95-855 Project Proposal Part 3

Park Kittipatkul, parkk@cmu.edu
Attila Csokai, acsokai@andrew.cmu.edu
Kevin Tropeck, kdtropeck@cmu.edu
Woon Chiat Sim, wsim@andrew.cmu.edu

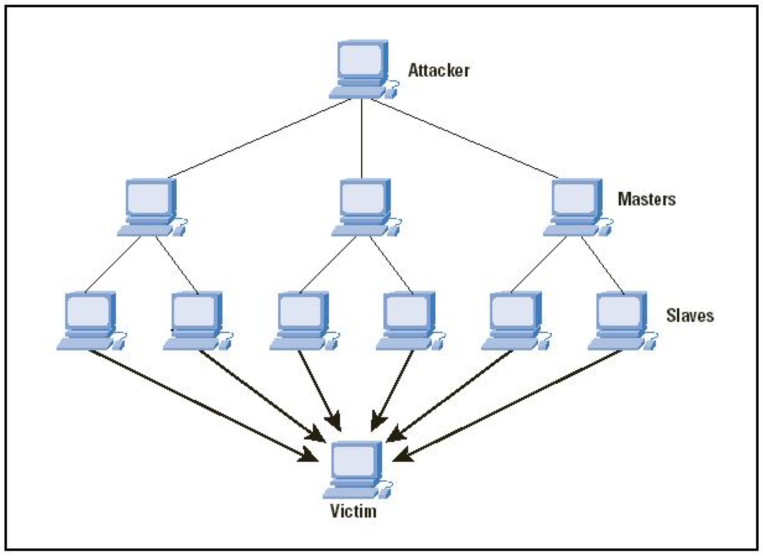
1. Problem Statement

On the surface, flow data entering and leaving a network looks the same. We can easily see the types of protocols used the IP addresses of hosts communicating and practically any statistics we may be interested in. However, the capture flow data in the right hands can reveal a lot more.

In this project, we will define various forms of malicious traffic, define its characteristics and isolate those traffic patterns entering and leaving an organization's network using SiLK filters. If possible, we will also attempt to match these malicious traffic patterns to known malware behaviors.

2. Definitions

Denial of Service (DoS). The goal of a DoS attack is to exhaust the resources of the target by overwhelming its communication requests. The end result is the inability of the host to respond to legitimate requests, therefore denying the service to all.



Source: http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_7-4/dos_attacks.html

Although there are many forms of DoS attacks, the most common are the SYN Flood, PING of Death, and SMURF. Using either of these methods, the attacker will take control of a number of computers, which in turn will be used to carry out the concentrated attack effort.

A ***SYN flood*** attack is a TCP/SYN connection request. The attacker sends multiple single packet SYN requests, usually using a spoofed source IP. Often, the forged source IP of the SYN flood is the victim's IP. Attackers using SYN flood, will usually have the SYN flag set only, as they do not care to complete the TCP handshake.

From the perspective of the host, the response to the spoofed source address is never responded to, leaving return SYN/ACK packets unanswered. The victimized host ends up with many half-open TCP connections, until a timeout results. Hosts can only handle a number of active half-open TCP connections.

PING Flood occurs when an attacker sends many ICMP echo requests to a specific host from multiple IP sources, again overwhelming the victim as it replies to the ICMP echo requests. In flow traffic, characteristics of PING flood attacks, identify there will be ICMP echo requests coming from one or multiple hosts to the same target with high frequency.

As a result the victimized host has many unanswered returned SYN/ACK packets, again leaving half-open TCP connections. With many ICMP requests coming in at one time, and trying to respond to these requests, this causes the host to become overwhelmed answering these ICMP requests.

SMURF is an amplified version of the ping flood attack. In this case, the attacker sends an ICMP echo request to a broadcast IP, hoping that the router will distribute the ping to all hosts multiplying the replies by the number of hosts that receive the PING. In this case the sign of a very high volume of unsolicited ICMP echo requests/replies are likely be indicative of the attack.

The host will have like responses to a PING flood or SYN flood, with multiple half-open TCP connections, but the incoming requests would be coming from a source address in the broadcast or network range (e.g. .255 or .0).

