is set when the sender wishes to acknowledge a sequence number.

3 Analysis and Countermeasures

3.1 Source Routing

Source Routing is a method IP employ's to allow packets to chose their own route to a destination and then follow the reverse of that route back. This has its purpose in limited situations such as the automatic route being dead. It's implemented in an optional field in the IP header called 'IP Options'. A list of addresses is included in this field indicating the route that the packet is to follow. There are two types of source routing; lose routing and strict routing. In loose source routing (LSRR) any number of intermediate gateways are allowed to be used to reach the next address, however in strict source routing (SSRR) the next address on the route must be on a directly connected network for delivery to be completed. Abuse of this feature lies in an attacker using source routing to use a secured host (e.g. a firewall or basting host) as an intermediate router and thus bypassing it. A connection can be tunneled through a firewall or around it. It is also possible to forge the source address of a source routed packet and set it equal to the address of an internal host on the target network. Once the packet has been delivered to the destination host any returning packets will be delivered along the reverse route and to the forged internal host address. Packets could go through multiple interfaces (e.g. duel homed hosts) otherwise invisible an outside attacker, i.e. an internal network. To defend

against this type of attack source routing should be disabled on any gateways into the internal networks or set packet filtering or firewall rules to reject any packets with source routing enabled.

3.2 Sequence Number Prediction

Sequence number prediction is the key to spoofing or hijacking TCP connections. It lets a hostile client maintain a fully open TCP connection while never needing to see any responses from the server. To understand better the importance of sequence numbers let us take a look at how a legitimate client would normally establish a connection with a server.

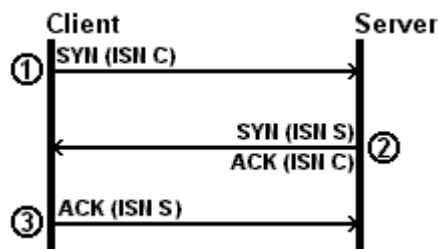


Figure 1

A client wishing to establish a TCP/IP connection with a listening port on a server must first initiate a three-way handshake. This must take place to set control information, such as sequence numbers and window sizes before any data can flow. To commence this handshake a client must choose and send its own Initial Sequence Number (ISN) to the server (Action 1 in Figure 1). The SYN flag of the TCP packet is set and the sequence number contains the clients ISN. Upon receiving this

packet the server knows a client wishes to connect. If the server wishes to allow the connection it will acknowledge the clients ISN, the ACK flag is set and send its own ISN, the SYN flag is set (Action 2 in Figure 1); otherwise a RST packet is sent closing down the half open connection. The client will receive the servers ISN and send an ACK packet acknowledges it (Action 3 in Figure 1). The handshake is complete and data transmission may now take place. Reviewing this we can determine that for data transmission to take place the client must only know the ISN of the server. The vulnerability lies in predicting this number and thus being able to spoof a fully open TCP connection to a server masquerading as a legitimate client (e.g. a trusted host).

To predict the servers ISN we must first know how it is generated. According to the TCP specifications the ISN will be a 'random' 32bit number which will be incremented by a fixed value every second. The specifications say to increment the ISN by 128,000 every second but different implementations vary. The ISN is also meant to be incremented by 64,000 every time a new connection occurs. However a change in the increment value is meaningless, as it's the repetition that is critical and open to abuse. The idea not to use truly random ISN is that if a connection arrived there would be no certainties that the sequence numbers would be different from a previous incarnation (e.g. if data from a previous connection stuck in a routing loop gets freed and sent back into the new version of its previous connection it would mess it up). Thus for an attacker to predict the next ISN of a host he will need to know three things, the value the ISN gets incremented by every second (INC_VAL), the last ISN used and the Round Trip Time (RTT). To get the current ISN the attacker can send a SYN packet to a listening TCP port and read

the response (step two in the three way handshake) this value can then be known as PREVIOUS_ISN. The attacker can presume that the INC_VAL is 128,000 as laid down by the specifications, but if it is different it can be found out by sampling two ISN's, getting their difference and dividing it by half the Round Trip Time as shown in this pseudo formula:

$$\text{INC_VAL} = (\text{SECOND_ISN} - \text{FIRST_ISN}) / (\text{RTT} / 2)$$

To then predict the next ISN the attacker could use the following pseudo formula:

$$\begin{aligned} \text{ISN_DISPLACEMENT} &= \text{INC_VAL} * (\\ &\quad \text{ROUND_TRIP_TIME} / 2) \\ \text{NEXT_ISN} &= \text{PREVIOUS_ISN} + \\ &\quad \text{ISN_DISPLACEMENT} \end{aligned}$$

This will work providing no new connections were received by the server prior to sampling the PREVIOUS_ISN as then it will be off by 64,000. The use of predicting the ISN will be shown in the next section called "Connection spoofing".

