NetSA Final Project Report

December 8, 2009

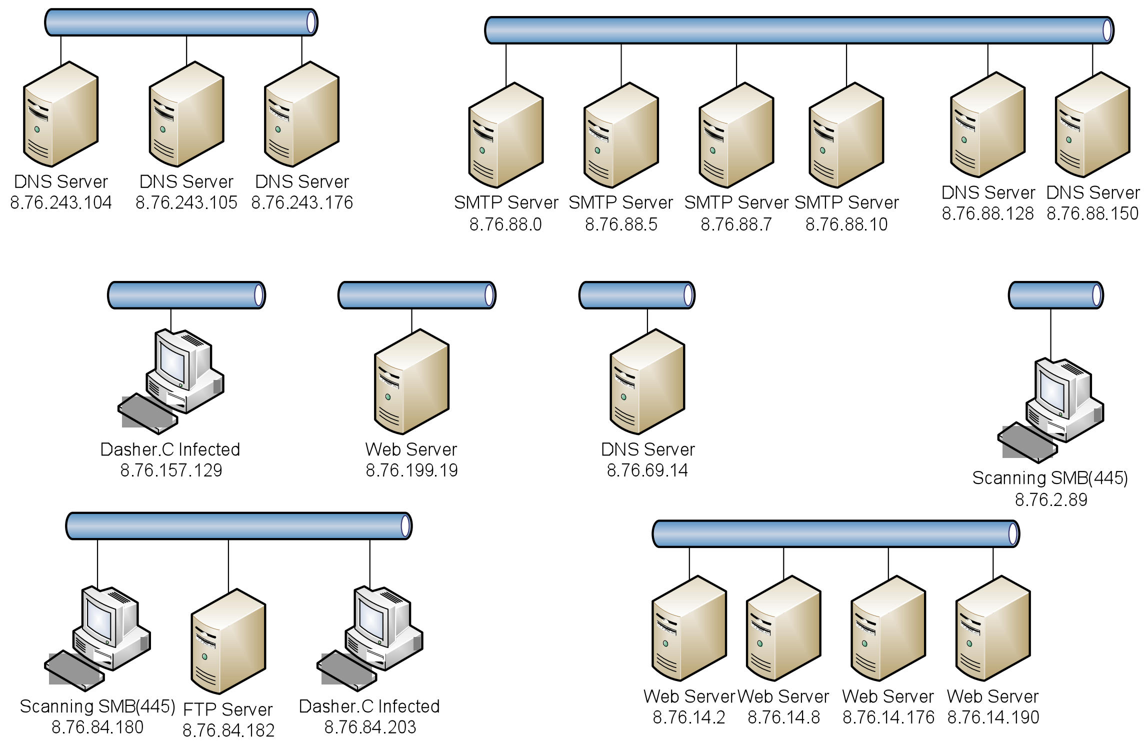
Phil Burdette, Ryan Easton, Zack Loether, Derrick Spooner

Table of Contents

Event Summary

The class B subnet that we analyzed was 8.76.0.0/16. During the course of the data collection period, this net block saw 25,457 unique traffic flows composed of 384,211 individual packets. This particular class B network was chosen for analysis over the others because of its size. It is large enough the provide interesting analysis points without being so huge that there would be too much data to analyze. Based upon the analysis conducted, we have come to the conclusion that this net block is probably entirely, or at least largely comprised of a corporate network. Some of our key findings included SMB scanning and possible worm infection. This was evident by performing flow analysis on the data packets that were captured. For an event analysis, we compared out net block's activity to a DDoS attack of a Belarusian news site. A similar attack on our network would be extremely noticeable do to the significant increase in web traffic to our web servers.

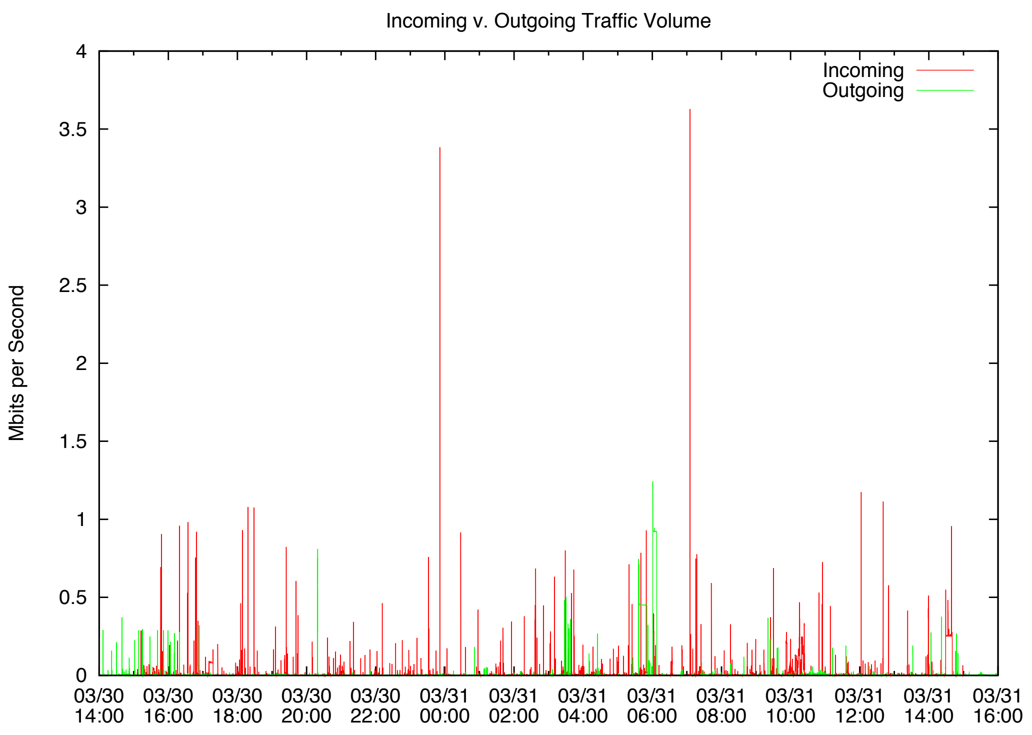
Network Profile



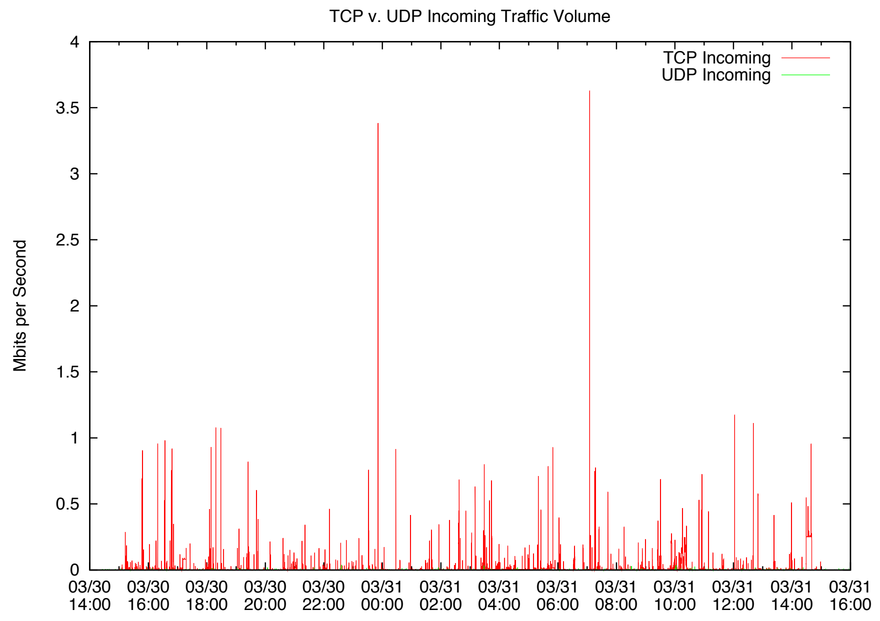
The network diagram above divides some of the top-talkers and notable hosts in our /16 into /24 blocks. We concluded that our /16 consists primarily of business hosts based on several observations. The most obvious feature of the data is that the bulk of the traffic volume is to and from web servers located in our subnet. The analysis also revealed that other top services were DNS and SMTP, which is mostly indicative of a business network. However, the traffic pattern does not follow any obvious workday so the web servers could be potentially international customer facing sites. Moreover, the subnet does not contain any obvious BitTorrent, P2P, gaming or instant messaging flows, which lends further evidence that it is a corporate network.

