Traffic Management Systems

An Impact Analyses

Shuchi Muley

Mark Allen

Contents

Problem Statement

In the busy world of the Internet, users are pretty much able to derive any content they can imagine. With the limited amount of bandwidth available to ISPs, the growing concern they may have is regarding controlling the traffic to make improvements for efficiency. This project will serve as the analysis of the technical impacts of implementing a traffic management system.

Approach

Our data analysis will consist of generalizing and profiling the data. The following categories are specifically what we're interested in (all other traffic will be disregarded):

- FTP
- Web (HTTP/HTTPS)
- Streaming Audio
- Streaming Video

After categorizing the data, we will classify each type of data based on its legitimacy. The data will be classified as either legitimate business traffic or non-legitimate traffic, based on the destination and type of traffic. For instance, a user may be streaming video from a valid news source (cnn.com) which is legitimate traffic. Alternatively, another user may be streaming video from an illegitimate source, such as youtube.com. Therefore, we will have categories of data divided by subsets of classifications.

- FTP
  - Legitimate FTP
  - Illegitimate FTP
- Web (HTTP/HTTPS)
  - Legitimate Web Traffic
  - Illegitimate Web Traffic
- Streaming Audio
  - Legitimate Streaming Audio
  - Illegitimate Streaming Audio
- Streaming Video
  - Legitimate Streaming Video
  - Illegitimate Streaming Video

After the classification, we will determine the impacts to the profiled data of multiple traffic management providers.

Traffic Management

The first traffic management software we will use is Zeus Technology's Zeus Extensible Traffic Manager (ZXTM). http://www.zeus.com/products/zxtm/index.html

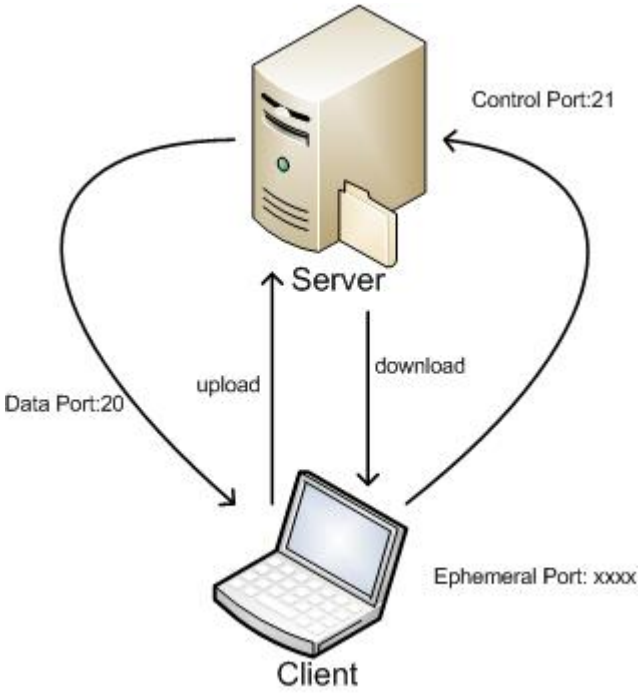
The second traffic management software is Secure Computing's Secure Web Smart Filter. http://www.securecomputing.com/index.cfm?skey=85

The third traffic management solution is ARA Network's Traffic Monitor.
http://www.aranetworks.com/solutions/traffic_monitor

Protocol Basics

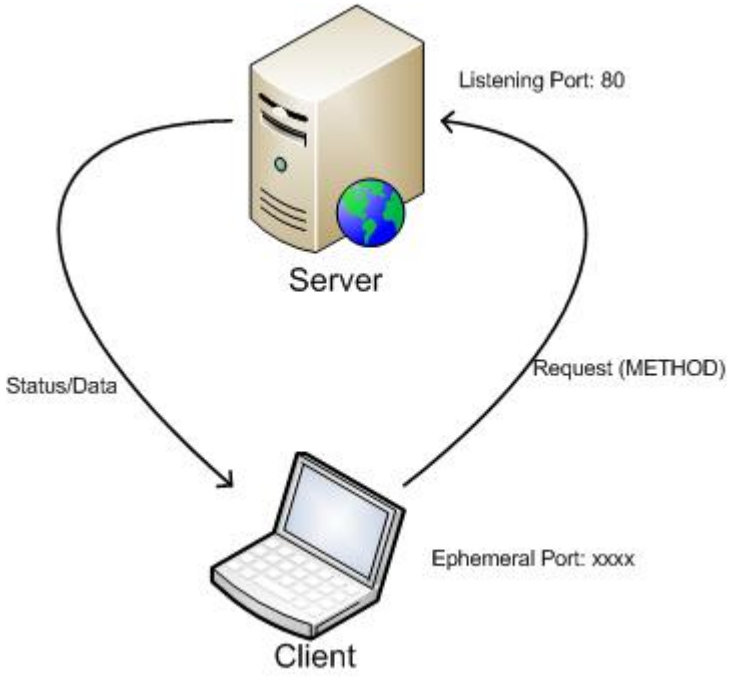
File Transfer Protocol (FTP)

File Transfer Protocol, or FTP, is simply used for transferring files between a client and server. All communication takes place over TCP. The server is the host listening on port 21 for control messages. The client initiates a connection over an ephemeral port decided on by the client based on its operating system and other factors. However, the data transfers take place over port 20. The term *uploading* refers to when a client sends a file to the server, while *downloading* refers to when a client receives a file from the server. In active FTP, the client opens a dynamic port, sends it to the server, and waits. The server then initiates the data connection over port 20 on the server and the dynamic port for the client. In passive mode, the server opens a dynamic port, sends it to the client, and waits. Here, the client binds to the dynamic port. For our purposes, we will only look at active mode FTP.



Hyper-Text Transfer Protocol (HTTP)

Hyper-Text Transfer Protocol is the protocol used by web hosts to serve resources, such as web pages and other content, to clients. HTTP is a connection-oriented protocol (TCP) and, by default, takes place over port 80. The client initiating the connection will use an ephemeral port to send the packet, but will bind to the server's listening port. Port 443 is used for HTTPS, which is Secure HTTP or HTTP with an encryption key for secure communication. The encryption used is typically a Secure Socket Layer (SSL) key, but sometimes Transport Layer Security (TLS) is used. In HTTP, the client performs an action against a resource on the server. The action, or method, can be one of the eight defined in RFC 2616: HEAD, GET, POST, PUT, DELETE, TRACE, OPTIONS, and CONNECT.



Streaming Audio

Research incomplete at time of report.

Streaming Video

Research incomplete at time of report.

Data Sets

For our analysis of the impacts of a traffic management system, we will be using two data sets from DatCat from the "Day In The Life", or DITL, Internet Project. Both are Internet2 traffic collected from the Abilene Network Juniper T-640 routers. Both data sets are NetFlow v5 data, sampled at 1/100. All non-multicast (IPv4) addresses are anonymized by setting the last 11 bits to zero. The first data set, was collected from January 9th, 2007 00:00 UTC to January 11th, 2007 00:00 UTC (2 days). The second data set was collected from March 19th, 2008 00:00 UTC to March 20th, 2008 00:00 UTC (1 day).

Internet2 Network Structure

The Internet2 structure consists of many partners which connect to each other over extremely high speed connections, allowing their users to run demanding applications to work together. The connections are only used to connect to each other and each partner has their own separate Internet connection. Because of this, one should be able to see duplicate (however juxtaposed) entries when analyzing the data traffic. This means one should be able to see a source IP as a destination IP in another set of traffic. However, the assumption for this rule is that there are full network captures. However, this may not be the case with the data that was obtained; the data was sampled at a rate of 1/100. With this sampling rate, we only get one out of every 100 flows that pass through the router. Since each router is also sampling independently of each other, meaning packet travel time and time set on the router were ignored, finding the same packets reversed somewhere else will prove to be quite difficult. Anonymization will also play an important role here. Since every IP address was anonymized by stripping the last eleven bits and setting them to zero, we will be unable to determine, or essentially show definitive proof, that two packets talking to the same host coming from the same subnet are from the same source. However, this is an assumption we have to take; otherwise, the data isn't useful.

