Christof Voemel

SANS Intrusion Detection Practical - SANS Parliament Square 2001

GCIA Practical Assignment Version 3.0

September 7, 2001

Outline:

Assignment 1 - Describe the State of Intrusion Detection:

Chronicles of the birth of LaBrea - History is happening right now

Abstract
A common practice of network defense is to set up honeypots in order to study the behavior of attackers, see for example the Honeynet Project at http://project.honeynet.org/
We describe in this essay the evolution of a complimentary technique represented by the "LaBrea" software for Linux. LaBrea is designed as a lightweight tool against automated scans and worm propagation, intended to slow down an attacker by tricking him first into connections with non-existent machines that are afterwards dropped without reset. Thus, the attacker will continue to send data never being acknowledged until the connection times out. LaBrea can be a valuable tool for an intrusion analyst monitoring a network as it potentially slows down the attacker and increases the time for reaction.
LaBrea, version 1.6 of August 28, is free under the GNU license and available from http://www.threenorth.com/LaBrea/

Essay
In July 2001, we see the powerful "Code Red" worm spreading over the internet exploiting a vulnerability in the Microsoft IIS 4.0/5.0 web server software.
For a summary of Code Red, we refer to
http://www.incidents.org/react/code_red.php
Characteristic for all variants of the worm is that once a system is infected, it starts scanning other systems on TCP port 80 to infect them, too.

As a reaction to the worm spread, Tom Liston proposes on July 31, 2001 to the readers of the Incidents.org Intrusions Mail list to install a sort of "decoy systems" to occupy the worm and slow it down.
http://www.incidents.org/archives/intrusions/msg01215.html
He points out that unused IP addresses could host these decoy systems which should be doing nothing else but waiting for the worm to try

to connect to their TCP port 80. Once a TCP SYN is received on port 80, the decoy host replies with a SYN/ACK and sets at the same time the Maximum Segment Size (MSS) for the connection to a small value. (The TCP MSS value specifies the maximum amount of TCP data in a single IP datagram that the local system can accept.) The infected system would now start to send the exploit code in small packets to the decoy host, but the decoy host is programmed to acknowledge only the initial SYN and none of the following packets.
This causes the TCP connection to time out after a fair amount of time, keeping the infected machine occupied and slowing down the spread of the worm. Furthermore, it wouldn't be necessary to really have a physical host on each unused IP number, it would be sufficient to use a generalized Network Address Translation to route all traffic to e.g. a single host impersonating all the decoys.

After some initial discussions, Mihnea Stoenescu announces two days later that she has successfully tried the concept, see
http://www.incidents.org/archives/intrusions/msg01239.html
"For a few hours I've been teergrubing CodeReds via three-way handshake on behalf of an entire C-block, by using only one host. At a rate of 6 hosts per minute hitting my block, I'm consuming circa 15 minutes of effective attack time every minute. A lot of hosts can be scanned in 15 minutes."

This is the start of the development of what is now known as LaBrea.
Only one day later, on August 3, Tom Liston, announces the release of CodeRedneck, a test code to bottleneck CodeRed scans. At the same time, he publishes two sample traces of bottlenecked connections, the first one timing out after 101.77, the second one after 93.60 seconds, compared to "normally [...] 10 seconds or less for someone to scan a Class C subnet (254 addresses)", see
http://www.incidents.org/archives/intrusions/msg01262.html
You can feel the joy of having discovered something new in his following post, only roughly two and a half hours after having announced CodeRedneck. This time, his traces show the 24 minutes-"teergrubbing" of a scanner trying to launch an exploit against portmapper on TCP port 111
http://www.incidents.org/archives/intrusions/msg01259.html

We show part of these traces here for illustration

```
Aug  3 16:58:41 ns CodeRedneck: Teergrubing: 203.161.230.82 4138 ->
my.local.ip.addr 111
Aug  3 16:58:41 ns CodeRedneck: Activity: 203.161.230.82 4138 ->
my.local.ip.addr 111
Aug  3 17:20:20 ns CodeRedneck: Activity: 203.161.230.82 4138 ->
my.local.ip.addr 111
[...]
Aug  3 17:22:20 ns CodeRedneck: Activity: 203.161.230.82 4138 ->
my.local.ip.addr 111 *
```

On August 8, the first version of LaBrea, "la brea" (Spanish) = the tar, is announced: http://www.incidents.org/archives/intrusions/msg01368.html
Bundling essentially the functionality of CodeRedneck on a Linux boot disk, this initial version of LaBrea can be used to start a small runtime environment for an improved version of CodeRedneck on a machine with minimal hardware requirements.
After configuration, the machine is IP-aliasing itself to all the previously unused IP numbers that shall be used as tarpits. While in reality dropping all inbound TCP connections (without sending the graceful "RST"), it teergrubs all connection attempts on every port to the tarpit machines, extending the capabilities of the original CodeRedneck. Another difference to the initial code is that LaBrea now uses TCP windows advertising in order to reduce the incoming data stream to minimal 10 bytes.

During the following two days, a number of people evaluate the new tool and post their experiences to the Incidents.org Intrusions Mail list.
The most important post for the further development of LaBrea comes from Micropterus Salmoides asking how to use LaBrea effectively

for a whole x.x.x.x/16 subnet.
http://www.incidents.org/archives/intrusions/msg01380.html

The difficulties in solving this problem are associated to routing packets to the tarpits on the LAN. In the initial design, the LaBrea host acted as a gateway to the "virtual tarpits", however, doing this for a whole subnet would exceed the capabilities of the lightweight machine for which LaBrea is intended.
This is illustrated in the reply given by Tom Liston
http://www.incidents.org/archives/intrusions/msg01386.html
"...unless you have something sitting behind the router answering ARP requests for an IP, it'll either deep-six the packet, or it'll reply with an ICMP unreachable. What LaBrea does is use IP aliasing to create "virtual machines" sitting on the IP addresses that you specify. [...]
I don't think a machine will handle that many aliases."

Only half an hour later, Tom Liston comes up with a solution to the problem: The capabilities of LaBrea have to be extended such that not only the IP addresses of non-existent hosts are spoofed but also their ARP replies!
http://www.incidents.org/archives/intrusions/msg01388.html
"Here's the deal: Via libpcap and libnet, it appears that the next step for LaBrea is possible. I should be able to write something that monitors arp who-has requests, fires back an arp is-at, still DROPs all the inbound packets and fires back a SYN/ACK. All based on a BPF filter... That means we can handle "Big Honkin' Chunks o' IP space!" (TM)"

While Tom Liston is working on the extension of LaBrea to ARP spoofing, Donald Smith reviews the source code and suggests substantial design modifications improving both memory consumption and performance.
The first modification concerns LaBrea's random number generator that is used to generate random IP id numbers and random TCP sequence numbers for the SYN/ACK reply to the SYN scanner.
Donald argues that random number generation is computationally expensive and therefore too slow. The second modification concerns the snaplenght of the link layer interface that is shortened from 64KB packet inspection to 1500B (standard Ethernet size).
The proposal for these modifications can be seen at
http://www.incidents.org/archives/intrusions/msg01408.html
and its acknowledgment by Tom Liston at
http://www.incidents.org/archives/intrusions/msg01413.html
(We remark that, while random number generation is at that time omitted because of the cost argument, it is re-introduced in a clever way in version 1.5 of LaBrea: random numbers are pre-generated and pulled from a list when needed, see http://www.threenorth.com/LaBrea/)

On August 15, Tom Liston announces the beta release of the new LaBrea, incorporating also ARP reply spoofing, see
http://www.incidents.org/archives/intrusions/msg01453.html
and finally one week later the full release as independent stand-alone program for compilation and execution under windows
http://www.incidents.org/archives/intrusions/msg01505.html
The final release has the additional feature of not only covering TCP SYN scans but also ICMP echo requests, making the false decoy hosts "ping"-able.

Here ends, currently, the evolution of LaBrea. Its importance as a tool for the intrusion analyst is clearly stated by Forrest Whitcher
http://www.incidents.org/archives/intrusions/msg01567.html
"I expect that this tool could be deployed in strategic places about the 'net to lengthen the odds / increase the attack cost at little expense to the defenders."
It is ideal if an analyst monitoring a network can rely on automated defenses slowing down the attacker and giving the analyst more time for reaction.

Personally, I think that the concept of LaBrea is very promising and will certainly evolve. With respect to Nmap fingerprinting, it is clearly possible to design virtual tarpits impersonating not any, but a specific machine with a specific operating system, thus representing "virtual honeypots". On the other hand, it is also clear that scanning techniques will be adapted to recognize the simple "connection choking" performed by LaBrea. A possible extension to LaBrea could for example incorporate random selective acknowledging of packets while still dropping the connection. It will be exciting to follow the further development of the "tar pit", history is happening right now!

Disclaimer:
In this short chronicle, I try to describe the exciting evolution of LaBrea as I could follow it by reading the posts to the Incidents.org Intrusions Mail list. I have tried my best to give credit to the contributions of everybody involved and apologize if I missed somebody's achievements.

References (for assignment 1)

LaBrea, version 1.6 of August 28, is available from
http://www.threenorth.com/LaBrea/

http://project.honeynet.org/

http://www.nmap.org/

http://www.incidents.org/react/code_red.php

http://www.threenorth.com/LaBrea/

http://www.incidents.org/archives/intrusions/msg01215.html
Date: Tue, 31 Jul 2001 08:47:06 -0500
From: "Tom Liston" <tliston@xxxxxxxxxxx>
Subject: Can we make life difficult for Code Red?

http://www.incidents.org/archives/intrusions/msg01239.html
Date: Thu, 2 Aug 2001 04:20:32 +0300 (EEST)
From: Mihnea Stoenescu <mihnea@xxxxx>
Subject: RE: Can we make life difficult for Code Red?

http://www.incidents.org/archives/intrusions/msg01262.html
Date: Fri, 3 Aug 2001 14:53:20 -0500
From: "Tom Liston" <tliston@xxxxxxxxxxx>
Subject: Yes, we CAN make life difficult for Code Red (and a whole lot more...)

http://www.incidents.org/archives/intrusions/msg01259.html
Date: Fri, 3 Aug 2001 17:36:07 -0500
From: "Tom Liston" <tliston@xxxxxxxxxxx>
Subject: CodeRedneck

http://www.incidents.org/archives/intrusions/msg01368.html
Date: Wed, 8 Aug 2001 15:06:16 -0500
From: "Tom Liston" <tliston@xxxxxxxxxxx>
Subject: New tool: LaBrea

http://www.incidents.org/archives/intrusions/msg01380.html
Date: Thu, 09 Aug 2001 09:04:36 -0400
From: "Micropterus Salmoides" <micro_salmoides@xxxxxxxxxxx>
Subject: RE: New tool: LaBrea

http://www.incidents.org/archives/intrusions/msg01386.html
Date: Thu, 9 Aug 2001 09:49:17 -0500
From: "Tom Liston" <tliston@xxxxxxxxxxx>

Subject: RE: New tool: LaBrea

http://www.incidents.org/archives/intrusions/msg01388.html
Date: Thu, 9 Aug 2001 10:19:27 -0500
From: "Tom Liston" <tliston@xxxxxxxxxxx>
Subject: RE: New tool: LaBrea

http://www.incidents.org/archives/intrusions/msg01408.html
Date: Fri, 10 Aug 2001 10:05:24 -0600
From: "Smith, Donald " <Donald.Smith@xxxxxxxxx>
Subject: RE: My tribulations with the new tool: LaBrea

http://www.incidents.org/archives/intrusions/msg01413.html
Date: Fri, 10 Aug 2001 15:38:03 -0500
From: "Tom Liston" <tliston@xxxxxxxxxxx>
Subject: LaBrea - Update

http://www.incidents.org/archives/intrusions/msg01453.html
Date: Wed, 15 Aug 2001 14:36:35 -0500
From: "Tom Liston" <tliston@xxxxxxxxxxx>
Subject: Son of a tarpit! A new LaBrea! Beta testers wanted.

http://www.incidents.org/archives/intrusions/msg01505.html
Date: Wed, 22 Aug 2001 13:58:38 -0500
From: "Tom Liston" <tliston@xxxxxxxxxxx>
Subject: Announcement: The NEW LaBrea - *SON OF A TARPIT*

http://www.incidents.org/archives/intrusions/msg01544.html
Date: Tue, 28 Aug 2001 09:45:32 -0500
From: "Tom Liston" <tliston@xxxxxxxxxxx>
Subject: Livin' la Vida LaBrea...

http://www.incidents.org/archives/intrusions/msg01567.html
Date: Wed, 29 Aug 2001 22:55:41 -0400
From: forrest whitcher <fw@xxxxxxxxxxxxx>
Subject: Re: Livin' la Vida LaBrea... Too bad CodeRed doesn't speak IPv6?

Introductory remarks to assignments 2 and 3

For the assignments 2 and 3, I have used the following lists of frequently seen port numbers:
http://www.iana.org/assignments/port-numbers
http://advice.networkice.com/advice/Exploits/Ports/default.htm
http://www.satx.rr.com/support/security/computer_ports.html
Whois lookups have been done with
http://www.arin.net/whois/index.html
The list of Common Vulnerabilities and Exposures is taken from
http://www.cve.mitre.org/cve/
As a web search engine, I used
http://www.google.com
Other references are given in the text and summarized at the end of the report.