Traffic Summary

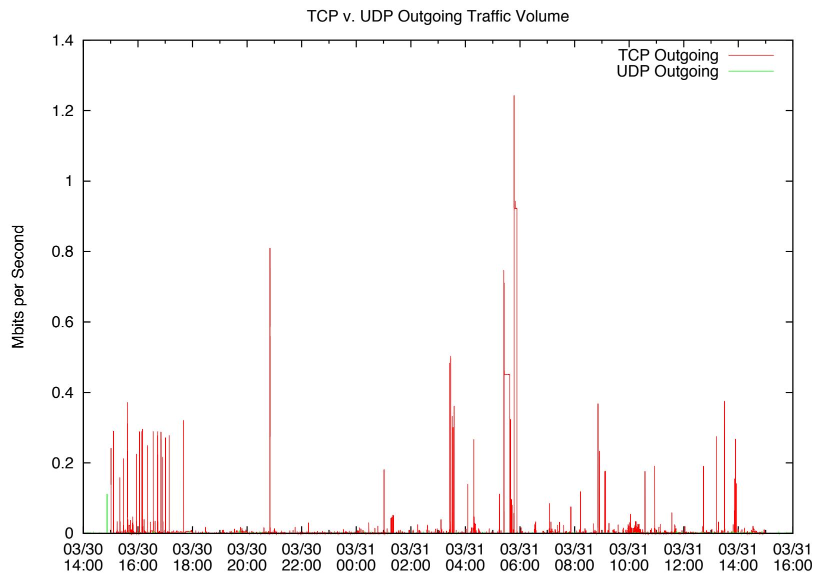
Below depicts a graph that compares the total volume of incoming traffic to the total volume of outgoing traffic. We see large spikes in the amount of incoming traffic at 12:00 Am and at approximately 7:00 AM.



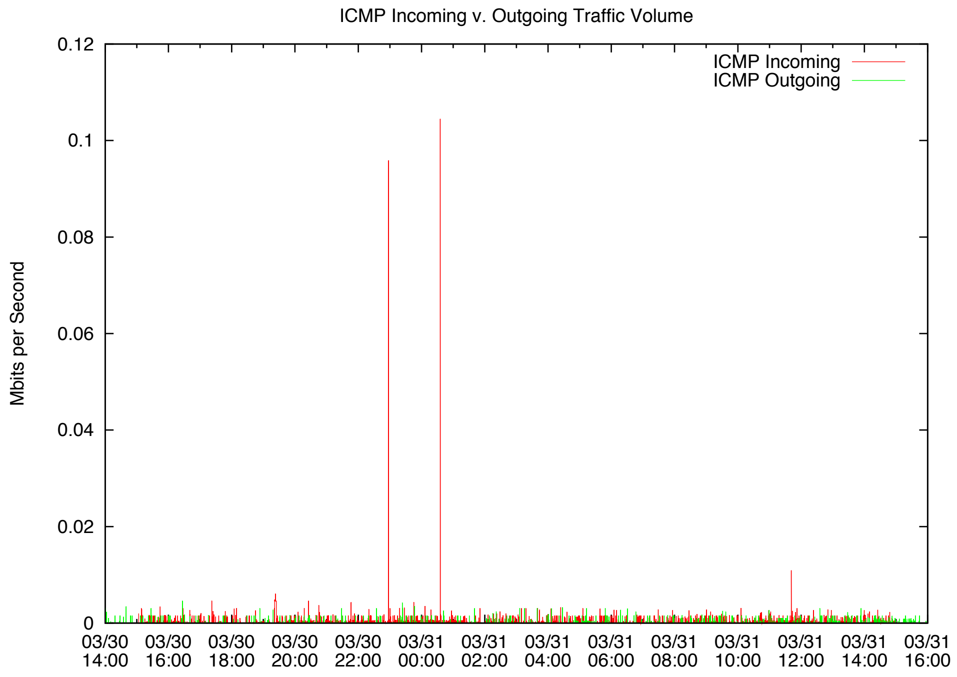
When we compare TCP v. UDP incoming traffic volume, there is a large amount of TCP traffic compared to UDP. The large spikes in overall data traffic seen above can be linked directly to TCP traffic. Comparing the Mbits per Second between the two graphs we see that they are approximately the same.



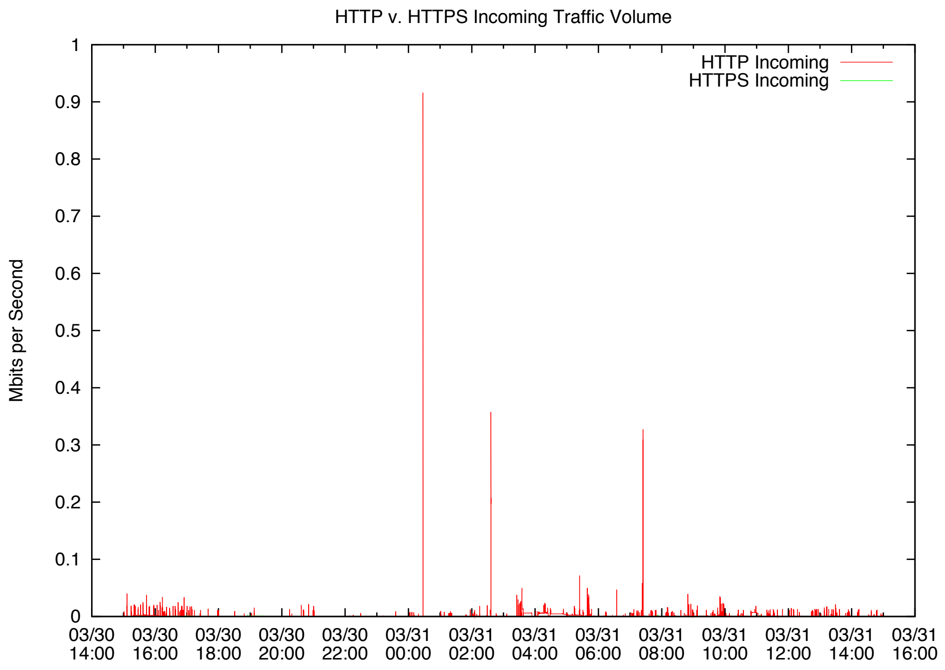
Outgoing TCP and UDP traffic differs greatly from incoming, as shown in the graph below. The amount of TCP outgoing traffic spikes at approximately 9:00 PM and then again around 6:00 AM. There are additional spikes around 6:00 AM leading us to believe a lot of downloading was occurring at that particular time.



