Global Information Assurance Certification Paper

Assignment 1- Network Detects (30 Points)

Detect #1

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/24-20:47:11.001095 192.168.0.50:1024 -> 192.168.0.5:143
TCP TTL:64 TOS:0x0 ID:3  DF
*****PA* Seq: 0xFB6B1A85   Ack: 0x5E953932   Win: 0x7D78
2A 20 6C 6F 67 69 6E 20 90 90 90 90 90 90 90 90  * login ........
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
<snip – identical to above for 51 lines>
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 EB 58 5E 31 DB 83 C3 08 83 C3 02 88 5E 26  ...X^1........^&
31 DB 83 C3 23 83 C3 23 88 5E A8 31 DB 83 C3 26  1...#..#.^.1...&
83 C3 30 88 5E C2 31 C0 88 46 0B 89 F3 83 C0 05  ..0.^.1..F......
31 C9 83 C1 01 31 D2 CD 80 89 C3 31 C0 83 C0 04  1....1.....1....
31 D2 88 56 27 89 F1 83 C1 0C 83 C2 1B CD 80 31  1..V'..........1
C0 83 C0 06 CD 80 31 C0 83 C0 01 CD 80 69 61 6D  ......1......iam
61 73 65 6C 66 6D 6F 64 69 66 79 69 6E 67 6D 6F  aselfmodifyingmo
6E 73 74 65 72 79 65 61 68 69 61 6D E8 83 FF FF  nsteryeahiam....
FF 2F 65 74 63 2F 70 61 73 73 77 64 78 72 6F 6F  ./etc/passwdxroo
74 3A 3A 30 3A 30 3A 72 30 30 74 3A 2F 3A 2F 62  t::0:0:r00t:/:/b
69 6E 2F 62 61 73 68 78 83 F3 FF BF 88 F8 FF BF  in/bashx........
20 62 61 68 0A 00                                 bah..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/24-20:51:41.192326 192.168.0.50:1028 -> 192.168.0.5:143
TCP TTL:64 TOS:0x0 ID:19  DF
*****PA* Seq: 0x5E556200   Ack: 0x705BFE95   Win: 0x7D78
2A 20 6C 6F 67 69 6E 20 90 90 90 90 90 90 90 90  * login ........
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
<snip – identical to above for 51 lines>
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 EB 58 5E 31 DB 83 C3 08 83 C3 02 88 5E 26  ...X^1........^&
31 DB 83 C3 23 83 C3 23 88 5E A8 31 DB 83 C3 26  1...#..#.^.1...&
83 C3 30 88 5E C2 31 C0 88 46 0B 89 F3 83 C0 05  ..0.^.1..F......
31 C9 83 C1 01 31 D2 CD 80 89 C3 31 C0 83 C0 04  1....1.....1....
31 D2 88 56 27 89 F1 83 C1 0C 83 C2 1B CD 80 31  1..V'..........1
C0 83 C0 06 CD 80 31 C0 83 C0 01 CD 80 69 61 6D  ......1......iam
61 73 65 6C 66 6D 6F 64 69 66 79 69 6E 67 6D 6F  aselfmodifyingmo
6E 73 74 65 72 79 65 61 68 69 61 6D E8 83 FF FF  nsteryeahiam....
FF 2F 65 74 63 2F 70 61 73 73 77 64 78 72 6F 6F  ./etc/passwdxroo
74 3A 3A 30 3A 30 3A 72 30 30 74 3A 2F 3A 2F 62  t::0:0:r00t:/:/b
69 6E 2F 62 61 73 68 78 83 F3 FF BF 88 F8 FF BF  in/bashx........
20 62 61 68 0A 00                                 bah..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/24-20:53:20.074414 192.168.0.5:6000 -> 192.168.0.50:1030
TCP TTL:64 TOS:0x0 ID:59785  DF
**S***A* Seq: 0x76758BED   Ack: 0x69BFAAC7   Win: 0x7D78
TCP Options => MSS: 1460
```

1. **Source of trace:**
   SANS IDNet in Monterey, October 2000. Marty Roesch collected them using Snort 1.6.3.
   http://www.snort.org/sans_packet_logs.htm
   It's the 0324@1732 log.
2. **Detect was generated by:**
   Snort 1.6.3.
   [timestamp] [src ip]:[src port] -> [dst ip]:[dst port]
   [TCP TTL] [TOS] [ID] [DF]
   [FLAGS] [Initial SEQ] [ACK] [Window size]

[Optional TCP Options]
[Hex]

3.Probability the source address was spoofed:

The probability of a spoofed address is not high, although the possibility exists. The attacker may want a response. However, the intent seems to be to create an account, which may indicate that the attacker doesn't care if there's a response, just whether it worked or not. Given the fact that only 2 packets are transmitted it is not likely to be a spoofed address.

4.Description of attack:

Attack against TCP port 143 - typically IMAP. This appears to be a buffer overflow attempt in order to create an account "r00t" on the machine with no password. No specific CVE could be found, although buffer overflows against IMAP daemons are not at all unheard of.

5.Attack mechanism:

Available logs indicate that the attacker did no previous reconnaissance of the target machine. Two identical packets were sent several minutes apart. The packets appear to be destined for the IMAP service running on port 143. A legitimate "login" command is issued, followed by a large amount of "90" hex padding. This padding is followed by some apparent garbage, which is followed by a message and the exploit code. This exploit code appears to attempt to add a user to the /etc/passwd file named "r00t" with no password.

6.Correlations:

Although several IMAP buffer overflows are known, this specific type is not known. Perhaps if the target daemon was known better correlations could be made. Current IMAP buffer overflows can be found here: http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=IMAP+buffer+overflows

7.Evidence of active targeting:

There is definitely evidence of active targeting, as there are two crafted packets to a single IP.

8.Severity:

Severity = (Criticality + Lethality) - (System countermeasures + Network countermeasures)
Criticality = 3; UNIX machine, may or not be a main mail server
Lethality = 5; root access could be gained across net

System Countermeasures = 4; newer OS patch status unknown
Network Countermeasures= 1; no known network defenses

Severity= 3

9.Defensive Recommendation:

Analyze the target machine's logs and inspect the /etc/passwd file for the new account

10.Multiple Choice:

```
03/24-20:47:11.001095 192.168.0.50:1024 -> 192.168.0.5:143
TCP TTL:64 TOS:0x0 ID:3  DF
*****PA* Seq: 0xFB6B1A85   Ack: 0x5E953932   Win: 0x7D78
2A 20 6C 6F 67 69 6E 20 90 90 90 90 90 90 90 90  * login ........
```

This packet shows evidence of . . .
A) Normal delivery of email
B) A buffer overflow attempt
C) A trojan command
D) An attempt at unauthorized access

Detect #2