Assignment 2 - Network Detects

Network Detect #1 - Scan for DNS servers

```
[**] IDS7/misc_SourcePortTraffic-53-tcp [**]
07/10-09:51:13.280591  type:0x800 len:0x3C
198.87.182.135:53 -> good.guy.xxx.xxx:53 TCP TTL:244 TOS:0x0 ID:39444
IpLen:20 DgmLen:40
******S* Seq: 0x726CB93E  Ack: 0xE35B332  Win: 0x28  TcpLen: 20
0x0000: 00 20 AF DC D7 D8 00 60 09 C4 16 7A 08 00 45 00  ......`...z..E.
```

```
0x0010: 00 28 9A 14 00 00 F4 06 37 84 C6 57 B6 87 C7 4C .(......7..W...L
0x0020: B1 0B 00 35 00 35 72 6C B9 3E 0E 35 B3 32 50 02 ...5.5rl.>.5.2P.
0x0030: 00 28 CD 06 00 00 1E 60 FF 6C D3 73              .(.....`.l.s

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

[**] IDS7/misc_SourcePortTraffic-53-tcp [**]
07/10-09:51:13.283591 type:0x800 len:0x3C
198.87.182.135:53 -> good.guy.xxx.xxx:53 TCP TTL:243 TOS:0x0 ID:39444
IpLen:20 DgmLen:40
******S* Seq: 0x726CB93E  Ack: 0xE35B332  Win: 0x28  TcpLen: 20
0x0000: 00 A0 24 14 AF F0 00 60 09 C4 16 7A 08 00 45 00 ..$....`...z..E.
0x0010: 00 28 9A 14 00 00 F3 06 38 7A C6 57 B6 87 C7 4C .(......8z.W...L
0x0020: B1 15 00 35 00 35 72 6C B9 3E 0E 35 B3 32 50 02 ...5.5rl.>.5.2P.
0x0030: 00 28 CC FC 00 00 8B B8 6E B3 7D 22              .(......n.}"

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

[**] IDS7/misc_SourcePortTraffic-53-tcp [**]
07/10-09:51:13.285192 type:0x800 len:0x3C
198.87.182.135:53 -> good.guy.xx.xxx:53 TCP TTL:243 TOS:0x0 ID:39444
IpLen:20 DgmLen:40
******S* Seq: 0x726CB93E  Ack: 0xE35B332  Win: 0x28  TcpLen: 20
0x0000: 00 C0 4F 20 9B 00 00 60 09 C4 16 7A 08 00 45 00  ..O...`...z..E.
0x0010: 00 28 9A 14 00 00 F3 06 38 77 C6 57 B6 87 C7 4C .(......8w.W...L
0x0020: B1 18 00 35 00 35 72 6C B9 3E 0E 35 B3 32 50 02 ...5.5rl.>.5.2P.
0x0030: 00 28 CC F9 00 00 1E 60 FF 6C D3 73              .(.....`.l.s

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

[**] IDS7/misc_SourcePortTraffic-53-tcp [**]
07/10-09:51:13.285459 type:0x800 len:0x3C
198.87.182.135:53 -> good.guy.xx.xxx:53 TCP TTL:243 TOS:0x0 ID:39444
IpLen:20 DgmLen:40
******S* Seq: 0x726CB93E  Ack: 0xE35B332  Win: 0x28  TcpLen: 20
0x0000: 00 A0 24 05 EB 8B 00 60 09 C4 16 7A 08 00 45 00 ..$....`...z..E.
0x0010: 00 28 9A 14 00 00 F3 06 38 76 C6 57 B6 87 C7 4C .(......8v.W...L
0x0020: B1 19 00 35 00 35 72 6C B9 3E 0E 35 B3 32 50 02 ...5.5rl.>.5.2P.
0x0030: 00 28 CC F8 00 00 7D 2A 40 3D 7D 2A              .(....}*@=}*

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

[**] IDS7/misc_SourcePortTraffic-53-tcp [**]
07/10-09:51:13.286057type:0x800 len:0x3C
198.87.182.135:53 -> good.guy.xxx.xxx:53 TCP TTL:243 TOS:0x0 ID:39444
IpLen:20 DgmLen:40
******S* Seq: 0x726CB93E  Ack: 0xE35B332  Win: 0x28  TcpLen: 20
0x0000: 00 60 6E 3B 08 93 00 60 09 C4 16 7A 08 00 45 00 .`n;...`...z..E.
0x0010: 00 28 9A 14 00 00 F3 06 38 73 C6 57 B6 87 C7 4C .(......8s.W...L
0x0020: B1 1C 00 35 00 35 72 6C B9 3E 0E 35 B3 32 50 02 ...5.5rl.>.5.2P.
0x0030: 00 28 CC F5 00 00 7A B1 54 48 92 C0              .(....z.TH..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
(...snip, not all traces shown, cf. the URL for the source of the trace)
```

1. Source of Trace.
This trace was obtained from the incidents.org archive page
http://www.incidents.org/archives/intrusions/msg01017.html
Date of posting: Tue, 10 Jul 2001 16:10:33 -0500
From: "Rich Phelps"
Subject: INCIDENT

2. Detect was generated by:
Snort IDS

3. Probability the source address was spoofed:

Although the IP destination addresses are (somehow poorly)
sanitized, this is clearly a scan and not a DoS.
We can see this either from the different MAC destination addresses
packet 1: 00 20 AF DC D7 D8, packet 2: 00 A0 24 14 AF F0, etc...
or from the IP destination address in hex format
packet 1: C7 4C B1 0B, packet 2: C7 4C B1 15, etc...
Since the scanner wants a response, his source address is not spoofed.

4. Description of attack:
We see a scan for TCP port 53 (0x35) with crafted packets.
We look at first at the IP headers
packet 1: 4500 0028 9A14 0000 F406 3784 C657 B687 C74C B10B
packet 2: 4500 0028 9A14 0000 F306 387A C657 B687 C74C B115
packet 3: 4500 0028 9A14 0000 F306 3877 C657 B687 C74C B118
etc...
We see clear indications of packet craft, all packets have the
same identification 0x9814, also the TTL is suspiciously high,
probably initially set to 255 by the sender.
Looking further at the TCP content, we see more such indications:
packet 1: 0035 0035 726C B93E 0E35 B332 5002 0028 CD06 0000 1E60 FF6C D373
packet 2: 0035 0035 726C B93E 0E35 B332 5002 0028 CCFC 0000 8BB8 6EB3 7D22
packet 3: 0035 0035 726C B93E 0E35 B332 5002 0028 CCF9 0000 1E60 FF6C D373
etc...
We see that all packets have the same sequence number 0x726C B93E
and also the same acknowledgment number 0x0E35 B332. The TCP window
size is always advertised as 0x28. The TCP header length
is said to be 20 octets, and from the IP headers we see that the total length
of the packet should be 0x28=40 octets. However, we see that the actual
TCP payload has a size of 26 octets!
This attack signature can be related to the "synscan 2.0" tool, see the URLs
given as correlations.

5. Attack mechanism:
The attacker is scanning for DNS servers for a later targeted exploit.
The Berkeley Internet Name Domain (BIND) package is the most widely used
implementation of Domain Name Service (DNS) and infamous for its history
of vulnerabilities discovered over the years.
At the time of writing this report, BIND vulnerabilities were ranking as
number one threat in the SANS Top Ten, with the recently
discovered vulnerabilities CVE-1999-0833 and CVE-1999-0009, see
http://www.sans.org/topten.htm

6. Correlations:
At the incidents.org archive page, see
http://www.incidents.org/archives/intrusions/msg00980.html
Date: Thu, 5 Jul 2001 17:36:10 -0600
From: "Smith, Donald "
Subject: synscan 2.0
and
http://www.incidents.org/archives/intrusions/msg01020.html
Date: Tue, 10 Jul 2001 17:17:35 -0600
From: "Smith, Donald "
Subject: RE: INCIDENT

7. Evidence of active targeting:
This is clearly a scan targeted at the subnet C7.4C.B1.* = 199.76.177.*
The 'oversized' TCP payload indicates that an immediate exploit attempt
could follow, once a DNS server has been detected.

8. Severity:
Criticality: 5 (DNS server is key element of the network infra structure)
Lethality: 5 (root compromise possible)
System Countermeasures: (I don't have enough details about the system,
 but I assume, the poster has applied all BIND patches before posting
 to incidents.org.)
Network counter measures: 4 (I don't have enough details about the

firewall, however, IDS logging with full packet capturing is in place.)
This gives a severity of 2.

9. Defensive recommendation:
Extensive recommendations are given at the SANS page
http://www.sans.org/topten.htm
Most important is to upgrade to the latest version of BIND and
to employ all patches. Most notably, new BIND versions can and should
be run as a non-privileged user.

10. Multiple choice test question:
Printed below is the hex-content of a TCP packet.
Which of the following characteristics indicates packet craft?

```
 Exhibit:
 4500 0028 9A14 0000 F406 3784 C657 B687
 C74C B10B 0035 0035 726C B93E 0E35 B332
 5002 0028 CD06 0000 1E60 FF6C D373
```

a)The protocol 0x06.
b)The combination of source port 0x35 and destination port 0x35.
c)The total length of the IP datagram 0x28.
d)The IP Type Of Service (TOS) value of 0x0.
C
Answer: C
The dumped IP datagram has actually a total length of 46 octets and
not 40 as stated.

Network Detect #2 - Unicode vulnerability scan or IDS test?

```
[**] [102:1:1] spp_http_decode: ISS Unicode attack detected [**]
09/05-02:40:54.291297 0:0:C:46:5C:D1 -> 0:E0:1E:8E:31:71 type:0x800 len:0x7D
61.183.107.135:1431 -> 130.216.191.21:80 TCP TTL:42 TOS:0x0 ID:4237
IpLen:20 DgmLen:111 DF
***AP*** Seq: 0xBD132FF5 Ack: 0x4EA173B2 Win: 0x5A00 TcpLen: 32
TCP Options (3) => NOP NOP TS: 13064 745178560 [Snort log]

[**] [102:1:1] spp_http_decode: ISS Unicode attack detected [**]
09/05-02:40:54.314890 0:0:C:46:5C:D1 -> 0:E0:1E:8E:31:71 type:0x800 len:0x7D
61.183.107.135:1435 -> 130.216.191.25:80 TCP TTL:42 TOS:0x0 ID:4239
IpLen:20 DgmLen:111 DF
***AP*** Seq: 0xBD1647BA Ack: 0x1F69FE01 Win: 0x5A00 TcpLen: 32
TCP Options (3) => NOP NOP TS: 13065 14788526 [Snort log]

[**] [102:1:1] spp_http_decode: ISS Unicode attack detected [**]
09/05-02:40:54.333607 0:0:C:46:5C:D1 -> 0:E0:1E:8E:31:71 type:0x800 len:0x71
61.183.107.135:1448 -> 130.216.191.38:80 TCP TTL:42 TOS:0x0 ID:4241
IpLen:20 DgmLen:99 DF
***AP*** Seq: 0xBD202F05 Ack: 0x6EEAE331 Win: 0x5A00 TcpLen: 20 [Snort log]

[**] [102:1:1] spp_http_decode: ISS Unicode attack detected [**]
09/05-02:40:54.550329 0:0:C:46:5C:D1 -> 0:E0:1E:8E:31:71 type:0x800 len:0x7D
61.183.107.135:1455 -> 130.216.191.45:80 TCP TTL:42 TOS:0x0 ID:4248
IpLen:20 DgmLen:111 DF
***AP*** Seq: 0xBD257B1D Ack: 0xE72BA754 Win: 0x5A00 TcpLen: 32
TCP Options (3) => NOP NOP TS: 13067 256051526 [Snort log]

[**] [102:1:1] spp_http_decode: ISS Unicode attack detected [**]
09/05-02:40:54.573106 0:0:C:46:5C:D1 -> 0:E0:1E:8E:31:71 type:0x800 len:0x7D
61.183.107.135:1460 -> 130.216.191.50:80 TCP TTL:42 TOS:0x0 ID:4250
IpLen:20 DgmLen:111 DF
***AP*** Seq: 0xBD299491 Ack: 0xB5A7F0FC Win: 0x5A00 TcpLen: 32
TCP Options (3) => NOP NOP TS: 13067 0 [Snort log]
```

packet dump:

```
[**] spp_http_decode: ISS Unicode attack detected [**]
09/05-02:40:54.333607 0:0:C:46:5C:D1 -> 0:E0:1E:8E:31:71 type:0x800 len:0x71
61.183.107.135:1448 -> 130.216.191.38:80 TCP TTL:42 TOS:0x0 ID:4241
IpLen:20 DgmLen:99 DF
***AP*** Seq: 0xBD202F05  Ack: 0x6EEAE331  Win: 0x5A00  TcpLen: 20
47 45 54 20 2F 73 63 72 69 70 74 73 2F 2E 2E 25  GET /scripts/..%
63 31 25 31 63 2E 2E 2F 77 69 6E 6E 74 2F 73 79  c1%1c../winnt/sy
73 74 65 6D 33 32 2F 63 6D 64 2E 65 78 65 20 48  stem32/cmd.exe H
54 54 50 2F 31 2E 30 0D 0A 0D 0A                 TTP/1.0....
```

1. Source of Trace.
http://www.incidents.org/archives/intrusions/msg01621.html
Date: Wed, 5 Sep 2001 09:03:24 +1200 (NZST)
From: Russell Fulton <r.fulton@xxxxxxxxxxxxxx>
Subject: Exploit attempt from

2. Detect was generated by:
Snort plugin spp_http_decode

3. Probability the source address was spoofed:
Http works over TCP port 80, thus a 3-way handshake is established
and the source address cannot be spoofed.

4. Description of attack:
We see a scan of servers on TCP port 80 associated to http.
The snort alert refers to a flaw in the Microsoft IIS Unicode translation
described by the Internet Security Systems (ISS) at
http://xforce.iss.net/alerts/advise68.php
Unicode is a standard providing a identifiers for every character in
every language to facilitate uniform computer representation of the
characters used all over the world, see http://www.unicode.org
The vulnerability of the IIS consists of the fact that (the unpatched)
IIS doesn't decode overlong Unicode representations until after path checking
and allows a directory transversal attack, see
http://www.securityfocus.com/archive/1/140091
While the obvious directory transversal
http://www.victim.com/../../winnt/system32/cmd.exe
doesn't work because IIS strips off "../..", the unpatched IIS doesn't
recognize the equivalent Unicode representation "..%C1%1C.." as directory
transversal and allows so the attacker to execute commands.
This vulnerability has been categorized as CAN-2000-0884.