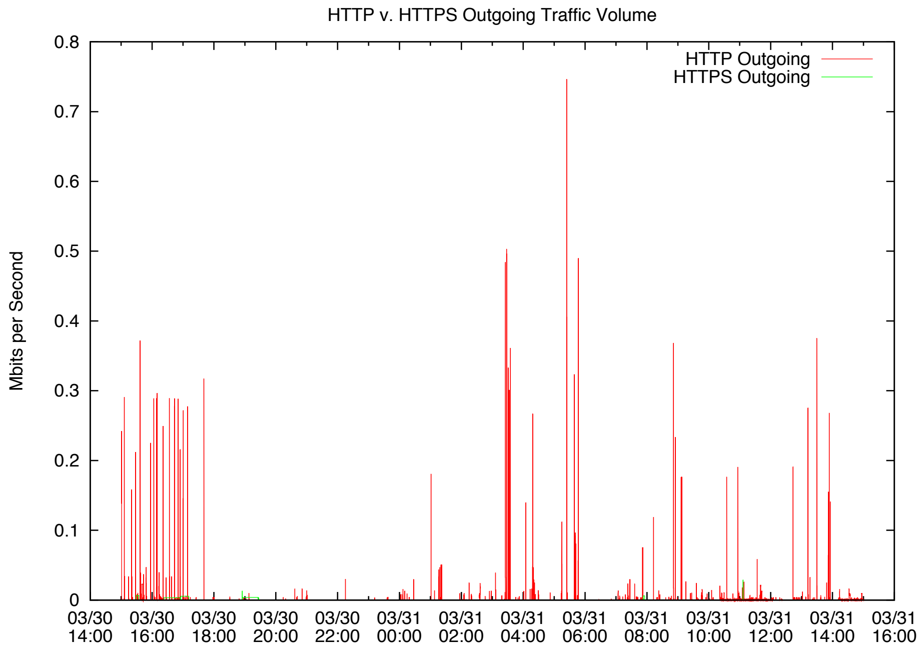
When we compare incoming ICMP traffic to outgoing, they both tend to create the same amount of traffic with the exception of two times throughout the day. At approximately 11:00 PM and 1:00 AM, the amount of inbound ICMP traffic spikes. We hypothesize that ping scanning was occurring and the firewall dropped all ICMP responses. That is why we do not see the same spikes in outgoing ICMP traffic.



Comparing the traffic volume of incoming HTTP traffic to HTTPS traffic we are able to identify very little HTTPS traffic and 3 spikes in HTTP traffic. The spikes in HTTP traffic occur at approximately 12:30 AM, 2:30 AM, and 7:30 AM.



The graph below analysis the traffic volume of outgoing HTTP and HTTPS in order to find any spikes in the data flow. We have identified quiet a few spikes in the amount of outgoing HTTP traffic, while HTTPS traffic is very minimal. There are many spikes between 3:00 PM and 6:00 PM, and then again at 3:30 PM and a little before 6:00 PM. The spikes in outgoing traffic are not aligned with the spikes in incoming traffic.



Web Traffic

One of the most common types of traffic on the Internet is that used by the World Wide Web. A corporate network is no different. The most commonly used ports for web traffic are TCP ports 80 and 443. Alternative ports exist, which are 8080 and 8443. In our netblock, however, we did not find any instances of traffic flows on either of the alternative ports. Most of the web traffic was on port 80. This is the most commonly used port for web traffic. Port 443 is the standard port for HTTPS. Many corporate networks include servers using HTTPS to allow employees and clients to securely access services.

```
$ rwfilter *.rw --dport=80 --daddr=8.76.0.0/16 --protocol=6 --pass=stdout | rwuniq
--fields=sip,dip,sport,dport,bytes,packets | sort -nr -t '|' -k 5 | head
rwuniq: Warning: Using default temporary directory /tmp
 144.44.24.208|   8.76.14.190|54087|   80|   873912|    20215|        1|
 144.44.24.208|   8.76.14.190|41750|   80|   804201|    18597|        1|
 144.44.24.208|   8.76.14.190|58891|   80|   707652|    16532|        1|
 144.44.24.208|     8.76.14.8|59326|   80|   522728|    12222|        1|
 144.44.24.208|     8.76.14.2|60592|   80|   512926|    12047|        1|
 144.44.102.38|   8.76.14.176|53909|   80|   491239|    11139|        1|
 173.94.111.43|   8.76.199.19| 1447|   80|   364438|      444|        1|
...
```

```
$ rwfilter *.rw --dport=443 --daddr=8.76.0.0/16 --protocol=6 --pass=stdout |
rwuniq --fields=sip,dip,sport,dport,bytes,packets | sort -nr -t '|' -k 5 | head
rwuniq: Warning: Using default temporary directory /tmp
 192.42.134.97|   8.76.148.13| 2630|  443|   122728|     2623|        1|
 192.42.134.97|   8.76.148.33| 4830|  443|   111552|     2270|        1|
 192.42.134.97|    8.76.148.4| 1710|  443|    62196|     1270|        1|
...
```

Port 83/TCP

Port 83 is not a normally used port on most networks. However, in our subnet the two flows with the higher amount of traffic originate from out net block and have a port of 83/TCP. Internet research found that the only service known to use TCP port 83 is known as mit-ml-dev. Unfortunately, the name of the service is all that is known. However, there is nothing stopping a system administrator from running services on non-default ports. There is a likely possibility that this service running on port 83 is actually a common protocol such as HTTP.

```
$ rwfilter *.rw --protocol=6 --pass=stdout | rwuniq --
fields=sip,dip,sport,dport,bytes,packets | sort -nr -t '|' -k 5 | head

    8.76.14.3|  192.42.30.184|   83| 4315|  43231477| 30769|FS PA   | 1|
   8.76.14.176|  192.42.30.184|   83| 4095|  43153717| 30712|FS PA   | 1|
  193.52.224.3|    8.76.12.249|   80|17931|  21081097| 14373| S PA   | 1|
 173.94.202.220|   8.76.177.166|   80|46912|   4120360|  2910|FS PA   | 1|
 193.52.237.155|    8.76.12.242|64219|26084|   2673276|  1898| S PA   | 1|
 149.162.10.222|   8.76.146.163|   80|55146|   2672034|  1794|FS PA   | 1|
...
```

Mail

Outgoing mail uses the default TCP port of 25 using SMTP. There is both mail coming into this netblock and leaving. There is, however, no traffic using POP3 or IMAP, the protocols used to retrieve email from a server. This is in keeping with the assertion that this netblock is a corporate network. A properly configured corporate network will show incoming and outgoing SMTP traffic from email going into and coming out of the network. However, it makes sense that there is no visible POP3 or IMAP flows because they will all be internal and behind a firewall.

```
$ rwfilter *.rw --dport=25 --daddr=8.76.0.0/16 --protocol=6 --pass=stdout | rwuniq
--fields=sip,dip,sport,dport,bytes,packets | sort -nr -t '|' -k 5 | head
rwuniq: Warning: Using default temporary directory /tmp
   144.44.34.4|   8.76.88.109|56716|   25|    16424|       20|        1|
   144.44.34.4|    8.76.88.68|61327|   25|     9942|       16|        1|
   144.44.34.4|    8.76.88.68|60466|   25|     5909|       13|        1|
 149.162.10.177|   8.76.217.226|56214|   25|     5797|       29|        1|
```

…

```
$ rwfilter *.rw --sport=25 --saddr=8.76.0.0/16 --protocol=6 --pass=stdout | rwuniq -
-fields=sip,dip,sport,dport,bytes,packets | sort -nr -t '|' -k 5 | head
rwuniq: Warning: Using default temporary directory /tmp
   8.76.88.10|149.101.220.128|   25|54274|      892|       14|        1|
   8.76.88.10| 193.52.228.35|   25|36532|      656|       10|        1|
    8.76.88.7| 193.52.228.35|   25|36541|      656|       10|        1|
    8.76.88.0| 149.92.211.34|   25|26539|      546|       10|        1|
```
…

File Sharing

NFS

The Network File System protocol, created by Sun Microsystems in 1984, allows users to access files across a network (RFC 3530[1]). We looked to see if we could locate any of this traffic on our class B address block sending or receiving information on port 2049.

```
[pfb@unix37 ext1]$ rwfilter *.rw --any-address=8.76.0.0/16 --print-volume-stat --
aport=2049 --pass=stdout | rwstats --count=100 --sip --dip
          sIP|           dIP|           Records|%_of_total|   cumul_%|
   8.76.43.42|   133.44.82.73|                 3| 75.000000| 75.000000|
  8.76.81.239|   133.44.82.73|                 1| 25.000000|100.000000|
```

We found two hosts, namely 8.76.43.42 and 8.76.81.239, sending packets over the NFS protocol. Digging down in more detail, we found TCP packets being sent to the NFS port on other machines.

```
[pfb@unix37 ext1]$ rwfilter *.rw --any-address=8.76.0.0/16 --print-volume-stat --
aport=2049 --pass=stdout | rwcut --fields=sip,dip,sport,dport,protocol,packets,dur
sIP|           dIP|sPort|dPort|pro|   packets|     dur|
   8.76.43.42|   133.44.82.73|   80| 2049|  6|         1|    0.000|
   8.76.43.42|   133.44.82.73| 7400| 2049|  6|         1|    0.000|
   8.76.43.42|   133.44.82.73| 7000| 2049|  6|         1|    0.000|
  8.76.81.239|   133.44.82.73| 3389| 2049|  6|         1|    0.000|
```

The flow that is most interesting is the host 8.76.43.42 sending traffic from port 80 over NFS.

