

Tools for Surviving the Cisco IOS Flaw

Michael J. Martin

This is an emergency broadcast article; please tune in if you have been affected -- or potentially affected -- by the latest Cisco IOS bug (CERT VU#411332). If you're like me, it's safe to say you have already been dealing with the latest Internet security attack. The IOS IPv4 denial-of-service attack was publicly announced last week. Read more about it here.

The vulnerability affects Cisco IOS-based routers and switches running 11.x thru 12.2.x. IOS version 12.3 and a number of 12.1 and 12.2 rebuilds are not affected. You've read the news; now learn how to deal with it. This article will review the facts, take a look at the attack, and discuss the options for protecting your network.

When the news broke, one security engineer said to me in a resigned tone, "It was just a matter of time." He was right. Bugs happen to good software. This is not the first Cisco IOS bug, and it will not be the last. This is the first, however, to have such a broad impact. That said, if you have deployed your network with some thought toward security and have implemented proper filtering, then you may well already be protected against the threat, or at least be able to deal with it easily.

The Attack

The attack is very simple and quite effective. It involves sending x number of crafted packets at a specific router interface. These crafted packets, with protocol type 53 (SWIPE), type 55 (Mobile IP), type 77 (SUNND) and type 103 (Protocol Independent Multicast), force the router into thinking the target interface's input queue is full. Consequently, the router stops processing inbound traffic on the interface. The attack methodology is to perform a denial-of-service attack on all of the router's interfaces, crippling the router as a forwarding device and in some cases making the device inaccessible to manage (that's why you should always have out-of-band access to your routers). The attack functions without attacking the transit gateways; only the target destination interface is affected by the traffic. The full Cisco Security Advisory is available at Cisco.com.

The attack is not detectable unless you are monitoring for it (using network intrusion detection or ACL filter logging) or inspecting packet dumps of the router interface input-buffer queues using `<show buffers input-interface {interface} {packet|dump}>`. We will cover packet identification in a moment.

The actual attack code, called shadowcode, can be downloaded from the Web. One such location is the Neohapsis Archive. The code has been made available to allow individuals to use it to develop detection signatures and test their environments after patches and/or upgrades have been installed, for certification purposes. Law prohibits the use of this code for malicious intent; misuse could leave parties liable for damages and potential prosecution.

How To Identify The Attack

For those who are unable to access the attack code, here are some sample TCPdump and Cisco buffer dump captures for each of the four protocol types. The packets generated by the tool utilize randomly generated routable and non-routable IP addresses. Each packet generated by an "attack run" (four packets per run) has a different non-correlateable source IP address.

These sample dumps are of the Protocol Independent Multicast variety. Notice the protocol in the prot field:

```
Buffer information for Middle buffer at 0x81E920C0
  data_area 0x52AF304, refcount 1, next 0x0, flags 0x200
  linktype 7 (IP), enctype 1 (ARPA), encsize 14, rxttype 1
  if_input 0x81F5B6F0 (Ethernet0/0), if_output 0x0 (None)
  inputtime 0x0, outputtime 0x0, oqnumber 65535
  datagramstart 0x52AF34A, datagramsize 263, maximum size 756
  mac_start 0x52AF34A, addr_start 0x52AF34A, info_start 0x0
```