5. Attack mechanism:
The basic attack mechanism has already been described under 4 and is
also explained at http://www.8thlayer.org/unicode.html
From there, we take one example to illustrate the dangers of the vulnerability:
http://www.target.site/scripts/..%c0%af../winnt/system32/newcmd.exe?/c
+echo+you+be+ultra+hacked+>+..\wwwroot\index.htm
It is remarkable that the representation of the "/" character varies
from one character set to another, in the above line it is "%c0%af" and not
"..%C1%1C..".
The question is whether the traces above show an attack or something else.
We see TCP packets logged of length "len:0x71" and "len:0x7D", unfortunately,
only the contents of one of the smaller packets is printed.
The dumped packet contains
GET /scripts/..%c1%1c../winnt/system32/cmd.exe HTTP/1.0...
which doesn't do anything malicious. By doing research on the web,
I found the (French) web site
http://www.hsc.fr/ressources/outils/idswakeup/
describing a utility performing IDS tests like the following:
133.186.155.192 -> 127.0.0.1 80/tcp GET /scripts/cmd.exe HTTP/1.0
Such a utility could be easily modified to execute also the above command.
For a complete analysis, I would need the packet dumps of the larger packets.
At this point, I would suppose that someone is testing the IDS, or perhaps
trying to find out whether an IDS is installed. (I realize that the command
"HTTP/1.0..." could also be used to trigger an error message and to

obtain the version banner of IIS. However, I think that this possibility is not very likely as the attacker wouldn't need to use an exploit to obtain this information.)

6. Correlations:
I don't know of any traces showing the same "harmless" commands.
I have already pointed out the possibility of an IDS test.
There have been a lot of Unicode attacks that can be found by searching the web. A complete write-up on Unicode together with the backgate kit can be found at the SANS page
http://www.incidents.org/react/unicode.php

7. Evidence of active targeting:
We see something like a scan, or as discussed, an IDS test. This might be targeted specifically at our network, however, we don't have evidence of an exploit attempt targeted at one of the machines.

8. Severity:
Criticality: 4 (Web server probe)
Lethality: 3-5 (not enough infos, depends whether the logs contain malicious packets that would allow to obtain control over the machine)
System Countermeasures: 4 (not enough infos, I assume that IIS is patched)
Network counter measures: 3-4 (not enough infos, but at least IDS is in place)
This gives a severity of 0-2, for a clear rating, more information is needed.

9. Defensive recommendation:
The Unicode bug has been completely fixed, refer to the following Microsoft security bulletin for the patch
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms00-057.asp

10. Multiple choice test question:
In the packet dump shown below, what is the significance of the content represented by the ASCII characters "..%c1%1c.."?
Exhibit:

```
09/05-02:40:54.333607 0:0:C:46:5C:D1 -> 0:E0:1E:8E:31:71 type:0x800 len:0x71
61.183.107.135:1448 -> 130.216.191.38:80 TCP TTL:42 TOS:0x0 ID:4241
IpLen:20 DgmLen:99 DF
***AP*** Seq: 0xBD202F05  Ack: 0x6EEAE331  Win: 0x5A00  TcpLen: 20
47 45 54 20 2F 73 63 72 69 70 74 73 2F 2E 2E 25  GET /scripts/..%
63 31 25 31 63 2E 2E 2F 77 69 6E 6E 74 2F 73 79  c1%1c../winnt/sy
73 74 65 6D 33 32 2F 63 6D 64 2E 65 78 65 20 48  stem32/cmd.exe H
54 54 50 2F 31 2E 30 0D 0A 0D 0A                 TTP/1.0....
```

a)Packets containing the characters "..%c1%1c.." circumvent the detection by the snort IDS, because snort can't speak Unicode.
b)The characters "..%c1%1c.." allowed to change a directory on an unpatched IIS.
c)The characters "..%c1%1c.." are the Unicode encoding of "69", the signature of the "Woodstock-virus".
d)The characters "..%c1%1c.." generate a buffer overflow in the Apache web server and allow the attacker to gain root privileges.
B
Answer: B
The unpatched IIS (4.0 and 5.0) doesn't recognize the Unicode representation "..%C1%1C.." of "../.." as directory transversal and allows so the attacker to execute commands.

Network Detect #3 - Portmap scan with immediately following attack attempt

```
Jun 13 13:24:52 66.96.216.175:3365 -> a.b.c.51:111 SYN ******S*
Jun 13 13:24:54 66.96.216.175:4159 -> a.b.c.59:111 SYN ******S*
Jun 13 13:24:54 66.96.216.175:4162 -> a.b.c.62:111 SYN ******S*
Jun 13 13:24:56 66.96.216.175:760 -> a.b.c.62:111 UDP
Jun 13 13:24:57 66.96.216.175:4180 -> a.b.c.80:111 SYN ******S*
```

```
Jun 13 13:24:54 66.96.216.175:4205 -> a.b.c.105:111 SYN ******S*
Jun 13 13:24:57 66.96.216.175:4252 -> a.b.c.152:111 SYN ******S*
Jun 13 13:24:54 66.96.216.175:4292 -> a.b.c.192:111 SYN ******S*
Jun 13 13:24:54 66.96.216.175:4301 -> a.b.c.201:111 SYN ******S*
Jun 13 13:24:54 66.96.216.175:4309 -> a.b.c.209:111 SYN ******S*
Jun 13 13:24:54 66.96.216.175:4312 -> a.b.c.212:111 SYN ******S*
(snip, not all scan traces shown, refer to URL below)
Jun 13 13:24:56 hosth portmap[1879]: connect from 66.96.216.175 to getport(status):
request from unauthorized host
Jun 13 13:24:56 hosth snort: RPC portmap request rstatd: 66.96.216.175:760 ->
a.b.c.62:111
```

1. Source of Trace.
This trace was obtained from the incidents.org archive page
http://www.incidents.org/archives/intrusions/msg00817.html
Date: Thu, 14 Jun 2001 12:43:30 -0400
From: Laurie Zirkle <lat@xxxxxxxxxx>
Subject: June 13, 2001 probes (part #1)

2. Detect was generated by:
Snort (scan report) and system logs.

3. Probability the source address was spoofed:
Zero, because the scanner watches the replies from his SYN scan.
Once he has discovered the potentially vulnerable machine a.b.c.62:111,
he immediately tries to connect to connect to statd.

4. Description of attack:
As of the date of this report, RPC weaknesses are the number 3 among the
SANS Top Ten Most Critical Security Threats (http://www.sans.org/topten.htm).
The portmap daemon manages RPC requests, in this case one to the stat daemon.
We look at the log message

```
Jun 13 13:24:56 hosth portmap[1879]: connect from 66.96.216.175 to getport(status):
request from unauthorized host
```

Normally, the "getport(status)" call causes the statd-server (a network
status monitor) to start listening on a socket, the corresponding port number
is returned by portmapper to the client that should then connect to this port
in order to send or receive file data.
With respect to the scans before, it is clear that here, this is an attack in
order to exploit one of several known vulnerabilities of statd,
for the most recent one CAN-2000-0666 see
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0666
for others, see the links on the SANS Top Ten page.
In the current case, the attacker fails because a secure portmapper is
installed which recognizes that the attacker is not authorized for
a remote connect to statd and logs the event.

5. Attack mechanism:
If the attacker succeeds to connect to statd, he can gain root privileges on
the system. By supplying crafted queries to the server, he can trick the
server into executing them as commands. Since the server runs with root
privileges, the host running this server can be compromised.
As an example, see the description of an exploit for the vulnerability
CAN-2000-0666 on
http://www.cert.org/advisories/CA-2000-17.html
and in the bugtraq archives, e.g. at
http://security-archive.merton.ox.ac.uk/bugtraq-200007/0209.html

6. Correlations:
Portmap attacks occur very frequently, see also the traces I analyzed for
assignment 3 in this report. Other correlations include
http://www.incidents.org/archives/intrusions/msg01060.html
Date: Mon, 16 Jul 2001 07:24:54 -0400
From: Laurie Zirkle <lat@xxxxxxxxxx>
Subject: July 14, 2001 probes
and

http://www.incidents.org/archives/intrusions/msg01210.html
Date: Mon, 30 Jul 2001 11:40:13 -0400
From: Laurie Zirkle <lat@xxxxxxxxxx>
Subject: July 28, 2001 probes (part 1)

7. Evidence of active targeting:
The exploit attempt is clearly targeted at the host identified to be possibly vulnerable from the preceding scan.

8. Severity:
Criticality: 2 (I assume that a.b.c.62 is a user desktop)
Lethality: 5 (attacker can gain root access)
System Countermeasures: 4 (secure portmapper installed)
Network counter measures: 2 (the firewall doesn't block access to port 111)
This gives a severity of 1.

9. Defensive recommendation:
Blocking port 111 from external access on the firewall has been recommended by SANS, see http://www.sans.org/newlook/resources/IDFAQ/blocking.htm
For Solaris operating systems, also the high ports 32770, 32771,... have to be blocked.

10. Multiple choice test question:
In the traces of the SYN scan shown below, which is the service that the attacker is looking for?

```
 Exhibit:
 Jun 13 13:24:54 66.96.216.175:4205 -> a.b.c.105:111 SYN ******S*
 Jun 13 13:24:57 66.96.216.175:4252 -> a.b.c.152:111 SYN ******S*
 Jun 13 13:24:54 66.96.216.175:4292 -> a.b.c.192:111 SYN ******S*
 Jun 13 13:24:54 66.96.216.175:4301 -> a.b.c.201:111 SYN ******S*
 Jun 13 13:24:54 66.96.216.175:4309 -> a.b.c.209:111 SYN ******S*
 Jun 13 13:24:54 66.96.216.175:4312 -> a.b.c.212:111 SYN ******S*
```

a)FTP.
b)RPCbind.
c)SNMP.
d)Back Orifice
B
Answer: B
RPCbind (portmapper) is listening on TCP port 111.

Network Detect #4 - Large Pings for MTU discovery

```
Jun  7 18:46:56 sparky kernel: Packet log: input DENY ppp0 PROTO=1
32.97.170.150:8 12.82.133.172:0 L=1500 S=0x00 I=12295 F=0x4000 T=244 (#45)
Jun  7 18:46:57 sparky kernel: Packet log: input DENY ppp0 PROTO=1
32.97.170.137:8 12.82.133.172:0 L=1500 S=0x00 I=8574 F=0x4000 T=244 (#45)
Jun  7 18:46:59 sparky kernel: Packet log: input DENY ppp0 PROTO=1
32.97.170.149:8 12.82.133.172:0 L=1500 S=0x00 I=33637 F=0x4000 T=244 (#45)
Jun  7 18:47:13 sparky kernel: Packet log: input DENY ppp0 PROTO=1
32.97.170.149:8 12.82.133.172:0 L=1500 S=0x00 I=37030 F=0x4000 T=244 (#45)
Jun  7 18:47:30 sparky kernel: Packet log: input DENY ppp0 PROTO=1
32.97.170.150:8 12.82.133.172:0 L=1500 S=0x00 I=16241 F=0x4000 T=244 (#45)
Jun  7 18:47:38 sparky kernel: Packet log: input DENY ppp0 PROTO=1
32.97.170.137:8 12.82.133.172:0 L=1500 S=0x00 I=13455 F=0x4000 T=244 (#45)
Jun  7 18:47:42 sparky kernel: Packet log: input DENY ppp0 PROTO=1
32.97.170.149:8 12.82.133.172:0 L=1500 S=0x00 I=40763 F=0x4000 T=244 (#45)
Jun  7 18:51:07 sparky kernel: Packet log: input DENY ppp0 PROTO=1
32.97.170.150:8 12.82.133.172:0 L=1500 S=0x00 I=39684 F=0x4000 T=244 (#45)
Jun  7 18:51:08 sparky kernel: Packet log: input DENY ppp0 PROTO=1
32.97.170.149:8 12.82.133.172:0 L=1500 S=0x00 I=65323 F=0x4000 T=244 (#45)
```

```
06/07-18:46:56.327073 32.97.170.150 -> 12.82.133.172
ICMP TTL:244 TOS:0x0 ID:12295 IpLen:20 DgmLen:1500 DF
```

```
Type:8  Code:0  ID:0   Seq:0  ECHO
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00.....
<snip>
continue to fill each of 3 packets to 1500 bytes

snort:
06/07-18:46:57.277134 32.97.170.137 -> 12.82.133.172
ICMP TTL:244 TOS:0x0 ID:8574 IpLen:20 DgmLen:1500 DF
Type:8  Code:0  ID:0   Seq:0  ECHO
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00....
<snip>
continue to fill each of 4 packets to 1500 bytes

snort:
06/07-18:46:59.657435 32.97.170.149 -> 12.82.133.172
ICMP TTL:244 TOS:0x0 ID:33637 IpLen:20 DgmLen:1500 DF
Type:8  Code:0  ID:0   Seq:0  ECHO
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00...
<snip>
continue to fill each of 4 packets to 1500 bytes
```

1. Source of Trace.
This trace was obtained from the incidents.org archive page
http://www.incidents.org/archives/intrusions/msg00698.html
Date: Sat, 09 Jun 2001 08:17:39 -0700
From: John Sage <jsage@xxxxxxxxxxxxxx>
Subject: 06/07/01 probes at FinchHaven: Big pings

2. Detect was generated by:
Linux IPChains and snort.

3. Probability the source address was spoofed:
The ICMP message type 8, code 0, is an echo request. If the sender wants the reply, the source address cannot spoofed.