SMB

[1] http://tools.ietf.org/html/rfc3530

We also explored the application layer network protocol Server Message Block, which is used to users to files and printers. Because SMB is a Microsoft protocol, we were unable to locate an official RFC.

```
[pfb@unix38 ext2]$ rwfilter *.rw --any-address=8.76.0.0/16 --print-volume-stat --
aport=445 --pass=stdout | rwstats --count=100 --sip --dip
         sIP|          dIP|          Records|%_of_total|   cumul_%|
           ...
  8.76.84.180|   133.44.225.4|              1|  1.449275| 62.318841|
   8.76.84.180|  133.44.220.46|              1|  1.449275| 63.768116|
   8.76.84.180| 133.44.151.132|              1|  1.449275| 65.217391|
   8.76.84.180|  133.44.164.30|              1|  1.449275| 66.666667|
   8.76.84.180| 133.44.217.208|              1|  1.449275| 68.115942|
   8.76.84.180|   133.44.57.76|              1|  1.449275| 69.565217|
   8.76.84.180|   133.44.253.2|              1|  1.449275| 71.014493|
           ...
```

In our netblock, we found a lot of SMB traffic originating from host 8.76.84.180. Exploring the host, we found the following:

```
pfb@unix36 ext2]$ rwfilter *.rw --any-address=8.76.84.180 --print-volume-stat --
aport=445 --pass=stdout | rwcut --fields=sip,dip,sport,dport,protocol,packets,dur
         sIP|          dIP|sPort|dPort|pro|   packets|     dur|
    |          Recs|         Packets|           Bytes|    Files|
   8.76.84.180|  133.44.111.72| 2020|  445|  6|         2|    2.933|
   8.76.84.180|  133.44.192.56| 2567|  445|  6|         2|    2.980|
   8.76.84.180|   133.44.57.76| 1889|  445|  6|         2|    2.946|
   8.76.84.180|  133.44.60.244| 1696|  445|  6|         2|    2.825|
   8.76.84.180| 133.44.151.132| 3176|  445|  6|         2|    3.014|
   8.76.84.180|  133.44.156.38| 3904|  445|  6|         2|    2.889|
   8.76.84.180|  133.44.81.234| 2237|  445|  6|         2|    2.870|
   8.76.84.180|   133.44.253.2| 3880|  445|  6|         2|    2.984|
   8.76.84.180|   133.44.37.92| 1508|  445|  6|         2|    3.002|
   8.76.84.180|   133.44.225.4| 3520|  445|  6|         2|    2.918|
   8.76.84.180| 133.44.188.229| 3013|  445|  6|         2|    3.000|
   8.76.84.180|  133.44.170.65| 3575|  445|  6|         2|    2.972|
   8.76.84.180|  133.44.247.82| 3578|  445|  6|         2|    2.982|
   8.76.84.180| 133.44.177.110| 1274|  445|  6|         2|    2.907|
   8.76.84.180|  133.44.164.30| 1529|  445|  6|         2|    2.960|
   8.76.84.180|  133.44.227.77| 3505|  445|  6|         2|    2.928|
   8.76.84.180|  133.44.111.27| 3985|  445|  6|         2|    3.014|
   8.76.84.180|  133.44.220.46| 2108|  445|  6|         2|    2.941|
   8.76.84.180|   133.44.47.32| 4226|  445|  6|         2|    2.963|
   8.76.84.180|   133.44.30.71| 3747|  445|  6|         2|    2.978|
   8.76.84.180| 133.44.217.208| 4264|  445|  6|         2|    2.992|
   8.76.84.180|  133.44.77.118| 4686|  445|  6|         2|    2.965|
   8.76.84.180|  133.44.170.69| 1800|  445|  6|         2|    2.970|
   8.76.84.180| 133.44.145.119| 3388|  445|  6|        2|    2.970|
   8.76.84.180| 133.44.184.130| 3275|  445|  6|        2|    2.942|
```

The host is attempting to connect to SMB ports on other machines outside of its network. The rate at which the packets are being sent implies that 8.76.84.180 is performing horizontal scanning on port 445.

```
[pfb@unix35 public]$ rwfilter *.rw --saddress=8.76.84.180 --print-volume-stat --
flags-all=S/SARF --pass=stdout | rwcut --
fields=sip,dip,sport,dport,protocol,packets,dur,flags
         sIP|          dIP|sPort|dPort|pro|   packets|     dur|   flags|
   8.76.84.180|  133.44.167.58| 4575|  445|  6|        2|    3.012| S      |
   8.76.84.180|  133.44.119.79| 4449|  445|  6|        2|    2.847| S      |
   8.76.84.180|192.177.166.219| 1155|  445|  6|        2|    2.923| S      |
          ...
```

A packet size of two indicates that the victim is responding to the TCP requests with indicating to the host that the port is open or closed. Here we see that the port is closed because an ICMP packet with destination port 771 is being returned. Port 771 indicates that ICMP is Destination Unreachable, Port Unreachable, because the decimal translation of the ICMP packet appears in the destination port field of the flow record.

```
[pfb@unix36 ext1]$ rwfilter *.rw --daddress=8.76.84.180 --print-volume-stat
--pass=stdout | rwcut
--fields=sip,dip,sport,dport,protocol,packets,dur,flags

                      sIP|     dIP|sPort|dPort|pro|packets|   dur|
133.44.167.58 |  8.76.84.180|    0|  771|  1|      2| 3.023|
133.44.119.79 |  8.76.84.180|    0|  771|  1|      2| 2.848|
133.44.125.234|  8.76.84.180|    0|  771|  1|      2| 2.953|
133.44.218.102|  8.76.84.180|    0|  771|  1|      2| 3.011|
133.44.206.179|  8.76.84.180|    0|  771|  1|      2| 2.999|
133.44.101.39 |  8.76.84.180|    0|  771|  1|      2| 2.972|
          ...
```

Expanding our search for the host in the first group of flows, we find host 8.76.2.89 is also performing horizontal port scanning on port 445.

```
[pfb@unix36 ext1]$ rwfilter *.rw --any-address=8.76.2.89 --print-volume-stat --
aport=445 --pass=stdout | rwcut --fields=sip,dip,sport,dport,protocol,packets,dur
         sIP|          dIP|sPort|dPort|pro|   packets|     dur|
     8.76.2.89|    133.44.2.83| 2688|  445|  6|        2|    2.925|
     8.76.2.89| 133.44.212.212| 2001|  445|  6|        2|    2.929|
     8.76.2.89|   133.44.65.27| 1116|  445|  6|        2|    2.950|
     8.76.2.89|    133.44.61.6| 4097|  445|  6|        2|    3.014|
     8.76.2.89|  133.44.202.13| 1156|  445|  6|        2|    3.020|
   8.76.2.89| 133.44.138.254| 2553|  445|  6|       2|   3.013|
         ...
```

Similar to above, we see responses from destination port 771 from the victims indicating that the port is closed. We were unable to find any evidence that any of the ports on the victims' machines were open.