Tools for Surviving the Cisco IOS Flaw

Michael J. Martin

network_start 0x52AF358, transport_start 0x52AF36C, caller_pc 0x80223384
source: 92.121.40.79, destination: 172.30.71.99, id: 0xEF28, ttl: 1, prot: 103

```
052AF300: AFACEFAD 0A30303A 30303A31 333A2025 /,o-.00:00:13: %
052AF310: 5359532D 352D434F 4E464947 5F493A20 SYS-5-CONFIG_I:
052AF320: 436F6E66 69677572 65642066 726F6D20 Configured from
052AF330: 6D656D6F 72792062 7920636F 0000000C memory by co....
052AF340: 65006365 052AF394 052A0004 DD0C8380 e.ce.*s...*..]...
052AF350: 00306525 6A430800 450000F9 EF280000 .0e%jC..E..yo(..
052AF360: 0167512C 5C79284F AC1E4763 00010203 .gQ,y(O,.Gc....
052AF370: 04050607 08090A0B 0C0D0E0F 10111213 .....
052AF380: 14151617 18191A1B 1C1D1E1F 20212223 ..... !"#
052AF390: 24252627 28292A2B 2C2D2E2F 30313233 $%&'()*+,-./0123
052AF3A0: 34353637 38393A3B 3C3D3E3F 40414243 456789:;<=>?@ABC
052AF3B0: 44454647 48494A4B 4C4D4E4F 50515253 DEFGHIJKLMNOPQRS
052AF3C0: 54555657 58595A5B 5C5D5E5F 60616263 TUVWXYZ[ ]^_`abc
052AF3D0: 64656667 68696A6B 6C6D6E6F 70717273 defghijklmnopqrs
052AF3E0: 74757677 78797A7B 7C7D7E7F 80818283 tuvwxyz{|}~.....
052AF3F0: 84858687 88898A8B 8C8D8E8F 90919293 .....
052AF400: 94959697 98999A9B 9C9D9E9F A0A1A2A3 ..... !"#
052AF410: A4A5A6A7 A8A9AAAB ACADAEAF B0B1B2B3 $%&'()*+,-./0123
052AF420: B4B5B6B7 B8B9BABB BCBD BEBF C0C1C2C3 456789:;<=>?@ABC
052AF430: C4C5C6C7 C8C9CACB CCDCCECF D0D1D2D3 DEFGHIJKLMNOPQRS
052AF440: D4D5D6D7 D8D9DADB DCDDDEDF E0E1E2E3 TUVWXYZ[ ]^_`abc
052AF450: E4000000 00000000 00000000 00000000 d.....
052AF460: 00000000 00000000 00000000 00000000 .....
052AF470: 00000000 00000000 00000000 00000000 .....
052AF480: 00000000 00000000 00000000 00000000 .....
052AF490: 00000000 00000000 00000000 00000000 .....
052AF4A0: 00000000 00000000 00000000 00000000 .....
052AF4B0: 00000000 00000000 00000000 00000000 .....
052AF4C0: 00000000 00000000 00000000 00000000 .....
052AF4D0: 00000000 00000000 00000000 00000000 .....
052AF4E0: 00000000 00000000 00000000 00000000 .....
052AF4F0: 00000000 00000000 00000000 00000000 .....
052AF500: 00000000 00000000 00000000 00000000 .....
052AF510: 00000000 00000000 00000000 00000000 .....
052AF520: 00000000 00000000 00000000 00000000 .....
052AF530: 00000000 00000000 00000000 00000000 .....
052AF540: 00000000 00000000 00000000 00000000 .....
052AF550: 00000000 00000000 00000000 00000000 .....
052AF560: 00000000 00000000 00000000 00000000 .....
052AF570: 00000000 00000000 00000000 00000000 .....
052AF580: 00000000 00000000 00000000 00000000 .....
052AF590: 00000000 00000000 00000000 00000000 .....
052AF5A0: 00000000 00000000 00000000 00000000 .....
052AF5B0: 00000000 00000000 00000000 00000000 .....
052AF5C0: 00000000 00000000 00000000 00000000 .....
052AF5D0: 00000000 00000000 00000000 00000000 .....
052AF5E0: 00000000 00000000 00000000 00000000 .....
052AF5F0: 00000000 00 .....

```

Tools for Surviving the Cisco IOS Flaw

Michael J. Martin

The TCPdump trace has translated the protocol ID into names when possible. Notice that 102 has been translated into pim v0:

```
21:09:57.719431 26.228.189.14 > 172.30.71.99: pim v0 [ttl 1]
0x0000 4500 00c6 abcb 0000 0167 4192 1ae4 bd0e      E.....gA.....
0x0010 ac1e 4763 0001 0203 0405 0607 0809 0a0b      ..Gc.....
0x0020 0c0d 0e0f 1011 1213 1415 1617 1819 1a1b      .....
0x0030 1c1d 1e1f 2021 2223 2425 2627 2829 2a2b      .....!"$%&'()*+
0x0040 2c2d 2e2f 3031 3233 3435 3637 3839 3a3b      ,-. /0123456789:;
0x0050 3c3d 3e3f 4041 4243 4445 4647 4849 4a4b      <=>?@ABCDEFGHIJK
0x0060 4c4d 4e4f 5051 5253 5455 5657 5859 5a5b      LMNOPQRSTUVWXYZ[
0x0070 5c5d 5e5f 6061 6263 6465 6667 6869 6a6b      ]^_`abcdefghijkl
0x0080 6c6d 6e6f 7071 7273 7475 7677 7879 7a7b      lmnopqrstuvwxyz{
0x0090 7c7d 7e7f 8081 8283 8485 8687 8889 8a8b      |}~.....
0x00a0 8c8d 8e8f 9091 9293 9495 9697 9899 9a9b      .....
0x00b0 9c9d 9e9f a0a1 a2a3 a4a5 a6a7 a8a9 aaab      .....
0x00c0 acad aeaf b0b1      .....
```