4. Description of attack:
We receive large ping packets on a modem that are denied by IPChains and logged by snort. Maliciously formed pings can be used to crash certain vulnerable systems. The infamous Ping O' Death is one example. However, this is absolutely not the case in our case. The packets have a size of 1500, the usual MTU of Ethernet and the DF (don't fragment bit) set. One gets immediately the idea that we see actually not an attack but an MTU discovery.

```
5. Attack mechanism:
32.97.170.150 pings 12.82.133.172 with a large packet of size 1500
with DF bit in order to find out the MTU of the connection between them.
Either 12.82.133.172 replies to the echo request, in this case the MTU
is at least 1500. If no reply is received by 32.97.170.150, or the
ICMP error message "fragmentation needed but DF flag set" is received,
the MTU is smaller than 1500. To discover the MTU of a connection
is beneficial because it prevents unnecessary fragmentation of the IP
datagrams in transit. In the IPChains logs, we see 32.97.170.150 repeating the
pings over several minutes because the firewall at
12.82.133.172 drops the echo requests.
John Sage (see the correlations) states that finding the MTU
"is fairly common from AIX machines."

6. Correlations:
Correlations can be found at
http://www.incidents.org/archives/intrusions/msg00720.html
Date: Mon, 11 Jun 2001 09:35:51 -0700
From: John Sage <jsage@xxxxxxxxxxxxxx>
Subject: Re: 06/07/01 probes at FinchHaven: Big pings
the GCIA practical of Marc Bayerkohler
http://www.sans.org/giactc/gcia.htm
and the complete thread starting with the post
http://www.incidents.org/archives/intrusions/msg01568.html
Date: Thu, 30 Aug 2001 00:56:08 -0400
From: Chris Brenton <cbrenton@xxxxxxxxxxx>
Subject: Weird Type 8 stuff

[**] MISC Large ICMP Packet [**]
06/08-15:11:42.610845 129.33.225.36 -> xxx.xxx.xxx.2
ICMP TTL:239 TOS:0x0 ID:62655 IpLen:20 DgmLen:1500 DF
Type:8  Code:0  ID:0   Seq:0  ECHO
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

[**] MISC Large ICMP Packet [**]
06/08-14:32:14.976177 162.75.1.14 -> xxx.xxx.xxx.2
ICMP TTL:243 TOS:0x0 ID:9793 IpLen:20 DgmLen:1496 DF
Type:8  Code:0  ID:0   Seq:0  ECHO
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

7. Evidence of active targeting:
The ping is directed at 12.82.133.172, that's active, but probably not
malicious targeting.

8. Severity:
Criticality: 2 (host on PPP link)
Lethality: 1 (false alarm, not an attack)
System Countermeasures: 4 (IPChains, snort, probably well patched)
Network counter measures: 1 (probably, the ISP doesn't have firewall, etc)
This gives a severity of -2.

9. Defensive recommendation:
ICMP traffic can not only be used for attacks but also for reconnaissance,
see e.g. the paper "ICMP Usage In Scanning" by Ofir Arkin, at
http://www.sys-security.com/archive/papers/ICMP_Usage_v3.0.pdf
For this reason, it is generally advisable to block most of the
ICMP traffic at the firewall, in fact, all apart from the ICMP
```

error message "fragmentation needed but DF flag set", see e.g.
http://www.faqs.org/faqs/computer-security/most-common-qs/section-18.html

10. Multiple choice test question:
In the ICMP echo request dumped below, which of the following fields
is the most "suspicious" one and could be a reason to inspect the packet?

```
 Exhibit:
 06/07-18:46:56.327073 32.97.170.150 -> 12.82.133.172
 ICMP TTL:244 TOS:0x0 ID:12295 IpLen:20 DgmLen:1500 DF
 Type:8  Code:0  ID:0   Seq:0  ECHO
```

a)The TCP Sequence number 0.
b)The ICMP code 0.
c)The Type Of Service 0.
d)The Datagram Length 1500.
D
Answer: D
ICMP traffic doesn't have a TCP sequence number, and TOS=0 and ICMP=0 are
common values for the respective fields. However, for a simple echo
request, the datagram is quite big. It is not necessarily malicious, as
it can serve to discover the MTU of a connection.

Network Detect - #5 Scan for compromised systems on TCP port 24452

```
Jun  9 08:52:06 hosth /kernel: Connection attempt to TCP a.b.c.62:24452
from 200.23.66.65:24452
Jun  9 08:56:11 hostmf /kernel: Connection attempt to TCP a.b.f.167:24452
from 200.23.66.65:24452
Jun  9 08:52:06 200.23.66.65:24452 -> a.b.c.4:24452 SYN ******S*
Jun  9 08:52:06 200.23.66.65:24452 -> a.b.c.9:24452 SYN ******S*
Jun  9 08:52:06 200.23.66.65:24452 -> a.b.c.33:24452 SYN ******S*
Jun  9 08:52:06 200.23.66.65:24452 -> a.b.c.62:24452 SYN ******S*
Jun  9 08:52:06 200.23.66.65:24452 -> a.b.c.101:24452 SYN ******S*
Jun  9 08:52:06 200.23.66.65:24452 -> a.b.c.121:24452 SYN ******S*
Jun  9 08:52:06 200.23.66.65:24452 -> a.b.c.182:24452 SYN ******S*
Jun  9 08:52:06 200.23.66.65:24452 -> a.b.c.209:24452 SYN ******S*
Jun  9 08:52:06 200.23.66.65:24452 -> a.b.c.212:24452 SYN ******S*
Jun  9 08:52:07 200.23.66.65:24452 -> a.b.d.233:24452 SYN ******S*
Jun  9 08:52:07 200.23.66.65:24452 -> a.b.e.25:24452 SYN ******S*
Jun  9 08:52:07 200.23.66.65:24452 -> a.b.e.42:24452 SYN ******S*
Jun  9 08:52:07 200.23.66.65:24452 -> a.b.e.48:24452 SYN ******S*
Jun  9 08:52:07 200.23.66.65:24452 -> a.b.e.79:24452 SYN ******S*
Jun  9 08:52:07 200.23.66.65:24452 -> a.b.e.100:24452 SYN ******S*
Jun  9 08:52:07 200.23.66.65:24452 -> a.b.e.101:24452 SYN ******S*
Jun  9 08:52:08 200.23.66.65:24452 -> a.b.e.128:24452 SYN ******S*
Jun  9 08:52:08 200.23.66.65:24452 -> a.b.e.229:24452 SYN ******S*
Jun  9 08:52:08 200.23.66.65:24452 -> a.b.e.238:24452 SYN ******S*
Jun  9 08:52:08 200.23.66.65:24452 -> a.b.f.37:24452 SYN ******S*
Jun  9 08:52:08 200.23.66.65:24452 -> a.b.f.190:24452 SYN ******S*
Jun  9 08:52:08 200.23.66.65:24452 -> a.b.f.192:24452 SYN ******S*
```

1. Source of Trace.
This trace was obtained from the incidents.org archive page
http://www.incidents.org/archives/intrusions/msg00713.html
Date: Sun, 10 Jun 2001 21:19:00 -0400
From: Laurie Zirkle <lat@xxxxxxxxxx>
Subject: June 9, 2001 probes (part #1)

2. Detect was generated by:
Snort and system logs.

3. Probability the source address was spoofed:
Zero, because the SYN scanner needs a response.

4. Description of attack:
The TCP port 24452 is not assigned as can be checked at
http://www.iana.org/assignments/port-numbers
This makes one immediately suspicious about a backdoor installed
at that port. From the time stamps that can be seen in the traces,
it is clear that the scan is automated. There are speculations about a
worm, see the URLs given as correlations. However, I haven't seen these
speculations being confirmed at the time of writing this report.
We see two log messages:

```
Jun  9 08:52:06 hosth /kernel: Connection attempt to TCP a.b.c.62:24452
from 200.23.66.65:24452
Jun  9 08:56:11 hostmf /kernel: Connection attempt to TCP a.b.f.167:24452
from 200.23.66.65:24452
```

By doing research on the web, I found the following URL
http://www.geocrawler.com/archives/3/165/1999/9/0/2672764/
producing logs in the same format.
According to this page, the FreeBSD-3.2 inetd has a "-l" flag which logs all
connection attempts, regardless of whether they are allowed, or denied.
With the additional commands

```
sysctl -w net.inet.udp.log_in_vain=1
sysctl -w net.inet.tcp.log_in_vain=1
```

also connections to invalid ports can be logged.

5. Attack mechanism:
With some research on the web, I found the following
page describing a sophisticated system compromise, see
http://www.rvglug.org/pipermail/rvglug/2001-February/000436.html
The attacker modified /etc/xinitd.conf by adding a root /bin/sh on
TCP port24452. Anybody connecting on this port can execute commands
with root privileges.

6. Correlations:
A more stealthy SYN/FIN scan on TCP port 24452 for compromised systems is
shown at the URL http://www.incidents.org/archives/intrusions/msg00540.html

```
[**] SCAN-SYN FIN [**]
03/09-15:22:06.846446 64.208.150.130:24452 -> 10.1.120.75:24452
TCP TTL:27 TOS:0x0 ID:39426
**S*F*** Seq: 0x37EE05EC   Ack: 0x490B5266   Win: 0x404
```

Another trace, this time a SYN scan like in our logs, is shown at
http://www.incidents.org/archives/intrusions/msg00747.html

7. Evidence of active targeting:
This is a noisy SYN scan not targeted at a particular machine.
From the timestamps it is clear that we see an automated scan, for
example by a script.

8. Severity:
Criticality: 2 (if there is a backdoor, it is probably at a user's desktop)
Lethality: 5 (attacker can become root, if backdoor is found)
System Countermeasures: 3 (we assume that
Network counter measures: 2 (firewall doesn't block access to this port)
This gives a severity of

9. Defensive recommendation:
Check for processes running on port 24452, for example with the "lsof"
command. In case you find a backdoor installed, proceed according to
the SANS guide "Incident Handling Step by Step: Unix Trojan Programs",
http://www.incidents.org/react/trojan.php
Generally, it is advisable to block traffic to "strange" ports
per default at a firewall, for example by denying everything that is not
explicitly allowed.

10. Multiple choice test question:
An IDS sensor logs scans of the ports listed below. Which of these
ports is probably NOT associated to trojan activity but used
by certain operating systems for regular services?

a)Port 24452.
b)Port 27374.
c)Port 31337.
d)Port 32770.
D
Answer: D.
The Sun Solaris portmapper not only listens on the well known port 111, but also on a high port greater or equal to 32770.
A backdoor placed at port 24452 was described in the above analysis, port 27374 is infamous for the Sub7 trojan, and the "ELEET" port 31337 is prominent for Back Orifice.

Assignment 3 - "Analyze This" Scenario

In the following, I give a security evaluation on the basis of 5 days of snort logs from the University of Maryland, Baltimore County.
The snort log data has been obtained from the data page of the CCS Security Officer and NIDS Handler, Andy Johnston
http://www.research.umbc.edu/~andy/
Further information on the layout of the IDS and the sensors collecting the data has been taken from a presentation given by the CCS Junior Security Officer, Robin Anderson
http://userpages.umbc.edu/~robin/Presentations/Snort/Snort_FINAL.ppt
According to the presentation, sensors and analysis boxes are separated and both automatically administrated by cron scripts.
Two types of sensors are in place:
A short term sensor is collecting data in intervals of 15 minutes and forwarding them to an analysis station that processes the data immediately. It generates alert data files and logs also selected packets with their full header as "out_of_spec" (OOS).
A long term sensor collects scan data in intervals of a day, these logs are processed by an analysis station that sends emails with a summary to the security officers.

The data sets I have investigated are from June 29 to July 3, 2001.
The files used for my analysis are the
5 Snort Alert Reports "alert.010629",...,"alert.010703"
containing alert logs in the "snort -A fast" output format and reports from the spp_portscan plugin,
the 5 Snort Scan reports "scans.010629",...,"scans.010703" containing details about UDP and TCP scans of the network,
and finally the 5 Snort "out_of_spec" (OOS) data sets
"oos_Jun.29.2001",...,"oos_Jul.3.2001"
containing the full headers of selected packets of interest.
A sample overview of the three different log formats is given in the appendix.

Only the OOS data is small enough to be investigated directly by a human, the alert and the scan logs are together of a size of approximately 42 MB. For data parsing and data inspection, I developed a perl script starting from the work of Chris Kuethe, GCIA.
My script process_scanalert.pl is substantially enhanced and has the following features:
- simultaneous parsing of alert and scan logs possible with correlation between the files
- list of occurring attack/scan types sorted by frequency
- list of targets sorted by frequency
- detection of fingerprint attempts in scan logs by unusual TCP flags
- list containing only those targets that have been fingerprinted
- detection of target addresses that are multicast/broadcast addresses
- list of attackers sorted by frequency
- list of fingerprinting attackers
- detection of spoofed source addresses (10.*, 172.16-31.*., 192.168.*)

- summary of snort spp_portscan plugin alerts
- list of scanned TCP ports, with marked fingerprints sorted by frequency
- list of scanned UDP ports sorted by frequency
- separate list of destination ports occurring in alert files,
  also sorted by [**] attack signature [**]
- list of attack-target pairs sorted by frequency
- list of paired communications, for example:
  151.21.227.249-MY.NET.137.31 *** <- 3 *** -> 1 ***

My script is given in the appendix.
Apart from using the script on whole data sets, it is also possible to use it on selected parts of the data. For example, the command line
cat alerts | grep -i 'trojan' | process_scanalert.pl -v > output
does a data analysis uniquely on all alerts of the form
[**] Possible trojan server activity [**].

For quick data browsing together with easy ARIN whois lookup of IP addresses, I also used SnortSnarf version 010821.1 on the alert logs. (http://www.silicondefense.com/software/snortsnarf/index.htm)
Some aspects of SnortSnarf need still to be fixed, for example, I had to substitute MY.NET by digits like 10.10 in order to allow SnortSnarf to parse the data. Also, SnortSnarf wasn't able to read scan logs and was therefore useless for correlations between scan and alert logs.

The following part of this report is structured as follows:
In the next section, we give an analysis of the alert log files. Secondly, an analysis of the scan logs is given and then a description of the treatment of the OOS data. Finally, a summary of the analysis is given. For each type of alert or other incident,
I investigate the packets sent by the concerned systems on MY.NET for indications of a possible compromise. I resolve attacker IP addresses on ARIN if this seems appropriate and give defensive recommendations.
I think that it is easier for an incident handler, if I list registration information about external source addresses in the analysis report of the attack they participate instead of displaying them in a separate table. The same holds for lists of possible compromises and defensive recommendations. I indicate if a host is concerned by several attacks and thus the defensive measures have to take account of several incidents.