AFP

The network protocol, Apple Filing Protocol, is a file service for Mac OS X. Looking at the flow records from our network we see that one host is using AFP.

```
[pfb@unix38 ext2]$ rwfilter *.rw --any-address=8.76.0.0/16 --print-volume-stat --aport=548 --pass=stdout | rwstats --count=100 --sip --dip
           sIP|            dIP|           Records|%_of_total|   cumul_%|
   8.76.84.224|  133.44.194.65|                 1|100.000000|100.000000|
```

Learning more about 8.76.84.224, we determine the following.

```
[pfb@unix38 ext2]$ rwfilter *.rw --any-address=8.76.0.0/16 --print-volume-stat --aport=548 --pass=stdout | rwcut --fields=sip,dip,sport,dport,protocol,packets,dur
           sIP|            dIP|sPort|dPort|pro|   packets|      dur|
    |            Recs|         Packets|             Bytes|     Files|
   8.76.84.224|  133.44.194.65|   80|  548|  6|         1|    0.000|
```

The host sent one packet of TCP traffic from port 80 to port 445 on another computer. This is interesting in that web traffic is attempting to connect to AFP.

Non-Internet Traffic

NetBIOS

The Network Basic Input/Output system is related to the session layer and runs over TCP or UDP in order to allow computers to communicate over a LAN (RFC 1001[2]). Different NetBIOS functions use different ports and protocols. For example, nbname will use port 137 and either TCP or UDP, nbdatagram will use port 138 over UDP, and nbsession will use port 139 over TCP.

```
[pfb@unix37 ext1]$ rwfilter *.rw --any-address=8.76.0.0/16 --print-volume-stat --aport=137,138,139 --pass=stdout | rwstats --count=100 --sip --dip
           sIP|            dIP|           Records|%_of_total|   cumul_%|
    8.76.65.91|  174.63.104.45|                 1|100.000000|100.000000|
```

One host on our network sent one netbios record to another host. We need more information to determine what protocol and what port were used.

[2] http://tools.ietf.org/html/rfc1001

```
[pfb@unix37 ext1]$ rwfilter *.rw --any-address=8.76.0.0/16 --print-volume-stat --
aport=137,138,139 --pass=stdout | rwcut --
fields=sip,dip,sport,dport,protocol,packets,dur
         sIP|          dIP|sPort|dPort|pro|   packets|     dur|
    |           Recs|         Packets|            Bytes|     Files|
    8.76.65.91|  174.63.104.45|  139| 3544|  6|        3|    8.301|
```

Here we see that the nbsession function was called because there was TCP traffic sent to from port 139. This is suspicious because netbios traffic should not leave the LAN for security reasons. (Note: There are some exceptions.)

Remote Control

RDP

The Remote Desktop Protocol, developed my Microsoft, allows a user to control another computer through a graphical interface. This is particularly common inside of LANs where administrators monitor servers remotely, as well as employee workstations.

```
[pfb@unix37 ext1]$ rwfilter *.rw --any-address=8.76.0.0/16 --print-volume-stat --
aport=3389 --pass=stdout | rwstats --count=100 --sip --dip
         sIP|          dIP|           Records|%_of_total|   cumul_%|
 144.44.240.121|    8.76.81.239|              449|  6.114667|  6.114667|
  144.44.188.89|    8.76.81.239|                3|  0.040855|  6.155522|
    8.76.81.239| 192.177.175.11|                3|  0.040855|  6.196378|
    8.76.81.239|  133.44.81.217|                3|  0.040855|  6.237233|
    8.76.81.239| 133.44.155.104|                3|  0.040855|  6.278088|
    8.76.81.239|   133.44.39.77|                3|  0.040855|  6.318943|
    8.76.81.239|   133.44.29.34|                3|  0.040855|  6.359798|
           ...
```

Here we see that host 8.76.81.239, which is inside of our network, is communicating with host 144.44.240.121, which is outside of our class B address space.

VNC

Another graphical desktop sharing system, Virtual Network Computing, uses the RFB protocol to allow users to remotely control hosts. RFB stands for the "remote framebuffer" and was designed as a simple protocol to facilitate cross platform remote communication.

```
[pfb@unix38 ext1]$ rwfilter *.rw --any-address=8.76.0.0/16 --print-volume-stat --
aport=5800,5900 --pass=stdout | rwstats --count=100 --sip --dip
         sIP|          dIP|           Records|%_of_total|   cumul_%|
   8.76.81.239|   133.44.33.60|              1|100.000000|100.000000|
```

Host 8.76.81.239 is communicating with host 133.44.33.60 over VNC. We must examine the host more closely to learn more.

```
[pfb@unix38 ext1]$ rwfilter *.rw --any-address=8.76.0.0/16 --print-volume-stat --
aport=5800,5900 --pass=stdout | rwcut --
fields=sip,dip,sport,dport,protocol,packets,dur
         sIP|          dIP|sPort|dPort|pro|   packets|     dur|
   8.76.81.239|   133.44.33.60| 3389| 5900|  6|         1|    0.000|
```

The communication originated from our class B network and was sent to port 5900. No other VNC communication was found in any of the flow data.

Messaging Services

Messaging communications could help to profile the network in some ways. Seeing very large amounts of traffic for these protocols could imply that many of the IP addresses belong to personal users. Therefore, we determined the different port numbers used by many common messaging programs and looked for their traffic. The commands and results follow.

Service	Port Number
MSN Messenger (messages)	1863
MSN Messenger (voice)	6901
MSN File Transfers	6891-6900
AIM	5190-5193
Yahoo Messenger (messages)	5050
Yahoo Messenger (voice)	5000, 5001
XMPP (including Google Talk)	5222
IRC	6667

```
$ rwfilter ext1/*.rw --any-address=8.76.0.0/16 --print-volume-stat --
aport=1863,6891,6892,6893,6894,6895,6896,6897,6898,6899,6900,6901,5190,51
91,5192,5193,5000,5001,5050,5222,6667 --pass=stdout | rwcut --
fields=sip,dip,sport,dport,protocol,packets,dur
         sIP|          dIP|sPort|dPort|pro|   packets|     dur|
    |           Recs|         Packets|             Bytes|     Files|
Total|        104723184|        918568027|        656996415962|        16|
 Pass|                8|               21|                1099|          |
 Fail|        104723176|        918568006|        656996414863|          |
    8.76.43.42|  133.44.255.88|   80| 6667|  6|         1|    0.000|
    8.76.43.42|  133.44.255.88| 7400| 6667|  6|         1|    0.000|
    8.76.43.42|  133.44.255.88| 7000| 6667|  6|         1|    0.000|
    8.76.43.42|  133.44.255.88|   80| 6667|  6|         1|    0.000|
   8.76.12.242|  174.63.204.30|49101| 1863| 17|        13|   52.797|
   8.76.81.239| 133.44.172.199| 3389| 5192|  6|        1|    0.000|
   8.76.81.239|  133.44.228.37| 3389| 1863|  6|        1|    0.000|
   8.76.134.64|   133.44.220.6| 1863|  445|  6|        2|    2.820|
```

```
$ rwfilter ext2/*.rw --any-address=8.76.0.0/16 --print-volume-stat --
aport=1863,6891,6892,6893,6894,6895,6896,6897,6898,6899,6900,6901,5190,51
91,5192,5193,5000,5001,5050,5222,6667 --pass=stdout | rwcut --
fields=sip,dip,sport,dport,protocol,packets,dur
         sIP|          dIP|sPort|dPort|pro|   packets|     dur|
    |            Recs|         Packets|             Bytes|     Files|
Total|        64224058|        625286420|       426480594766|        8|
 Pass|               2|                4|                471|         |
 Fail|        64224056|        625286416|       426480594295|         |
   8.76.69.161|  133.44.255.88|   80| 6667|  6|        1|    0.000|
   8.76.12.240| 149.92.143.230|44656| 1863| 17|        3|   84.884|
```

As we can see, there were some instances of communication for MSN Messenger, which operates on UDP protocols, as well as plenty of IRC traffic. The IRC traffic exists mostly in older applications, so it seems that there are several older systems within our network.