Here are the SWIPE protocol 53 dumps (Cisco):

```
Buffer information for Middle buffer at 0x81E93DF8
  data_area 0x52B1384, refcount 1, next 0x0, flags 0x200
  linktype 7 (IP), enctype 1 (ARPA), encsize 14, rxtype 1
  if_input 0x81F5B6F0 (Ethernet0/0), if_output 0x0 (None)
  inputtime 0x0, outputtime 0x0, oqnumber 65535
  datagramstart 0x52B13CA, datagramsize 263, maximum size 756
  mac_start 0x52B13CA, addr_start 0x52B13CA, info_start 0x0
  network_start 0x52B13D8, transport_start 0x52B13EC, caller_pc 0x80223384

  source: 98.142.169.75, destination: 172.30.71.99, id: 0xC579, ttl: 1, prot: 53
```

```
052B1380: AFACEFAD 0A2A4D61 72202031 2030303A /,o-.*Mar 1 00:
052B1390: 30303A31 312E3536 323A2025 4C494E4B 00:11.562: %LINK
052B13A0: 2D332D55 50444F57 4E3A2049 6E746572 -3-UPDOWN: Inter
052B13B0: 66616365 20457468 65726E65 00000008 face Etherne....
052B13C0: 2C206368 052B1414 052B0004 DD0C8380 , ch.+...+..]...
052B13D0: 00306525 6A430800 450000F9 C5790000 .0e%jC..E..yEy..
052B13E0: 0135F3FB 628EA94B AC1E4763 00010203 .5s{b.)K,.Gc....
052B13F0: 04050607 08090A0B 0C0D0E0F 10111213 .....
052B1400: 14151617 18191A1B 1C1D1E1F 20212223 ..... !"#
052B1410: 24252627 28292A2B 2C2D2E2F 30313233 $%&'()*+,-./0123
052B1420: 34353637 38393A3B 3C3D3E3F 40414243 456789:;<=>?@ABC
052B1430: 44454647 48494A4B 4C4D4E4F 50515253 DEFGHIJKLMNOPQRS
052B1440: 54555657 58595A5B 5C5D5E5F 60616263 TUVWXYZ[]^_`abc
052B1450: 64656667 68696A6B 6C6D6E6F 70717273 defghijklmnopqrs
052B1460: 74757677 78797A7B 7C7D7E7F 80818283 tuvxyz{|}~.....
052B1470: 84858687 88898A8B 8C8D8E8F 90919293 .....
052B1480: 94959697 98999A9B 9C9D9E9F A0A1A2A3 ..... !"#
052B1490: A4A5A6A7 A8A9AAAB ACADAEAF B0B1B2B3 $%&'()*+,-./0123
052B14A0: B4B5B6B7 B8B9BABB BCBDBEBF C0C1C2C3 456789:;<=>?@ABC
052B14B0: C4C5C6C7 C8C9CACB CCCCCECF D0D1D2D3 DEFGHIJKLMNOPQRS
052B14C0: D4D5D6D7 D8D9DADB DCDDDEDF E0E1E2E3 TUVWXYZ[]^_`abc
052B14D0: E4000000 00000000 00000000 00000000 d.....
052B14E0: 00000000 00000000 00000000 00000000 .....
```

Tools for Surviving the Cisco IOS Flaw

Michael J. Martin

```
052B14F0: 00000000 00000000 00000000 00000000 .....
052B1500: 00000000 00000000 00000000 00000000 .....
052B1510: 00000000 00000000 00000000 00000000 .....
052B1520: 00000000 00000000 00000000 00000000 .....
052B1530: 00000000 00000000 00000000 00000000 .....
052B1540: 00000000 00000000 00000000 00000000 .....
052B1550: 00000000 00000000 00000000 00000000 .....
052B1560: 00000000 00000000 00000000 00000000 .....
052B1570: 00000000 00000000 00000000 00000000 .....
052B1580: 00000000 00000000 00000000 00000000 .....
052B1590: 00000000 00000000 00000000 00000000 .....
052B15A0: 00000000 00000000 00000000 00000000 .....
052B15B0: 00000000 00000000 00000000 00000000 .....
052B15C0: 00000000 00000000 00000000 00000000 .....
052B15D0: 00000000 00000000 00000000 00000000 .....
052B15E0: 00000000 00000000 00000000 00000000 .....
052B15F0: 00000000 00000000 00000000 00000000 .....
052B1600: 00000000 00000000 00000000 00000000 .....
052B1610: 00000000 00000000 00000000 00000000 .....
052B1620: 00000000 00000000 00000000 00000000 .....
052B1630: 00000000 00000000 00000000 00000000 .....
052B1640: 00000000 00000000 00000000 00000000 .....
052B1650: 00000000 00000000 00000000 00000000 .....
052B1660: 00000000 00000000 00000000 00000000 .....
052B1670: 00000000 00 .....