Physical Layout

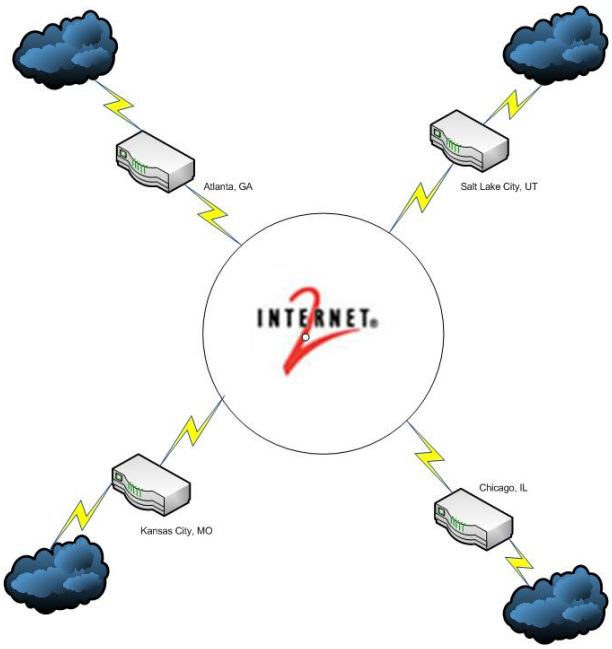
The physical layout of the Internet2 network is a collection of several, logically different networks. The networks are advanced IP networks, dynamic Circuit networks, and core optical networks. The map below illustrates where the partners are physically located.

Figure 1 - Internet2 Network Physical Layout (http://www.internet2.edu/pubs/networkmap.pdf)

The image above shows many cities that contain partners for Internet2. The cities that we actually had in our data sets were Atlanta, GA; Chicago, IL; Houston, TX; Kansas City, MO; Los Angeles, CA; New York, NY; and Salt Lake City, UT.

Logical Layout

The image below is a logical representation of the Internet2 layout. As mention, each site has it's own Internet connection, but connects to each other over the Internet2 routers. This creates a ring of sites



3ROX

One partner is especially noteworthy for the purposes of this project. 3ROX, or Three Rivers Optical Exchange, is based at Carnegie Mellon University. The primary focus of the Exchange is to provide high-capacity, cost effective network connectivity to the university, local educational institutions, government, and business in Western Pennsylvania and parts of West Virginia. Some of the more well-known sites, besides CMU, include the Pennsylvania State University, Pittsburgh Supercomputing Center, the University of Pittsburgh, and West Virginia University. This partner is not in our data, but since the author's attend Carnegie Mellon University, they felt it was worth mentioning.

Data Analyses

In analyzing the data, we split the data based on file name, which was, in essence, split by time. Half of the files contained a 12, for time 12:00:00, and the rest contained 13, for time 13:00:00. In retrospect, we should have split the analyses by city.

FTP Analyses – "12"

First, we took filtered out all "pass" files looking for any TCP connections that took place over port 21, the default port for FTP control traffic:

```
rwfilter pass-*12.rw --pass=12.ftp.rw --aport=21 --proto=6 --print-vol --print-file
```

In order to figure out the major conversations, we had to sort. We can sort either by bytes or by packets. First, we chose to sort by bytes:

```
rwsort 12.ftp.rw --out=12.ftp.rw.SORTED --fields=bytes
```

In order to view the data, we cut it using rwcut. We only wanted the major players, so we added the "tail" to get the bottom 10 (which is actually the top 10 based on bytes):

```
rwcut 12.ftp.rw.SORTED --fields=sip,sport,dip,dport,bytes,packets,stime,etime | tail
```

sIP	dIP	sPort	dPort	Bytes	packets	sTime	eTime
163.221.8.0	21	193.233.8.0	50458	15000	10	2008/10/12T12:51:58.835	2008/10/12T12:52:33.402
163.221.8.0	21	193.233.8.0	50458	15000	10	2008/10/12T12:51:54.361	2008/10/12T12:52:33.060
128.113.24.0	21	193.233.8.0	63181	15620	11	2008/10/12T12:52:19.249	2008/10/12T12:52:50.711
134.174.136.0	41159	221.140.64.0	21	15620	11	2008/10/12T12:39:19.821	2008/10/12T12:40:16.812
163.221.8.0	21	193.233.8.0	50458	16500	11	2008/10/12T12:52:26.856	2008/10/12T12:52:58.305
163.221.8.0	21	193.233.8.0	50458	16500	11	2008/10/12T12:52:07.962	2008/10/12T12:53:02.856
134.174.136.0	41159	221.140.64.0	21	17040	12	2008/10/12T12:43:20.818	2008/10/12T12:44:06.819
128.113.24.0	21	193.233.8.0	63181	19880	14	2008/10/12T12:07:21.382	2008/10/12T12:08:10.216
128.113.24.0	21	193.233.8.0	63181	26980	19	2008/10/12T12:50:31.535	2008/10/12T12:51:16.384
128.113.24.0	21	193.233.8.0	63181	51120	36	2008/10/12T12:51:20.243	2008/10/12T12:52:18.567

We can see from the output that there is a major FTP conversation between 163.221.8.0, 128.113.24.0 and 192.233.8.0. At first, one might conclude, since we only have the first 21 bits and the data is sampled, that there is no conclusive proof that the conversation is between two specific hosts; it may be between multiple hosts on both sides. However, for the purposes of this paper, we will assume that it is between two hosts, due in large part to the ephemeral port being the same, 50,458, and 63181 for the

duration of the resulting conversation. This large conversation made me wonder how long the conversation occurred and the total bytes and packets.

rwfilter 12.ftp.rw --aport=21 --saddr=163.221.8.0 --daddr=193.233.8.0 --proto=6 --pass=12.ftp.large1.rw

rwfilter 12.ftp.rw --aport=21 --saddr=128.113.24.0 --daddr=193.233.8.0 --proto=6 --pass=12.ftp.large2.rw

This gives me a file containing only the FTP conversation between these two pair of 2 addresses. I then want to output the information for easy sorting in Microsoft Excel:

rwcut 12.ftp.large1.rw --fields=sip,dip,sport,dport,packets,bytes,dur

Based on the output, I was able to obtain the following information:

File 1: IP: 163.221.8.0

Packets: 1300
Bytes: 350,019
Duration: 3 hours

rwcut 12.ftp.large2.rw --fields=sip,dip,sport,dport,packets,bytes,dur

File 2: IP: 128.113.24.0

Packets: 789
Bytes: 457,940
Duration: 1.1 hours

The timing, bytes, and packets are random enough to suggest the source IP is sending a large file or set of files over a long period of time to the destination IP. After running the following commands:

rwfilter pass*.12.rw --sadd=163.221.8.0 --daddr=193.233.8.0 --proto=6 --pass=stdout | rwcut --fields=sip,dip,sport,dport,packets,bytes,dur

rwfilter pass*.12.rw --sadd=128.113.24.0 --daddr=193.233.8.0 --proto=6 --pass=stdout | rwcut --fields=sip,dip,sport,dport,packets,bytes,dur

The output yielded the same results as when specifying any port as 21. This means the only 2 communication these two IP address was over FTP (aside from any web traffic).