Database Services

In addition to many of the other services, it is also important to observe the database services that may be occurring through our network. Activity on these ports will expose different servers that may be used for businesses within our networks. Therefore, we searched for and documented the results for the services being used for SQL Server, Oracle as well as MySQL.

SQL Server[3]

Services	Ports
SQL Server	156, 1433, 1434
SQL Services	118
Analysis Server	2725
OLAP Services	2393, 2394

```
$ rwfilter ext1/*.rw --any-address=8.76.0.0/16 --print-volume-stat --
aport=156,1433,1434,118,2725,2393,2394 --pass=stdout | rwcut --
fields=sip,dip,sport,dport,protocol,packets,dur
         sIP|          dIP|sPort|dPort|pro|   packets|     dur|
```

[3]http://www.microsoft.com/smallbusiness/support/articles/ref_net_ports_ms_prod.mspx

```
    |          Recs|        Packets|            Bytes|    Files|
Total|       104723184|       918568027|      656996415962|       16|
 Pass|             17|             17|              688|        |
 Fail|       104723167|       918568010|      656996415274|        |
   214.75.58.59|   8.76.84.203| 1433| 6000|  6|        1|   0.000|
   214.75.58.61|   8.76.84.203| 1433| 6000|  6|        1|   0.000|
   214.75.58.90|   8.76.84.203| 1433| 6000|  6|        1|   0.000|
   214.75.59.24|   8.76.84.203| 1433| 6000|  6|        1|   0.000|
   214.75.59.19|   8.76.84.203| 1433| 6000|  6|        1|   0.000|
   214.75.59.20|   8.76.84.203| 1433| 6000|  6|        1|   0.000|
   214.75.59.32|   8.76.84.203| 1433| 6000|  6|        1|   0.000|
    214.75.59.7|   8.76.84.203| 1433| 6000|  6|        1|   0.000|
   214.75.59.36|   8.76.84.203| 1433| 6000|  6|        1|   0.000|
   214.75.59.80|   8.76.84.203| 1433| 6000|  6|        1|   0.000|
   214.75.59.89|   8.76.84.203| 1433| 6000|  6|        1|   0.000|
  214.75.56.214|   8.76.84.203| 1433| 6000|  6|        1|   0.000|
  193.66.118.10|  8.76.157.129| 1433| 6000|  6|        1|   0.000|
 215.149.60.129|   8.76.84.203| 1433| 6000|  6|        1|   0.000|
 215.149.60.128|   8.76.84.203| 1433| 6000|  6|        1|   0.000|
  193.66.118.10|  8.76.157.129| 1433| 6000|  6|        1|   0.000|
    8.76.81.239| 133.44.86.168| 3389|  118|  6|        1|   0.000|
```

```
$ rwfilter ext2/*.rw --any-address=8.76.0.0/16 --print-volume-stat --
aport=156,1433,1434,118,2725,2393,2394 --pass=stdout | rwcut --
fields=sip,dip,sport,dport,protocol,packets,dur
            sIP|            dIP|sPort|dPort|pro|   packets|      dur|
    |          Recs|        Packets|            Bytes|    Files|
Total|        64224058|       625286420|      426480594766|        8|
 Pass|              2|              4|              184|        |
 Fail|        64224056|       625286416|      426480594582|        |
    8.76.84.224| 133.44.201.53|   80| 2393|  6|        1|   0.000|
 144.44.166.166|    8.76.14.96| 1433|   80|  6|        3|   8.896|
```

Oracle[4]

Service	TCP
Sql*net	66
Sql*net 2	1521
Sql*net 1	1525
Listener port	1526
Tlisrv	1527
Coauthor	1529
Oracle Remote	1571

[4] http://www.chebucto.ns.ca/~rakerman/oracle-port-table.html

Database	
Oraclenames	1575
Oracle Net8 Cman	1630
Oracle-em1	1748
Oracle-em2	1754
Oracle-VP2	1808
Oracle-VP1	1809
Oracle Net8 Cman Admin	1830
Oracle GIOP	2481
Oracle GIOP SSL	2482
Oracle TTC	2483
Oracle TTC SSL	2484
OEM Agent	3872
Oracle RTC-PM port	3891
Oracle dbControl Agent	3938

```
$ rwfilter ext1/*.rw --any-address=8.76.0.0/16 --print-volume-stat --
aport=66,1521,1524,1526,1527,1529,1571,1575,1630,1748,1754,1808,1809,1830,
2481,2482,2483,2484,3872,3891,3938 --pass=stdout | rwcut --
fields=sip,dip,sport,dport,protocol,packets,dur
           sIP|            dIP|sPort|dPort|pro|   packets|      dur|
     |            Recs|          Packets|              Bytes|     Files|
Total|        104723184|        918568027|       656996415962|        16|
 Pass|                7|               15|               1539|          |
 Fail|        104723177|        918568012|       656996414423|          |
    8.76.43.42|  133.44.123.64| 7400| 1808|  6|         1|    0.000|
    8.76.43.42|  133.44.123.64| 7000| 1808|  6|         1|    0.000|
  192.42.134.97|   8.76.178.112| 1575| 8000| 17|         1|    0.059|
   8.76.178.112|  192.42.134.97| 8000| 1575| 17|         1|    0.000|
  192.42.134.97|   8.76.148.150| 1748|   80|  6|         5|    0.146|
   8.76.148.150|  192.42.134.97|   80| 1748|  6|         4|    0.087|
    8.76.84.180|  133.44.250.91| 1630|  445|  6|         2|    2.907|
```

```
rwfilter ext2/*.rw --any-address=8.76.0.0/16 --print-volume-stat --
aport=66,1521,1524,1526,1527,1529,1571,1575,1630,1748,1754,1808,1809,1830,
2481,2482,2483,2484,3872,3891,3938 --pass=stdout | rwcut --
fields=sip,dip,sport,dport,protocol,packets,dur,bytes
           sIP|            dIP|sPort|dPort|pro|   packets|      dur|     bytes|
     |            Recs|          Packets|              Bytes|     Files|
Total|         64224058|        625286420|       426480594766|         8|
 Pass|               19|               89|              21189|          |
 Fail|       64224039|       625286331|      426480573577|        |
  192.42.134.97|    8.76.150.1| 1526|   80|  6|        5|    0.453|     1171|
    8.76.150.1|  192.42.134.97|   80| 1526|  6|        5|    0.373|      699|
  192.42.134.97|  8.76.146.137| 1529|   80|  6|        5|    0.357|     1226|
   8.76.146.137|  192.42.134.97|   80| 1529|  6|        5|    0.269|      598|
 144.44.166.166|    8.76.14.96| 1524|   80|  6|        3|    8.875|      144|
 144.44.166.166|    8.76.14.96| 1630|   80|  6|        3|    8.947|      144|
 144.44.166.166|    8.76.14.96| 2481|   80|  6|        3|    9.048|      144|
   8.76.84.224|  133.44.123.64|   80| 1808|  6|        1|    0.000|       40|
  192.42.134.97|    8.76.18.140| 1748|   80|  6|       17|   12.283|     1075|
   8.76.18.140|  192.42.134.97|   80| 1748|  6|       11|   12.191|    13399|
   8.76.18.140|  192.42.134.97|   80| 1748|  6|        9|   79.554|      360|
   8.76.84.180|  133.44.164.30| 1529|  445|  6|        2|    2.960|       96|
   8.76.84.224| 133.44.100.205|   80| 3872|  6|        1|    0.000|       40|
 144.44.166.166|    8.76.14.96| 1521|   80|  6|        3|    9.049|      144|
 144.44.166.166|    8.76.14.96| 1527|   80|  6|        3|    9.049|      144|
 144.44.166.166|    8.76.14.96| 1529|   80|  6|        3|    9.148|      144|
 144.44.166.166|    8.76.14.96| 1630|   80|  6|        3|    8.948|      144|
  192.42.134.97|    8.76.144.67| 2484| 8000| 17|        1|    0.000|       44|
 149.162.180.62|    8.76.199.30| 1527|   80|  6|        6|    0.664|     1433|
```