1.Attack/Scan alerts prioritized by number of occurrences

```
118870  UDP SRC and DST outside network
15201  Possible trojan server activity
4977  High port 65535 tcp - possible Red Worm - traffic
3029  Spp_portscan detect
1648  External RPC call
1646  connect to 515 from outside
1053  Watchlist 000220 IL-ISDNNET-990517
467  SMB Name Wildcard
301  Queso fingerprint
271  SYN-FIN scan!
220  WinGate 1080 Attempt
124  Port 55850 tcp - Possible myserver activity - ref. 010313-1
118  Watchlist 000222 NET-NCFC
75  NMAP TCP ping!
66  Null scan!
58  TCP SRC and DST outside network
40  High port 65535 UDP - possible Red Worm - traffic
33  SUNRPC highport access!
29  connect to 515 from inside
24  Russia Dynamo - SANS Flash 28-jul-00
8  Attempted Sun RPC high port access
3  Back Orifice
2  STATDX UDP attack
1  TCP SMTP Source Port traffic
```

1 ICMP SRC and DST outside network
1 Tiny Fragments - Possible Hostile Activity

Many of these attacks have been seen before, compare e.g. with the work of Byron Thatcher, March 2001 available from http://www.sans.org/giactc/gcia.htm
The most remarkable difference in our data sets is perhaps that we note significantly more trojan server activity.
Using the link analysis feature of my analysis script to identify two-sided communications, I can identify a significant number of suspicious hosts on MY.NET that need to be checked or supervised for possible compromises.

Analysis of the attack alerts in detail:

*UDP SRC and DST outside network
These packets were sent to the following destination ports: 53 (DNS), 137(NetBIOS nbtstat), 138(NetBIOS master browser), and 5779. The port 5779 was unknown to me, moreover, 103758 occurrences were logged, 87% of all alerts of this kind. Research on the source addresses and inspection of the target addresses showed that most of the traffic came from Yahoo and was sent to class D multicast addresses.
Among the remaining detects, those coming from the reserved private address space 192.168.* and 172.16.* called my attention.
However, an inspection of the logs revealed that this was probably related to ordinary name resolution traffic

```
07/01-14:30:25.212260 [**] UDP SRC and DST outside network [**]
 192.168.0.102:137 -> 162.129.20.10:53
07/02-12:50:42.143742 [**] UDP SRC and DST outside network [**]
 172.16.25.169:137 -> 172.16.3.101:137
```

Apparently, the site uses NAT. As a conclusion, the packet logging should not alert on source addresses from the private address space in use to avoid false alarms. Also, I would ask to verify that filtering of the incoming packets on these addresses is in place in order to avoid spoofing. Further, a firewall has to block all NetBIOS requests from the outside.

*Possible trojan server activity
A lot of ports are associated to alerts of this kind, I will not display them all. 85% of all detects were associated to TCP destination port 27374 which is heavily used by the SubSeven v2.1 Windows trojan, for information see e.g. the corresponding CERT advisory http://www.cert.org/incident_notes/IN-2001-07.html
As an example, I show the proceeding for this case, the other cases of trojan activity can be treated similarly.
The most active scanner was 24.159.128.162 (Charter Communications) with 306 scans of the form

```
07/02-19:49:42.521287 [**] Possible trojan server activity [**]
 24.159.128.162:3601 -> MY.NET.111.52:27374
07/02-19:49:42.532642 [**] Possible trojan server activity [**]
 24.159.128.162:3603 -> MY.NET.111.54:27374
07/02-19:49:43.577797 [**] Possible trojan server activity [**]
 24.159.128.162:3607 -> MY.NET.111.58:27374
```

for the trojan.
Badly enough, some hosts like the following one are responding:

```
07/02-19:49:43.577842 [**] Possible trojan server activity [**]
 MY.NET.111.58:27374 -> 24.159.128.162:3607
```

This machine is possibly infected, as a typical reaction of the SubSeven trojan is to report his presence when triggered by a packet to its port.
A good strategy for trojan detection is therefore to look for machines engaging in two-sided communication with others. This link analysis is not only effective for the SubSeven trojan but can also be applied to other alerts. From processing all trojan alerts with our script and looking for machines exchanging packets in both directions with others, we obtain the following, unfortunately extremely long, list of 614 possibly compromised hosts that need to be checked. In addition to possibly installing the necessary patches, it might be reasonable to block trojan ports on the

firewall for both incoming and outgoing traffic, and perhaps some fine-tune the snort rules in order to reduce the number of false positives.
The list of possibly compromised host in the subnet MY.NET consists of:

99.61, 99.60, 99.53, 99.52, 99.46, 99.42, 99.230, 99.187, 99.174, 99.171, 99.169, 99.165, 99.161, 99.159, 98.121, 97.249, 97.248, 97.246, 97.244, 97.241, 97.240, 97.239, 97.237, 97.230, 97.228, 97.220, 97.212, 97.207, 97.204, 97.182, 97.170, 97.164, 85.62, 75.82, 75.223, 75.220, 75.216, 75.207, 75.196, 75.158, 75.144, 75.135, 75.126, 75.118, 75.114, 75.110, 7.121, 68.11, 68.1, 60.63, 60.59, 60.55, 60.51, 60.39, 60.38, 60.22, 60.14, 60.1, 54.1, 53.98, 53.90, 53.86, 53.75, 53.70, 53.58, 53.54, 53.50, 53.47, 53.43, 53.39, 53.34, 53.226, 53.222, 53.210, 53.206, 53.202, 53.198, 53.174, 53.167, 53.159, 53.155, 53.151, 53.150, 53.147, 53.143, 53.139, 53.135, 53.131, 53.122, 53.114, 53.110, 53.107, 53.103, 53.1, 5.77, 5.65, 5.41, 5.40, 5.38, 5.36, 5.16, 5.13, 5.121, 5.109, 5.104, 5.1, 27.1, 253.10, 232.49, 230.69, 230.61, 230.45, 230.41, 230.37, 230.29, 230.25, 230.173, 230.169, 230.161, 230.153, 230.149, 230.137, 230.117, 230.109, 227.81, 227.77, 227.65, 227.53, 227.193, 227.189, 227.185, 227.181, 227.173, 227.169, 227.137, 227.129, 226.201, 224.97, 224.49, 224.45, 224.41, 224.37, 224.29, 224.185, 224.181, 224.17, 224.133, 224.121, 224.105, 224.101, 222.97, 222.85, 222.81, 222.77, 222.69, 222.45, 222.33, 222.29, 222.25, 222.245, 222.241, 222.229, 222.197, 222.177, 222.173, 222.17, 222.157, 222.129, 222.125, 222.113, 222.105, 221.249, 221.245, 221.241, 221.201, 219.1, 217.98, 217.89, 217.73, 217.69, 217.62, 217.57, 217.50, 217.45, 217.33, 217.141, 217.134, 217.133, 217.130, 217.109, 217.106, 216.97, 216.73, 216.5, 216.33, 216.25, 216.169, 216.165, 216.161, 216.157, 216.149, 216.145, 216.141, 216.129, 216.125, 216.117, 216.113, 215.77, 215.73, 215.69, 215.65, 215.57, 215.49, 215.45, 215.37, 215.249, 215.241, 215.233, 215.221, 215.217, 215.213, 215.205, 215.117, 214.65, 214.57, 214.53, 214.213, 214.201, 214.197, 214.189, 214.185, 214.177, 214.165, 214.153, 214.125, 214.121, 214.105, 213.97, 213.45, 213.41, 213.33, 213.25, 213.241, 213.237, 213.229, 213.221, 213.217, 213.209, 213.193, 213.185, 213.181, 213.161, 213.153, 213.145, 213.137, 213.133, 213.129, 213.113, 213.109, 212.69, 212.41, 212.37, 212.213, 212.165, 212.149, 212.145, 212.137, 212.125, 212.117, 212.113, 212.109, 212.101, 21.9, 21.81, 21.77, 21.5, 21.49, 21.41, 21.32, 21.28, 21.20, 21.16, 21.1, 209.77, 209.73, 209.61, 209.57, 209.41, 209.37, 209.181, 209.173, 209.169, 209.161, 209.133, 209.125, 209.117, 208.53, 208.49, 208.41, 208.133, 208.13, 208.125, 208.121, 208.117, 208.109, 207.253, 207.249, 207.245, 207.241, 207.237, 207.193, 207.185, 207.173, 206.85, 206.69, 206.53, 206.49, 206.33, 206.29, 206.21, 206.17, 206.157, 206.153, 206.149, 206.145, 206.141, 206.129, 206.109, 206.105, 205.97, 205.93, 205.77, 205.65, 205.5, 205.49, 205.41, 205.37, 205.241, 205.233, 205.105, 205.101, 204.205, 204.197, 203.85, 203.81, 203.69, 203.61, 203.57, 203.5, 203.49, 203.41, 203.189, 203.185, 203.173, 203.161, 203.153, 203.145, 203.141, 203.133, 203.13, 203.125, 200.97, 200.35, 200.180, 200.171, 200.169, 200.167, 200.165, 200.163, 200.161, 200.159, 200.157, 200.155, 200.153, 200.151, 200.149, 200.146, 200.144, 200.142, 200.140, 200.138, 200.134, 200.132, 200.130, 2.203, 195.23, 185.66, 184.42, 184.39, 182.93, 182.92, 182.237, 182.136, 182.121, 182.108, 181.37, 181.33, 180.192, 180.185, 18.25, 18.21, 179.86, 179.82, 179.79, 179.78, 179.70, 179.67, 179.54, 165.162, 165.126, 163.99, 163.42, 163.22, 163.107, 163.103, 162.98, 162.83, 162.81, 162.70, 162.51, 162.49, 162.39, 162.35, 162.242, 162.235, 162.231, 162.226, 162.127, 162.123, 162.119, 162.117, 162.108, 162.104, 162.100, 157.5, 157.49, 157.4, 157.241, 157.216, 157.212, 157.204, 157.176, 156.124, 156.121, 153.45, 153.237, 153.209, 153.205, 153.201, 153.197, 153.196, 153.188, 153.177, 153.161, 153.157, 153.145, 153.141, 153.137, 153.125, 153.116, 152.48, 152.248, 152.216, 152.185, 152.181, 152.173, 152.172, 152.17, 152.169, 152.16, 152.157, 152.149, 152.144, 152.109, 151.98, 151.79, 151.75, 151.71, 151.190, 150.98, 150.54, 150.42, 150.38, 150.2, 150.127, 150.119, 15.72, 15.69, 146.21, 146.17, 146.16, 145.91, 145.90, 145.88, 145.85, 145.84, 145.82, 145.53, 145.246, 145.227, 145.223, 145.211, 145.179, 145.178, 145.171, 145.155, 145.154, 145.1, 144.25, 143.238, 143.154, 143.147, 141.229, 140.191, 140.151, 140.143, 140.134, 14.2, 139.69, 139.29, 139.24, 139.229, 139.121, 138.59, 138.51, 138.46, 138.30, 138.14, 138.10, 137.1, 130.86, 130.187, 130.162, 130.127, 130.123, 130.122, 130.11, 121.26, 120.28, 120.1, 116.47, 116.28, 115.95, 115.51, 115.31, 112.25, 112.24, 112.20, 111.91, 111.78, 111.75, 111.69, 111.58, 111.21, 111.195, 111.185, 111.174, 111.173, 111.166, 111.161, 111.152, 111.145, 111.143, 111.139, 111.130, 111.116, 111.114, 111.111, 109.9, 109.17, 109.16, 109.13, 106.69, 106.4, 106.228, 106.200, 106.20, 106.192, 106.188, 106.149, 106.126, 106.1, 105.204, 105.189, 105.185, 105.120, 104.213, 104.208, 104.200, 104.128, 104.117, 104.105, 100.84, 100.73, 100.65, 100.57, 100.56, 100.52, 100.45, 100.32, 100.227, 100.224, 100.214, 100.201, 100.180, 100.172, 100.165, 100.120, 100.1, 10.87, 10.83, 10.79, 10.59, 10.15

*High port 65535 tcp - possible Red Worm - traffic
We see 4977 alerts for the Adore (originally: Red) Worm, not to be confused with Code Red. Information on this Linux Worm can be found at
http://www.sans.org/y2k/adore.htm
The top source for this alert was 192.207.123.2 (Philips Laboratories) with 4918 alerts from 09:59:05 to 10:04:46 on June 29 connecting to MY.NET.99.51 on port 23. My guess would be that someone from the university was at Philips and tried to connect home via telnet.
More suspicious is traffic to port 25 SMTP, as the Adore worm v.02 is known to send emails in order to announce the infection of a machine.

```
06/29-19:30:34.861744 [**] High port 65535 tcp - possible Red Worm -
 traffic [**] MY.NET.253.24:65535 -> 195.121.6.51:25
07/01-19:31:00.295100 [**] High port 65535 tcp - possible Red Worm -
 traffic [**] MY.NET.253.24:65535 -> 18.7.21.83:25
```

SMTP connections are to an address in the Netherlands and MIT.
This is is suspicious, I would investigate this host or at least observe it for more activity of this kind. Furthermore, link analysis shows a two-sided connection between 38.202.60.19 and MY.NET.253.52, this host needs also to be looked at. It could also be worth to block port 65535 at the firewall.

*Output from the portscan module will be treated together with the scan logs

*External calls to portmapper tcp/udp 111 should generally be blocked at the firewall, because of a number of well documented vulnerabilities associated to RPC services, see e.g.
http://www.cert.org/incident_notes/IN-2001-01.html
http://www.cert.org/advisories/CA-2001-05.html
http://www.cert.org/advisories/CA-2000-17.html
As of the date of this report, RPC weaknesses are the number 3 among the SANS Top Ten Most Critical Securit Threats (http://www.sans.org/topten.htm).
Blocking port 111 has been recommended by SANS, see
http://www.sans.org/newlook/resources/IDFAQ/blocking.htm
The top scanner with 433 alerts on July 1 is 211.23.6.234, the company CHTD, Chunghwa Telecom Co.,Ltd. from Taiwan.
This server tried also a STATDX UDP attack on MY.NET.6.15 has been reported of attempted connection to port 515, see
http://www.incidents.org/archives/intrusions/msg00960.html
Such a behavior could be a good reason to put the machine on a black list, denying at the firewall by default every connection from it.