Looking back at one of the previous results, we also see the following:

134.174.136.0	41159	221.140.64.0	21	15620	11	2008/10/12T12:39:19.821	2008/10/12T12:40:16.812
134.174.136.0	41159	221.140.64.0	21	17040	12	2008/10/12T12:43:20.818	2008/10/12T12:44:06.819

Since the previous conversation enveloped any further communication in this conversation, we decided to take a second look.

rwsort 12.ftp.second.large.rw --fields=stime | rwcut --fields=sip,dip,sport,dport,packets,bytes,dur

The output:

sIP	dIP	sPort	dPort	packets	bytes	duration
134.174.136.0	221.140.64.0	41159	21	4	5680	33.901
134.174.136.0	221.140.64.0	41159	21	1	1420	0
134.174.136.0	221.140.64.0	41159	21	9	12780	56.396
134.174.136.0	221.140.64.0	41159	21	5	7100	35.866
134.174.136.0	221.140.64.0	41159	21	2	2840	2.283
134.174.136.0	221.140.64.0	41159	21	11	15620	56.991
134.174.136.0	221.140.64.0	41159	21	1	1420	0
134.174.136.0	221.140.64.0	41159	21	5	7100	31.892
134.174.136.0	221.140.64.0	41159	21	6	8520	48.581
134.174.136.0	221.140.64.0	41159	21	7	9940	24.911
134.174.136.0	221.140.64.0	41159	21	12	17040	46.001
134.174.136.0	221.140.64.0	41159	21	1	1420	0
134.174.136.0	221.140.64.0	41159	21	4	5680	29.327
134.174.136.0	221.140.64.0	41159	21	9	12780	48.469
134.174.136.0	221.140.64.0	41159	21	1	1420	0
134.174.136.0	221.140.64.0	41159	21	9	12780	47.683
134.174.136.0	221.140.64.0	41159	21	2	2840	29.811
134.174.136.0	221.140.64.0	41159	21	7	9940	47.276
134.174.136.0	221.140.64.0	41159	21	3	4260	14.677
134.174.136.0	221.140.64.0	41159	21	5	7100	19.671
134.174.136.0	221.140.64.0	41159	21	2	2840	6.819

134.174.136.0	221.140.64.0	41159	21	8	11360	50.087
134.174.136.0	221.140.64.0	41159	21	3	4260	41.518
134.174.136.0	221.140.64.0	41159	21	2	2840	29.061
134.174.136.0	221.140.64.0	41159	21	2	1460	1.826

All the packets are transferred over ephemeral port 41159. However, since the data stream for FTP traffic runs over port 20, we need to analyze the data for port 20.

rwfilter pass-*12.rw --pass=12.port20.rw --aport=20 --proto=6 --print-vol --print-file

rwsort 12.port20.rw --out=12.port20.rw.SORTED --fields=bytes

rwcut 12.port20.rw.SORTED --fields=sip,sport,dip,dport,bytes,packets,stime,etime | tail

The output:

Sip	Sport	Dip	Dport	Bytes	Packets	Stime	Etime
130.14.24.0	20	140.109.56.0	57952	1227000	818	2008/10/12T12:25:42.156	2008/10/12T12:26:41.809
130.14.24.0	20	140.109.56.0	57952	1227000	839	2008/10/12T12:46:42.130	2008/10/12T12:47:41.808
130.14.24.0	20	140.109.56.0	57952	1272000	848	2008/10/12T12:27:42.037	2008/10/12T12:28:41.796
130.14.24.0	20	140.109.56.0	57952	1317000	878	2008/10/12T12:05:41.068	2008/10/12T12:06:40.829
130.14.24.0	20	140.109.56.0	57952	1368000	912	2008/10/12T12:40:42.056	2008/10/12T12:41:41.701
130.14.24.0	20	140.109.56.0	57952	1378500	919	2008/10/12T12:43:42.098	2008/10/12T12:44:41.402
130.14.24.0	20	140.109.56.0	57952	1416000	944	2008/10/12T12:23:42.137	2008/10/12T12:24:41.751
130.14.24.0	20	140.109.56.0	57952	1419000	946	2008/10/12T12:22:41.985	2008/10/12T12:23:41.707
130.14.24.0	20	140.109.56.0	57952	1461000	974	2008/10/12T12:04:41.053	2008/10/12T12:05:40.783
130.14.24.0	20	140.109.56.0	57952	1479000	986	2008/10/12T12:06:41.031	2008/10/12T12:07:40.765

Based on this output, we see one major file transfer conversation going on here. In order to get the full details of the session though, we need to filter on the port 20 file for these two addresses:

rwfilter 12.port20.rw --saddr=130.14.24.0 --daddr=140.109.56.0 --pass=stdout | rwcut --fields=sip,sport,dip,dport,bytes,packets,stime,etime

Output:

sIP	sPort	dIP	dPort	bytes	packets	sTime	eTime
130.14.24.0	20	140.109.56.0	57942	1138500	759	2008/10/12T12:00:41.115	2008/10/12T12:01:40.325
130.14.24.0	20	140.109.56.0	57942	913500	609	2008/10/12T12:01:43.017	2008/10/12T12:02:41.734
130.14.24.0	20	140.109.56.0	57942	1141500	761	2008/10/12T12:02:41.104	2008/10/12T12:03:40.820
130.14.24.0	20	140.109.56.0	57942	1111500	741	2008/10/12T12:03:41.122	2008/10/12T12:04:40.653
130.14.24.0	20	140.109.56.0	57942	1461000	974	2008/10/12T12:04:41.053	2008/10/12T12:05:40.783
130.14.24.0	20	140.109.56.0	57942	1317000	878	2008/10/12T12:05:41.068	2008/10/12T12:06:40.829
130.14.24.0	20	140.109.56.0	57942	1479000	986	2008/10/12T12:06:41.031	2008/10/12T12:07:40.765
130.14.24.0	20	140.109.56.0	57942	1063500	709	2008/10/12T12:07:42.048	2008/10/12T12:08:41.659
130.14.24.0	20	140.109.56.0	57942	796500	531	2008/10/12T12:08:41.083	2008/10/12T12:09:22.005
130.14.24.0	20	140.109.56.0	57951	118500	79	2008/10/12T12:09:26.863	2008/10/12T12:09:40.729
130.14.24.0	20	140.109.56.0	57951	870000	580	2008/10/12T12:09:42.029	2008/10/12T12:10:38.249
130.14.24.0	20	140.109.56.0	57951	652500	435	2008/10/12T12:10:44.825	2008/10/12T12:11:38.482
130.14.24.0	20	140.109.56.0	57952	3000	2	2008/10/12T12:11:41.591	2008/10/12T12:11:41.591
130.14.24.0	20	140.109.56.0	57952	789000	526	2008/10/12T12:11:42.015	2008/10/12T12:12:41.734
130.14.24.0	20	140.109.56.0	57952	490500	327	2008/10/12T12:12:42.012	2008/10/12T12:13:41.549
130.14.24.0	20	140.109.56.0	57952	420000	280	2008/10/12T12:13:42.128	2008/10/12T12:14:40.915
130.14.24.0	20	140.109.56.0	57952	193500	129	2008/10/12T12:14:42.112	2008/10/12T12:15:40.530
130.14.24.0	20	140.109.56.0	57952	213000	142	2008/10/12T12:15:43.236	2008/10/12T12:16:41.793
130.14.24.0	20	140.109.56.0	57952	370500	247	2008/10/12T12:16:42.202	2008/10/12T12:17:41.582
130.14.24.0	20	140.109.56.0	57952	291000	194	2008/10/12T12:17:41.315	2008/10/12T12:18:39.712
130.14.24.0	20	140.109.56.0	57952	516000	344	2008/10/12T12:18:41.943	2008/10/12T12:19:41.702
130.14.24.0	20	140.109.56.0	57952	747000	498	2008/10/12T12:19:42.596	2008/10/12T12:20:41.641
130.14.24.0	20	140.109.56.0	57952	672000	448	2008/10/12T12:20:42.046	2008/10/12T12:21:41.797
130.14.24.0	20	140.109.56.0	57952	1144500	763	2008/10/12T12:21:42.033	2008/10/12T12:22:41.810
130.14.24.0	20	140.109.56.0	57952	1419000	946	2008/10/12T12:22:41.985	2008/10/12T12:23:41.707
130.14.24.0	20	140.109.56.0	57952	1416000	944	2008/10/12T12:23:42.137	2008/10/12T12:24:41.751
130.14.24.0	20	140.109.56.0	57952	817500	545	2008/10/12T12:24:42.075	2008/10/12T12:25:41.736
130.14.24.0	20	140.109.56.0	57952	1227000	818	2008/10/12T12:25:42.156	2008/10/12T12:26:41.809
130.14.24.0	20	140.109.56.0	57952	342000	228	2008/10/12T12:26:42.071	2008/10/12T12:26:56.805
130.14.24.0	20	140.109.56.0	57966	654000	436	2008/10/12T12:26:59.217	2008/10/12T12:27:41.817
130.14.24.0	20	140.109.56.0	57966	1272000	848	2008/10/12T12:27:42.037	2008/10/12T12:28:41.796
130.14.24.0	20	140.109.56.0	57966	1044000	696	2008/10/12T12:28:42.083	2008/10/12T12:29:41.760
130.14.24.0	20	140.109.56.0	57966	1198500	799	2008/10/12T12:29:42.362	2008/10/12T12:30:41.702
130.14.24.0	20	140.109.56.0	57966	690000	460	2008/10/12T12:30:42.047	2008/10/12T12:31:41.617
130.14.24.0	20	140.109.56.0	57966	669000	446	2008/10/12T12:31:42.070	2008/10/12T12:32:41.762
130.14.24.0	20	140.109.56.0	57966	849000	566	2008/10/12T12:32:42.136	2008/10/12T12:33:41.438
130.14.24.0	20	140.109.56.0	57966	759000	506	2008/10/12T12:33:43.323	2008/10/12T12:34:41.745
130.14.24.0	20	140.109.56.0	57966	1050000	700	2008/10/12T12:34:42.207	2008/10/12T12:35:41.779
130.14.24.0	20	140.109.56.0	57966	384000	256	2008/10/12T12:35:42.061	2008/10/12T12:36:09.090
130.14.24.0	20	140.109.56.0	57971	394500	263	2008/10/12T12:36:11.983	2008/10/12T12:36:41.785