```
03/25-14:15:28.028445 192.168.0.54:1052 -> 192.168.0.200:139
TCP TTL:128 TOS:0x0 ID:334  DF
*****PA* Seq: 0xD62F9E13   Ack: 0x1A6572   Win: 0x4407
00 00 01 20 FF 53 4D 42 73 00 00 00 00 18 07 C8  ... .SMBs.......
00 00 00 00 00 00 00 00 00 00 00 00 00 FF FE  ................
00 00 40 00 0D 75 00 E4 00 04 11 32 00 00 00 00  ..@..u.....2....
00 00 00 18 00 18 00 00 00 00 00 D4 00 00 00 A7  ................
```

```
00 40 F2 47 A7 02 58 55 1A 83 DF A6 97 AE DB 76  .@.G..XU.......v
75 C7 81 80 36 D7 BC F0 09 FF 02 3F CB D9 E5 11  u...6......?....
C5 7D C6 C4 45 83 B3 30 1C 57 4E B4 F7 67 02 3C  .}..E..0.WN..g.<
49 00 41 00 64 00 6D 00 69 00 6E 00 69 00 73 00  I.A.d.m.i.n.i.s.
74 00 72 00 61 00 74 00 6F 00 72 00 00 00 4D 00  t.r.a.t.o.r...M.
4F 00 44 00 55 00 4C 00 4F 00 39 00 39 00 00 00  O.D.U.L.O.9.9...
57 00 69 00 6E 00 64 00 6F 00 77 00 73 00 20 00  W.i.n.d.o.w.s. .
32 00 30 00 30 00 30 00 20 00 32 00 31 00 39 00  2.0.0.0. .2.1.9.
35 00 00 00 57 00 69 00 6E 00 64 00 6F 00 77 00  5...W.i.n.d.o.w.
73 00 20 00 32 00 30 00 30 00 30 00 20 00 35 00  s. .2.0.0.0. .5.
2E 00 30 00 00 00 00 00 04 FF 00 20 01 08 00 01  ..0........ ....
00 31 00 00 5C 00 5C 00 31 00 39 00 32 00 2E 00  .1..\.\.1.9.2...
31 00 36 00 38 00 2E 00 30 00 2E 00 32 00 30 00  1.6.8...0...2.0.
30 00 5C 00 49 00 50 00 43 00 24 00 00 00 3F 3F  0.\.I.P.C.$...??
3F 3F 3F 00                                      ???.
```

^^^^^ Many duplicate packets

1.Source of trace:

SANS IDNet in Monterey, October 2000. Marty Roesch collected them using Snort 1.6.3.
http://www.snort.org/sans_packet_logs.htm (It's the 0324@1732 log.)

2.Detect was generated by:

Snort 1.6.3.
[timestamp] [src ip]:[src port] -> [dst ip]:[dst port]
[TCP TTL] [TOS] [ID] [DF]
[FLAGS] [Initial SEQ] [ACK] [Window size]
[Optional TCP Options]
[Hex]

3.Probability the source address was spoofed:

The attack is intended to receive information back; the address was not spoofed.

4.Description of Attack:

The same source ip is used with incrementing source ports (indicating that the packets were not crafted). The packets appears to try to connect to IPC$ - a NT daemon (Inter-Process Communication).

5.Attack Mechanism:

A unique signature is found. The strings "Administrator" and "MODULO99" and "IPC$" appear in the payload. IPC can be exploited to gather information and gain access to a Windows NT machine. The presence of "modulo" suggests activity involving the SAM database (more information follows). The additional presence of "Administrator" should cause significant alarm. This is quite definitely bad and should be detected and blocked if possible.

6.Correlations:

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

This link details the SYSKEY vulnerability in NT where a SAM database is obtained and the passwords cracked. The cracking is made easy because they are stored as two 8-bit strings instead of one 16-bit string. I believe the "modulo" in the payload refers to these 8-bit strings. A patch to this has been available for quite some time and is available from Microsoft (linked from the Razor page).

There are two potential CVE entries for this exploit, one with NT and one with 2000. They can both be found here:

http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=SYSKEY

7.Evidence of active targeting:

There is definitely evidence of active targeting, as this is a specific exploit directed at a specific machine with a valid source ip.

8.Severity:

Severity = (Criticality + Lethality) - (System countermeasures + Network countermeasures)
Criticality = 4; NT Server - probably a significant asset

Lethality = 5; user/password database could be compromised

System Countermeasures = 4; Modern OS without recent patches
Network Countermeasures= 1; No known network defenses

Severity= 4

9.Defensive Recommendation:

Verify the condition of the target host - is it patched? If not, passwords must all be changed immediately. Logs should be scoured and continually monitored for suspicious activity.

10.Multiple Choice:

```
03/25-14:15:28.028445 192.168.0.54:1052 -> 192.168.0.200:139
TCP TTL:128 TOS:0x0 ID:334  DF
*****PA* Seq: 0xD62F9E13   Ack: 0x1A6572   Win: 0x4407

Based on the above packet:
A) It's a typical NetBIOS data connection
B) It's typical NetBIOS enumeration
C) It is part of a port 139 scan
D) It could be the Ping O' Death
```

Detect #3