Port Scans look for open and or listening ports on hosts. The IP address of the scanner is usually spoofed, as it is interested in getting the results of the scan back. Port scans are usually small 1 – 3 packets/flow with a 40 bytes/packet. Another characteristic of a port scan is the high ratio of SYN requests versus ACK responses and follow-up connections.

> A ***Horizontal Scan*** will scan multiple destination IP addresses with a single or few destination ports.
> A ***Vertical Scan*** will scan a single destination IP address and multiple destination ports.
> A ***General Scan*** will scan just a few destinations IP addresses/ ports and may have to rely on the timing between sent SYN packets.

Spamming is generally defined as the mass mailing of unsolicited email to hosts which did not solicit those emails. The traffic characteristics of spamming traffic are to send a high volume of email in a short period of time, though the actual spamming activity may span hours or days. Generally, legitimate clients should not be sending mail to many distinct SMTP servers. We attempt to differentiate between SMTP clients and servers by looking at whether they respond to communication attempts on port 25. Also, a spammer should generally not be communicating on other ports other than port 25, and many of attempts to connect may meet with failures, as some spammer blindly attempt to connect to SMTP servers without actual knowledge of the validity of the server IP.

Worms – worms will try to copy themselves to other hosts and network shares without user interaction. Worms will use horizontal port scans as one method to locate other hosts and network shares. Furthermore, a worm scan will have a constant packet size and or flow. Worms may lie dormant for a while and begin scanning the destination IP only after some time has elapsed. The type of flow data generated is usually very worm specific, depending on what, who and how it is trying to infect.

Beaconing and Command and Control - beaconing is a signaling method by which a host indicates its presence. For our interest, beaconing is used by infected hosts to signal their presence and availability to the master. The signaling characteristics include regular timings per source and destination IP address pairs using DNS, IRC, and HTTP protocols as possible beaconing signals. More advanced malware may not be so regular to avoid detection. On average, the beaconing data will be fairly constant in terms of byte size and flow. Command and Control (C&C), is command script from a worms' author to the worms instructing them to perform certain tasks. C&C can generally be included as part of beaconing responses or in a separate message.

3. **Approach**

In our approach, we have looked at traffic patterns and characteristics of the malware defined above and have modeled our SiLK flow analysis in order to manipulate our data so that the results match those characteristics.

SYN flood

Attackers using SYN flood typically sends large amount of SYN packets to a small number of destinations, attempting to overflow the destinations' ability to establish connection. The attacker does not respond to the SYN/ACK sent by the victims. To identify such behavior we can look for:

```
--sync=1 --ack=0 --fin=0  (looking for flow where only the SYN
flag is set)
--proto=6 (TCP traffic only)
--pass=stdout

rwfilter datapath --syn=1 --ack=0 --fin=0 --proto=6 --pass=stdout
| rwuniq --fields=sip --all-counts --flows=high number
```

To create more targeted flow analysis, we can specify ports that are likely to be targets of a SYN flood attack. One such port is destination port 80. Furthermore, we can also attempt to see how many distinct destination IP's exhibit characteristics of such an attack. The following rwfilter command would help us here:

```
rwfilter /afs/andrew/course/95/855/ext3/2008* --syn=1 --ack=0 --
fin=0 --proto=6 --dport=80 --pass=stdout | rwuniq --fields=sip --
all-counts --flows=500 --dip-distinct
```

SYN Scanning

One common way to perform a port scan is to send a SYN packet to the port to be scanned. If the destination responds with a SYN/ACK, it means that the there is a service listening on that port. We can use the method similar to identifying SYN floods to identify possible scanners. However, the volume will usually be much lower than that of a flood:

```
rwfilter ../2008* --pass=stdout --packets=1 --flags-all=S/SFAPR |
rwstats --flows --count=30 --sip
```

To gain a higher confidence in classifying whether a particular source IP is a scanner, we can investigate the distinct destination and ports that it is "scanning". For example, if we're interested in a possible scanner with an IP of 6.204.20.223, we can use the following filter:

```
rwfilter ../200805151400 --pass=stdout --flags=S/SFAPRU --proto=6
--saddress=6.204.20.223 | rwuniq --fields=dip,dport | sort -k 1
```

Spamming

In order to identify spammers, we first narrow down a list of IPs that connects to multiple SMTP servers while not acting as SMTP servers themselves. To further narrow down the list, we examine the shortlisted IPs and see if they communicate using other protocols besides SMTP. It would be kind of strange if they are not SMTP servers and they mainly communicate in SMTP. Also, we also check if their peers rejected many of their attempted SMTP connections. If so, it is a sign that they are blindly trying to connect to possible SMTP servers without legitimate knowledge.