```

TCPdump (SWIPE):

```
21:09:57.718818 32.38.132.58 > 172.30.71.99: ip-proto-53 178 [ttl 1]
0x0000 4500 00c6 1544 0000 0135 0bde 2026 843a      E....D...5...&.:
0x0010 ac1e 4763 0001 0203 0405 0607 0809 0a0b      ..Gc.....
0x0020 0c0d 0e0f 1011 1213 1415 1617 1819 1a1b      .....
0x0030 1c1d 1e1f 2021 2223 2425 2627 2829 2a2b      .....!"$%&'()*+
0x0040 2c2d 2e2f 3031 3233 3435 3637 3839 3a3b      ,-. /0123456789:;
0x0050 3c3d 3e3f 4041 4243 4445 4647 4849 4a4b      <=>?@ABCDEFGHIJK
0x0060 4c4d 4e4f 5051 5253 5455 5657 5859 5a5b      LMNOPQRSTUVWXYZ[
0x0070 5c5d 5e5f 6061 6263 6465 6667 6869 6a6b      ]^_`abcdefghijklmnop
0x0080 6c6d 6e6f 7071 7273 7475 7677 7879 7a7b      lmnopqrstuvwxyz{
0x0090 7c7d 7e7f 8081 8283 8485 8687 8889 8a8b      |}~.....
0x00a0 8c8d 8e8f 9091 9293 9495 9697 9899 9a9b      .....
0x00b0 9c9d 9e9f a0a1 a2a3 a4a5 a6a7 a8a9 aaab      .....
0x00c0 acad aeaf b0b1 .....

```

The IP Mobility protocol 55 dumps (Cisco):

```
Buffer information for Middle buffer at 0x81E943D0
  data_area 0x52B1A04, refcount 1, next 0x0, flags 0x200
  linktype 7 (IP), enctype 1 (ARPA), encsize 14, rxtype 1
  if_input 0x81F5B6F0 (Ethernet0/0), if_output 0x0 (None)
  inputtime 0x0, outputtime 0x0, oqnumber 65535
  datagramstart 0x52B1A4A, datagramsize 263, maximum size 756
  mac_start 0x52B1A4A, addr_start 0x52B1A4A, info_start 0x0
  network_start 0x52B1A58, transport_start 0x52B1A6C, caller_pc 0x80223384