*connect to 515 from outside and inside
The BSD lpd printing service called LPRng using printer port tcp 515 has a well known recent vulnerability
http://www.kb.cert.org/vuls/id/382365
which is exploited for example by the Red Hat Linux Ramen worm.
A documentation on Ramen together with instructions for detection and removal can be found at
http://www.sans.org/y2k/ramen.htm
The most suspicious traffic of this kind were three alerts of the form

```
alerts0629_0703:06/29-01:41:54.127208  [**] connect to 515 from outside [**]
255.255.255.255:31337 -> MY.NET.134.110:515
```

and 9 packets to broadcast addresses like the following

```
alerts0629_0703:06/30-10:39:03.237615  [**] connect to 515 from outside [**]
150.183.110.179:1140 -> MY.NET.134.0:515
```

The first thing to say is that the firewall should block spoofed source addresses like 255.255.255.255. Secondly, it would be advisable to block also broadcast attempts from the outside.
Furthermore, it is again possible to shun the most persistent scanners

like 150.183.110.179 (Korea Institute of Science and Technology).
The following systems on MY.NET tried to connect to external 515 ports and should therefore be investigated on the presence of the worm:
100.234, 179.78, and 219.42.

*Watchlist 000220 IL-ISDNNET-990517 and Watchlist 000222 NET-NCFC
These are alerts on connections from hosts infamous for suspicious activities, the first list contains addresses from Israel, the second one addresses belonging to the
Computer Network Center Chinese Academy of Sciences.
Similar traffic has been observed by Bakos and Zeltser, see
http://ouah.bsdjeunz.org/George_Bakos.html
http://www.zeltser.com/sans/idic-practical/
Among the ports observed in the communications was the most prominent TCP port 1214. Traffic of this kind was for example seen at
http://www.incidents.org/archives/intrusions/msg00527.html
An interpretation of this traffic as being associated to the "KaZaA" filesharing program has been given at
http://www.incidents.org/archives/intrusions/msg00530.html

*SMB Name Wildcard
The port 137 is used by NetBIOS nbtstat. In order to avoid network reconnaissance, or exploits like the 911 worm
(http://www.cert.org/incident_notes/IN-2000-03.html),
it should be blocked by the firewall.

*Queso fingerprint
Generally, systems trying a fingerprint reconnaissance should be put on a watchlist and further connections should be logged and investigated.
A typical queso fingerprint can be seen at
http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids29&view=research
However, the alert rule on these fingerprints in the file RULES.SAMPLE of the snort software

```
alert tcp any any -> 192.168.1.0/24 any (msg:"Queso fingerprint";flags: S12;)
```

states quite general characteristics, namely the TCP flags: S12.
The number of false positives can be quite high, see
http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids29&view=event
The top host triggering these alerts is 199.183.24.194 (Red Hat Software).
These strange alerts have the form

```
06/29-03:17:13.082973 [**] Queso fingerprint [**] 199.183.24.194:49498 ->
 MY.NET.253.42:25
06/29-03:53:52.142546 [**] Queso fingerprint [**] 199.183.24.194:58557 ->
 MY.NET.253.42:25
06/29-03:56:49.232870 [**] Queso fingerprint [**] 199.183.24.194:60028 ->
 MY.NET.253.41:25
06/29-05:10:09.454394 [**] Queso fingerprint [**] 199.183.24.194:51954 ->
 MY.NET.253.43:25
06/29-05:19:39.838918 [**] Queso fingerprint [**] 199.183.24.194:55902 ->
 MY.NET.253.43:25
```

(... not all traces shown, 138 alerts, 3 destination hosts.)
This is not exactly a scan as only three destinations are involved.
Below, I inspect the OOS data in order to decide whether this activity is malicious. My conclusion will be that the alerts above are false positives.

*SYN-FIN scan!
These scans were considered to be stealthy, as some IDS seemed to get confused by the SF flags set together. Today, this isn't true anymore.
Scans will be discussed thoroughly when the scan logs are investigated.

*WinGate 1080 Attempt
WinGate is a proxy that could be abused to hide the identity of an attacker, so they keep looking for it. Further information on this kind of probe can be found for example at
http://www.whitehats.com/info/IDS175
Firewalls that are able to distinguish between connections initiated from the

inside and connections initiated from the outside can be configured to block reconnaissance traffic from the outside to the proxy.

*Port 55850 tcp - Possible myserver activity - ref. 010313-1
Due to the reference 010313, I examined the alert file of March 13. However, TCP port 55850 didn't occur in this file. Unfortunately, the corresponding OOS data for this day was no longer available for inspection. By research on the internet, I found the following page
http://archives.neohapsis.com/archives/incidents/2000-10/0136.html
about the MyServer DDoS Agent at UDP (not TCP!) port 55850.
In essence, I am confused by this snort alert need more information to handle these detect. I see three two-sided communications involving hosts of my network, this would be my point to start, once that I have confirmed that this alert makes sense. The concerned hosts are MY.NET.217.154, MY.NET.253.52, and MY.NET.5.29.

*NMAP TCP ping!
NMAP is a stealthy OS fingerprinting and reconnaissance tool that is documented at
http://www.insecure.org/nmap/
We see several reconnaissance pings, the most active scanner is 207.238.101.253 (Business Internet, Inc.) doing a scan with 30 detects over the whole period of five days investigated. Further connections from scanners like this one should be monitored for further correlations and attempts of exploits.

*Null scan
Another stealthy scan using crafted packets with no TCP flags set.
Most prominent is a slow scan from 213.66.109.65 (Telia Network services ISP) directed at MY.NET.217.130, a server reported already above of suspicious trojan activity associated to port 27374.
From the traces below, it appears as if the traces could be also related to gnutella file sharing (port 6346). In essence, this host needs definitely to be looked at.

```
07/01-08:20:41.428158 [**] Null scan! [**] 213.66.109.65:3400 ->
 MY.NET.217.130:6346
07/01-17:16:44.410608 [**] Null scan! [**] 213.66.109.65:4065 ->
 MY.NET.217.130:6346
```

(not all traces printed...)
Traces like the above are posted at
http://www.sans.org/y2k/052000.htm
more information about gnutella is given at
http://www.sans.org/y2k/gnutella.htm

*TCP SRC and DST outside network
These traces seem mostly to be related to gnutella traffic.
We see 37 instances of the form

```
07/03-19:45:33.226183 [**] TCP SRC and DST outside network [**]
 24.180.140.132:6346 -> 213.122.166.185:21979
```

It is well known that gnutella clients can spoof their source address, see http://www.sans.org/y2k/gnutella.htm
It is likely, that we see an example of this kind of traffic here.
By filtering outgoing traffic from MY.NET and dropping all connections that do not have a valid source address within MY.NET, we should be able to make this kind of traffic disappear. Also, snort rules could perhaps be adjusted to ignore this kind of traffic.

*High port 65535 udp - possible Red Worm - traffic
The Red Worm has been explained above,
Link analysis shows the host MY.NET.70.242 involved in two-sided communications with 207.65.152.69 and 195.179.0.30. It should be checked for possible compromise.

*Attempted Sun RPC high port access
As stated in the SANS ID FAQ,
http://www.sans.org/newlook/resources/IDFAQ/blocking.htm

Solaris 2.x portmapper listens not only on TCP port 111, and UDP port 111, but also on a port greater than 32770. Thus, the vulnerabilities exploitable for port 111 described above could also exploited via the port 32771 registered in our locks. One solution is to filter for such ports on the firewall, another is the installation of a secure portmapper. Refer to the cited SANS page for details.

*SUNRPC highport access!

In the alert files, 33 SUNRPC highport accesses are logged, to the hosts MY.NET.217.6, MY.NET.60.39, and MY.NET.1.6. Each of them could be possibly compromised and needs to be checked. In particular, MY.NET.60.39 is associated to 6 different alert signatures
(8 Attempted Sun RPC high port access, 5 WinGate 1080 Attempt,
3 Queso fingerprint, 3 Back Orifice, 1 SUNRPC highport access,
1 Possible trojan server activity) thus seeming to be a valuable target.

*Russia Dynamo - SANS Flash 28-jul-00

As indicated by the title, traffic of this kind has first occurred last year and refers to trojan-compromised Windows machines sending large amounts of data to a Russian IP address (194.87.6.X). The original flash is available at
http://archives.neohapsis.com/archives/sans/2000/0068.html
By now, machines should be patched against this trojan.
However, we see two-sided traffic between MY.NET.104.111 and 194.87.6.201 and also between MY.NET.182.120 and 194.87.6.229. Therefore, I recommend checking MY.NET.104.111 and MY.NET.182.120.

*Back Orifice

Port 31337 ("ELEET") is quite an infamous port for back doors, used by, among others, the trojan Back Orifice.
This trojan is well known and documented, see for example
http://www.networkice.com/advice/Phauna/RATs/Back_Orifice/default.htm
One of the most popular techniques is to use an infected computer for reconnaissance on others ("Bouncing"), see e.g.
http://www.networkice.com/advice/Underground/Hacking/Methods/Technical/Bounce/default.htm
Looking at the traces on our network, this appears to be the case
The host 207.41.14.11 (US Courts, NETBLK-SPRINT-CF290E-1) is running several probes clearly targeted against one of our hosts, MY.NET.60.39.
We show only selected ones:

```
07/03-14:26:06.894016 [**] SUNRPC highport access! [**] 207.41.14.11:40809
 -> MY.NET.60.39:32771
07/03-14:28:17.630340 [**] Attempted Sun RPC high port access [**]
 207.41.14.11:1755 -> MY.NET.60.39:32771
07/03-14:31:32.863038 [**] Back Orifice [**] 207.41.14.11:1765 ->
 MY.NET.60.39:31337
07/03-14:31:59.231822 [**] Back Orifice [**] 207.41.14.11:1755 ->
 MY.NET.60.39:31337
07/03-14:32:10.369533 [**] Attempted Sun RPC high port access [**]
 207.41.14.11:1765 -> MY.NET.60.39:32771
07/03-15:00:12.364831 [**] WinGate 1080 Attempt [**] 207.41.14.11:48644 ->
 MY.NET.60.38:1080
07/03-15:01:42.643866 [**] Back Orifice [**] 207.41.14.11:3697 ->
 MY.NET.60.39:31337
```

Apart from observing carefully MY.NET.60.39 for possible infections, I would contact the responsible for 207.41.14.11 and ask him to check his computer.

*STATDX UDP attack

A well known input validation problem in rpc.statd is documented by CERT, see
http://www.cert.org/advisories/CA-2000-17.html
(CVE entries: CVE-1999-0018, CVE-1999-0019)
Two attackers try to exploit this on MY.NET.6.15, see the logs below and the remarks on the external calls to portmapper tcp/udp 111 above.

```
07/01-09:00:37.454441 [**] STATDX UDP attack [**] 211.23.6.234:835 ->
 MY.NET.6.15:32776
06/30-12:17:02.627140 [**] External RPC call [**] 210.90.168.5:3217 ->
```

MY.NET.6.15:111
06/30-12:17:02.869023 [**] STATDX UDP attack [**] 210.90.168.5:836 ->
MY.NET.6.15:32776
MY.NET.6.15 needs to be checked for a possible compromise.

*TCP SMTP Source Port traffic
The following packet triggered the alert because TCP port 25 is usually only used for mailing.
07/03-11:16:43.255384 [**] TCP SMTP Source Port traffic [**]
207.88.135.158:25 -> MY.NET.5.73:807
The destination port 807 is clearly suspicious, however, I have no explanation for this packet. I would suggest to contact someone at 207.88.135.158 (Concentric Network Corporation), in particular because only processes running as root can be configured to use ports below 1024.

*ICMP SRC and DST outside network
Neither IP source nor destination address of the following ICMP packet belong to MY.NET.
06/29-22:52:03.053264 [**] ICMP SRC and DST outside network [**]
172.128.115.252 -> 24.0.217.198
The question is, why they are seen on MY.NET. One explanation is that someone on MY.NET is crafting ICMP packets. This could be prevented by blocking outgoing traffic not originating from MY.NET.

*Tiny Fragments - Possible Hostile Activity
Tiny fragments are a clear indication of packet craft and can be used to crash IP stacks or to circumvent payload inspection by firewalls that don't reassemble IP datagrams before inspection. We see only one packet on our network.
06/30-15:03:17.324734 [**] Tiny Fragments - Possible Hostile Activity [**]
202.39.78.125 -> MY.NET.98.144
A reasonable threshold for triggering this kind of alert together with further alerts of this kind is given at
http://archives.neohapsis.com/archives/snort/2000-05/0103.html
Snort has a preprocessor "defrag" that handles fragmented IP datagrams, see
http://www.snort.org/docs/faq.html#1.5
It should be installed.

2. Scan logs prioritized by number of occurrences

192203	SYN **S*****
150324	UDP
301	SYN (non-standard flags)
275	SYNFIN (SF flags)
93	INVALIDACK (ACK set, not 'normal', no SPAU or FULLXMAS)
84	NOACK (A flag is missing)
70	NULL (none of SFRPAU)
33	UNKNOWN (unusual combinations of flags, see spp_portscan.c)
26	VECNA (one of the following combinations: P, U, PU, FP, FU)
10	XMAS (FPU flags)
6	FULLXMAS (SFRPAU flags)
5	FIN (F flag)
2	SPAU (SPAU flags)
2	NMAPID (SFPU flags)

Snort goes through several steps in order to classify a scan, this process can be seen from the inspection of the file 'spp_portscan.c' in the snort distribution (available from http://www.snort.org/).
Standard TCP SYN **S***** scans and UDP scans are by far the most common types of scans. All other scans are TCP scans with particular combinations of flag bits set. They are reported by the spp_portscan module as stealth scans.
We don't list all encountered combinations of flag bits,
but certainly a lot of the 256 possible pattern were actually seen.
Special combinations of TCP flags can be used for fingerprinting,
or to circumvent detection by an IDS,
therefore, special attention should be payed to them. On the other hand, it is

dangerous to simply ignore standard SYN **S***** scans just because they are not considered stealthy. With everybody paying attention to special TCP flags and slow scans, it appears almost as the safest bet to launch a quick SYN **S***** scan and getting ignored because of the overwhelming amount of traffic.

At first, we list the top ten attackers and the top ten targets. Afterwards, we give the same information for those attacks using TCP fingerprinting. Finally, we list the most popular UDP/TCP address ports, and we show also the most popular TCP destination ports for stealth scans.