We set different thresholds for determining how many SMTP connections an IP must make before we investigate it, and how many distinct destination SMTP servers it connects to before we classify it as a possible spammer. In general, the steps taken are as follows:

1. Determine the set of IPs that make more than a threshold (e.g. 10) of SMTP connections. The following command does that:

```
rwfilter ../200809161400 --pass=stdout --flags-all=SAF/SAF --
packets=5- --dport=25 | rwuniq --fields=sip --flows=10 --no-title
--delimited=" " | awk '{print $1}' | rwsetbuild stdin
200809161400.smtp.clients.set
```

2. Next, we need to find the set of possible SMTP servers. In general, we determine whether an IP is acting as a server by seeing if the first packet it sends out contains SYN_ACK flags. Again, we set a minimum number of such flows before we classify an IP as a server. For example, the following command will list a set of IP and the possible server ports:

```
rwfilter ../200809161400 --pass=stdout --packets=5- --flags-
initial=SA/SFA  | rwuniq --fields=sip,sport --flows=5
```

To find a specific type of server, we filter for the source port of the connections. To generate a IP set of possible SMTP servers, we perform the following:

```
rwfilter ../200809161400 --pass=stdout --packets=5- --flags-
initial=SA/SFA --sport=25  | rwuniq --fields=sip,sport --flows=5
--no-title --delimited=" " | awk '{print $1}' | rwsetbuild stdin
00809161400.smtp.servers.set
```

3. Using the two IP sets generated above, we can determine a set of IPs for possible spammers by determining the IPs that appear in the set obtained from (1) and not (2).

```
rwsettool --difference 200809161400.smtp.clients.set
00809161400.smtp.servers.set --
output=200809161400.possible.spammers.set
```

4. We can see the set of IPs by doing a rwcatset on it:

```
rwcatset 200809161400.possible.spammers.set
```

5. We can further investigate each IP by investigating the number of distinct destination IPs it is connecting to, for example, if we suspect a IP 192.117.128.200 of being a spammer:

```
rwfilter ../200809161400 --pass=stdout --flags-all=SAF/SAF --
packets=5- --dport=25 --saddress=192.117.128.200 | rwuniq --
fields=dip
```

6. To see further confirm whether the suspect is a spammer, we can also look at the statistics of its communication. If it is a spammer, the SMTP communication should outweigh the rest of the other communication by a significant amount. To see the breakdown, we can use rwstat, e.g.:

```
rwfilter ../200809161400 --pass=stdout --flags-all=SAF/SAF --
saddress=192.117.128.200 | rwstats –dport –count=10
```

7. As noted above, a spammer will most likely receive many RST for its attempted connections to SMTP servers. To see if a suspect received my RST to its attempts:

```
rwfilter ../200809161400 --pass=stdout --flags-initial=R/SAFR --
daddress=192.117.128.200 --sport=25| rwcut --fields=sip,dip
```

Worms Spreading and Beaconing

Worms spreading can utilize many approaches, including finding exploits on well-known services, finding previously installed known-worms or backdoors.

Since many of the worms and backdoors utilize high port numbers, we started by trying to gather information about high port to high port connections in our data set.