```

Tools for Surviving the Cisco IOS Flaw

Michael J. Martin

source: 70.220.239.66, destination: 172.30.71.99, id: 0x1B79, ttl: 1, prot: 55

```
052B1A00: AFACEFAD 0A30303A 30303A31 323A2025 /,o-.00:00:12: %
052B1A10: 4C494E45 50524F54 4F2D352D 5550444F LINEPROTO-5-UPDO
052B1A20: 574E3A20 4C696E65 2070726F 746F636F WN: Line protoco
052B1A30: 6C206F6E 20496E74 65726661 00000009 l on Interfa....
052B1A40: 74686572 052B1A94 052B0004 DD0C8380 ther.+...+..]...
052B1A50: 00306525 6A430800 450000F9 1B790000 .0e%jC..E..y.y..
052B1A60: 013773B5 46DCEF42 AC1E4763 00010203 .7s5FoB,.Gc....
052B1A70: 04050607 08090A0B 0C0D0E0F 10111213 .....
052B1A80: 14151617 18191A1B 1C1D1E1F 20212223 ..... !"#
052B1A90: 24252627 28292A2B 2C2D2E2F 30313233 $%&'()*+,-./0123
052B1AA0: 34353637 38393A3B 3C3D3E3F 40414243 456789:;<=>?@ABC
052B1AB0: 44454647 48494A4B 4C4D4E4F 50515253 DEFGHIJKLMNOPQRS
052B1AC0: 54555657 58595A5B 5C5D5E5F 60616263 TUVWXYZ[ ]^_`abc
052B1AD0: 64656667 68696A6B 6C6D6E6F 70717273 defghijklmnopqrs
052B1AE0: 74757677 78797A7B 7C7D7E7F 80818283 tuvwxyz{|}~.....
052B1AF0: 84858687 88898A8B 8C8D8E8F 90919293 .....
052B1B00: 94959697 98999A9B 9C9D9E9F A0A1A2A3 ..... !"#
052B1B10: A4A5A6A7 A8A9AAAB ACADAEAF B0B1B2B3 $%&'()*+,-./0123
052B1B20: B4B5B6B7 B8B9BABB BCBDDBEF C0C1C2C3 456789:;<=>?@ABC
052B1B30: C4C5C6C7 C8C9CACB CCDCCECF D0D1D2D3 DEFGHIJKLMNOPQRS
052B1B40: D4D5D6D7 D8D9DADB DCDDDEDF E0E1E2E3 TUVWXYZ[ ]^_`abc
052B1B50: E4000000 00000000 00000000 00000000 d.....
052B1B60: 00000000 00000000 00000000 00000000 .....
052B1B70: 00000000 00000000 00000000 00000000 .....
052B1B80: 00000000 00000000 00000000 00000000 .....
052B1B90: 00000000 00000000 00000000 00000000 .....
052B1BA0: 00000000 00000000 00000000 00000000 .....
052B1BB0: 00000000 00000000 00000000 00000000 .....
052B1BC0: 00000000 00000000 00000000 00000000 .....
052B1BD0: 00000000 00000000 00000000 00000000 .....
052B1BE0: 00000000 00000000 00000000 00000000 .....
052B1BF0: 00000000 00000000 00000000 00000000 .....
052B1C00: 00000000 00000000 00000000 00000000 .....
052B1C10: 00000000 00000000 00000000 00000000 .....
052B1C20: 00000000 00000000 00000000 00000000 .....
052B1C30: 00000000 00000000 00000000 00000000 .....
052B1C40: 00000000 00000000 00000000 00000000 .....
052B1C50: 00000000 00000000 00000000 00000000 .....
052B1C60: 00000000 00000000 00000000 00000000 .....
052B1C70: 00000000 00000000 00000000 00000000 .....
052B1C80: 00000000 00000000 00000000 00000000 .....
052B1C90: 00000000 00000000 00000000 00000000 .....
052B1CA0: 00000000 00000000 00000000 00000000 .....
052B1CB0: 00000000 00000000 00000000 00000000 .....
052B1CC0: 00000000 00000000 00000000 00000000 .....
052B1CD0: 00000000 00000000 00000000 00000000 .....
052B1CE0: 00000000 00000000 00000000 00000000 .....
052B1CF0: 00000000 00 .....

```

TCPdump (IP Mobility):

Tools for Surviving the Cisco IOS Flaw

Michael J. Martin

```
21:09:57.719217 58.202.31.133 > 172.30.71.99: mobile: [] (bad checksum 515) [ttl
1]
0x0000 4500 00c6 f6de 0000 0137 7452 3aca 1f85      E.....7tR:...
0x0010 ac1e 4763 0001 0203 0405 0607 0809 0a0b      ..Gc.....
0x0020 0c0d 0e0f 1011 1213 1415 1617 1819 1a1b      .....
0x0030 1c1d 1e1f 2021 2223 2425 2627 2829 2a2b      .....!#$%&'()*+
0x0040 2c2d 2e2f 3031 3233 3435 3637 3839 3a3b      ,-. /0123456789:;
0x0050 3c3d 3e3f 4041 4243 4445 4647 4849 4a4b <=>?@ABCDEFGHIJK
0x0060 4c4d 4e4f 5051 5253 5455 5657 5859 5a5b      LMNOPQRSTUVWXYZ[
0x0070 5c5d 5e5f 6061 6263 6465 6667 6869 6a6b      ]^_`abcdefghijkl
0x0080 6c6d 6e6f 7071 7273 7475 7677 7879 7a7b      lmnopqrstuvwxyz{
0x0090 7c7d 7e7f 8081 8283 8485 8687 8889 8a8b      |}~.....
0x00a0 8c8d 8e8f 9091 9293 9495 9697 9899 9a9b      .....
0x00b0 9c9d 9e9f a0a1 a2a3 a4a5 a6a7 a8a9 aaab     .....
0x00c0 acad aeaf b0b1
```