Top Ten Scan Targets		Top Ten Attackers	
23505	MY.NET.219.42	49655	MY.NET.160.114
5068	MY.NET.110.33	30156	211.207.15.190
4839	MY.NET.180.76	23501	66.68.62.229
4828	MY.NET.71.248	19508	217.81.194.157
4700	MY.NET.178.222	15814	205.188.233.121
4366	MY.NET.108.13	14921	205.188.233.153
4007	MY.NET.145.166	13110	148.223.228.15
3850	MY.NET.145.197	12087	61.222.34.170
3785	MY.NET.60.39 (5 fps)	10867	207.236.81.82
3496	MY.NET.106.184	9583	205.188.244.249

Top Ten Fingerprinted Targets		Top Ten Fingerprinting Attackers	
112	MY.NET.70.97 fps	269	211.180.236.194 fps
84	MY.NET.70.149 fps	138	199.183.24.194 fps
54	MY.NET.253.41 fps	75	24.66.152.186 fps
45	MY.NET.253.42 fps	27	193.226.113.248 fps
44	MY.NET.253.43 fps	11	66.41.18.185 fps
37	MY.NET.150.225 fps	9	213.66.109.65 fps
18	MY.NET.5.29 fps	9	24.169.190.158 fps
15	MY.NET.217.130 fps	7	148.63.17.58 fps
15	MY.NET.219.50 fps	7	64.198.133.235 fps
14	MY.NET.150.133 fps	6	217.0.71.249 fps

Top Ten TCP ports		Top Ten FPrinted TCP Ports			Top Ten UDP ports	
92370	21	269	111	(269 fps)	58845	6970
38227	53	146	25	(146 fps)	38150	27005
10463	1214 (105 fps)	105	1214	(105 fps)	11615	6112
5914	27374	76	6346	(76 fps)	7070	7778
5199	47017	19	0	(19 fps)	2454	7000
2904	25 (146 fps)	12	21536	(12 fps)	1768	7003
2027	6346 (76 fps)	9	20	(9 fps)	1607	53
1872	111 (269 fps)	4	143	(4 fps)	849	21077
1634	515	4	3456	(4 fps)	592	27243
299	6347 (2 fps)	4	113	(4 fps)	385	4665

Popular scan targets give an idea which systems might get next the attention of attackers, this is particularly true for those subject to stealthy scans. Active attackers can be put on a watchlist or completely blocked by a firewall.
Popular destination ports indicate which sorts of exploits an attacker wants to use, this makes it easier to look for them. Generally, we see a lot of scans to the ports that also appeared in our alert logs and have been analyzed above. If a port suddenly gets popular without the corresponding vulnerability being published e.g. to bugtraq, it is perhaps advisable to block it at the firewall. TCP port 1214 has a lot of scans associated to it, and I am not completely convinced that all of this is related to the KaZaA filesharing program, as was given as a possible explanation in the analysis of the watchlist detects above.
Looking at the top attackers, we see MY.NET.160.114 sending UDP packets from UDP port 777. The corresponding TCP port 777 is associated to the "Undetected" trojan, and I would strongly advise to check this system for a possible compromise. If we apply link analysis to the scan logs, we see MY.NET.98.144 in two-sided communication with 15 other hosts, where 11

are from 62.27.42.*. I would suppose that some kind of server is running on MY.NET.98.144 and check it for possible backdoors. The other host involved in heavy two-sided communication with 66.68.62.229 (Roadrunner Southwest) is MY.NET.219.42. Looking at the scan logs, I see on July 1 the host MY.NET.219.42 starting an exhaustive port scan on 66.68.62.229

```
Jul  1 12:39:15 MY.NET.219.42:4879 -> 66.68.62.229:569 SYN **S*****
Jul  1 12:39:15 MY.NET.219.42:4884 -> 66.68.62.229:1379 SYN **S*****
Jul  1 12:39:15 MY.NET.219.42:4885 -> 66.68.62.229:438 SYN **S*****
Jul  1 12:39:15 MY.NET.219.42:4887 -> 66.68.62.229:426 SYN **S*****
Jul  1 12:39:15 MY.NET.219.42:4888 -> 66.68.62.229:578 SYN **S*****
...(not all traces shown, 1170 packets)
```

Nine hours later, the revenge comes promptly from 66.68.62.229 scanning back

```
Jul  1 21:20:05 66.68.62.229:63030 -> MY.NET.219.42:1 SYN **S*****
Jul  1 21:20:05 66.68.62.229:63031 -> MY.NET.219.42:2 SYN **S*****
Jul  1 21:20:05 66.68.62.229:63032 -> MY.NET.219.42:3 SYN **S*****
Jul  1 21:20:05 66.68.62.229:63033 -> MY.NET.219.42:4 SYN **S*****
Jul  1 21:20:05 66.68.62.229:63034 -> MY.NET.219.42:5 SYN **S*****
...(not all traces shown, 23501 packets)
```

I would check who was working on MY.NET.129.42, and also check for a backdoor as the same host was also subject to "connect to 515 from outside and inside" alerts.

When I analyzed scan and alert logs together in order to find hidden correlations, I realized that this was not necessary as all scanning activity had been reliably recorded as summarizing alerts by the spp_portscan module. This is a clear indication that the different sensors 'see' the same packet flow, and that the scan logs give the details to the spp_portscan alerts.

3.Inspection of OOS data for suspicious activity

The OOS logs contain the full packet headers of selected events that triggered alerts. For this reason, they can be used in some cases to distinguish between false and true positives.
As an example, I investigate the packets coming from Red Hat triggering the Queso fingerprint alerts.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/29-01:26:46.348740 199.183.24.194:60628 -> MY.NET.253.41:25
TCP TTL:54 TOS:0x0 ID:61718  DF
21S***** Seq: 0x53A51F0B   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 327609885 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/29-01:31:49.590043 199.183.24.194:35578 -> MY.NET.253.42:25
TCP TTL:54 TOS:0x0 ID:26439  DF
21S***** Seq: 0x669E03CB   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 327640204 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/29-01:48:51.093254 199.183.24.194:39612 -> MY.NET.253.41:25
TCP TTL:54 TOS:0x0 ID:42772  DF
21S***** Seq: 0xA77C5491   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 327742341 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/29-03:15:27.976400 199.183.24.194:49498 -> MY.NET.253.42:25
TCP TTL:54 TOS:0x0 ID:4056  DF
21S***** Seq: 0xEE7BEB0A   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 328262866 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/29-03:32:02.403299 199.183.24.194:53712 -> MY.NET.253.43:25
TCP TTL:54 TOS:0x0 ID:60432  DF
21S***** Seq: 0x2D5F8528   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 328362296 0 EOL EOL EOL EOL
```

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/29-03:52:07.265564 199.183.24.194:58557 -> MY.NET.253.42:25
TCP TTL:54 TOS:0x0 ID:23100  DF
21S***** Seq: 0x79A165F3   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 328482767 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
(more logs of this kind)
```

Investigating these packets, the only non-common thing I can see are the TCP flags 21S*****. However, this is not necessarily an indication for packet craft.
Looking at the TCP destination port 25 (SMTP) and the time of the packets, I would suppose that we see here actually Red Hat mailing list activity with mails being sent to the mail servers MY.NET.253.41, MY.NET.253.42, and MY.NET.253.43.
Doing research on the web, I found the SANS writeup on ECN http://www.incidents.org/detect/ecn.php.
With this as a background, I classify the alerts as false positives.
I would ask to tune the snort rules appropriately such that the number of false positives is reduced. A possible modification of the "Queso fingerprint" signature is given at
http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids29&view=signatures
There it is suggested to take also the TTL field into account:

```
alert TCP $EXTERNAL any -> $INTERNAL any
(msg:"IDS29/scan_probe-Queso Fingerprint attempt"; ttl:>225; flags: S12;)
```

4. Summary
In the previous sections, I have given an analysis of the incidents logged during the period of five days from June 29 to July 3, 2001, at the University of Maryland, Baltimore County.

The number of severe incidents was very high and to a large part related to trojan and worm activity. I have identified a large number of possibly compromised host that need immediately to be investigated.
Furthermore, I found that the firewall rule set needs to be adjusted such that some possibly vulnerable services are protected from external access.
I also indicated several IP source addresses, including those which are obviously spoofed, which should be generally denied at the firewall.
Perhaps, a firewall policy "Deny everything that is not explicitly allowed" should be enforced.
However, I also found a large number of false alarms given by the IDS.
Clearly, the snort rule set has to be tuned and I have given some suggestions during the previous sections. By following my recommendations, it is possible to enhance substantially the security of MY.NET and, as a positive side effect, to reduce the number of alerts given by the IDS.

References (for assignment 2 and 3)
----

http://advice.networkice.com/advice/Exploits/Ports/default.htm
http://archives.neohapsis.com/archives/incidents/2000-10/0136.html
http://archives.neohapsis.com/archives/sans/2000/0068.html
http://archives.neohapsis.com/archives/snort/2000-05/0103.html
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0666
http://ouah.bsdjeunz.org/George_Bakos.html
http://security-archive.merton.ox.ac.uk/bugtraq-200007/0209.html
http://userpages.umbc.edu/~robin/Presentations/Snort/Snort_FINAL.ppt
http://www.8thlayer.org/unicode.html
http://www.arin.net/whois/index.html
http://www.cert.org/advisories/CA-2000-17.html
http://www.cert.org/advisories/CA-2001-05.html
http://www.cert.org/incident_notes/IN-2000-03.html),
http://www.cert.org/incident_notes/IN-2001-01.html
http://www.cert.org/incident_notes/IN-2001-07.html

http://www.cve.mitre.org/cve/
http://www.faqs.org/faqs/computer-security/most-common-qs/section-18.html
http://www.geocrawler.com/archives/3/165/1999/9/0/2672764/
http://www.google.com
http://www.hsc.fr/ressources/outils/idswakeup/
http://www.iana.org/assignments/port-numbers
http://www.incidents.org/archives/intrusions/msg00527.html
http://www.incidents.org/archives/intrusions/msg00530.html
http://www.incidents.org/archives/intrusions/msg00540.html
http://www.incidents.org/archives/intrusions/msg00698.html
http://www.incidents.org/archives/intrusions/msg00713.html
http://www.incidents.org/archives/intrusions/msg00720.html
http://www.incidents.org/archives/intrusions/msg00747.html
http://www.incidents.org/archives/intrusions/msg00817.html
http://www.incidents.org/archives/intrusions/msg00960.html
http://www.incidents.org/archives/intrusions/msg00980.html
http://www.incidents.org/archives/intrusions/msg01017.html
http://www.incidents.org/archives/intrusions/msg01020.html
http://www.incidents.org/archives/intrusions/msg01060.html
http://www.incidents.org/archives/intrusions/msg01210.html
http://www.incidents.org/archives/intrusions/msg01568.html
http://www.incidents.org/archives/intrusions/msg01621.html
http://www.incidents.org/detect/ecn.php.
http://www.incidents.org/react/unicode.php
http://www.insecure.org/nmap/
http://www.kb.cert.org/vuls/id/382365
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms00-057.asp
http://www.networkice.com/advice/Phauna/RATs/Back_Orifice/default.htm
http://www.networkice.com/advice/Underground/Hacking/Methods/Technical/Bounce/default.htm

http://www.research.umbc.edu/~andy/
http://www.rvglug.org/pipermail/rvglug/2001-February/000436.html
http://www.sans.org/giactc/gcia.htm
http://www.sans.org/newlook/resources/IDFAQ/blocking.htm
http://www.sans.org/topten.htm
http://www.sans.org/topten.htm).
http://www.sans.org/y2k/052000.htm
http://www.sans.org/y2k/adore.htm
http://www.sans.org/y2k/gnutella.htm
http://www.sans.org/y2k/ramen.htm
http://www.satx.rr.com/support/security/computer_ports.html
http://www.securityfocus.com/archive/1/140091
http://www.silicondefense.com/software/snortsnarf/index.htm)
http://www.snort.org/).
http://www.snort.org/docs/faq.html#1.5
http://www.sys-security.com/archive/papers/ICMP_Usage_v3.0.pdf
http://www.unicode.org
http://www.victim.com/../../winnt/system32/cmd.exe
http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids29&view=event
http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids29&view=research
http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids29&view=signatures
http://www.whitehats.com/info/IDS175
http://www.zeltser.com/sans/idic-practical/
http://xforce.iss.net/alerts/advise68.php

Appendix 1 : Samples from the data files for assignment 3:
--

alert.010629:

```
*********************************************************
                Snort Alert Report at Sat Jun 30 00:06:59 2001
*********************************************************
```

```
06/29-00:15:17.926017  [**] UDP SRC and DST outside network [**] 134.192.131.107:137 ->
134.192.1.5:53
06/29-00:15:19.429287  [**] UDP SRC and DST outside network [**] 134.192.131.107:137 ->
134.192.1.5:53

scans.010629:
*********************************************************
                Snort Scan Report at Sat Jun 30 00:11:04 2001
*********************************************************
Jun 29 00:15:58 151.21.227.249:2441 -> MY.NET.133.215:111 SYN **S*****
Jun 29 00:15:58 151.21.227.249:2445 -> MY.NET.133.219:111 SYN **S*****
Jun 29 00:16:00 151.21.227.249:2166 -> MY.NET.133.21:111 SYN **S*****

oos_Jun.29.2001:
*********************************************************
                oos_Jun.29.2001
*********************************************************
Initializing Network Interface ep0
snaplen = 68
Entering readback mode....
06/29-00:04:09.908220 216.211.22.92:32983 -> MY.NET.219.50:1848
TCP TTL:14 TOS:0x0 ID:7936  DF
**SFRPAU Seq: 0x6972664F   Ack: 0x9E1B5EA0   Win: 0xB2D5
TCP Options => NOP NOP TS: 511509506 1852142215

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/29-00:12:39.667322 65.0.7.50:33297 -> MY.NET.253.125:80
TCP TTL:49 TOS:0x0 ID:0  DF
21S***** Seq: 0x43AD9D61   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 19731182 0 EOL EOL EOL EOL

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

Appendix 2: A script for snort log file parsing: process_scanalert.pl

