draft-shepard-tcp-reassign-port-number-00.txt

Tim Shepard
shep@alum.mit.edu

August 2, 2004

Most TCP connections are not vulnerable

The existence of most TCP connections is unknown to an attacker who cannot see the traffic.

(And if an attacker can see the traffic, then nothing short of IPSEC's ESP or AH will be a sufficient defense.)

Solution

For TCP connections that are today vulnerable, make them unknown to the attacker.

- Don't publish the port numbers involved (in SNMP or wherever).
- Use good random port numbers (16-bits each, or nearly 16-bits each if some values are excluded).

But how then can you establish a connection to a server running on a well-known port and keep the port number unknown from an off-path attacker?

Inspiration: CHAOSNET'S port number assignment

In the CHAOSNET (the first networking protocol suite that I learned) each end got to choose it's own port number.

(The name of the service that you want to connect to was carried in an ASCII string in the first packet sent to the server.)

But TCP uses a well-known port number to identify the target service of a connection request.

Reassign Port Option, on SYN segment

- TCP option, 2 bytes (option code and length)
- carried on the SYN (the first segment, client to server)
- *Server, I would like you to pick a new random port number for your end of this TCP connection (which is to your port 5223).*

Reassign Port Option, on SYN ACK segment

- 4 bytes (the same option code, length, and 2 bytes of port number)
- carried on the SYN ACK segment, and includes the original (well-known) port number
- *OK client, as you requested, for your connection request to my port number 5223 I have reassigned the port number on my end.*
- port number in the TCP header of the SYN ACK is already the new port number

The option is carried on any segment with a SYN bit on, and never carried on any segment with the SYN bit off.

Features

- An attacker now has a search space of over four billion combinations just to match the port numbers correctly, even if the attacker knows both IP addresses involved.
- Conceptually simple, both operationally and analytically.
- No extra round trips.
- Backwards compatible (if the server doesn't implement, the option is ignored).
- Implementation should be straightforward. Ask me in a week or two about patches for NetBSD-current (and perhaps Linux-2.6.8).

How long for an attack to succeed

With two unknown port numbers, and max window scaling, the space an attacker has to search in is $2^{16} \times 2^{16} \times 4$.

At 40-bytes per packet, would take 90 minutes at 1 Gbps to cover the entire space with RSTs. (Expected time would be half that, 45-minutes.)

Issues

- firewalls: Simple stateless firewalls that pass any TCP segment that has the ACK bit on should be unaffected. State tracking firewalls will probably cause the connection to fail. Fixing them should be straightforward (if you can code and recompile).

- NATs: Will need to be upgraded. (Will need to be upgraded anyway to fix the original vulnerability. If they strip the option on the SYN, will not cause failure.)

- What else?