Finally, the SUN ND packet dumps, protocol ID 77 (Cisco):

```
Buffer information for Middle buffer at 0x81E946BC
  data_area 0x52B1D44, refcount 1, next 0x0, flags 0x200
  linktype 7 (IP), enctype 1 (ARPA), encsize 14, rxtype 1
  if_input 0x81F5B6F0 (Ethernet0/0), if_output 0x0 (None)
  inputtime 0x0, outputtime 0x0, oqnumber 65535
  datagramstart 0x52B1D8A, datagramsize 263, maximum size 756
  mac_start 0x52B1D8A, addr_start 0x52B1D8A, info_start 0x0
  network_start 0x52B1D98, transport_start 0x52B1DAC, caller_pc 0x80223384

  source: 47.162.67.171, destination: 172.30.71.99, id: 0x1A4F, ttl: 1, prot:77

052B1D40: AFACEFAD 0A30303A 30303A31 323A2025 /,o-.00:00:12: %
052B1D50: 4C494E45 50524F54 4F2D352D 5550444F LINEPROTO-5-UPDO
052B1D60: 574E3A20 4C696E65 2070726F 746F636F WN: Line protoco
052B1D70: 6C206F6E 20496E74 65726661 0000000A l on Interfa....
052B1D80: 61737445 052B1DD4 052B0004 DD0C8380 astE.+T.+...].
052B1D90: 00306525 6A430800 450000F9 1A4F0000 .0e%jC..E..y.O..
052B1DA0: 014D379B 2FA243AB AC1E4763 00010203 .M7./"C+,.Gc....
052B1DB0: 04050607 08090A0B 0C0D0E0F 10111213 .....
052B1DC0: 14151617 18191A1B 1C1D1E1F 20212223 ..... !"#
052B1DD0: 24252627 28292A2B 2C2D2E2F 30313233 $%&'()*+,-./0123
052B1DE0: 34353637 38393A3B 3C3D3E3F 40414243 456789:;<=>?@ABC
052B1DF0: 44454647 48494A4B 4C4D4E4F 50515253 DEFGHIJKLMNOPQRS
052B1E00: 54555657 58595A5B 5C5D5E5F 60616263 TUVWXYZ[]^_`abc
052B1E10: 64656667 68696A6B 6C6D6E6F 70717273 defghijklmnopqrs
052B1E20: 74757677 78797A7B 7C7D7E7F 80818283 tuvwxyz{|}~.....
052B1E30: 84858687 88898A8B 8C8D8E8F 90919293 .....
052B1E40: 94959697 98999A9B 9C9D9E9F A0A1A2A3 ..... !"#
052B1E50: A4A5A6A7 A8A9AAAB ACADAEAF B0B1B2B3 $%&'()*+,-./0123
052B1E60: B4B5B6B7 B8B9BABB BCBDDBEBF C0C1C2C3 456789:;<=>?@ABC
052B1E70: C4C5C6C7 C8C9CACB CCCCCECF D0D1D2D3 DEFGHIJKLMNOPQRS
052B1E80: D4D5D6D7 D8D9DADB DCDDDEDF E0E1E2E3 TUVWXYZ[]^_`abc
052B1E90: E4000000 00000000 00000000 00000000 d.....
052B1EA0: 00000000 00000000 00000000 00000000 .....
052B1EB0: 00000000 00000000 00000000 00000000 .....
052B1EC0: 00000000 00000000 00000000 00000000 .....
052B1ED0: 00000000 00000000 00000000 00000000 .....
```

Tools for Surviving the Cisco IOS Flaw

Michael J. Martin

```
052B1EE0: 00000000 00000000 00000000 00000000 .....
052B1EF0: 00000000 00000000 00000000 00000000 .....
052B1F00: 00000000 00000000 00000000 00000000 .....
052B1F10: 00000000 00000000 00000000 00000000 .....
052B1F20: 00000000 00000000 00000000 00000000 .....
052B1F30: 00000000 00000000 00000000 00000000 .....
052B1F40: 00000000 00000000 00000000 00000000 .....
052B1F50: 00000000 00000000 00000000 00000000 .....
052B1F60: 00000000 00000000 00000000 00000000 .....
052B1F70: 00000000 00000000 00000000 00000000 .....
052B1F80: 00000000 00000000 00000000 00000000 .....
052B1F90: 00000000 00000000 00000000 00000000 .....
052B1FA0: 00000000 00000000 00000000 00000000 .....
052B1FB0: 00000000 00000000 00000000 00000000 .....
052B1FC0: 00000000 00000000 00000000 00000000 .....
052B1FD0: 00000000 00000000 00000000 00000000 .....
052B1FE0: 00000000 00000000 00000000 00000000 .....
052B1FF0: 00000000 00000000 00000000 00000000 .....
052B2000: 00000000 00000000 00000000 00000000 .....
052B2010: 00000000 00000000 00000000 00000000 .....
052B2020: 00000000 00000000 00000000 00000000 .....
052B2030: 00000000 00 .....