Destination IP with top high port flow counts:

```
rwfilter ./2008* --sport=1024- --dport=1024- --pass=stdout |
rwstats --dip --top --count=50
```

Source IP with top high port flow counts:

```
rwfilter ./2008* --sport=1024- --dport=1024- --pass=stdout |
rwstats --sip --top --count=50
```

Destination port with top high port flow counts

```
rwfilter ./2008* --sport=1024- --dport=1024- --pass=stdout |
rwstats --dport --top --count=10
```

Source port with top high port flow counts

```
rwfilter ./2008* --sport=1024- --dport=1024- --pass=stdout |
rwstats --sport --top --count=10
```

From the initial analysis, we found that the top ports that constitutes the flow counts are 3127, 3128, 3124

Periodicity

One of the characteristics of early worms is the periodic connection to one or more command servers, either to report presence or to obtain commands and updates. One possible way of spotting such traffic is to determine the intervals between connections for a pair of source and destination IPs. This is not foolproof, because of the following:

i. The malware may be reporting to more than one command server, and hence, looking at specific intervals between fixed pair of IPs may not reveal the periodicity.

ii. The malware may be intelligent enough to vary the intervals between communications.

iii. Some legitimate software may also exhibit similar behavior. For example, virus signature updates and software updates.

However, it is still useful to be able to isolate such behaviors for further investigations. If given un-anonymized datasets, we may be able to further determine if the server being connected is legitimate. In any case, to determine the set periodicity, we perform the following:

1. Sort the data set by start time of the flow.
2. For each pair of communicating IPs, record the start time.
3. Determine the deltas between connections.
4. Bin the deltas (possible in bin intervals of 1 sec).
5. See if there are significant occurrences in certain interval bins for a pair of IPs.

Modification of the above can look at the end time to start time intervals instead.

4. Dataset

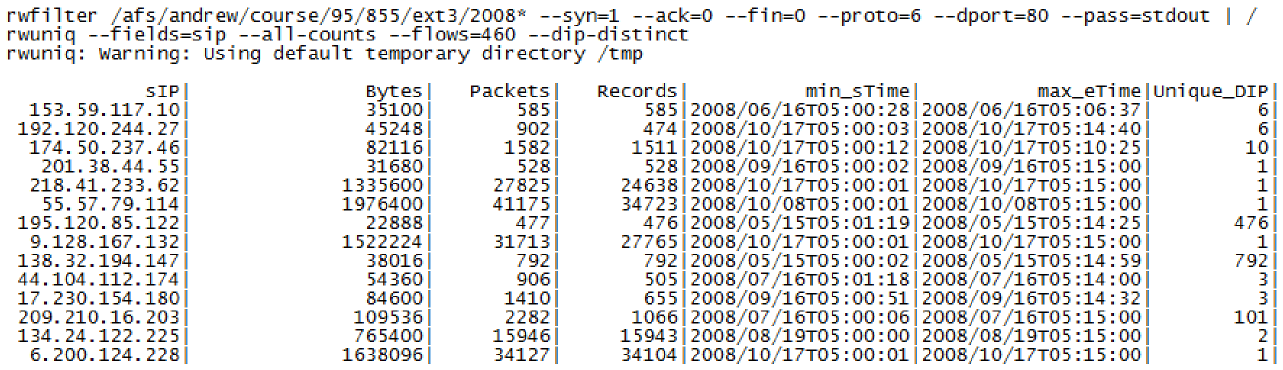
The data used from the analysis is obtained from the anonymized packet traces from the WIDE-TRANSIT 150 megabit Ethernet connection. The captures were 15 minutes in duration and un-sampled. The packet traces were converted to flow data records of the form understood by SiLK using the `rwp2yaf2silk` tool. The packet captures used were for the following dates: 2008/05/15, 2008/06/16, 2008/07/16, 2008/08/19, 2008/09/16, 2008/10/08 and 2008/10/17.

5. Proof-of-Concept and Results

Below we present the results of our investigations.