MySQL – Port 3306

```
 rwfilter ext1/*.rw --any-address=8.76.0.0/16 --print-volume-stat --aport=3306 --
pass=stdout | rwcut --fields=sip,dip,sport,dport,protocol,packets,dur
            sIP|            dIP|sPort|dPort|pro|   packets|      dur|
     |            Recs|         Packets|              Bytes|     Files|
Total|       104723184|       918568027|       656996415962|        16|
 Pass|               0|               0|                  0|          |
 Fail|       104723184|       918568027|       656996415962|          |
```

```
 rwfilter ext2/*.rw --any-address=8.76.0.0/16 --print-volume-stat --aport=3306 --
pass=stdout | rwcut --fields=sip,dip,sport,dport,protocol,packets,dur
            sIP|            dIP|sPort|dPort|pro|   packets|      dur|
     |            Recs|         Packets|              Bytes|     Files|
Total|        64224058|       625286420|       426480594766|         8|
 Pass|               0|               0|                  0|          |
 Fail|        64224058|       625286420|       426480594766|          |
```

As we can see, there are definitely a couple of SQL Server and Oracle databases. However, there were no MySQL databases. Through the reporting, we were able to distinguish several Oracle servers. There were several different ports used, but this is common with Oracle as the application runs through the different ports for authentication and connections. Within the SQL Server data, we can see that there was a lot of data from port 1433 to port 6000. Since port 6000 is commonly associated with X11, this could be common traffic. However, since there are many different computers connecting it, there could be something interesting there. We will go into more detail about our findings later in this paper.

P2P

BitTorrent

The typical BitTorrent client uses the default port range of 6888-6900 over both TCP and UDP[5]. Our analysis did not reveal any torrent traffic over this port range. However, the results were still inconclusive as some BitTorrent clients allow users to specify their own ports or randomize them each time the program starts. Unfortunately, detecting Torrent traffic using flow data requires a more in-depth understanding of your network in order to realize and detect what sort of flows are anomalous. Simply searching for ports that can easily be randomized is not sufficient to conclude that there is absolutely no torrent traffic on our network.

Dasher.C

While searching for hosts running Microsoft SQL Server we came across a strange pattern of traffic. We encountered multiple traffic flows coming from a TCP source port of 1433 and going to a destination port of 6000. TCP port 1433 is used by Microsoft SQL Server and TCP port 6000 is used by X11. This seems like an odd combination so we did some more digging and discovered a worm called Win32/Dasher.C that uses these ports also. Dasher.C will use a source port of 6000 and send out SYN requests to hosts on ports 1433 and 445. These are the ports used by SQL Server and Microsoft-DS, respectively. If Dasher.C receives a valid response that the port is open it will attempt to exploit a vulnerability on that service. All of the records we found using port 6000 all have that port as the destination and the source is either 1433 or 445. All of the flows are only a single packet in length and all have the RST/ACK flag, meaning they are response packets. This suggests a few possible explanations. First, the original SYN packets might have not been collected for some reason. Another explanation would be that when the SYN packets were sent out they came from an address outside our subnet with a spoofed source address. If this had happened, when the outside addresses responded with their RST/ACK packet it would go to the spoofed address instead of the real source address.

```
$ rwfilter *.rw --aport=6000 --pass=stdout | rwuniq --
fields=sip,dip,sport,dport,bytes,packets,protocol,flags
rwuniq: Warning: Using default temporary directory /tmp
          sIP|          dIP|sPort|dPort|    bytes|  packets|pro|  flags|  Records|
  214.75.59.80|   8.76.84.203| 1433| 6000|       40|        1|  6|  R A  |        1|
  214.75.58.61|   8.76.84.203| 1433| 6000|       40|        1|  6|  R A  |        1|
```

[5] http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

```
  214.75.59.24|   8.76.84.203| 1433| 6000|      40|       1|  6|  R A   |       1|
  214.75.59.36|   8.76.84.203| 1433| 6000|      40|       1|  6|  R A   |       1|
  214.75.59.20|   8.76.84.203| 1433| 6000|      40|       1|  6|  R A   |       1|
  214.75.59.89|   8.76.84.203| 1433| 6000|      40|       1|  6|  R A   |       1|
  214.75.59.32|   8.76.84.203| 1433| 6000|      40|       1|  6|  R A   |       1|
 215.149.60.128|   8.76.84.203| 1433| 6000|      40|       1|  6|  R A   |       1|
 193.66.118.10|  8.76.157.129| 1433| 6000|      40|       1|  6|  R A   |       2|
 214.75.56.214|   8.76.84.203| 1433| 6000|      40|       1|  6|  R A   |       1|
 215.149.60.129|   8.76.84.203| 1433| 6000|      40|       1|  6|  R A   |       1|
  214.75.58.59|   8.76.84.203| 1433| 6000|      40|       1|  6|  R A   |       1|
   214.75.59.7|   8.76.84.203| 1433| 6000|      40|       1|  6|  R A   |       1|
  214.75.58.90|   8.76.84.203| 1433| 6000|      40|       1|  6|  R A   |       1|
  214.75.59.19|   8.76.84.203| 1433| 6000|      40|       1|  6|  R A   |       1|
 193.66.118.10|  8.76.157.129|  445| 6000|      40|       1|  6|  R A   |       2|
```

Suggestions for Improvement

After our analysis of the data, there are many things that can be extrapolated from this data. We have attempted to do what we can, but there are many more things that we could have missed. However, we have collected several suggestions, which are an aggregate of our findings. We have divided these recommendations into three categories: Security, Availability, and Policy.

Security

First, we found some NetBIOS traffic passing through the network. Since the data was collected on a Tier 1 router, it shows that the traffic was passing back and forth on the Internet. We suggest that this traffic be blocked because there is no need for it to travel across the Internet. Instead, it should exist only within a private network.

Second, network administrators should frequently analyze their networks in order to detect irregular traffic. Tools such as Snort should be used in order to detect occurrences such as DDoS attacks, viruses, or any other undesired traffic within their networks. Our network did a decent job of blocking much of this traffic, but we were still able to identify some questionable traffic.

Third, the network administrators should also use some heuristic filtering on their network based on ICMP traffic in order to prevent outsiders from scanning ports. As part of our analysis, we have identified a couple scan attempts that could have been used to profile a network, in addition to the analysis that we were able to perform. Specifically, since we have identified that our network mostly consists of corporate networks, it is crucial to protect to confidentiality of a network. Allowing people to use ICMP scans could allow hackers access to information that would aid them with intrusion attempts. We suggest using intelligent filtration methods to block particular IP addresses would be very beneficial in terms of security.

Availability

Since our network contains several corporate servers, it likely that these servers need to be available at all times. The non-functional requirements of those servers are not known due to the anonymity of the data, but assumptions can be made. We recommend that the ISP's as well as the network administrators of the corporations use network-monitoring tools such as Nagios in order to ensure the stability of the network. This could be utilized in order to use better load balancing of a network or block attacks that might be occurring. In any case, many of these servers will need to be available on a consistent basis, so necessary steps should be taken.

Policy

We have identified many outdated services as well an outdated virus in our analysis. This demonstrates to us that many computers on this network have not been updated with their virus definitions or other security updates. It is possible that these computers do not have any security software installed, but this would fall into the same recommendation of ours. Our recommendation to these computers is to mandate updates in security software. If this happens, then many of the holes in the currently running software will be patched as well as provide many additional preventative measures. We raise this issue because our network is primarily corporate. If these vulnerabilities are not patched properly, then there will be a significant risk for the company. Updating their security software will at least give the company a fighting chance against many common vulnerabilities.