Prevention from guessing the NEXT_ISN lies in randomizing the increment. This goes against the initial reasons not to do so. However, the chances of a lost packet coming from an old connection of a current client with the same sequence number (or at least one that falls within the TCP Window of the server) actually arriving are very minute. Therefore making the ISN a truly random 32bit number will defeat all sequence number prediction attacks. To do this some

form of cryptographic hash should be employed (these are non-reversible due to mathematical one way functions being utilized). However the issue of lost packets must still be considered, S. Bellovin has developed a method which allows truly random ISN's to be produced while never bearing any relation to previous incarnations. The randomizing involves partitioning the sequence number space. Every connection will have its own separate sequence number space therefore avoiding old packets fouling things up. Sequence numbers will still be incremented as before but there will be no relationship between the numbers. A pseudo formula for this is:

$$\text{ISN} = \text{M} + \text{F}(\text{boot time}, \text{random number})$$

This should work if M is a 4 microsecond timer (allowing a separate sequence number space for each connection) and F is a cryptographic one way hash. F must not be computable to the attacker or the ISN could still be predicted. Therefore F should be a hash of a random number combined with the machines boot time.

3.3 Connection Spoofing

Connection spoofing is an active attack involving masquerading as a legitimate host and forging a full bi-directional TCP connection with a server. There are two types of spoof attacks, blind and non-blind. In a non-blind attack the illegitimate host is on the same subnet as either the server or client which is being spoofed and can see all responses such as the servers ISN sent to the legitimate client. Therefore there is no need to predict the servers ISN as it can be sniffed off the wire during stage two of the three way handshake (Action 2 in Figure 2). Blind spoof attacks take place through separate interfaces (e.g. across the Internet) and the attacker cannot see any responses that the

server makes to the legitimate client, therefore the servers ISN must be predicted.

For this attack to succeed the legitimate client (depicted as CT in Figure 2) must be rendered disabled so as not to reply to any packets from the server. If CT receives a SYN/ACK packet from a connection it did not initiate (Action 2 in Figure 2) it will immediately respond with a reset (RST) packet, tearing down the half open connection. To countermeasure this problem the attacker can employ another vulnerability inherent in the TCP protocol called "SYN flooding". What this means is that the attacker CX can send numerous SYN packets to CT and overflow its backlog queue of pending connection requests. With this backlog limit reached CT will discard all further incoming SYN requests until the pending queue is dealt with. Therefore once CT has been "SYN flooded" it will not respond with a RST packet once it receives the SYN packet from the server (Action 2 in Figure 2).

With the legitimate host out of the way the attacker can move on to the main section of the attack, see Figure 2. Firstly the attacker must predict the ISN that the server will use for this next connection (using the methods dealt with above). Then they must initiate the connection by forging a SYN packet with the source address of the client being spoofed and an arbitrary ISN and send it to the server (Action 1 in Figure 2). The server will respond to this connection request by acknowledging the clients ISN and

sending its own ISN (Action 2 in Figure 2). The legitimate client (CT) will receive this packet but with its backlog queue full of pending requests from the "SYN flood" it will be discarded. The attacker must then acknowledge the servers ISN by sending an ACK packet with the predicted ISN plus one (because it is being acknowledged) see Action 3 in Figure 2.

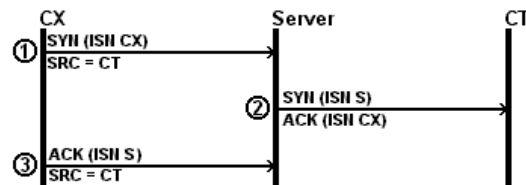


Figure 2

If the attacker is correct in the prediction of the servers ISN the connection will be fully open and data transfer may continue. If the predicted sequence number is less than the expected value the packet is considered a retransmission and is discarded. If the predicted sequence number is greater than the expected value but within the range of the servers TCP receive window it will be presumed to be a future load of data and will be held awaiting the arrival of the missing data. If the predicted sequence number is greater than the expected value and not within the servers TCP receive window it will be dropped and a packet with the expected sequence number will be sent out to the client.

This attack is normally held against services with trusting host relationships like the Berkley r services (e.g. rlogin). These services rely on address bases authentication. As IP addresses are easily forged and fully open TCP connections can be spoofed an illegitimate client masquerading as a trusted client can connect to the rlogin daemon on a server and execute commands.