```
21:38:15.084929 eth0 < MY.NET.250.210 > MY.NET.250.209: igmp-0 [v0][|igmp] (frag
10931:1480@0+)
                  4500 05dc 2ab3 2000 8002 f32b 80ce fad2
                  80ce fad1 0000 0000 0000 0000 0000 0000
                  0000 0000 0000 0000 0000 0000 0000 0000
                  0000 0000 0000 0000 0000 0000 0000 0000
                  0000 0000 0000 0000 0000 0000 0000 0000
                  0000 0000 0000 0000 0000 0000 0000 0000
                  0000 0000 0000 0000 0000 0000 0000 0000
                  0000 0000 0000 0000 0000 0000 0000 0000
                  0000 0000 0000 0000 0000 0000 0000 0000
<snip - normal fragmentation>
21:38:15.133727 eth0 < MY.NET.250.210 > MY.NET.250.209: (frag 10931:800@59200)
                  4500 0334 2ab3 1ce8 8002 f8eb 80ce fad2
                  80ce fad1 0000 0000 0000 0000 0000 0000
                  0000 0000 0000 0000 0000 0000 0000 0000
                  0000 0000 0000 0000 0000 0000 0000 0000
                  0000 0000 0000 0000 0000 0000 0000 0000
                  0000 0000 0000 0000 0000 0000 0000 0000
                  0000 0000 0000 0000 0000 0000 0000 0000
                  0000 0000 0000 0000 0000 0000 0000 0000
                  0000 0000 0000 0000 0000 0000 0000 0000
21:38:15.134189 eth0 > MY.NET.250.209 > MY.NET.250.210: icmp: MY.NET.250.209 protocol 2
unreachable [tos 0xc0]
                  45c0 0240 1d81 0000 ff01 a43a 80ce fad1
                  80ce fad2 0302 3865 0000 0000 4500 ea74
                  2ab3 0000 8002 f32b 80ce fad2 80ce fad1
                  0000 0000 0000 0000 0000 0000 0000 0000
                  0000 0000 0000 0000 0000 0000 0000 0000
                  0000 0000 0000 0000 0000 0000 0000 0000
                  0000 0000 0000 0000 0000 0000 0000 0000
                  0000 0000 0000 0000 0000 0000 0000 0000
                  0000 0000 0000 0000 0000 0000 0000 0000
```

1.Source of trace:

my network in a lab environment.

2.Detect was generated by:

tcpdump did the sniffing, igmpnuke (http://www.tlsecurity.net/DoS/igmpnuke.htm) was the tool used to

generate the traffic.

tcpdump format:

[timestamp] [interface] [src ip] > [dst ip]: [protocol*] (frag: [id]:[length]@[offset][+**]) [message*] [tos*] [hex]

* denotes that the field may or may not be present

** more fragments bit - may or may not be present

3.Probability the source address was spoofed:

Entirely likely candidate for a spoofed ip address. However, the tool used did not spoof the source ip.

4.Description of attack:

A large IGMP packet is sent to a target machine. Windows 95/98/NT machines are vulnerable to this and will BSOD or simply reboot. CVE 1999-0918 deals with this here:

http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0918

5.Attack mechanism:

The attack simply sends any number of large IGMP packets to the target machine, causing a reboot or BSOD. A patch is available from Microsoft and can be found on their main page (http://www.microsoft.com). Follow links to your OS and look under "fixes." IGMPNuke was used to generate this attack. It has configurable fields for destination address, packet size, and the number of packets to send.

6.Correlations:

This particular exploit was discovered in September, 1999 and has since been patched. It was widely known (http://www.securityfocus.com/vdb/bottom.html?vid=514)and discussed then and does not seem to be in major use today.

7.Evidence of active targeting:

By the nature of the exploit, a specific host is targeted.

8.Severity:

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

Criticality = 1; 95/98 user desktops

Lethality = 4; machine reboots

System Countermeasures = 3; older OS without recent patches

Network Countermeasures= 1; no network defenses in place

Severity= 1

9.Defensive Recommendation:

Keep security fixes and patches up to date. Block IGMP on your outer perimeter, as it should not usually enter or leave a network - make exceptions as necessary.

10.Multiple Choice:

```
21:38:15.133727 eth0 < MY.NET.250.210 > MY.NET.250.209: (frag 10931:800@59200)
                    4500 0334 2ab3 1ce8 8002 f8eb 80ce fad2

What protocol type is the above packet?
A) ICMP
B) IGMP
C) type 80
D) type 243
```

Detect #4

21:09:41.849269 eth0 > MY.NET.250.209 > MY.NET.250.210: (frag 1109:9@65520)
4500 001d 0455 1ffe ff**01** a04b 80ce fad1
80ce fad2 0800 0000 0000 0000 00
^^^^^^^^^^ LARGE numbers of duplicate packets

Source of trace:

My network in a lab environment.

Detect was generated by:

Jolt2 was used to generate the attack: (http://rootshell.com/archive-j457nxiqi3gq59dv/200005/jolt2.txt.html)

tcpdump format:
[timestamp] [interface] [src ip] > [dst ip]: (frag: [id]:[length]@[offset])
[hex]

Probability the source address was spoofed:

Highly likely candidate for source spoofing, as the response is not of concern to the attacker. In the case of this specific attack, an option for a fake source ip is available, but does not appear to work correctly in my tests.

Description of Attack:

A security bulletin was released by Microsoft in May, 2000 and has since been patched. GIAC started seeing it's signature around the same time frame. It operates by exploiting fragmentation-reassembly in Microsoft Operating Systems, as well as several major firewalls.

Attack Mechanism:

TCP fragments with ID: 1109 TTL:255 Length:9 and offset: 65520 are sent to the target host. The target host immediately goes to 100% CPU utilization in an attempt to reassemble the nonexistent packet. The attack appears to send on the order of 150 of these identical packets per second to the target machine. The target machine returns to normal operation immediately upon the cessation of the attack if the attack last only a few minutes. Longer-duration attacks can render the machine inoperable, requiring a reboot.

Correlations:

Unbeknownst to me when I chose this detect - there is extensive information available on it.

Microsoft released a security bulletin regarding the attack here:
http://www.microsoft.com/technet/security/bulletin/ms00-029.asp

There is a CVE on it here:
http://cve.mitre.org/cgi-bin/cvename.cgi?name=0305

The DoS was discussed in a paper by Michael Castro here:
http://www.sans.org/infosecFAQ/jolt2.htm

Evidence of active targeting:

By the nature of the exploit, a specific host is targeted.

Severity:

Severity = (Criticality + Lethality) - (System countermeasures + Network countermeasures)
Criticality = 1; 95/98 user desktops
Lethality = 4; machine completely freezes

System Countermeasures = 3; older OS without recent patches
Network Countermeasures= 1; no network defenses in place

Severity= 1

Defensive Recommendation:

Maintain current patches on all Operating Systems. Stateful firewalls should drop these packets immediately, as there are no previous packets to assemble it with. Blocking fragmented packets on the routers will also work. Console logging may be turned off, as many firewalls will eat up CPU cycles logging all of these packets.

Multiple Choice:

21:09:41.849269 eth0 > MY.NET.250.209 > MY.NET.250.210: (frag 1109:9@65520)
4500 001d 0455 1ffe ff01 a04b 80ce fad1
80ce fad2 0800 0000 0000 0000 00

This attack obviously tries to exploit:

A) Invalid ICMP message types
B) Fragmentation reassembly errors
C) Invalid packet length errors
D) a buffer overflow

Assignment 2 - Evaluate an Attack (20 points)

Windows and Mac versions of Napster can be downloaded from http://www.napster.com
Linux versions can be found at http://freshmeat.net/search/?q=napster

Napster has come under fire for many reasons as of late. Whatever the criticism of it, it is a network protocol. As with any protocol/utility, it can be abused. Napster use is extremely widespread, especially on University campuses. I have seen this abuse firsthand at the University of Missouri - Columbia. The University owns a 45Mbit-FD link to the Internet. At given times, 35 Mbit-FD can be consumed by Napster traffic. That is, mp3 files shared via napster. In given environments, Napster can be considered a Denial Of Service.

Many attacks are both detectable and preventable. IDS sensors can detect many attacks, or one may notice that a machine has been compromised, or the machine may not respond due to a denial of service attack. An attack is preventable either through OS patches, firewall and router configuration, or other such devices (personal firewalls). Most attacks can be classified as malicious and involve active targeting. Napster is not, by it's nature, malicious. There is nothing illegal about using napster itself. Denial of Service attacks are malicious as well. Napster does not necessarily fit the bill as an attack. It is not malicious, it is not illegal, so what is it? I argue that Napster does not intend by it's nature to cause a denial of service, but that it can and does through abuse. If Napster is considered to be a denial of service attack in certain environments, then it should be treated as such in those environments.

My environment at a University is one such example. The University does not have the funding to purchase high-end traffic shaping appliances, so another solution must be constructed. There is need to both more accurately detect how widespread its usage is, and to kill connections if absolutely necessary. The University does not want to block Napster entirely, simply to preserve a working Internet connection for all. A logical choice for the detect and defeat of Napster is an IDS system with the capability to knock down connections (session-snipe) on an immediate basis. Snort was the only logical solution as this point. The question now becomes how to use Snort to detect and defeat Napster correctly.

I have observed many different client ports in use; this leads to to believe that any filtering based on client ports, especially the default ports, is useless. I have observed many different destination IP's used for Napster servers. Aside from the main block of "Napster Inc." servers there are countless OpenNAP servers to contend with (http://www.napigator.com/list.php). Destination ports are somewhat reliable, although I wouldn't trust them, either, as there are pages dedicated to the listing of SOCKS proxies, which can (but may not) circumvent these filters (http://proxys4all.cgi.net/). So what identifies all Napster traffic, no matter the client version, the server (Napster Inc. or OpenNAP), and no matter if a proxy is used? How does a client talk to a server and vice-versa?

Napster utilizes the Napster protocol, which can be found at http://opennap.sourceforge.net/napster.txt . Napster uses TCP to communicate, each message to/from the server is in the form <length><type><data> where type is specified in the protocol specification. I keyed in on these types to identify Napster correctly. Snort comes with several rules to detect Napster traffic. They are (taken from 10102kany.rules):