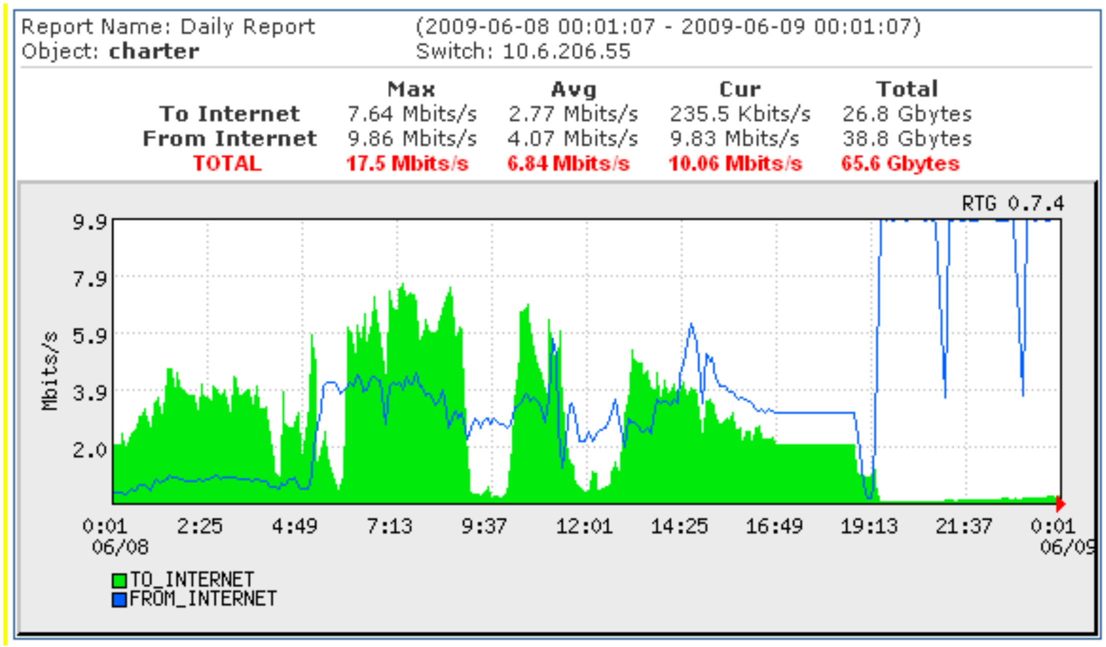
Event Analysis

In early June 2009, a Belarusian news site, Charter97.org, suffered a distributed denial-of-service attack that lasted several days. Arbor Networks' security researcher Jose Nazario believes the attack was fueled by a geopolitical conflict between Russia and Belarus[6]. Apparently, leading up to the attack, Belarus refused to recognize the independence of rebel Georgian territories much to the dismay of Russia who fears that its influence in former Soviet Union nations is slowly deteriorating[7]. In response to this, Russia withheld a very large loan installment from Belarus and banned the importation of Belarusian dairy goods into Russia.[8] Belarus then demonstrated their intent to align themselves politically and economically with the European union by requesting their financial support.

[6] http://asert.arbornetworks.com/2009/06/ddos-floods-in-belarus-political-motivations/
[7] http://www.etaiwannews.com/etn/news_content.php?id=971327&lang=eng_news
[8] http://www.etaiwannews.com/etn/news_content.php?id=971327&lang=eng_news

Apparently not satisfied with their initial punishment tactics, Nazario believes that Russia, or individuals acting for Russia, orchestrated a DDoS attack against Charter97.org, which is a news site for the human rights organization of the same name that has been vocal in favor of democratic movements in Belarus.[9] The attack itself consisted of initial short attacks that the administrators at Charter97 were able to mitigate, however the attackers increased the intensity of the attack and the site was unavailable for several days.[10] According to their own reports, Charter97 believed that over 5000 bots were involved in the attack.[11] This attack was yet another incident that indicates that political events have cyber-world repercussions. Figure 1 shows a chronology of the attack that was taken from another Belarusian site. During the initial attack, Charter97 administrators were able to keep the site reasonably online, however around 19:13 the attackers escalated the number of hosts to 5000 and the site was rendered completely unavailable.

Figure 1 - Charter97.org DDoS Chronology from http://www.electroname.com/photos/20090609_ddos.png



Report Name: Daily Report (2009-06-08 00:01:07 - 2009-06-09 00:01:07)
Object: **charter** Switch: 10.6.206.55

	Max	Avg	Cur	Total
To Internet	7.64 Mbits/s	2.77 Mbits/s	235.5 Kbits/s	26.8 Gbytes
From Internet	9.86 Mbits/s	4.07 Mbits/s	9.83 Mbits/s	38.8 Gbytes
TOTAL	**17.5 Mbits/s**	**6.84 Mbits/s**	**10.06 Mbits/s**	**65.6 Gbytes**

Since sample point F is collected from a trans-Pacific link, evidence of this DDoS attack would most likely not appear in the data set since Russia and Belarus are

[9] http://en.wikipedia.org/wiki/Charter_97
[10] http://www.theregister.co.uk/2009/06/12/belarus_ddos/
[11] http://www.charter97.org/en/news/2009/6/10/18992/

located in northern Asia and Eastern Europe. The bot-net was most likely centered in Russia, and traffic from any US hosts probably would have passed a trans-Atlantic line. However, for the sake of analysis, let us imagine the scenario that our network contains the web server(s) for Charter97.org. Figure 2 displays all traffic to and from web servers within our subnet. We can assume that the initial traffic levels at 0:01 on Figure 1 represent the average traffic level for Charter97. Figure 2 indicates that the total volume of web traffic on our subnet rarely exceeds 1 Mbps compared to a much higher throughput for Charter97. In other words, had this attack occurred against our web servers, the impact of the DDoS would have been much more obvious and effective. If our infrastructure was only designed to support such a low level of traffic, potentially fewer bots might have been needed to initiate the attack against our web servers. In terms of manifesting itself in our dataset, the magnitude of the Charter97 attack would have been immediately apparent even just by looking at the total link saturation. Specifically, depending on the method used for this attack, it could have manifested itself in the form of single packet SYN flows, RST flows, or an ICMP flood. Unfortunately, there was not adequate media source material regarding this particular attack to determine the actual attack vector.

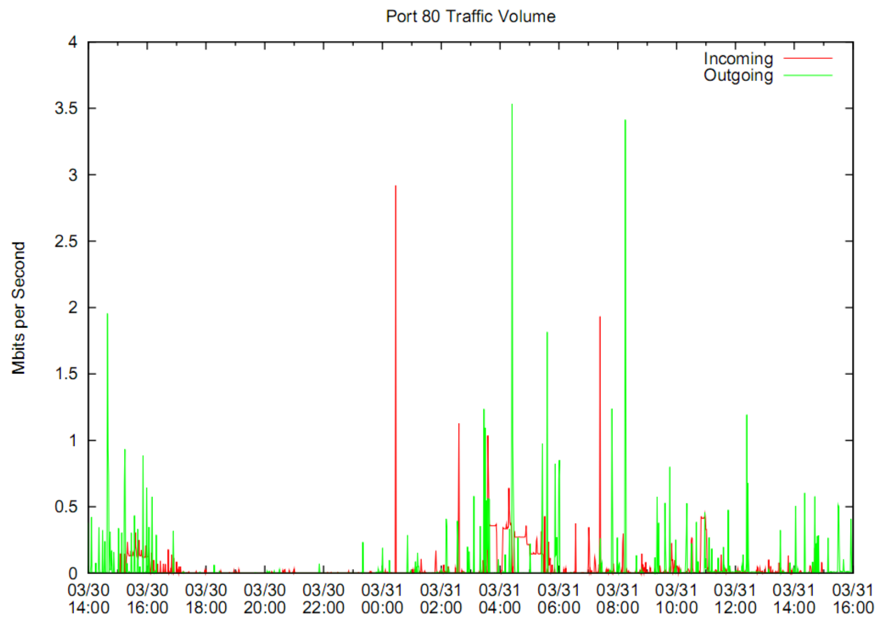


Figure 2 - Traffic going to and from all web servers on our subnet