130.14.24.0	20	140.109.56.0	57971	1005000	670	2008/10/12T12:36:42.049	2008/10/12T12:37:36.387
130.14.24.0	20	140.109.56.0	57971	1500	1	2008/10/12T12:36:58.439	2008/10/12T12:36:58.439
130.14.24.0	20	140.109.56.0	57972	25500	17	2008/10/12T12:37:39.113	2008/10/12T12:37:41.572
130.14.24.0	20	140.109.56.0	57972	885000	590	2008/10/12T12:37:42.082	2008/10/12T12:38:41.762
130.14.24.0	20	140.109.56.0	57972	936000	624	2008/10/12T12:38:42.069	2008/10/12T12:39:41.756
130.14.24.0	20	140.109.56.0	57972	1119000	746	2008/10/12T12:39:42.106	2008/10/12T12:40:41.742
130.14.24.0	20	140.109.56.0	57972	1368000	912	2008/10/12T12:40:42.056	2008/10/12T12:41:41.701
130.14.24.0	20	140.109.56.0	57972	1126500	751	2008/10/12T12:41:42.114	2008/10/12T12:42:41.632
130.14.24.0	20	140.109.56.0	57972	1072500	715	2008/10/12T12:42:42.095	2008/10/12T12:43:41.704
130.14.24.0	20	140.109.56.0	57972	1378500	919	2008/10/12T12:43:42.098	2008/10/12T12:44:41.402
130.14.24.0	20	140.109.56.0	57972	924000	616	2008/10/12T12:44:42.783	2008/10/12T12:45:41.649
130.14.24.0	20	140.109.56.0	57972	186000	124	2008/10/12T12:45:42.118	2008/10/12T12:45:50.343
130.14.24.0	20	140.109.56.0	57981	669000	446	2008/10/12T12:45:53.354	2008/10/12T12:46:41.728
130.14.24.0	20	140.109.56.0	57981	1258500	839	2008/10/12T12:46:42.130	2008/10/12T12:47:41.808
130.14.24.0	20	140.109.56.0	57981	607500	405	2008/10/12T12:47:41.932	2008/10/12T12:48:41.407
130.14.24.0	20	140.109.56.0	57981	885000	590	2008/10/12T12:48:42.251	2008/10/12T12:49:41.753
130.14.24.0	20	140.109.56.0	57981	715500	477	2008/10/12T12:49:42.125	2008/10/12T12:50:41.687
130.14.24.0	20	140.109.56.0	57981	1054500	703	2008/10/12T12:50:42.185	2008/10/12T12:51:41.755
130.14.24.0	20	140.109.56.0	57981	1168500	779	2008/10/12T12:51:41.975	2008/10/12T12:52:41.748
130.14.24.0	20	140.109.56.0	57981	1140000	760	2008/10/12T12:52:42.088	2008/10/12T12:53:41.708
130.14.24.0	20	140.109.56.0	57981	943500	629	2008/10/12T12:53:42.139	2008/10/12T12:54:41.439
130.14.24.0	20	140.109.56.0	57981	721500	481	2008/10/12T12:54:42.035	2008/10/12T12:55:41.551
130.14.24.0	20	140.109.56.0	57981	1150500	767	2008/10/12T12:55:42.098	2008/10/12T12:56:41.612
130.14.24.0	20	140.109.56.0	57981	1003500	669	2008/10/12T12:56:42.038	2008/10/12T12:57:38.753
130.14.24.0	20	140.109.56.0	57992	3000	2	2008/10/12T12:57:40.795	2008/10/12T12:57:40.795
130.14.24.0	20	140.109.56.0	57992	882000	588	2008/10/12T12:57:42.168	2008/10/12T12:58:41.665
130.14.24.0	20	140.109.56.0	57992	999000	666	2008/10/12T12:58:42.108	2008/10/12T12:59:41.655
130.14.24.0	20	140.109.56.0	57992	1144500	763	2008/10/12T12:59:42.069	2008/10/12T13:00:41.591

Over the course of 98 seconds, the source sent about 56,472,000 bytes in 37648 packets.

We then wanted to see what else was going on, if anything, between these 2 hosts. We would expect to see some kind of control information being passed over port 21. We filtered all “pass” files for these 2 addresses in order to get a clear picture of what’s going on.

rwfilter pass*.12.rw --sadd=130.14.24.0 --daddr=140.109.56.0 --proto=6 --pass=stdout | rwcut --fields=sip,dip,sport,dport,packets,bytes,dur

This yielded different results than above

The duration remained the same. However, the number of bytes sent was 56,477,059 in 37653 packets.