Countermeasures against this form of attack lie in modifying gateways between networks to reject (and log) packets that claim to come from the internal network but do in fact come from outside. Also any trusting relationship between computers should be restricted to internal hosts and not ones outside the internal network (e.g. the Internet). Initial Sequence Numbers should not be predictable as outlined in the previous section. The vulnerability in TCP to spoof a blind connection is so inherent in its design (all methods in the attack are based around the implementation of TCP and its reaction to certain events) that its only defense lies in third party objects (e.g. routers, gateways and firewalls).

3.4 Connection Hijacking

Connection Hijacking is the method used to take over and control a fully open TCP/IP connection between two hosts. An example would be if a client has a telnet connection to the server an attacker could come along and hijack the connection and begin executing commands on the server. This is particularly worrying as it evades most authentication methods such as password login, Secure Key (S/Key) and One Time Password. To perform this attack the intruder must be on the same subnet as the client or the server. This is so packets can be sniffed off the wire to see the current sequence numbers being used. The attack consists of creating a desynchronized state at each end of the connection so neither host may exchange data any longer. A

desynchronized state evolves when an established connection no longer maintains accurate sequence and acknowledgment numbers for the other host, e.g.

```
SERVER_SEQ != CLIENT_ACK
```

```
CLIENT_SEQ != SERVER_ACK
```

If during a connection this state is achieved the connection will remain stable but data exchange is impossible. This is where the vulnerability lies. To create a desynchronized state between a connection the attacker must monitor the connection and upon an appropriate time send a variable amount of "null data" to both the client and the server masquerading as the other. Null data is bytes of data in the TCP packet, but ones chosen so as not to effect the application using this connection. These packets will use the correct acknowledgement numbers as sniffed off the wire but will change the sequence number to allow for the null data. As the length of the null data was variable in each case the sequence numbers will no longer match the real connections ones thus throwing it into a desynchronized state. With the connection desynchronized data transfer may not take place, as any data packets sent to either the client or the server will not match the acknowledgment number expected. However the attacker knows the valid sequence and acknowledgement numbers and can forge packets that will be accepted by the hosts. The connection has now been hijacked and the attacker may maintain data transfer to both the client and the server. Another variant of this attack is packet insertion where one half of the connection is desynchronized and packets are inserted on the stream by an attacker who then resynchronizes the connection. This is a more advanced attack and can be used with the routing protocols such as ARP and RIP to avoid any conflicts (ACK storms) arising

before it is possible to resynchronize the connection but this is beyond the scope of this paper.

Prevention against this type of attack is limited. Methods like the Kerberos scheme, (which encrypts data on the application layer) prevent any intrusion or modification of the data. The implementation of Intrusion Detection Systems (IDS's) and state full inspection firewalls could combat or detect these attacks as they go live.

4 Glossary

Robert Morris - Famous for the 'internet worm' of 1987 which pioneered such tactics as trusted host abuse and the first example of a 'buffer overflow' to compromise a host. He was caught however less than 24 hours after it was let loose and began to spread uncontrollably taking down the most of the net.

Kevin Mitnick - Famous for getting sent to jail for four years after breaking into Tsutomu Shimomura's computers on Christmas day 1994 while using the first known examples of a blind spoof attack to abuse trusting host relations. Somewhat of an underground legend he has recently been released. The two sides to the story can be found at kevinmitnick.com and takedown.com

Gateway - A machine which offers a network path to another network. For example a router could be a gateway for the internal network out to the Internet.

Dual Homed Host - A host which has two or more interfaces (e.g. network cards) connection to different networks. This host can act as a gateway between.

rlogin - A service developed by Berkley University which allows a command shell to be spawned on a remote machine using only the source address of the client as authentication.

Buffer Overflow - The generic term given to a type of attack where specially crafted machine code is forcefully inserted into a target machines memory and the execution flow of that system is altered to execute this code.

5 References

- Windows IP Source Routing Vulnerability - *Network Associates, Inc.*
- Security Problems in the TCP/IP Protocol Suite - *S.M. Bellovin*
- Sequence Number Attacks - *Rik Farrow*
- Simple Active Attack Against TCP - *Laurent Joncheray*
- IP Spoofing Demystified - *daemon9, route, infinity*
- A short overview of IP spoofing - *Brecht Claerhout*

Harmony Security provides computer and network security research and consultancy. We are focused on delivering new ideas and solutions to this field. Areas of concern include network and system vulnerabilities, malware and cryptography.