```

TCPdump (SUN ND):

```
21:09:57.719408 118.105.171.64 > 172.30.71.99: nd 178 [ttl 1]
0x0000 4500 00c6 f3ee 0000 014d afd1 7669 ab40      E.....M..vi.@
0x0010 ac1e 4763 0001 0203 0405 0607 0809 0a0b      ..Gc.....
0x0020 0c0d 0e0f 1011 1213 1415 1617 1819 1a1b      .....
0x0030 1c1d 1e1f 2021 2223 2425 2627 2829 2a2b      .....! "#$%&'()*+
0x0040 2c2d 2e2f 3031 3233 3435 3637 3839 3a3b      ,-. /0123456789:;
0x0050 3c3d 3e3f 4041 4243 4445 4647 4849 4a4b      <=>?@ABCDEFGHIJK
0x0060 4c4d 4e4f 5051 5253 5455 5657 5859 5a5b      LMNOPQRSTUVWXYZ[
0x0070 5c5d 5e5f 6061 6263 6465 6667 6869 6a6b      ]^_`abcdefghijklmnop
0x0080 6c6d 6e6f 7071 7273 7475 7677 7879 7a7b      lmnopqrstuvwxyz{
0x0090 7c7d 7e7f 8081 8283 8485 8687 8889 8a8b      |}~.....
0x00a0 8c8d 8e8f 9091 9293 9495 9697 9899 9a9b      .....
0x00b0 9c9d 9e9f a0a1 a2a3 a4a5 a6a7 a8a9 aaab      .....
0x00c0 acad aeaf b0b1 .....

```

The packets contain garbage data, because the idea of the attack is to simply fill the buffer. The search characteristics are the protocol ID and the destination address of the router. These would normally forward the packet, then actually process it for local use.

What To Do If You're Attacked

As I mentioned earlier, the router does not exhibit signs of the attack until it's too late. The attack can be detected by a network intrusion detection system with the proper signature. Most NIDS vendors have a signature available, so check your vendor's Web site. You can also detect the attack using a filtering ACL. I have covered the implementation and use of logging filtering ACLs in a previous article. You can check your router's interface input buffers for wedged packets with protocol ID 53, 55, 77 and 103. The command is <show buffers input-interface {interface} {packet|dump}>. If you find attack packets in the buffer, you will need to reboot the router to clear them. If this is not possible, Cisco recommends increasing the interface inbound hold queue, using the interface configuration sub-

Tools for Surviving the Cisco IOS Flaw

Michael J. Martin

command <hold-queue <value (75 is the default, 150 should be fine)> in>. This will enlarge the queue and allow the interface to start forwarding traffic again until you can reboot the router at a convenient time.

Preventative Measures, Workarounds And Updated Software

The easiest fix is to update your software to a rebuilt 12.2x or 12.3 image. Cisco is offering free IOS upgrades to users (see the Cisco Advisory). But there is a small downside. In many cases, you will need to upgrade the FLASH and DRAM in your router to support the IOS image. This costs money, involves router downtime and invokes the pain of going through the upgrade and IOS reload process. If upgrading the IOS is not a short-term option, you can install interface IP access-group filters on each router interface. Cisco recommends two filtering approaches:

Block Only The Attack Traffic

Implement infrastructure ACLs (or what I commonly refer to as explicit permit filtering)

The "attack block" ACL is very simple:

```
Access-list 101 deny tcp any any fragments log-input
Access-list 101 deny udp any any fragments log-input
Access-list 101 deny 53 any any log-input
Access-list 101 deny 55 any any log-input
Access-list 101 deny 77 any any log-input
Access-list 101 deny 103 any any log-input
Access-list 101 permit ip any any
```

This list needs to be installed as an inbound IP access group on all router interfaces. It serves two functions. First, it provides protection against inbound attacks to your network from locally adjacent networks under your control and remote networks on the Internet. Second, it enforces a "good net citizen policy" on your network, which disables those individuals on your network with evil intent or simply no good sense. This approach is ideal for networks with low-end routers under high utilization. There have been a number of complaints from users about performance impact of these patch ACLs, however. Administrators need to weigh the risks versus performance. But if these ACLs impact your performance, then, honestly, it is time for a new router anyway.