```
alert TCP any any <> any 6699 (msg:"Napster Client Data"; flags: PA; content: ".mp3"; nocase; )
alert TCP any any <> any 8888 (msg:"Napster 8888 Data"; flags: PA; content: ".mp3"; nocase; )
alert TCP any any <> any 7777 (msg:"Napster 7777 Data"; flags: PA; content: ".mp3"; nocase; )
alert TCP any any <> any 6666 (msg:"Napster 6666 Data"; flags: PA; content: ".mp3"; nocase; )
alert TCP any any <> any 5555 (msg:"Napster 5555 Data"; flags: PA; content: ".mp3"; nocase; )
alert TCP any any <> any 4444 (msg:"Napster 4444 Data"; flags: PA; content: ".mp3"; nocase; )
alert TCP any any <> any 8875 (msg:"Napster Server Login"; flags: PA; content: "anon@napster.com"; )
```

As you can see, these rules depend on ".mp3" in the payload of the packet with a specific port on either end. Although somewhat unlikely, normal traffic can appear on these ports. Also, if one of these ports is used and a file is sent, alerts pile up extremely quickly. The final rule will correctly identify a Napster, Inc. server login in its default install, but does not deal with OpenNAP servers or a non-deafult Windows client install. While this may be sufficient for many current users, it may not be in the near future, as Napster Inc. recently signed a contract to restrict access to their service on a subscription basis and I believe users will flock to OpenNAP servers. In addition, a simple reconfig on the client will bypass this filter. To handle the situation of easily-evaded filters I analyzed Napster traffic in consideration of the protocol. As I mentioned, I used the type field to identify traffic.

I used Ethereal (http://freshmeat.net/projects/ethereal/homepge/) to analyze the traffic. I used both Windows and a Linux Napster Clients. The Linux client sends data differently than the Windows client, but both have the length and type fields intact. Specifically, the Linux client I analyzed sends the length and type in one packet and the data in the following, while the Windows client sends the length, type, and data in one packet. This does not affect the signature, as only the type is of concern. The new (now BETA) Snort rules I wrote are as follows:

alert TCP any any <> any any (msg:"Napster Login"; flags: PA; content: "|00 0200|"; offset: 1; depth: 3;)

alert TCP any any <> any any (msg:"Napster Nick Check (New User Login Attempt)"; flags: PA; content: "|00 0700|"; offset: 1; depth: 3;)

alert TCP any any <> any any (msg:"Napster Download Request"; flags: PA; content: "|00 cb00|"; offset: 1; depth: 3;)

alert TCP any any <> any any (msg:"Napster Upload Request"; flags: PA; content: "|00 5f02|"; offset: 1; depth: 3;)

A Napster logon looks like this to a Snort log: The packet successfully triggered a "Napster Login" alert.

```
11/10-00:38:19.255526 128.206.250.209:2365 -> 64.124.41.236:8888
TCP TTL:64 TOS:0x0 ID:36727  DF
*****PA* Seq: 0xB2ABBD80   Ack: 0x7C04D1FA   Win: 0x7D78
TCP Options => NOP NOP TS: 9668023 7890647
2B 00 02 00                                        +...
```

A Napster download request looks like this to a Snort log: The packet successfully triggered a "Napster Download Request" alert.

```
11/10-00:38:34.529297 128.206.250.209:2365 -> 64.124.41.236:8888
TCP TTL:64 TOS:0x0 ID:36746  DF
*****PA* Seq: 0xB2ABE1EB   Ack: 0x7C04D78E   Win: 0x7D78
TCP Options => NOP NOP TS: 9669550 7890819
46 00 CB 00                                        F...
```

Notice the boldfaced payload bytes - these are the message types corresponding to login and download request. These are both from gnapster, as only the length and field are in this packet. A Windows clients packet would be much larger as other information would follow this payload.

Here is an upload request from a Windows client: The packet successfully triggered a "Napster Upload Request" alert.