SYN Floods

Using the approach we defined earlier, we attempt to look for SYN floods. In this case, we have attempted to further restrict the results by specifying higher flows to port 80 with unique destination IP addresses.



```
rwfilter /afs/andrew/course/95/855/ext3/2008* --syn=1 --ack=0 --fin=0 --proto=6 --dport=80 --pass=stdout | /
rwuniq --fields=sip --all-counts --flows=460 --dip-distinct
rwuniq: Warning: Using default temporary directory /tmp

            sIP|                 Bytes|    Packets|    Records|          min_sTime|          max_eTime|Unique_DIP|
  153.59.117.10|                 35100|        585|        585|2008/06/16T05:00:28|2008/06/16T05:06:37|         6|
 192.120.244.27|                 45248|        902|        474|2008/10/17T05:00:03|2008/10/17T05:14:40|         6|
  174.50.237.46|                 82116|       1582|       1511|2008/10/17T05:00:12|2008/10/17T05:10:25|        10|
   201.38.44.55|                 31680|        528|        528|2008/09/16T05:00:02|2008/09/16T05:15:00|         1|
  218.41.233.62|               1335600|      27825|      24638|2008/10/17T05:00:01|2008/10/17T05:15:00|         1|
   55.57.79.114|               1976400|      41175|      34723|2008/10/08T05:00:01|2008/10/08T05:15:00|         1|
 195.120.85.122|                 22888|        477|        476|2008/05/15T05:01:19|2008/05/15T05:14:25|       476|
  9.128.167.132|               1522224|      31713|      27765|2008/10/17T05:00:01|2008/10/17T05:15:00|         1|
 138.32.194.147|                 38016|        792|        792|2008/05/15T05:00:02|2008/05/15T05:14:59|       792|
 44.104.112.174|                 54360|        906|        505|2008/07/16T05:01:18|2008/07/16T05:14:00|         3|
 17.230.154.180|                 84600|       1410|        655|2008/09/16T05:00:51|2008/09/16T05:14:32|         3|
 209.210.16.203|                109536|       2282|       1066|2008/07/16T05:00:06|2008/07/16T05:15:00|       101|
 134.24.122.225|                765400|      15946|      15943|2008/08/19T05:00:00|2008/08/19T05:15:00|         2|
  6.200.124.228|               1638096|      34127|      34104|2008/10/17T05:00:01|2008/10/17T05:15:00|         1|
```

SYN Scans

Using the approach we defined earlier, we attempt to look for SYN scanners.

```
rwfilter ../2008* --pass=stdout --packets=1 --flags-all=S/SFAPR |
rwstats --flows --count=30 --sip


sIP|             Records|%_of_total|   cumul_%|
  110.233.8.118|               65536|  7.872249|  7.872249|
  36.108.241.10|               65536|  7.872249| 15.744498|
   192.75.196.6|               65284|  7.841978| 23.586476|
 219.122.177.91|               56259|  6.757887| 30.344363|
   37.40.77.187|               45819|  5.503823| 35.848186|
  6.200.124.228|               34081|  4.093843| 39.942030|
   6.164.31.121|               32729|  3.931440| 43.873469|
 192.110.27.144|               31735|  3.812039| 47.685509|
   55.57.79.114|               29232|  3.511377| 51.196886|
    65.78.11.29|               26368|  3.167350| 54.364236|

<snip>

 142.181.50.233|                1434|  0.172253| 85.356771|
  214.10.97.153|                1207|  0.144986| 85.501757|
 192.192.130.54|                1122|  0.134776| 85.636533|
   6.204.20.223|                1024|  0.123004| 85.759537|
 209.210.16.203|                 939|  0.112794| 85.872331|
  193.104.10.76|                 829|  0.099580| 85.971911|

<snip>
```

Further investigation into a specific IP may reveal some interesting information:

```
rwfilter ../2008*  --pass=stdout  -flags-all=S/SFAPRU --proto=6  --
saddress=6.204.20.223 | rwuniq --fields=dip,dport | sort -k 1

dIP          |dPort|    Records|
193.64.60.0|   80|          1|
193.64.60.0| 7212|          1|
193.64.60.0| 8000|          1|
193.64.60.0| 8080|          1|
193.64.60.1|   80|          1|
193.64.60.1| 7212|          1|
193.64.60.1| 8000|          1|
193.64.60.1| 8080|          1|
193.64.60.2|   80|          1|
193.64.60.2| 7212|          1|
193.64.60.2| 8000|          1|
193.64.60.2| 8080|          1|
193.64.60.3|   80|          1|
193.64.60.3| 7212|          1|
193.64.60.3| 8000|          1|
193.64.60.3| 8080|          1|
<snip>
```

We can see the IP seems to be performing a horizontal scan for a small number of web and proxy related ports (80, 7212, 8000 and 8080). It is not conclusive but the “scanner” maybe looking for open proxies.

