

CERT Advisory 95:01 Response

Document ID: 13604

Please provide your [feedback](#) on this document.

Introduction

This document shows you a simple way to respond to the CERT advisory CA- 95:01, 1/23/95. We've also included an article that summarizes the vulnerabilities cited in the CERT advisory.

Step 1

The defense is simply to set up your Internet firewall router to deny packets from outside your network that claim to have a source address inside your network. Here's a sample configuration for a network with internal source addresses of 131.108.x.x and 198.92.93.x:

```
access-list 101 deny IP 131.108.0.0 0.0.255.255 0.0.0.0 255.255.255.255
access-list 101 deny IP 198.92.93.0 0.0.0.255 0.0.0.0 255.255.255.255
[...the rest of your firewall goes here...]
```

and so on, where access list 101 describes all possible source addresses on your network.

Important: if you use only the two line example described above without any other access-list commands, all traffic will stop on your interface because packets that don't match one of the access list permit/deny lines are implicitly denied.

If you only want source address spoofing protection and nothing else, add the line

```
access-list 101 permit ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
```

to the end of the earlier example, which will allow all remaining traffic to flow to and from the Internet. This is not your best solution, however, since there are many possible attacks other than the IP spoofing fixed here. The CCO information service and various USENET mailing lists have articles on this topic. You can telnet to cco.cisco.com or point your WWW browser at <http://cco.cisco.com> on dial-in (408) 526-8070.

Step 2

Once you have defined an appropriate access list you can apply it to the vulnerable interfaces:

```
interface serial 0
description interface facing the big bad Internet
ip access-group 101 in
```

for a router running 9.21 or later, where your interface serial 0 faces the Internet.

If you don't have 9.21 and your Internet firewall is a two port router, you don't need an upgrade. Just apply access-list 101 as described above to the LAN interface and not the serial interface, like in this example:

```
interface ethernet 0
description LAN port on my Internet router
ip access-group 101
```

With this defense, any packets coming from the Internet that claim to be from your network are thrown out, preventing the style of attack described below.

If you have a multi-port router as your firewall and are running a version prior to 9.21, the access-lists required to prevent spoofing can cause internal connectivity problems. We recommend upgrading to 9.21 or later, or consulting the TAC for assistance. Also, in order to prevent other forms of spoofing attack from outside, all Internet firewalls should have this global command:

```
no ip source-routing
```

The following article appeared on comp.security.misc and sums up the references cited in the CERT advisory concerning the cited vulnerabilities.

Steve Bellovin's Comments on the Nature of the Attack

There's a great deal of confusion about what kind of attack the recent CERT advisory is referring to. Let me try to clear things up.

The specific attack is a sequence number guessing attack, originally described by R.T. Morris in Bell Labs Computer Science Technical Report #117, February 25, 1985. I generalized (and publicized) the attack in my 1989 paper "Security Problems in the TCP/IP Protocol Suite", Computer Communications Review 19:2, April 1989, pp. 32-48. Both his attack and my generalizations are special cases of a more general attack, IP source address spoofing, in which the attacker illegitimately uses a trusted machine's IP address in conjunction with some protocol (such as rsh) that does address-based authentication.

In order to understand the particular case of sequence number guessing, you have to look at the 3-way handshake used in the TCPopen sequence. Suppose client machine A wants to talk to rsh server B. It sends the following message:

```
A->B: SYN, ISSa
```

That is, it sends a packet with the SYN ("synchronize sequence number") bit set and an initial sequence number ISSb.

B replies with

```
B->A: SYN, ISSb, ACK(ISSa)
```

In addition to sending its own initial sequence number, it acknowledges A's. Note that the actual numeric value ISSa must appear in the message.

A concludes the handshake by sending

```
A->B: ACK(ISSb)
```

The initial sequence numbers are intended to be more or less random. More precisely, RFC 793 specifies that the 32-bit counter be incremented by 1 in the low-order position about every 4 microseconds. Instead, Berkeley-derived kernels increment it by 128 every second, and 64 for each new connection. Thus, if you open a connection to a machine, you know to a very high degree of confidence what sequence number it will use for its next connection. And therein lies the attack.

X first opens a real connection to its target B -- say, to the mail port or the TCP echo port. This gives ISSb. It then impersonates A and sends

```
A->B: SYN, ISSx
```

B's response to X's original SYN

```
B->A: SYN, ISSb', ACK(ISSx)
```

goes to the legitimate A, about which more anon. X never sees that message but can still send

```
A->B: ACK(ISSb')
```

using the predicted value for ISSb'. If the guess is right — and usually it will be — B's rsh server thinks it has a legitimate connection with A, when in fact X is sending the packets. X can't see the output from this session, but it can execute commands as more or less any user — and in that case, the game is over and X has won.

There is a minor difficulty here. If A sees B's message, it will realize that B is acknowledging something it never sent, and will send a RST packet in response to tear down the connection. There are a variety of ways to prevent this; the easiest is to wait until the real A is down (possibly as a result of enemy action, of course).

There are several possible defenses. Most obvious is to take advantage of topological knowledge: don't let packets purporting to be from a local machine arrive on an outside interface. That works very well if you only trust local machines. If trust is granted to outside machines (say, via .rhosts files) and if the attacker can identify the patterns of trust (which isn't that difficult), the topological solution won't work. In that case, you have to block all protocols that use TCP and address-based authentication. (UDP is a separate can of worms.)

Best of all, don't use address-based authentication; it's a disaster waiting to happen. The only real solution is cryptographic authentication.

—Steve Bellovin

For further information, see the two papers cited above:

ftp://ftp.research.att.com/dist/internet_security/117.ps.Z

ftp://ftp.research.att.com/dist/internet_security/ipext.ps.Z

All contents are Copyright © 1992—2001 Cisco Systems Inc. All rights reserved. [Important Notices](#) and [Privacy Statement](#).

Updated: Aug 01, 2007

Document ID: 13604