```
11/10-00:55:26.398731 64.124.41.236:8888 -> 128.206.250.209:2365
TCP TTL:48 TOS:0x0 ID:51735  DF
*****PA* Seq: 0x7C04D975   Ack: 0xB2ABE287   Win: 0x7C70
```

```
TCP Options => NOP NOP TS: 7993346 9770262
3C 00 5F 02 62 6F 64 69 65 33 20 22 5C 68 6F 6D  <._.bodie3 "\hom
65 32 5C 6E 65 77 6D 70 33 5C 43 4F 4D 45 44 59  e2\newmp3\COMEDY
5C 52 6F 62 69 6E 20 57 69 6C 6C 69 61 6D 73 20  \Robin Williams
2D 20 41 6C 63 6F 68 6F 6C 2E 6D 70 33 22 20 38  - Alcohol.mp3" 8
```

Following is a Napster nick check. I have found several inconsistencies with the definition of the protocol - this being one of them. The specification states that the following happens for a new user: nick check (type 7), nick ack (type 8), new user login. Instead, the new user login data is truncated onto a nick check, which isn't supposed to happen according to the specification. Here is the nick check under gnapster:

```
11/10-09:57:16.428100 128.206.250.209:2611 -> 207.195.111.2:8888
TCP TTL:64 TOS:0x0 ID:56387  DF
*****PA* Seq: 0xF0ECA3AF   Ack: 0xCADF78BA   Win: 0x7D78
TCP Options => NOP NOP TS: 13021740 3549238
08 00 07 00                                      ....
```

My rules demonstrate that Napster, no matter the platform, the version, the port, or the proxy, can be detected successfully. They also demonstrate that Napster can be defeated with Snort's "Flexible Response." Flexible Response sends a rst packet to either one end of the connection, both ends, of several various icmp messages to either/both ends.

Assignment 3 - Analysis Process (20 Points)

Note: The general format is taken from a previous practical done by Marc Gregoire.

To: MY.NET
From: Brent Deterding
Subject: Security Analysis of MY.NET

Over the past month, MY.NET was monitored for suspicious activity using Snort, a free IDS (http://www.snort.org). A report detailing this activity follows. Data collection, overall analysis, detailed analysis, a summary, and recommendations for the future follow.

Data Collection

Snort (http://www.snort.org) was used to monitor traffic. Enough data to provide a reliable picture of security at MY.NET was gathered, although a high level of granularity was not achieved. Two main types of reports were available; alerts and scans. This report treats scans and attacks separately in most instances.

Overall Analysis - Alerts

Several tables presenting general information of interest regarding alerts follow. Table 1 presents the distribution of attack methods encountered.

Table #1: The distribution of attack methods

%	# of attacks	method
48.60	9775	Watchlist 000222 NET-NCFC
15.31	3079	WinGate 1080 Attempt
13.58	2731	SYN-FIN scan!
13.12	2638	Watchlist 000220 IL-ISDNNET-990517
4.84	973	Attempted Sun RPC high port access
2.26	455	SNMP public access
0.87	174	SMB Name Wildcard
0.46	92	Null scan!
0.33	67	NMAP TCP ping!
0.16	32	SUNRPC highport access!
0.16	32	Probable NMAP fingerprint attempt
0.14	29	Queso fingerprint
0.09	18	External RPC call
0.04	9	
0.02	4	TCP SMTP Source Port traffic
0.01	2	Possible wu-ftpd exploit - GIAC000623
0.01	2	site exec - Possible wu-ftpd exploit - GIAC000623
0.00	1	Happy 99 Virus

This table clearly outlines the large amount of questionable traffic entering MY.NET. This amount of traffic is not necessarily uncharacteristic of a network this size. Each of these attacks will be further analyzed later. Specific hosts responsible for these attacks are to be noted. The following table shows when one host targeted another host using the same method repeatedly.

Table #2: The number of attacks from same host to same destination using same method

# of attacks	from	to	method

```
=============================================================
   1847     159.226.63.190    MY.NET.253.43      Watchlist 000222 NET-NCFC
   1828     159.226.63.190    MY.NET.253.42      Watchlist 000222 NET-NCFC
   1827     159.226.63.190    MY.NET.253.41      Watchlist 000222 NET-NCFC
   1150     159.226.45.108    MY.NET.6.7         Watchlist 000222 NET-NCFC
   872     159.226.114.129   MY.NET.162.199     Watchlist 000222 NET-NCFC
   808     212.179.58.174    MY.NET.157.200     Watchlist 000220 IL-ISDNNET-990517
   512     205.188.179.33    MY.NET.217.42      Attempted Sun RPC high port access
   507     159.226.45.3      MY.NET.253.43      Watchlist 000222 NET-NCFC
   307     212.179.66.2      MY.NET.221.94      Watchlist 000220 IL-ISDNNET-990517
   272     212.179.29.150    MY.NET.53.28       Watchlist 000220 IL-ISDNNET-990517
   266     212.179.66.2      MY.NET.181.87      Watchlist 000220 IL-ISDNNET-990517
   255     159.226.63.200    MY.NET.253.43      Watchlist 000222 NET-NCFC
   253     159.226.63.200    MY.NET.253.41      Watchlist 000222 NET-NCFC
   247     159.226.63.200    MY.NET.253.42      Watchlist 000222 NET-NCFC
   230     212.179.127.45    MY.NET.202.58      Watchlist 000220 IL-ISDNNET-990517
   170     212.179.27.111    MY.NET.206.154     Watchlist 000220 IL-ISDNNET-990517
   143     MY.NET.101.160    MY.NET.101.192     SMB Name Wildcard
   136     212.179.58.2      MY.NET.98.168      Watchlist 000220 IL-ISDNNET-990517
   130     159.226.45.3      MY.NET.163.32      Watchlist 000222 NET-NCFC
   126     212.179.61.244    MY.NET.5.29        Watchlist 000220 IL-ISDNNET-990517
   120     205.188.153.115   MY.NET.218.218     Attempted Sun RPC high port access
   112     MY.NET.98.172     MY.NET.101.192     SNMP public access
   110     205.188.153.109   MY.NET.219.26      Attempted Sun RPC high port access
   108     159.226.5.77      MY.NET.100.230     Watchlist 000222 NET-NCFC
   108     159.226.45.3      MY.NET.6.7         Watchlist 000222 NET-NCFC
   100     MY.NET.98.109     MY.NET.101.192     SNMP public access
```

Of interest are the top attackers. The large number of attacks from these host indicate a strong interest in specific targets within MY.NET. More information should be gathered regarding these hosts. This information will follow shortly. Also of interest are the attacks from MY.NET hosts. This may indicate that they have been compromised. To develop this theory further, hosts internal to MY.NET initiating attacks was analyzed in the following table.

Table #3: The distribution of MY.NET attack methods

```
====================================
        #  of
  %    attacks   method
====================================
79.37    477      SNMP public access
20.63    124      SMB Name Wildcard
```

Further analysis of the source of these attacks reveals . . .

Table #4: The number of attacks from an internal host to same destination using same method