130.14.24.0	21	140.109.56.0	57889	1	52	2008/10/12T12:45:51.097	2008/10/12T12:45:51.097
130.14.24.0	80	140.109.56.0	1493	1	507	2008/10/12T12:49:00.493	2008/10/12T12:49:00.493
130.14.24.0	80	140.109.56.0	1495	3	4500	2008/10/12T12:49:04.985	2008/10/12T12:49:05.675

FTP Analyses – "13"

First, we took filtered out all "pass" files looking for any TCP connections that took place over port 21, the default port for FTP control traffic:

```
rwfilter pass-*13.rw --pass=13.ftp.rw --aport=21 --proto=6 --print-vol --print-file
```

In order to figure out the major conversations, we had to sort. We can sort either by bytes or by packets. First, we chose to sort by bytes:

```
rwsort 13.ftp.rw --out=13.ftp.rw.SORTED --fields=bytes
```

In order to view the data, we cut it using rwcut. We only wanted the major players, so we added the "tail" to get the bottom 10 (which is actually the top 10 based on bytes):

```
rwcut 13.ftp.rw.SORTED --fields=sip,sport,dip,dport,bytes,packets,stime,etime | tail
```

sIP	sPort	sIP	dPort	Bytes	packets	sTime	eTime
163.221.8.0	21	193.233.8.0	50458	24000	16	2008/10/12T13:37:07.663	2008/10/12T13:38:04.387
163.221.8.0	21	193.233.8.0	50458	25500	17	2008/10/12T13:39:39.679	2008/10/12T13:40:35.260
163.221.8.0	21	193.233.8.0	50458	25500	17	2008/10/12T13:40:21.827	2008/10/12T13:41:16.853
163.221.8.0	21	193.233.8.0	50458	25500	17	2008/10/12T13:45:07.208	2008/10/12T13:45:58.897
163.221.8.0	21	193.233.8.0	50458	25500	17	2008/10/12T13:48:44.808	2008/10/12T13:49:40.118
163.221.8.0	21	193.233.8.0	50458	25500	17	2008/10/12T13:41:24.067	2008/10/12T13:42:19.523
148.219.120.0	3464	83.212.216.0	21	26780	19	2008/10/12T13:29:15.084	2008/10/12T13:30:03.394
163.221.8.0	21	193.233.8.0	50458	28500	19	2008/10/12T13:41:48.339	2008/10/12T13:42:39.841
163.221.8.0	21	193.233.8.0	50458	30000	20	2008/10/12T13:48:25.844	2008/10/12T13:49:18.339
163.221.8.0	21	193.233.8.0	50458	31500	21	2008/10/12T13:44:07.664	2008/10/12T13:45:04.778

We can see from the output that there is a major FTP control conversation between 163.221.8.0 and 192.233.8.0. At first, one might conclude, since we only have the first three octets and the data is sampled, that there is no conclusive proof that the conversation is between two specific hosts; it may be between multiple hosts on both sides. However, for the purposes of this paper, we will assume that it is between two hosts, due in large part to the ephemeral port being the same, 50,458, for the duration of

the resulting conversation. This large conversation made me wonder how long the conversation occurred and the total bytes and packets.

rwfilter 13.ftp.rw --aport=21 --sadd=163.221.8.0 --daddr=193.233.8.0 --proto=6 --pass=13.ftp.large.rw

This gives me a file containing only the FTP control conversation between these 2 addresses. I then want to output the information for easy sorting in Microsoft Excel:

rwcut 13.ftp.large.rw --fields=sip,dip,sport,dport,packets,bytes,stime,etime

Based on the output, I was able to obtain the following information:

Packets: 1,873
Bytes: 1,441,067
Duration: 24 hours

The timing, bytes, and packets are random enough to suggest the source IP is sending a lot of control information over a long period of time to the destination IP. I was also able to run the following command:

rwfilter pass*.13.rw --sadd=163.221.8.0 --daddr=193.233.8.0 --proto=6 --pass=stdout | rwcut --fields=sip,dip,sport,dport,packets,bytes,stime,etime

I also ran the command and saved the file as "13.ftp.second.large.rw". The output yielded the same results as when specifying any port as 21. This means the only 2 communication these two IP address was over FTP (aside from any web traffic).

Looking back at one of the previous results, we also see the following:

148.219.120.0	3464	83.212.216.0	21	26780	19	2008/10/12T13:29:15.084	2008/10/12T13:30:03.394

Since the previous conversation enveloped any further communication in this conversation, we decided to take a second look.

rwsort 13.ftp.second.large.rw --fields=stime | rwcut --fields=sip,dip,sport,dport,packets,bytes,stime,etime

The output:

sIP	dIP	sPort	dPort	packets	bytes	sTime	eTime
148.219.120.0	83.212.216.0	2625	21	1	108	2008/10/12T13:16:59.129	2008/10/12T13:16:59.129
148.219.120.0	83.212.216.0	3464	21	1	1460	2008/10/12T13:28:47.084	2008/10/12T13:28:47.084
148.219.120.0	83.212.216.0	3464	21	4	3838	2008/10/12T13:28:48.809	2008/10/12T13:29:03.937
148.219.120.0	83.212.216.0	3464	21	1	1460	2008/10/12T13:28:58.993	2008/10/12T13:28:58.993

148.219.120.0	83.212.216.0	3464	21	1	1460	2008/10/12T13:29:03.911	2008/10/12T13:29:03.911
148.219.120.0	83.212.216.0	3464	21	5	5933	2008/10/12T13:29:07.516	2008/10/12T13:29:35.817
148.219.120.0	83.212.216.0	3464	21	3	4380	2008/10/12T13:29:08.668	2008/10/12T13:30:01.781
148.219.120.0	83.212.216.0	3464	21	16	21127	2008/10/12T13:29:14.338	2008/10/12T13:30:04.753
148.219.120.0	83.212.216.0	3464	21	19	26780	2008/10/12T13:29:15.084	2008/10/12T13:30:03.394
148.219.120.0	83.212.216.0	3464	21	5	7300	2008/10/12T13:29:21.180	2008/10/12T13:30:02.210
148.219.120.0	83.212.216.0	3464	21	14	20440	2008/10/12T13:29:42.747	2008/10/12T13:30:30.702
148.219.120.0	83.212.216.0	3464	21	1	1460	2008/10/12T13:30:06.779	2008/10/12T13:30:06.779
148.219.120.0	83.212.216.0	3464	21	4	5840	2008/10/12T13:30:09.730	2008/10/12T13:30:15.459

As you can see, the first packet sent is 108 bytes, over ephemeral port 2625 occurred at 1:16:59 pm. However, this is the only piece of that conversation we have. The rest of the packets are over ephemeral port 3464. The conversation lasted about one minute and 22 seconds.

However, since the data stream for FTP traffic runs over port 20, we need to analyze the data for port 20.

rwfilter pass-*13.rw --pass=13.port20.rw --aport=20 --proto=6 --print-vol --print-file

rwsort 13.port20.rw --out=13.port20.rw.SORTED --fields=bytes

rwcut 13.port20.rw.SORTED --fields=sip,sport,dip,dport,bytes,packets,stime,etime | tail

The output:

Sip	Sport	Dip	Dport	Bytes	Packets	Stime	Etime
130.14.24.0	20	131.215.16.0	53899	3256500	2171	2008/10/12T13:49:08.729	2008/10/12T13:49:36.917
130.14.24.0	20	131.215.16.0	52260	3628500	2419	2008/10/12T13:48:37.011	2008/10/12T13:49:07.444
130.14.24.0	20	131.215.16.0	53899	4750500	3167	2008/10/12T13:49:36.992	2008/10/12T13:50:17.879
130.14.24.0	20	131.215.16.0	47147	4845000	3230	2008/10/12T13:50:19.191	2008/10/12T13:51:02.933
130.14.24.0	20	131.215.16.0	47147	5143500	3429	2008/10/12T13:50:19.486	2008/10/12T13:51:02.980
130.14.24.0	20	131.215.16.0	52260	5610000	3740	2008/10/12T13:47:39.098	2008/10/12T13:48:36.922
130.14.24.0	20	131.215.16.0	53899	6331500	4221	2008/10/12T13:49:08.610	2008/10/12T13:50:04.920
130.14.24.0	20	131.215.16.0	53899	6429000	4286	2008/10/12T13:49:09.190	2008/10/12T13:50:04.937