Spamming

We took the approach described earlier to look for spammers and wrote a script (**find_spammers.py**). Running the script against a SiLK-formatted dataset generated the following results:

./find_spammers.py ../200805151400

<snip>

```
---> Possible spammer's IP: 207.98.32.161
            dIP|   Records|
207.175.206.133|         1|
 207.16.229.140|         1|
   48.32.84.215|         1|
   194.108.6.70|         1|
221.186.211.202|         1|
   76.162.84.78|         1|
  82.255.24.111|         1|
   195.200.19.4|         1|
 192.192.123.53|         1|
    48.3.228.56|         1|
 192.192.123.52|         1|
  66.193.199.46|         1|
  82.255.24.152|         1|
   221.50.7.183|         1|
 48.216.152.218|         1|
 202.101.102.50|         1|
  78.137.82.108|         1|
   76.180.16.59|         1|
  220.209.81.60|         1|
 67.116.241.209|         1|
221.186.155.119|        21|
    220.78.92.3|         1|
 201.68.118.196|         1|
    65.53.68.41|          1|
  221.185.76.47|          1|
202.101.102.152|          1|
 201.213.53.183|          1|
 202.101.97.109|          1|
221.213.193.127|          1|
202.101.103.149|          1|
   79.34.164.85|          1|
    65.53.68.36|          1|
 221.77.148.198|          1|
    71.40.6.150|          1|
 221.186.155.19|         22|
  64.227.43.150|          1|
   68.3.130.151|          1|
  211.43.166.30|          1|
 202.101.97.105|          1|
     5.76.72.49|          1|
    66.62.14.21|          1|
202.101.102.145|          1|
   86.50.234.96|          1|
   66.32.249.17|          1|
  202.101.96.73|          1|

Protocol distribution:
             25|                   86| 97.727273| 97.727273|
          22942|                    1|  1.136364| 98.863636|
           4490|                    1|  1.136364|100.000000|

RSTs received from:
            sIP|
221.186.155.119|
221.186.155.119|
   48.32.84.215|
  220.209.81.60|
  220.209.81.60|
    71.40.6.150|
 48.216.152.218|
221.186.155.119|
 202.40.249.140|
  66.193.199.46|
 67.116.241.209|
  211.43.166.30|
 221.186.155.19|
 221.186.155.19|
```

<snip>

From the lists of results generated by script, we see that the anonymized IP `207.98.32.161` sent many SMTP requests to SMTP servers, “spoke” mainly SMTP, and received many RSTs from different servers. This fits the profile of a spam agent that we have outlined above. Note that is result is from a period of 15 minutes, and if more data was collected for a longer period, we may have stronger confirmation.

Again, this result is non-conclusive, and still quite manual, since we have to “eyeball” the results. Future work can involve the generation scripts that collate the different “signals” and generate an alert when most of the conditions are met.

Worms

From discussed approach on finding connections on high ports to high ports, we ranked out top IP addresses and top port that constitute the connection.

```
rwfilter ./2008* --sport=1024- --dport=1024- --pass=stdout | rwstats --dip --top --count=50
INPUT SIZE: 3242607 records for 1100719 unique keys
DESTINATION IP Key: Top 50 flow counts
            dIP|        Records|%_of_total|   cumul_%|
    163.7.48.83|         106360|  3.280077|  3.280077|
  160.87.172.28|          51999|  1.603617|  4.883694|
 209.22.177.208|          40744|  1.256520|  6.140214|
    95.7.189.72|          35037|  1.080519|  7.220733|
 217.11.114.177|          34087|  1.051222|  8.271955|
208.172.168.147|          33236|  1.024978|  9.296933|
208.172.168.153|          28782|  0.887619| 10.184552|
 192.120.254.26|          25672|  0.791709| 10.976261|
  8.179.206.104|          24500|  0.755565| 11.731826|
```