Now let's look at a more effective method for dealing with this exploit. You can install explicit permit L3/L4 choke filters on all of your router interfaces. In a previous article, I recommended two approaches: The Strainer Method and The Buzz Saw. The Strainer Method works from an explicit permit and implicit deny. The networks and protocols that are desirable are defined in a static access control list (SACL) and installed on a router interface as an IP access group. The SACL can permit traffic to pass at the L3 (network) layer or the L4 (protocol/transport) layer. Here are two simple examples, first L3 only:

```
Access-list 100 permit ip 172.30.100.0 0.0.0.255 any
Access-list 100 permit ip 172.30.101.0 0.0.0.255 any
Access-list 100 permit ip 172.30.102.0 0.0.0.255 172.30.103.0 0.0.0.255
```

This example allows any host on 172.30.100/24 and 101/24 to access any network. Users on the network 172.30.102/24 can only access 172.30.103/24. Everything else is implicitly denied. As a result, this SACL provides protection against the use of BOGON, RFC1918 and spoofed SRC addresses. This SACL would not, however, provide any protection against the current Cisco IOS bug, since included under the IP umbrella are protocols 53, 55, 77 and 103.

Tools for Surviving the Cisco IOS Flaw

Michael J. Martin

The better method -- but with higher administrative overhead -- is to explicitly define each protocol and, if need be, each inbound or outbound service. Here is another example:

```
Access-list 100 permit tcp 172.30.66.0 0.0.0.255 any
Access-list 100 permit tcp 172.30.67.0 0.0.0.255 any
Access-list 100 permit udp 172.30.66.0 0.0.0.255 any
Access-list 100 permit udp 172.30.67.0 0.0.0.255 any
Access-list 100 permit icmp 172.30.66.0 0.0.0.255 any
```

This SACL does provide protection against the IOS interface denial-of-service attack. Each network prefix defines the IP protocols it is permitted to use. Here TCP, UDP and ICMP are permitted, and everything else is implicitly denied. The Buzz Saw method works using the explicit permit/deny model. Specific protocol-based traffic is explicitly denied across the available service port range (0 to 65535) to block and log specific protocols and port attacks. For more detail on this approach, see my article, SACLs: filtering suggestions and ideas.

Now we have covered how the attack works, what it looks like, and how you should go about securing your environment to defend against it. I got a call from a friend who works for a carrier and was bemoaning the fact that he had to install an ACL on every router interface. You can simplify this process a great deal by using a script. If you are facing this problem, here's a script to get you started: Download the Aclinstall script.

This script, called Aclinstall, is an expect language script that builds a local source, local destination, or simple "any any" explicit permit ACL for the following IP protocols: TCP, UDP, ICMP (the ones that you need only), OSPF, EIGRP, IGRP, GRE, ESP and AHP. You define the protocols you wish to use within the script. The tool also installs the ACLs on a single router interface and saves the configuration.

Here is the command line syntax:

- <target host> = The IP address of hostname of the router on which you are running the script
- <user> = Username (using TACACS or RADIUS)
- <pass> = The VTY password (you should use TACACS) or username password
- <enable> = The enable password for the router
- <acl#> = The access-list number. The tool builds extended ACLs, so the range is 100-199.
- <install L2 interface> = The router interface on which the IP access group will be installed
- <install interface L3 address> = The IP address of the router interface on which the access group will be installed
- <interface filter dir (in|out)> = This is the important flag. If you use the "-in" flag, the ACL will be created with the defined network prefix expected in the destination address portion of the IP packet. Use this flag if you are creating ACLs to filter traffic flowing into your network from an unknown external network. This can be return traffic being sent by external hosts to internal hosts that have opened outbound connections, or external hosts opening inbound connections. If you use the "-out" flag, the ACL will be created with the defined network prefix expected in the source

Tools for Surviving the Cisco IOS Flaw

Michael J. Martin

address portion of the IP packet. Use this flag if you are creating ACLs to filter traffic out of your network.

- `<src/dst network prefix|any>` = The network prefix you wish to match. The tool assumes that you're filtering on one base prefix. If you need to create prefix entries for multiple networks, the tool will need to be run to add each prefix. The "any" attribute can be used in place of a specific network prefix if you only want to filter by protocol.
- `<filter orientation -in|-out>` = The filter direction you want the access group installed on the interface
- `<network wildcard mask>` = The network mask, in wildcard format, of the defined network prefix. This is only needed if a network prefix is defined.

One thing about security issues is that people like me write articles and tell you that you should have configured your router a certain way, and if you had, you would not be in the fix you find yourself in today. But everyone got sniped by this bug. In an ideal world, everything would be implemented correctly, but the world is not ideal. Look at this event as a learning experience and implement proper ACLs on your routers and switches.