130.14.24.0	20	131.215.16.0	52260	6990000	4660	2008/10/12T13:48:04.997	2008/10/12T13:49:04.922
130.14.24.0	20	131.215.16.0	52260	7374000	4916	2008/10/12T13:48:04.988	2008/10/12T13:49:04.923

Based on this output, we see one major file transfer conversation going on here. In order to get the full details of the session though, we need to filter on the port 20 file for these two addresses:

rwfilter 13.port20.rw --saddr=130.14.24.0 --daddr=131.215.16.0 --pass=stdout | rwcut --fields=sip,sport,dip,dport,bytes,packets,stime,etime

Output:

sIP	sPort	dIP	dPort	bytes	packets	sTime	eTime
130.14.24.0	20	131.215.16.0	52260	85500	57	2008/10/12T13:47:07.932	2008/10/12T13:47:21.491
130.14.24.0	20	131.215.16.0	52260	1672500	1115	2008/10/12T13:47:08.348	2008/10/12T13:48:04.943
130.14.24.0	20	131.215.16.0	52260	1641000	1094	2008/10/12T13:47:09.128	2008/10/12T13:48:04.952
130.14.24.0	20	131.215.16.0	52260	5610000	3740	2008/10/12T13:47:39.098	2008/10/12T13:48:36.922
130.14.24.0	20	131.215.16.0	52260	7374000	4916	2008/10/12T13:48:04.988	2008/10/12T13:49:04.923
130.14.24.0	20	131.215.16.0	52260	6990000	4660	2008/10/12T13:48:04.997	2008/10/12T13:49:04.922
130.14.24.0	20	131.215.16.0	52260	3628500	2419	2008/10/12T13:48:37.011	2008/10/12T13:49:07.444
130.14.24.0	20	131.215.16.0	52260	1500	1	2008/10/12T13:48:45.693	2008/10/12T13:48:45.693
130.14.24.0	20	131.215.16.0	52260	315000	210	2008/10/12T13:49:05.002	2008/10/12T13:49:07.811
130.14.24.0	20	131.215.16.0	52260	300000	200	2008/10/12T13:49:05.012	2008/10/12T13:49:07.758
130.14.24.0	20	131.215.16.0	53899	6331500	4221	2008/10/12T13:49:08.610	2008/10/12T13:50:04.920
130.14.24.0	20	131.215.16.0	53899	3256500	2171	2008/10/12T13:49:08.729	2008/10/12T13:49:36.917
130.14.24.0	20	131.215.16.0	53899	1500	1	2008/10/12T13:49:09.181	2008/10/12T13:49:09.181
130.14.24.0	20	131.215.16.0	53899	6429000	4286	2008/10/12T13:49:09.190	2008/10/12T13:50:04.937
130.14.24.0	20	131.215.16.0	53899	4750500	3167	2008/10/12T13:49:36.992	2008/10/12T13:50:17.879
130.14.24.0	20	131.215.16.0	53899	1548000	1032	2008/10/12T13:50:04.982	2008/10/12T13:50:18.257
130.14.24.0	20	131.215.16.0	53899	1573500	1049	2008/10/12T13:50:04.989	2008/10/12T13:50:18.201
130.14.24.0	20	131.215.16.0	53899	648	1	2008/10/12T13:50:18.201	2008/10/12T13:50:18.201
130.14.24.0	20	131.215.16.0	47147	1914000	1276	2008/10/12T13:50:19.142	2008/10/12T13:50:36.906
130.14.24.0	20	131.215.16.0	47147	4845000	3230	2008/10/12T13:50:19.191	2008/10/12T13:51:02.933

130.14.24.0	20	131.215.16.0	47147	5143500	3429	2008/10/12T13:50:19.486	2008/10/12T13:51:02.980
130.14.24.0	20	131.215.16.0	47147	1328	1	2008/10/12T13:50:26.766	2008/10/12T13:50:26.766
130.14.24.0	20	131.215.16.0	47147	2911500	1941	2008/10/12T13:50:37.000	2008/10/12T13:51:02.666

Over the course of 3 minutes and 29 seconds, the source sent about 66,324,476 bytes in 44,217 packets.

We then wanted to see what else was going on, if anything, between these 2 hosts. We would expect to see some kind of control information being passed over port 21. We filtered all "pass" files for these 2 addresses in order to get a clear picture of what's going on.

```
rwfilter pass*.13.rw --sadd=130.14.24.0 --daddr=131.215.16.0 --proto=6 --pass=stdout | rwcut --fields=sip,dip,sport,dport,packets,bytes,stime,etime
```

Unfortunately, this yielded the same results as above, which indicates that there was no other communication occurring between these two hosts (aside from any web traffic). Since the data is sampled at 1:100 the command channel likely was small enough to not be in the sample set.

Web Analyses – "12"

Now we do further profiling of the data to find out the major clients and servers in the internet, serving a high load.

```
rwuniq passweb*12.rw --fields=1 | sort -r -t "|" -k 2 |less
```

sIP	Records
128.30.48.0	207598
65.55.208.0	87982
140.211.160.0	87472
72.164.152.0	85542
130.14.24.0	68057
129.105.40.0	63870
204.179.120.0	51201
216.178.32.0	46614
68.142.208.0	46292
128.223.8.0	46077
65.55.104.0	41636
65.54.80.0	37872
64.107.248.0	37407
74.6.16.0	35922
128.112.128.0	34312
65.55.184.0	33666
128.112.136.0	32369
156.56.240.0	30233

65.55.8.0	26847
66.249.64.0	26601
128.112.152.0	25107
72.14.200.0	23481
163.18.104.0	22397
65.55.192.0	21258
208.111.168.0	20439
209.62.184.0	19317
131.95.120.0	19204
171.66.120.0	17888
65.55.24.0	17660
207.46.16.0	17383
65.55.200.0	16965
207.46.160.0	16901
204.153.48.0	16802
146.137.96.0	16322
140.234.24.0	14969
18.7.16.0	14895

Let's further trim down the data for flows greater than 68057.

rwuniq passweb*12.rw --field=1 --flow=68057

sIP	Records
140.211.160.0	87472
130.14.24.0	68057
65.55.208.0	87982
128.30.48.0	207598
72.164.152.0	85542

We filter the data further by finding out the number of bytes and packers sent across from the client.

rwuniq passweb*12.rw --field=1 --bytes --packets --flow=68057

sIP	Bytes	Packets	Records
140.211.160.0	477717550	374779	87472
130.14.24.0	148595262	123933	68057
65.55.208.0	8740601	143310	87982
128.30.48.0	254423770	239903	207598
72.164.152.0	158779537	143828	85542

Filtering further

rwuniq passweb*12.rw --field=1,3 --bytes --packets --flow=68057

sIP	sPort	Bytes	Packets	Records
72.164.152.0	80	158698241	143738	85471
128.30.48.0	80	253913837	236503	204923
140.211.160.0	80	476931122	370825	85044

Since 128.30.48.0 has the maximum Records, we will try to identify the conversation taking place for that IP. Considering 128.30.48.0 as the source IP or client,

rwfilter passweb*12.rw --pass=stdout --saddr=128.30.48.0 | rwuniq --fields=2 | sort -r -t "|" -k 2 |head

dIP	Records
64.34.192.0	5683
85.19.80.0	2543
153.91.0.0	1565
209.56.112.0	1302
192.149.56.0	1223
80.76.152.0	950
97.65.48.0	932
74.6.16.0	902
81.149.136.0	828

This data tells us that with 128.30.48.0 client was majorly served by the above Servers. With 64.34.192.0, 5683 records were transmitted.