```
============================================================
  # of
 attacks  from              to               method
============================================================
   124     MY.NET.101.160    MY.NET.101.192     SMB Name Wildcard
   111     MY.NET.98.172     MY.NET.101.192     SNMP public access
   111     MY.NET.98.109     MY.NET.101.192     SNMP public access
   106     MY.NET.97.217     MY.NET.101.192     SNMP public access
   28     MY.NET.97.154     MY.NET.101.192     SNMP public access
   24     MY.NET.98.114     MY.NET.101.192     SNMP public access
   16     MY.NET.98.171     MY.NET.101.192     SNMP public access
   15     MY.NET.98.191     MY.NET.101.192     SNMP public access
```

13	MY.NET.97.244	MY.NET.101.192	SNMP public access
13	MY.NET.98.181	MY.NET.101.192	SNMP public access
11	MY.NET.98.159	MY.NET.101.192	SNMP public access
10	MY.NET.98.201	MY.NET.101.192	SNMP public access
8	MY.NET.98.148	MY.NET.101.192	SNMP public access
3	MY.NET.97.246	MY.NET.101.192	SNMP public access
3	MY.NET.97.206	MY.NET.101.192	SNMP public access
3	MY.NET.98.177	MY.NET.101.192	SNMP public access
2	MY.NET.98.190	MY.NET.101.192	SNMP public access

I believe all internal SNMP attacks are false positives. It is not unusual to see SMB name wildcard alerts from internal sources, as they can be generated by commands such as nbtscan for Linux. However, 124 seperate SMB attempts from one source is suspicious. The presence of a large number of SNMP alerts could well be an SNMP push type service. May utilities, such as Big Brother, use an SNMP pull type model, where one machine polls many for SNMP information. The obverse could certainly be true of another application. The fact that they appear to be from the same subnet (MY.NET.97.0/23) indicates that this could very well be true (a service network?).

When considering the external network once again, the most frequently attacked hosts are as follows:

Table #5: The percentage and number of attacks to one certain host

%	# of attacks	to	method
13.41	2697	MY.NET.253.43	Watchlist 000222 NET-NCFC
10.94	2201	MY.NET.253.42	Watchlist 000222 NET-NCFC
10.76	2164	MY.NET.253.41	Watchlist 000222 NET-NCFC
6.39	1286	MY.NET.6.7	Watchlist 000222 NET-NCFC
4.34	872	MY.NET.162.199	Watchlist 000222 NET-NCFC
4.02	808	MY.NET.157.200	Watchlist 000220 IL-ISDNNET-990517
2.55	512	MY.NET.217.42	Attempted Sun RPC high port access
2.26	455	MY.NET.101.192	SNMP public access
1.53	307	MY.NET.221.94	Watchlist 000220 IL-ISDNNET-990517
1.35	272	MY.NET.53.28	Watchlist 000220 IL-ISDNNET-990517
1.32	266	MY.NET.181.87	Watchlist 000220 IL-ISDNNET-990517
1.31	264	MY.NET.100.230	Watchlist 000222 NET-NCFC
1.22	246	MY.NET.5.29	Watchlist 000220 IL-ISDNNET-990517
1.14	230	MY.NET.202.58	Watchlist 000220 IL-ISDNNET-990517
0.85	170	MY.NET.206.154	Watchlist 000220 IL-ISDNNET-990517
0.71	143	MY.NET.101.192	SMB Name Wildcard
0.68	136	MY.NET.98.168	Watchlist 000220 IL-ISDNNET-990517
0.65	130	MY.NET.163.32	Watchlist 000222 NET-NCFC
0.60	120	MY.NET.218.218	Attempted Sun RPC high port access
0.55	110	MY.NET.219.26	Attempted Sun RPC high port access

The targeted hosts should be analyzed for core devices to determine what is highly targeted. These targets were targeted by the following machines . . .

Table #6: Top 10 Attackers

159.226.63.190	10920
210.61.144.125	4784
168.187.26.157	4290

159.226.45.108	2346	
159.226.114.1	1751	
159.226.114.129	1746	
212.179.58.174	1615	
159.226.45.3	1558	
159.226.63.200	1556	
212.179.66.2	1144	

Overall Analysis - Scans

Several tables presenting general information regarding scans of interest follow.

Table #7: Statistics on main port-scanning sources.

195.114.226.41	25752	26160	0	RIPE Network Coordination Centre
35.10.82.111	25110	25290	0	Sprint Canada, Inc.
206.186.79.9	14452	15425	11	@HOME
24.17.189.83	12751	13675	0	RIPE Network Coordination Centre
212.141.100.97	11897	12627	3	Netname
210.61.144.125	4508	4822	54	VideoTron Ltee
24.201.118.67	4377	4461	0	Iowa Sate Univ.
129.186.93.133	3404	3760	0	RIPE Network Coordination Centre
194.165.230.250	2909	3219	0	RIPE Network Coordination Centre
168.187.26.157	2598	2972	0	Kuwait Ministry of Communications
MY.NET.1.13	2362	0	2542	
MY.NET.1.3	2237	0	2760	
63.248.55.245	2000	0	9849	Flashcom Inc.
MY.NET.1.4	1997	0	2279	
MY.NET.1.5	1991	0	2294	
216.198.45.10	1964	2006	0	STIC.NET
212.170.19.199	1450	1648	0	RIPE Network Coordination Centre
128.171.57.194	800	867	0	Univ. of Hawaii
62.158.107.236	757	770	0	RIPE Network Coordination Centre
4.54.37.160	654	727	0	BBN Planet
213.25.136.60	624	663	0	RIPE Network Coordination Centre
24.94.176.113	573	589	0	ServiceCo LLC - RoadRunner
207.19.142.78	487	518	1	Baltimore County Public Library
130.149.41.70	467	563	0	Technische Universitaet Berlin
210.55.227.13	361	416	0	NetName
24.3.39.44	312	0	312	@HOME
213.188.8.45	274	299	0	RIPE Network Coordination Centre
212.41.61.40	237	291	0	RIPE Network Coordination Centre
210.100.192.254	207	225	0	Netname
24.180.134.156	180	62070	62	@HOME
128.211.224.100	155	0	81945	Purdue Univ.
128.211.209.31	140	0	23780	Purdue Univ.
24.23.198.174	103	108	0	@HOME

These scans were looking for something specific. The most commonly probed ports were:

Table #8: Top 10 Destination Ports

21	96515	FTP
27374	27362	?
53	22384	DNS
25	13132	SMTP
1080	8717	HTTP
23	8292	TELNET
80	6256	HTTP
6699	2785	NAPSTER

7001	2258	Half-Life Multiplayer game
77	2104	rje?
32771	2054	RPC
12346	1910	NetBus
1097	1671	?
123	906	?
9704	1326	RPC statd exploit
2430	477	?

This list contains mostly standard and commonly probed ports. It appears that there is some legitimate traffic, such as port 53 (DNS) traffic, and the legitimacy of Napster can still be argued. The following table shows which hosts are most often scanned.

Table #9: Top 10 Targeted Hosts

MY.NET.253.4	14124
MY.NET.253.43	5419
MY.NET.253.42	4416
MY.NET.253.41	4287
MY.NET.6.7	2588
MY.NET.1.2	2105
MY.NET.162.199	1746
MY.NET.157.200	1615
MY.NET.217.42	1054
MY.NET.2.1	971

Do not be mislead into thinking that these hosts constitute the majority of scanned hosts - most hosts are scanned few times, but there are many of them that are scanned. These machine were targeted by the following machines . . .

Table #10: Top 10 Scanners

141.213.191.50	3805
63.248.55.245	3471
128.253.179.58	2290
206.186.79.9	1520
128.211.224.100	1417
24.17.189.83	1264
212.141.100.97	1138
24.180.134.156	988
195.114.226.41	627
168.187.26.157	564

Internal hosts scanning other internal is cause for alarm. Internal hosts that initiated scans are as follows.

Table #11: Port scans from internal hosts.