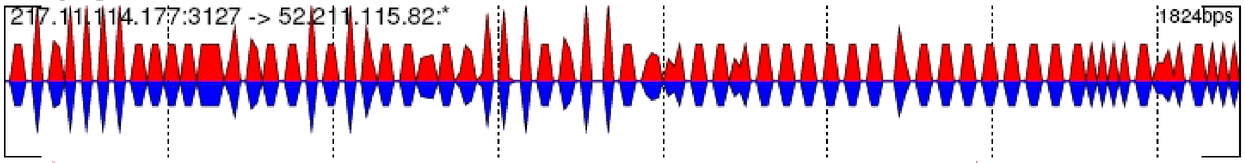
```
rwfilter ./2008* --sport=1024- --dport=1024- --daddress=160.87.172.28 --pass=stdout | rwstats --dport --top --count=10
INPUT SIZE: 51999 records for 263 unique keys
DESTINATION PORT Key: Top 10 flow counts
 dPort|        Records|%_of_total|   cumul_%|
  3127|          23650| 45.481644| 45.481644|
  3128|          23034| 44.297006| 89.778650|
  3124|           4399|  8.459778| 98.238428|
 12000|            144|  0.276928| 98.515356|
  2121|             75|  0.144234| 98.659590|
  8080|             45|  0.086540| 98.746130|
  4121|             37|  0.071155| 98.817285|
```

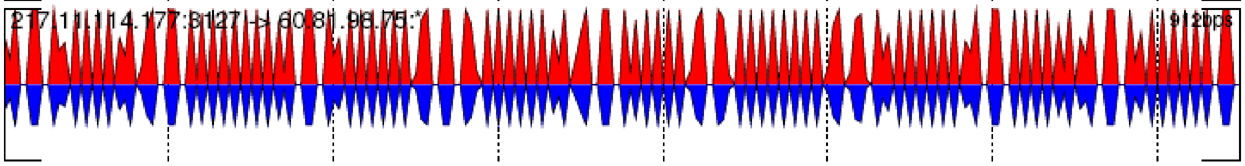
Port 3127 which constitutes most of the high port flows is commonly used by myDoom virus as a backdoor port. We dig down more on connections on that ports.

```
rwfilter ./2008* --sport=3127 --dport=1024- --pass=stdout | rwstats --dip --top --count=50
INPUT SIZE: 152318 records for 3667 unique keys
DESTINATION IP Key: Top 50 flow counts
            dIP|        Records|%_of_total|   cumul_%|
  8.179.206.104|          24492| 16.079518| 16.079518|
    95.7.189.72|          17568| 11.533765| 27.613283|
 209.22.177.208|          16331| 10.721648| 38.334931|
    9.9.247.197|           9609|  6.308512| 44.643443|
    73.16.75.75|           4883|  3.205793| 47.849236|
   36.178.31.73|           4104|  2.694363| 50.543600|
 213.85.234.101|           3096|  2.032590| 52.576189|
 217.88.135.184|           2213|  1.452881| 54.029071|
 220.129.136.52|               2136|  1.402329| 55.431400|
```

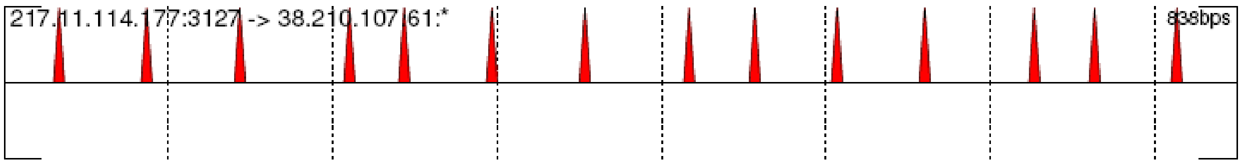
After that, we tried to plot the graph from the data we filtered previously to get the sense of the connection and timing of the usage of that port.

The graph below shows selection of the results.

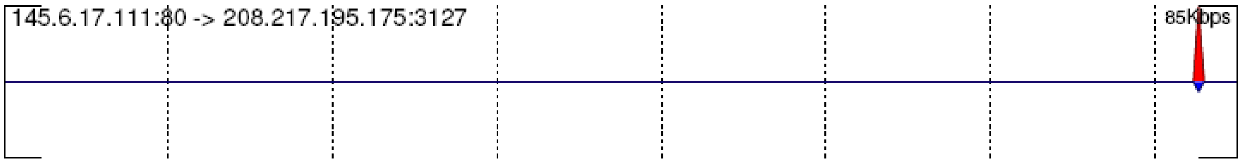




The two graphs above show consistent interval of connection on port 3127 which can be assumed as malicious connectivity connecting to myDoom backdoor.