Similarly, considering 128.30.48.0 as the server, we will find out the major client it served.

rwfilter passweb*12.rw --pass=stdout --daddr=128.30.48.0 | rwuniq --fields=1 | sort -r -t "|" -k 2 |head

sIP	Records
74.6.16.0	866
209.56.112.0	618
202.115.120.0	602
69.50.136.0	535
70.166.128.0	402
222.197.32.0	323
70.167.24.0	253

128.101.64.0	234
72.204.16.0	202

Let's consider the first entry. Filtering out the conversation between the IP addresses 74.6.16.0 and 128.38.48.0,

rwfilter pass*.12.rw --sadd=74.6.16.0 --daddr=128.30.48.0 --proto=6 --pass=stdout |rwfilter --input-pipe=stdin --dport=80 --fail=12.web.fail.rw --print-vol

	Recs	Packets	Bytes	Files
Total	866	1044	73885	1
Pass	866	1044	73885	
Fail	0	0	0	

This tells us that the only conversation between these two IP's took place on port 80.

When we try to run the above command with the next set of IP addresses, we observed that for IP: 69.50.136.0, we got the following data.

rwfilter pass*.12.rw --sadd=69.50.136.0 --daddr=128.30.48.0 --proto=6 --pass=stdout |rwfilter --input-pipe=stdin --dport=80 --fail=12.web.fail.rw --print-vol

	Recs	Packets	Bytes	Files
Total	535	582	405168	1
Pass	86	119	5282	
Fail	449	463	399886	

This shows that there is something else going on, on these IP addresses.

To further evaluate, we run the following command

rwfilter pass*.12.rw --sadd=69.50.136.0 --daddr=128.30.48.0 --proto=6 --pass=stdout |rwfilter --input-pipe=stdin --aport=80,443 --fail=stdout|rwcut --fields=sip,dip,sport,dport,packets|less

We see that the conversation is happening on port 443 and 80. Port 443 is for SSL traffic so we can ignore that data. Running the same set of commands on a couple of other IP addresses, we find that the conversation is generally happening on port 80 and 443. Again, since the data is annonymized, some uncertainty exists.

Next, we try to look further into the bytes and packets sent by two IP addresses. We run the following command with 128.30.48.0 as client and 64.34.192.0 as server.

rwfilter passweb-*12.rw --pass=stdout --saddr=128.30.48.0 --daddr=64.34.192.0 |rwuniq --fields=1,2 --bytes --packets |head

sIP	dIP	Bytes	Packets
128.30.48.0	64.34.192.0	6430970	5938

This helps us in knowing the amount of data transferred between the two systems.

Web Analyses – "13"

In analyzing the web traffic,

rwfilter passweb-ATLA.13.rw --pass=13.atl.web.rw --aport=80,443 --proto=6 --print-vol

Results:

	Recs	Packets	Bytes	Files
Total	412923	1315535	904240231	1
Pass	400284	1273722	867758060	
Fail	12639	41813	36482171	

Instead of looking at the "pass" traffic, I wanted to see what else would be included in data since we have over 12,000 records failing the filter.

rwfilter passweb-ATLA.13.rw --pass=13.atl.web.rw --aport=80,443 --proto=6 --print-vol --fail=13.atl.failweb.rw

We then used rwstats to see what other ports were being used:

rwstats 13.atl.failweb.rw --sport --count=10

Results:

sPort	Records	%_of_total	cumul_%
8080	3299	26.101749	26.101749
10942	94	0.74373	26.845478
10941	87	0.688346	27.533824
3143	36	0.284833	27.818657
44083	27	0.213624	28.032281
12310	11	0.087032	28.119313

43458	11	0.087032	28.206345
2593	10	0.07912	28.285466
3937	9	0.071208	28.356674
54830	9	0.071208	28.427882

As we can see, the largest port being used in the “fail” file is 8080, a commonly used port for web traffic when trying to circumvent firewalls, proxies, or other administrative restrictions. The other ports listed are ephemeral ports when connecting to the port 8080 servers. Based on this, we will not worry about filtering the data beyond “passweb”.

Now, let’s get some general stats.

```
rwuniq passweb*13.rw --fields=1 | sort -r -t "|" -k 2 |less
```

Output (partial):

sIP	Records
128.30.48.0	217467
72.164.152.0	95110
140.211.160.0	92939
65.55.208.0	92858
204.179.120.0	66058
130.14.24.0	65177
129.105.40.0	62652
216.178.32.0	52987
65.55.104.0	43164
208.111.168.0	42079
65.54.80.0	39496
163.18.104.0	39423
65.55.184.0	39086

74.6.16.0	38794
64.107.248.0	37274
72.14.200.0	34928
65.55.192.0	33865
65.55.8.0	32016
209.62.184.0	31088

We can see the largest source IP is 128.30.48.0. Since this is Source IP, this must be a web server. I wanted to see what ports it was serving out though, to determine if it was just some basic web pages or something else:

rwfilter passweb*.13.rw --daddr=128.30.48.0 --proto=6 --pass=stdout | rwuniq --field=4 | sort -r -t "|" -k 2 | less

The results show 12,754 records over port 80 and 2 records over port 443. Everything else was an ephemeral port and there was nothing over port 8080. Curious about the 443 traffic, I decided to look into it more.

rwfilter passweb*.13.rw --daddr=128.30.48.0 --dport=443 --proto=6 --pass=stdout | rwcut --fields sip,sport,dip,dport,bytes,packets

Results:

sIP	sPort	dIP	dPort	bytes	packets
99.226.152.0	60030	128.30.48.0	443	40	1
157.89.72.0	51962	128.30.48.0	443	40	1

As we can see, it was two random IP subnets sending over a base packet over 443. Although this presents the question, what were those 2 source IPs doing, possibly scanning?

For 99.226.152.0:

rwfilter pass*.13.rw --saddr=99.226.152.0 --proto=6 --pass=stdout | rwcut --fields=sip,sport,dip,dport,bytes,packets,stime,etime

Results:

sIP	sPort	dIP	dPort	bytes	packets	sTime	eTime
99.226.152.0	10068	129.255.0.0	33695	680	2	2008/10/12T13:13:57.314	2008/10/12T13:14:01.739
	3724	66.71.56.0	2068	40	1	2008/10/12T13:14:53.663	2008/10/12T13:14:53.663

99.226.152.0							
99.226.152.0	10068	129.255.0.0	33695	3000	2	2008/10/12T13:15:58.938	2008/10/12T13:16:20.395
99.226.152.0	10068	129.255.0.0	33695	52	1	2008/10/12T13:18:06.455	2008/10/12T13:18:06.455
99.226.152.0	10068	129.255.0.0	33695	52	1	2008/10/12T13:19:15.515	2008/10/12T13:19:15.515
99.226.152.0	10068	129.255.0.0	33695	1500	1	2008/10/12T13:19:18.756	2008/10/12T13:19:18.756
99.226.152.0	49357	137.28.224.0	25886	280	7	2008/10/12T13:19:52.548	2008/10/12T13:19:57.564
99.226.152.0	10068	129.255.0.0	33895	1500	1	2008/10/12T13:20:01.749	2008/10/12T13:20:01.749
99.226.152.0	10068	129.255.0.0	33895	4500	3	2008/10/12T13:20:54.493	2008/10/12T13:21:26.198
99.226.152.0	10068	129.255.0.0	33695	1500	1	2008/10/12T13:21:37.147	2008/10/12T13:21:37.147
99.226.152.0	10068	129.255.0.0	33895	1500	1	2008/10/12T13:22:39.921	2008/10/12T13:22:39.921
99.226.152.0	3724	129.130.176.0	58314	715	1	2008/10/12T13:23:23.393	2008/10/12T13:23:23.393
99.226.152.0	10068	129.255.0.0	33895	1500	1	2008/10/12T13:23:52.972	2008/10/12T13:23:52.972
99.226.152.0	10068	129.255.0.0	33695	52	1	2008/10/12T13:24:32.412	2008/10/12T13:24:32.412
99.226.152.0	10068	129.255.0.0	29708	52	1	2008/10/12T13:38:40.232	2008/10/12T13:38:40.232
99.226.152.0	10068	129.255.0.0	34213	66	1	2008/10/12T13:37:59.654	2008/10/12T13:37:59.654
99.226.152.0	10068	129.255.0.0	29708	4500	3	2008/10/12T13:38:54.480	2008/10/12T13:39:08.842
99.226.152.0	3724	204.38.192.0	3641	40	1	2008/10/12T13:46:52.471	2008/10/12T13:46:52.471
99.226.152.0	61922	129.130.208.0	3724	615	1	2008/10/12T13:49:43.761	2008/10/12T13:49:43.761
99.226.152.0	60030	128.30.48.0	443	40	1	2008/10/12T13:10:29.612	2008/10/12T13:10:29.612
99.226.152.0	51411	152.14.8.0	80	40	1	2008/10/12T13:30:06.828	2008/10/12T13:30:06.828