MY.NET.1.13	117
MY.NET.1.3	143
MY.NET.1.4	122
MY.NET.1.5	119
MY.NET.225.42	10

Detailed analysis of specific alerts and activities of specific activities

Detailed Analysis - Alerts

Watchlist 000222 NET-NCFC

This is traffic from the Chinese Academy of Science and is mostly destined for port 25 (SMTP) on host MY.NET.253.43. This may be legitimate mail traffic, it should be investigated further by analyzing the mail logs on MY.NET.253.43.

WinGate 1080 Attempt

WinGate is a proxy server. Once found, a wingate proxy can be used to leapfrog to other hosts, obscuring the attackers identity slightly. The target host should be investigated, this is possibly legitimate traffic.

SYN-FIN Scan!, Null scan!, NMAP TCP ping!, Probable NMAP fingerprint attempt, and Queso fingerprint

These are reconnaissance scans.

Watchlist 000220 IL-ISDNNET-990517

This alerts on traffic on hosts from Israel which have demonstrated ill-intent towards the security community. Port 6699 is the primary port used, which is indicative of Napster traffic.

Attempted Sun RPC high port access

A large amount on traffic from source port 4000 to destination port 32771 is alerted because RPC services live at 32771 typically. The constant source port 4000 causes me to think that this may just be ICQ traffic. An investigation of a targeted host such as MY.NET.217.42 is in order.

SNMP public access

The presence of a large number of SNMP alerts could well be an SNMP push type service. May utilities, such as Big Brother, use an SNMP pull type model, where one machine polls many for SNMP information. The obverse could certainly be true of another application. The fact that they appear to be from the same subnet (MY.NET.97.0/23) indicates that this could very well be true (a service network?).

SMB Name Wildcard

It is not unusual to see SMB name wildcard alerts from internal sources, as they can be generated by commands such as nbtscan for Linux. However, 124 seperate SMB attempts from one source is suspicious. Further investigation is recommended.

SUNRPC highport access!

MY.NET.211.2 was the primary target of this traffic, which is intended to access RPC services which live on ports 32xxx typically. The target host should be investigated immediately.

External RPC call

MY.NET.6.15 was the primary target of this traffic, which is intended to access the portmapper, which controls RPC services. The target host should be investigated immediately.

TCP SMTP Source Port traffic

This is traffic originating from port 25. Several were to port 25, suggesting legitimate mail traffic. Others were directed back to high-numbered ports, suggesting possibly a telnet connection to the server.

Possible wu-ftpd exploit - GIAC000623

site exec - Possible wu-ftpd exploit - GIAC000623

This exploit affected thousands of RedHat machines worldwide. It uses the default ftp server installed (wu-ftpd) to gain unauthorized access. The following internal hosts were targeted by this exploit.

MY.NET.99.104 - 1 follow-up connection - investigate.
MY.NET.150.24 - several follow-up connections - investigate immediately.
MY.NET.202.202 - several follow-up connections - investigate immediately.
MY.NET.202.190 - 1 follow-up connection - investigate.

Happy 99 virus
MY.NET.6.35 and MY.NET.179.80 should be scanned for virus' immediately.

Summary and Recommendations

It is important to note that data was incomplete. However, a fairly accurate assessment can still be made as to the overall security of MY.NET . Some hosts are more than likely compromised and must be taken care of immediately, while others need to be investigated, but are not necessarily compromised. MY.NET sees a lot of port scans of many varieties and size. These scans were executed to gather information for possible exploitation at a later date. The sources of these scans merits note as well. Some basic steps should be taken to enhance security at MY.NET, such as strong passwords (SNMP public for example), better access control on the perimeter (firewall redesign?), blocking malicious hosts, better tracking and blocking of certain services (telnet, ftp, http, smtp), and a process to review security on a regular basis. A VPN solution should be considered as well. An Acceptable Use Policy should be implemented as soon as possible to have policy backing up actions. This policy should directly address what is not acceptable, such as any outside services unless previously authorized. Updated patches should be installed and kept up to date as well. Updated patches would have prevented exploits such as wu-ftpd. As part of regular evaluation of security, a dedicated IDS sensor should be put into place that is reliable enough to keep accurate and complete logs. Ideally, one should be placed on either side of the firewall. The firewall should be analyzed for possible holes and its policy brought up to date with the AUP immediately.

Assignment 4 - Analysis Process (20 Points)

Assignment #3 was daunting to say the least when I began. I first gathered the data I would need and divided it into three types general types - Snort*, SOOS*, and SnortSca. I then wrote several perl scripts to parse out relevant information. I didn't know exactly what to do with this information once I had it. I attempted to insert it into an Oracle database but was largely unsuccessful. I did not have time to mess around with this any longer and parsed it in a way such that MS Access could read it well. This did not give me the flexibility with the data that I required (due to my inexperience with databases), so I returned to simple perl and shell scripts to meet my needs. I also came to realize that there was a much better way to divide the data - into scans and alerts. I did this and began my analysis.

First I had to get an idea of what type and how many attacks MY.NET was seeing. I remembered running along snort_stat long ago and went looking for it again. With some modification it gave me 6 reports that I could use. Table #1 is one of them, showing information of the number and types of attacks encountered. Table #2 was also generated from this same method. Table #3 was output from snort_stat as well with only MY.NET records input into it. Table #4 and #5 were also the result of snort_stat. The information is table #6 was gleaned from perl scripts while snort_stat was used for table #7 as well. Tables #8,9,10 were all generated from the same script that generated table #6. The scripts I used follow.

This is the modified snort_stat.pl I used: Only relavent portions included.