```
------------------------------------------------------------------------
#!/usr/local/bin/perl -ws
#
# use: process_scanalert.pl -v -l=1 samplescan.data  > analysis.output
#
unless
(defined($c)||defined($d)||defined($o)||defined($p)||defined($s)||defined($t)||defined($u)
||defined($v)){
            print "use the following action flags\n";
     print "\t-c \tprint the communicating hosts\n";
            print "\t-d \tprint the target hosts\n";
            print "\t-f \twatch for fingerprinting attempts\n";
            print "\t-o \tprint the attacked/scanned ports\n";
            print "\t-p \tprint the attacker/target pair\n";
            print "\t-s \tprint the attacking hosts\n";
            print "\t-t \tprint the attack type\n";
     print "\t-u \tprint summary from spp_portscan\n";
            print "\t-v \tbe verbose and print everything\n";
     print "----";
            print "\t-l=n\tconnection threshold before printing\n";
            exit 1;
            }
if(defined($l)){$thresh=$l;}else{$thresh=0;}
$vv=0; #set $vv=1 for very verbose output for script debugging

#############################
# Data parsing and analysis #
#############################
while (<>){
  #format of scan.YYMMDD
```

```
  if($_ =~
/^([a-zA-Z]{3}\s+\d{1,2})\s([0-9\:]{8})\s([MYNET0-9\.]+)\:([0-9\w]+)\s\-\>\s([MYNET0-9\.]+
)\:([0-9\w]+)\s(\w+[\w\s\*]*)/){
    chomp;
    $date=$1;$time=$2;$src_addr=$3;$src_port=$4;
    $dst_addr=$5;$dst_port=$6;$scantype=$7;
    $pkey = "$src_addr-$dst_addr";
    $pkeyl = "$src_addr:$src_port-$dst_addr:$dst_port";
    ++$asrc{$src_addr}; #count the same source address
    ++$adst{$dst_addr}; #count the same dest address
    if($scantype =~ /UDP/){
      ++$pdst_udp{$dst_port};} #count the same dest port
    else{++$pdst_tcp{$dst_port};}
    ++$type{$scantype}; #count the same scan type
    ++$pair{$pkey};     #count the same connection direction (IP only)
    ++$pairl{$pkeyl};   #count the same connection direction (IP:port)
    #look for OS fingerprint scans
    unless (($scantype =~ /SYN\s\*\*S\*\*\*\*\*/)||($scantype =~ /UDP/)||($scantype =~
/FIN\s\*\*\*F\*\*\*\*/)){
      ++$fsrc{$src_addr};
      ++$fdst{$dst_addr};
      if($scantype !~ /UDP/){++$fpdst_tcp{$dst_port};}
      ++$fpr{$pkey};
      ++$ftyp{$scantype};}
  }
  #format of alert.YYMMDD
  elsif($_ =~ /^[0-9\/]{5}\-[0-9\:\.]{15}\s+\[\*\*\]/){
    #look for portscans
    if($_ =~ /portscan/i){
      #we investigate only the summary of the portscans
      if($_ =~
/^([0-9\/]{5})\-([0-9\:\.]{15})\s+\[\*\*\]\s+spp_portscan\:\sEnd\sof\sportscan\sfrom\s([MY
NET0-9\.]+)\s+\(TOTAL\sHOSTS\:([0-9]*)\s+TCP\:([0-9]*)\s+UDP\:([0-9]*)\)\s+\[\*\*\]/){
 $date=$1;$time=$2;$portscan_dst=$3;
 $nmb_portscanners=$4;$nmb_scanTCPpacks=$5;$nmb_scanUDPpacks=$6;
 ++$type{"Spp_portscan detect"};
 $nmb_portscan{$portscan_dst}+=$nmb_portscanners;
 if($nmb_scanTCPpacks != 0){
   $portscan_tcp{$portscan_dst}+=$nmb_scanTCPpacks;}
 if($nmb_scanUDPpacks != 0){
   $portscan_udp{$portscan_dst}+=$nmb_scanUDPpacks;}
      }
      else{if($vv){print"Ignore intermediate output from spp_portscan $_ \n";}}
    }
    #parse 'normal' alerts
    else{ #need (.*?) stingy instead of greedy matching here...
      if($_ =~
/^([0-9\/]{5})\-([0-9\:\.]{15})\s+\[\*\*\]\s+(.*?)\s+\[\*\*\]\s([MYNET0-9\.]+)\:([0-9\w]+)
\s\-\>\s([MYNET0-9\.]+)\:([0-9\w]+)/){
 $date=$1;$time=$2;$attacktype=$3;$src_addr=$4;$src_port=$5;
 $dst_addr=$6;$dst_port=$7;
 $pkey = "$src_addr-$dst_addr";
 $pkeyl = "$src_addr:$src_port-$dst_addr:$dst_port";
 ++$asrc{$src_addr};
 ++$adst{$dst_addr};
 ++$pdst{$dst_port}; #cannot distinguish udp/tcp here easily
 if((! defined($portlist{$attacktype}))||
          ($portlist{$attacktype} !~ /$dst_port/)){
   $portlist{$attacktype} .= " $dst_port";}
 ++$type{$attacktype};
 ++$pair{$pkey};     #count the same connection direction (IP only)
 ++$pairl{$pkeyl};}   #count the same connection direction (IP:port)
      elsif($_ =~
/^([0-9\/]{5})\-([0-9\:\.]{15})\s+\[\*\*\]\s+(.*?)\s+\[\*\*\]\s([MYNET0-9\.]+)\s+\-\>\s([M
YNET0-9\.]+)/){
 #signatures that don't involve ports, e.g. ICMP traffic
```

```
 $date=$1;$time=$2;$attacktype=$3;$src_addr=$4;$dst_addr=$5;
 $pkey = "$src_addr-$dst_addr";
 ++$asrc{$src_addr};
 ++$adst{$dst_addr};
 ++$type{$attacktype};
 ++$pair{$pkey};
      }
      else{
 if($v){
   print"Ignore line $_ \n";}}
    }
  }
  else{print "!!! Ignore unknown data: $_ \n";}
}

############################
# Print various statistics #
############################
#Print different scan signatures
if (($t)||($v)&&(defined(%type))){
  print "\n\Attack/Scan Types\n=================\n\n" if ($v) ;
  foreach $key (sort { $type{$b} <=> $type{$a} }keys(%type)){
    if($v){ if ($key =~ /\s*(.*?)\s+/){ $totaltype{$1}+=$type{$key};}}
    if((($f)||($v))&&defined($ftyp{$key})){ $fp="\t\($ftyp{$key} fps\)";}
    else{ $fp=""; }
    print "$type{$key} \t$key $fp\n" if($type{$key} >= $thresh);
  }
  print "\n\Total Scan Types\n================\n\n" if ($v) ;
  foreach $key (sort { $totaltype{$b} <=> $totaltype{$a} }keys(%totaltype)){
    print "$totaltype{$key} \t$key \n" if($totaltype{$key} >= $thresh);
  }
}
#Print target hosts
if (($d)||($v)&&(defined(%adst))){
  print "\n\nTargets\n=======\n\n" if ($v) ;
  foreach $key (sort { $adst{$b} <=> $adst{$a} } keys(%adst)){
    if((($f)||($v))&&($fdst{$key})){$fp="\t($fdst{$key} fps)";}else{$fp="";}
    if(($key =~/\.255\s*$/)||($key =~/\.0\s*$/)){ $BCAST=" BCAST ";}else{ $BCAST="";}
    if($key =~/^\s*(224|225|226|227|228|229|230|231|232|233|234|235|236|237|238|239)\./){
$MCAST=" MCAST ";}else{ $MCAST="";}
    print "$adst{$key} \t$key $fp $MCAST $BCAST\n" if($adst{$key} >= $thresh);
  }
}
if (($f)||($v)&&(defined(%fdst))){
  print "\n\nFingerprinted targets\n=====================\n\n" if ($v) ;
  foreach $key (sort { $fdst{$b} <=> $fdst{$a} } keys(%fdst)){
    $fp="fps";
    if(($key =~/\.255\s*$/)||($key =~/\.0\s*$/)){ $BCAST=" BCAST ";}else{ $BCAST="";}
    if($key =~/^\s*(224|225|226|227|228|229|230|231|232|233|234|235|236|237|238|239)\./){
$MCAST=" MCAST ";}else{ $MCAST="";}
    print "$fdst{$key} \t$key $fp $MCAST $BCAST\n" if($adst{$key} >= $thresh);
  }
}
#Print attacking hosts
if (($s)||($v)&&(defined(%asrc))){
  print "\n\nAttackers by frequency\n======================\n\n" if ($v) ;
  foreach $key (sort {$asrc{$b} <=> $asrc{$a}} keys(%asrc)){
    if((($f)||($v))&&($fsrc{$key})){$fp="\t($fsrc{$key} fps)";}else{$fp="";}
    if(($key =~/^\s*10\./)||($key
=~/^\s*172\.(16|17|18|19|20|21|22|23|24|25|26|27|28|29|30|31)\./)||($key
=~/^\s*192\.168\./)){ $SPOOFED=" SPOOF ";}else{ $SPOOFED="";}
    print "$asrc{$key} \t $key $fp $SPOOFED\n" if($asrc{$key} >= $thresh);}
  if($vv){print "\n\nAttackers by IP\n=====================\n\n" if ($v) ;
      foreach $key (sort {$a cmp $b }keys(%asrc)){
      if((($f)||($v))&&($fsrc{$key})){$fp="\t($fsrc{$key} fps)";}else{$fp="";}
      print "$asrc{$key} \t$key $fp\n" if($asrc{$key} >= $thresh);}}
```

```
}
if (($f)||($v)&&(defined(%fdst))){
  print "\n\nFingerprinting attackers\n========================\n\n" if ($v) ;
  foreach $key (sort { $fsrc{$b} <=> $fsrc{$a} } keys(%fsrc)){
    $fp="fps";
    if(($key =~/^\s*10\./)||($key
=~/^\s*172\.(16|17|18|19|20|21|22|23|24|25|26|27|28|29|30|31)\./)||($key
=~/^\s*192\.168\./)){ $SPOOFED=" SPOOF ";}else{ $SPOOFED="";}
    print "$fsrc{$key} \t $key $fp $SPOOFED\n" if($fsrc{$key} >= $thresh);
  }
}
#Print summary from spp_portscan
if (($u)||($v)&&(defined(%nmb_portscan))){
  print "\n\nspp_portscan statistics\n======================\n\n" if ($v) ;
  print "* Nmb. of portscans\n";
  foreach $key (sort { $nmb_portscan{$b} <=> $nmb_portscan{$a} } keys(%nmb_portscan)){
    print"$nmb_portscan{$key} \t$key \n" if($nmb_portscan{$key} >= $thresh);}
  if(defined(%portscan_tcp)){
    print "* Nmb. of tcp packets\n";
    foreach $key (sort { $portscan_tcp{$b} <=> $portscan_tcp{$a} } keys(%portscan_tcp)){
      print"$portscan_tcp{$key} \t$key \n" if($portscan_tcp{$key} >= $thresh);}
  }
  if(defined(%portscan_udp)){
    print "* Nmb. of udp packets\n";
    foreach $key (sort { $portscan_udp{$b} <=> $portscan_udp{$a} } keys(%portscan_udp)){
      print"$portscan_udp{$key} \t$key \n" if($portscan_udp{$key} >= $thresh);}
  }
}
#Print attacked/scanned ports
if (($o)||($v)&&(defined(%pdst_tcp))){
  print "\n\nScanned TCP ports\n=================\n\n" if ($v) ;
  foreach $key (sort { $pdst_tcp{$b} <=> $pdst_tcp{$a} } keys(%pdst_tcp)){
    if((($f)||($v))&&($fpdst_tcp{$key})){$fp="\t($fpdst_tcp{$key} fps)";}else{$fp="";}
    print "$pdst_tcp{$key} \t$key $fp\n" if($pdst_tcp{$key} >= $thresh);
  }
}
if (($o)||($v)&&(defined(%pdst_udp))){
  print "\n\nScanned UDP ports\n================\n\n" if ($v) ;
  foreach $key (sort { $pdst_udp{$b} <=> $pdst_udp{$a} } keys(%pdst_udp)){
    print "$pdst_udp{$key} \t$key \n" if($pdst_udp{$key} >= $thresh);}
}
if (($o)||($v)&&(defined(%pdst))){
  print "\n\nPorts in alert files\n===================\n\n" if ($v) ;
  foreach $key (sort { $pdst{$b} <=> $pdst{$a} } keys(%pdst)){
    print "$pdst{$key} \t$key \n" if($pdst{$key} >= $thresh);}
}
if (($o)||($v)&&(defined(%portlist))){
  print "\n\nPorts by attack\n===============\n\n" if ($v) ;
  foreach $key (sort keys(%portlist)){
    print "$key \t $portlist{$key}  \n";}
}
#Attacker-Target pairs
if (($p)||($v)&&(defined(%pair))){
  print "\n\nAttacks/Targets\n===============\n\n" if ($v) ;
  foreach $key (sort { $pair{$b} <=> $pair{$a} } keys(%pair)){
    if((($f)||($v))&&($fpr{$key})){$fp="\t($fpr{$key} fps)";}else{$fp="";}
    print "$pair{$key} \t$key $fp\n" if($pair{$key}>= $thresh);
  }
}
#Communicating parties
if (($c)||($v)&&(defined(%pair))){
  print "\n\nCommunicating parties\n====================\n\n" if ($v) ;
  foreach $key (sort { $pair{$b} <=> $pair{$a} } keys(%pair)){
    if($key =~ /([MYNET0-9\.]+)\-([MYNET0-9\.]+)/){
      $src=$1;$dst=$2;
      $revkey="$dst-$src";
```

```
      if(defined($pair{$revkey})){
 if($src le $dst) {$commkey="$src-$dst";}
 else {$commkey="$dst-$src";}
 if((($f)||($v))&&($fpr{$key})){$fp="\t(fp)";}else{$fp="";}
 if($key eq $commkey)
   {$commpair{$commkey} .= " -> $pair{$key} $fp \*\*\* ";}
 else
   {$commpair{$commkey} .= " <- $pair{$key} $fp \*\*\* ";}
      }
    }
  }
  foreach $commkey (sort keys(%commpair)){
    print "$commkey  \*\*\* $commpair{$commkey} \n";
  }
}
```