This IP has a unique traffic pattern. Some of the largest flows were taking place over ports 10068 and 3724 (source port) and 33695 (destination port). I tried to do some research on these ports to see if they could tell me what was going on. According to http://portforward.com/cportsnotes/battlenet/wow.htm, Port 3724 is used for the online game World of WarCraft. The other ports did not appear to be specific to an application or game. However, since they are so high, they may have been used for other online games. This would make sense with our data sets: college students are highly active in online games. One can postulate that the users of the games knew each other and knew of their network over the Internet2 connection and setup the gaming server for their needs.

For 157.89.72.0:

rwfilter pass*.13.rw --saddr=157.89.72.0 --proto=6 --pass=stdout | rwuniq --fields=3 | sort -r -t "|" -k 2 | head

Results:

sPort	Records
1304	205
4550	204
3872	15
3394	15
2528	15
1548	12
49186	8
3125	8
59082	6

Here, we can see the source port is either 1304 or 4550. 1304 is a non-standard port, but 4550, according to a few web sites, is used for webcam traffic. This is something we should see when we do streaming video.

Getting back to the original largest web server, I wanted to see some other information about it. First, the client with the most packets then the one based on bytes:

rwfilter passweb*.13.rw --pass=stdout --daddr=128.30.48.0 --dport=80 --proto=6 | rwcut --fields=sip,sport,dip,dport,packets,bytes | sort -r -t "|" -k 5 | head

sIP	sPort	dIP	dPort	packets	bytes
74.6.16.0	59129	128.30.48.0	80	83	4316
74.6.16.0	59129	128.30.48.0	80	64	3328
74.6.16.0	59129	128.30.48.0	80	46	2392
74.6.16.0	36358	128.30.48.0	80	17	884
211.69.200.0	1419	128.30.48.0	80	14	560
74.6.16.0	37923	128.30.48.0	80	13	676
211.69.200.0	1419	128.30.48.0	80	13	520
202.115.88.0	2165	128.30.48.0	80	11	464
211.69.200.0	1419	128.30.48.0	80	11	440

rwfilter passweb*.13.rw --pass=stdout --daddr=128.30.48.0 --dport=80 --proto=6 | rwcut --fields=sip,sport,dip,dport,packets,bytes | sort -r -t "|" -k 6 | head

sIP	sPort	dIP	dPort	packets	bytes
74.6.16.0	59129	128.30.48.0	80	83	4316
74.6.16.0	59129	128.30.48.0	80	64	3328
74.6.16.0	59129	128.30.48.0	80	46	2392
163.28.32.0	64018	128.30.48.0	80	1	1400
74.6.16.0	36358	128.30.48.0	80	17	884
163.28.32.0	50532	128.30.48.0	80	1	851
211.69.200.0	1416	128.30.48.0	80	1	745
99.233.176.0	50112	128.30.48.0	80	1	683
163.28.32.0	50748	128.30.48.0	80	1	683

So we can see the IP subnet of 74.6.16.0is sending the most packets and the most bytes. We noticed the results weren't the same for both commands though. For the results based on bytes, we see a few IPs receiving 1 packet with 683-1400 bytes (the IPs differ and the byte sizes differ). We're unsure what was going on between these two, but perhaps there was a relatively large web page being served.

So I then wanted to see what was the largest web server that handled only HTTPS traffic.

rwfilter passweb*.13.rw --dport=443 --proto=6 --pass=stdout | rwuniq --fields=dip | sort -r -t "|" -k 2 | head

dIP	Records
65.55.184.0	39898
65.55.192.0	22702
65.55.200.0	21041
65.55.8.0	20771
207.46.16.0	14672
65.55.24.0	14598
65.55.48.0	14588
65.54.224.0	13091
65.55.152.0	12630

Based on these results, I then need to determine if any of these IPs appear to serve unsecure web traffic as well. To determine this I ran the following command for each of these IPs:

Rm 13.temp.rw && rwfilter passweb*.13.rw --daddr=<IP> --proto=6 --dport=80,8080 --pass=13.temp.rw --print-vol

The failures of this command will be 443 traffic and ephemeral port traffic (which I am ignoring). We didn't use --dport=443, because then the failures would be 80, 8080, and ephemeral port traffic, which won't give us the desired results. Unfortunately, running this for each of these 10 IPs yielded no results were the passes were zero, or even close to zero.

Here, a bash script or c-shell script would be beneficial to run against all results in the above command (not just the top 10).

Analyses Summary

In analyzing the FTP traffic, we found that there were many instances of control traffic that had no corresponding data traffic. We also found the reverse was true, instances of data traffic with no control traffic. However, the two major roadblocks here are the anonymization and the sampling. By only sampling 1/100 flows, we are essentially missing 99% of the flows. This means locating the complementary flows between FTP control and FTP data was very difficult. The anonymization was also a problem. The data was anonymized in such a unique way that we cannot be certain two flows with the same source IP and destination IP are in fact from the same source client and destination client. We can make that assumption (which we did), but there is no empirical proof. This made classifying the FTP traffic very difficult.

The web traffic was also uniquely affected by the sampling and more so by the anonymization. We were able to get some good conversations, but again, it's very difficult to get the clear understanding of what's going on when you can't be certain that two flows next to each other in a time series are from the same host. Perhaps one person found a site, then shared it with everyone on the same subnet. We'll never know the truth due to the anonymization.

Project Closure

Technically, this project isn't closed. However, for the purposes of this coursework, it is. We faced many challenges, as evident in the incompleteness and the analyses summary. There is also much more work to be done.

Challenges

Logistically, we lost focus frequently. We would begin down a path and then work on a tangent question. We would follow that tangent for awhile, which would prompt questions about different traffic. Not all of this is evident in this report, but many different lines of problems were investigated. Other challenges faced were regarding not having someone manage the project and ensure the resources were staying on task. Resource availability was also an issue; one of us is a full time employee that works well over 50 hours a week and the other is a full time student with many classes. Trying to find time to meet and work together was a problem. Technically, I already spoke about the issues with sampling and anonymization. Based on the layout of the network, we may have chosen a poor data set for this problem statement. We also don't have the technical skills needed to work on a project this size.

Future Work

There are a few things left to do on this project. The first is the continuation of the classification and the categorization. The next step is reviewing and investigating the traffic management systems. This will help us determine the impact they would have on the traffic. After completing a more final report, there should be some data validation to ensure the commands run were technically sound and the

report is accurate. Having someone with more technically abilities, both in Unix and with the SiLK tool suite, would help greatly with this project.

Conclusions

In conclusion, this project was an excellent beginning point the benefit of Network Situational Awareness and how the SiLK tools work as an enabler of the NetSA concept.