The graph above shows interval attempt to connect to port 3127 but there were no reply to the request. This is very likely be the scanning attempt for an open backdoor port.



Port 3127 is not used only for backdoor. The graph above shows what is likely to be legitimate connection. The client uses 3127 as ephemeral port to connect to web server on port 80.

Beaconing

We have also written a python script to calculate the time interval between the flows of distinct pairs of source, destination IPs. It classifies the intervals by seconds and display the number of times only if > 5, adjustable) significant intervals that are more than 4 (also adjustable) seconds apart appears for each pair.

Usage:

Only flows with small packet counts were selected, and they were sorted by stime:

```
rwfilter ../200809161400 --pass=stdout --proto=6 -packets=3-6 |
rwsort --field=sip,dip,stime >
200809161400.low.packet.flows.sort.rw
```

The results are fed to the script:

```
./intervals.py 200809161400.low.packet.flows.sort.rw
```

The script only displays intervals that occurs more that 10% for each pair of IPs. The results of the script looks like this:

<snip>

```
Src:  95.128.77.51         Dst:  215.53.11.143
        Seconds        Count         Percent
***           14         8          10.4
***           12         14          18.2
***           13         9          11.7
***           10         10          13.0
***           11         24          31.2


Src:  162.87.245.33         Dst:  4.118.134.226
        Seconds        Count         Percent
***           60         10          52.6


Src:  203.85.117.153         Dst:  138.195.67.7
        Seconds        Count         Percent
***           23         14          51.9


Src:  216.168.93.253         Dst:  138.195.67.7
        Seconds        Count         Percent
***            6         6          12.0
***            8         8          16.0
```

<snip>

From the above example, we see that there is 10 flows between 162.87.245.33 and 4.118.134.226 every 60 seconds. Again, this is non-conclusive but we can use this as a starting point for further investigation into the communications between that pair of IPs. Again, since we are only using 15 minutes worth of data for each day, many of the patterns may not show themselves.

6. Threat Assessment

Through understanding the characteristics of common malicious traffic and data analysis of filter sets that were drawn together based on these theories we were able to focus in on particular events that led to our findings, which can be considered real threats to an unprotected network.

Our flow analysis was able to provide some evidence of different malicious traffic within the representative data, which consisted of fifteen minutes intervals (per day). Considering the results and the small time interval of the analysis, it would seem to indicate that the amount of malicious traffic may be much larger than what we have detected.

The flow analysis gave insight to patterns and indications of exemplary data typical of malicious activities that normally would go undetected without flow analysis. A threat is only detectable when traffic can be parsed and identified to find indications of such. The fact of the matter is that some malicious traffic normally goes undetected on a network, unless a problem is discovered. Our analysis and findings give indication that preemptive understanding should be taken into consideration in the signature base or indicative flow of types of malicious traffic in order to find or understand the threat and the vulnerability that may exist to the threat.

7. References

[1] Symantec, "Worm W32.Spybot.ABDO,"
http://securityresponse.symantec.com/avcenter/venc/data/w32.spybot.abdo.html, December 2005
[2] ETH, "DDoS Attack Detection Based on Netflow Logs,"
ftp://www.tik.ee.ethz.ch/pub/students/2003-So/SA-2003-35.pdf,
July 2003

[3] Securityfocus, "Detecting Worms and Abnormal Activities with NetFlow, Part 1"
http://www.securityfocus.com/print/infocus/1796
August 2004

[4] Securityfocus "Detecting Worms and Abnormal Activities with NetFlow, Part 2"
http://www.securityfocus.com/print/infocus/1802
September 2004

[5] USENIX, "More Netflow Tools: For Performance and Security "
http://www.usenix.org/event/lisa04/tech/full_papers/gates/gates_html/
November 2004

[6] TCP Port 3127
http://www.linklogger.com/TCP3127.htm