```
# process whatever comes in
while (<>)
{
      if ($opt_a)
      {
            # process data from a snort alert file
            chomp();
            # if the line is blank, go to the next one

            if ( $_ eq "" )
            {
                  next;
            }

            $a = <>;
            chomp($a);

            $sig = $_;
            $a =~ m/^(\d+)\/(\d+)\-
(\d+)\:(\d+)\:(\d+\.\d+).*\](.*)\[\*\*\]\s(\w{1,16}\.\w{1,16}\.\d{1,3}\.\d{1,3}):(\d{1,5}
)\s\-\>\s(\w{1,16}\.\w{1,16}\.\d{1,3}\.\d{1,3}):(\d{1,5})/ox;

            $month=$1; $day=$2; $hour=$3; $minute=$4; $second=$5; $sig=$6; $saddr=$7;
$host="localhost"; $sport=$8; $daddr=$9; $dport=$10;

#           print "month: $1\tday: $2\thour: $3\tminute: $4\tsecond: $5\tsip: $7\t
attack: $6\t sprt: $8\tdip: $9\tdport: $10\n";
      }
```

I used the following perl scripts throughout assignment #3 for various purposes - they were frequently modified to suite any individual need.

```
#!/usr/bin/perl

#system("cat rawdata/Snort* | egrep -v \"spp_portscan\" > data-attacks");

open (allunsorted, ">allunsorted");
open (uniqed, "uniqed");
open (data, "data-portscan");

while(<data>)
{
        if ($_ =~ m/.*from (MY\.NET\.\d{1,3}\.\d{1,3}).*\n/)
        {
                print allunsorted "QQQ$1QQQ\n";
        }
}

system("cat allunsorted | sort > allsorted");
system("cat allsorted | uniq > uniqed");

while(<uniqed>)
{
      $ip = $_;
      chomp($ip);
      $count = 0;

        system("echo \"$ip -- \`cat allsorted | grep $ip | wc -l\`\" >> results");
}
```

```
#~/usr/bin/tcsh

cat Snort* | grep status > data-portscan
awk -F"from|:" '{print $5}' data-portscan > ssip2
cat ssip2 | sort | uniq > ssip
./ssip-sort.pl


#!/usr/bin/perl

$DEBUG = 0;

open (finaldata, ">scanners");
open (ssip, "ssip");

while(<ssip>)
{
        open (data, "ssip2");
        $ip = $_;
        chomp($ip);
        $count = 0;

        while(<data>)
        {
#               print "IP is $ip";
#               print "\$_ is $_";
#               print "count is $count\n";

                if (/$ip/)
                {
                        $count++;
#                       print "$ip\t\t\t$count\n";
                }
        }
        print finaldata "$ip -- $count\n";
}
```

```
#!/usr/bin/perl

#system("cat rawdata/Snort* | egrep -v \"spp_portscan\" > data-attacks");

open (allsorted, "allsorted");
open (allunsorted, ">allunsorted");
open (uniqed, "uniqed");
open (data, "data-individual");
open (finaldata, ">attackers");

while(<data>)
{
        if($_ =~ m/.*\](.*)\[\*\*\] (\w{1,16}\.\w{1,16}\.\d{1,3}\.\d{1,3}):(\d{1,5}) ->
(\w{1,16}\.\w{1,16}\.\d{1,3}\.\d{1,3}):(\d{1,5}).*\n/)
        {
                print allunsorted "$3\n";
        }

        if ($_ =~ m/.*\s(\w{1,16}\.\w{1,16}\.\d{1,3}\.\d{1,3}):(\d{1,5}) ->
(\w{1,16}\.\w{1,16}\.\d{1,3}\.\d{1,3}):(\d{1,5}).*\n/)
        {
                print allunsorted "QQQ$4QQQ\n";
        }
}

system("cat allunsorted | sort > allsorted");
system("cat allsorted | uniq > uniqed");

while(<uniqed>)
{
      $ip = $_;
      chomp($ip);
      $count = 0;

      open (allsorted, "allsorted");
      while(<allsorted>)
      {
            if (/$ip/)
            {
                        $count++;
            }
      }
     print finaldata "$ip -- $count\n";
}

#!/usr/bin/perl

open (temp, ">results");

while(<temp>)
{
```

```
        $_ =~ tr/Q//d;
}
```

```
#!/usr/bin/perl

open (finaldata1, ">1");
open (data, "data-portend");
open (finaldata2, ">sorted-portscan1");

while(<data>)
{

        if($_ =~ m/.*from (\w{1,16}\.\w{1,16}\.\d{1,3}\.\d{1,3}).*\:(\d+).*\:(\d+).*\:(\d+).*/)
        {
                print finaldata1 "$1\n";
                print finaldata2 "$1\t$2\t$3\t$4\n";
        }
}

system("cat 1 | sort | uniq > 2");

open (data1, "2");
open (final, ">sorted-portfinal");

while(<data1>)
{
        open (data2, "sorted-portscan1");

        $hosts=0;
        $tcp=0;
        $udp=0;

        if($_ =~ m/(\w{1,16}\.\w{1,16}\.\d{1,3}\.\d{1,3}).*/)
        {
                $sip1 = $1;

                while(<data2>)
                {
                        if($_ =~ m/(\w{1,16}\.\w{1,16}\.\d{1,3}\.\d{1,3})\t(\d+)\t(\d+)\t(\d+).*/)
                        {
                                $sip2 = $1;
                                $hosts2 = $2;
                                $tcp2 = $3;
                                $udp2 = $4;
                        }

                        if($sip1 =~ /$sip2/)
                        {
                                $hosts += $hosts2;
                                $tcp += $tcp2;
                                $udp += $udp2;
                        }
                }
                print final "$sip1\t$hosts\t$tcp\t$udp\n";
```

```
        }
}
#!/usr/bin/perl

$DEBUG = 0;

open (finaldata, ">sorted-snort.sql");
open (data, "data-attacks");


while(<data>)
{

        if($_ =~ m/(^\d{2}\/\d{2})-(\d{2}:\d{2}:\d{2}\.\d{6}).*\](.*)\[\*\*\]
(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}):(\d{1,5}) -> (\w{1,16}\.\w{1,16}\.\d{1,3}\.\d{1,3}):(\d{1,5}).*\n/)
        {
                print "date is $1\n";
                print "time is $2\n";
                print "event is $3\n";
                print "src ip is $4\n";
                print "src port is $5\n";
                print "dst ip is $6\n";
                print "dst port is $7\n\n";
                print finaldata "$1\t$2\t$3\t$4\t$5\t$6\t$7\n"
        }
}
```