ISP Best Practices

Addressing a DDoS Attack on a Host

Hervey Allen
Network Startup Resource Center

June 28, 2010
PacNOG 7 Conference
Pago Pago, American Samoa

Distributed Denial of Service Attack
Summary of an Attack

One of many typical DDoS scenarios…

- Took place in 2008 against a University of Oregon host.
- Host included (at the time) nsrc.org as well 12 other sites, several not in uoregon.edu domain.

- I'll talk about:
  - Symptoms
  - Figuring out the attack
  - Mitigating the attack
  - Other possible resolutions

Overview: What is a “DDoS”

DDoS ➔ “Distributed Denial of Service” Attack

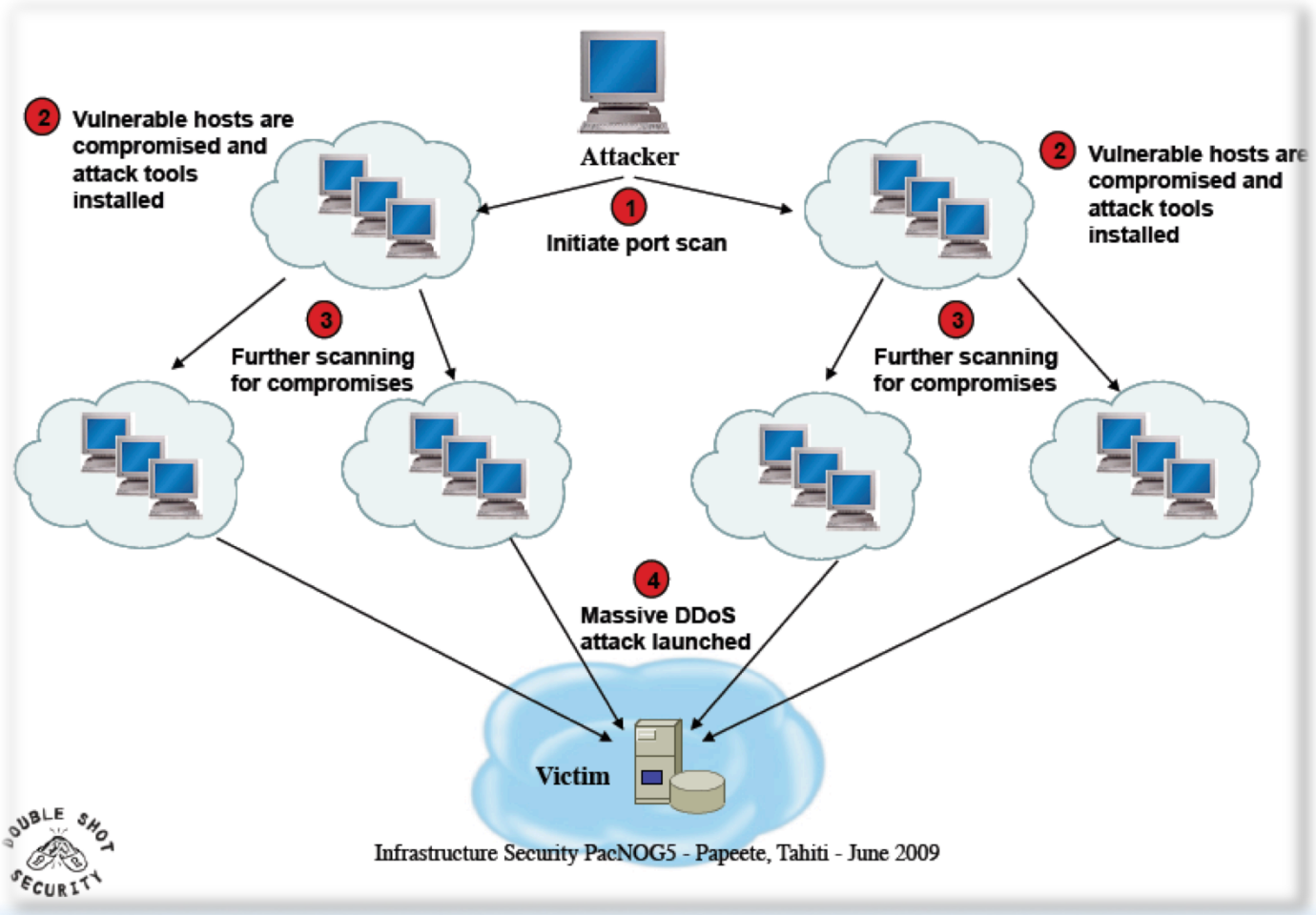
DOS ➔ “Denial of Service” Attack

“A **denial-of-service attack** (**DoS attack**) or **distributed denial-of-service attack** (**DDoS attack**) is an attempt to make a computer resource unavailable to its intended users. It generally consists of the concerted efforts of a person or people to prevent an Internet site or service from functioning efficiently or at all, temporarily or indefinitely.”*

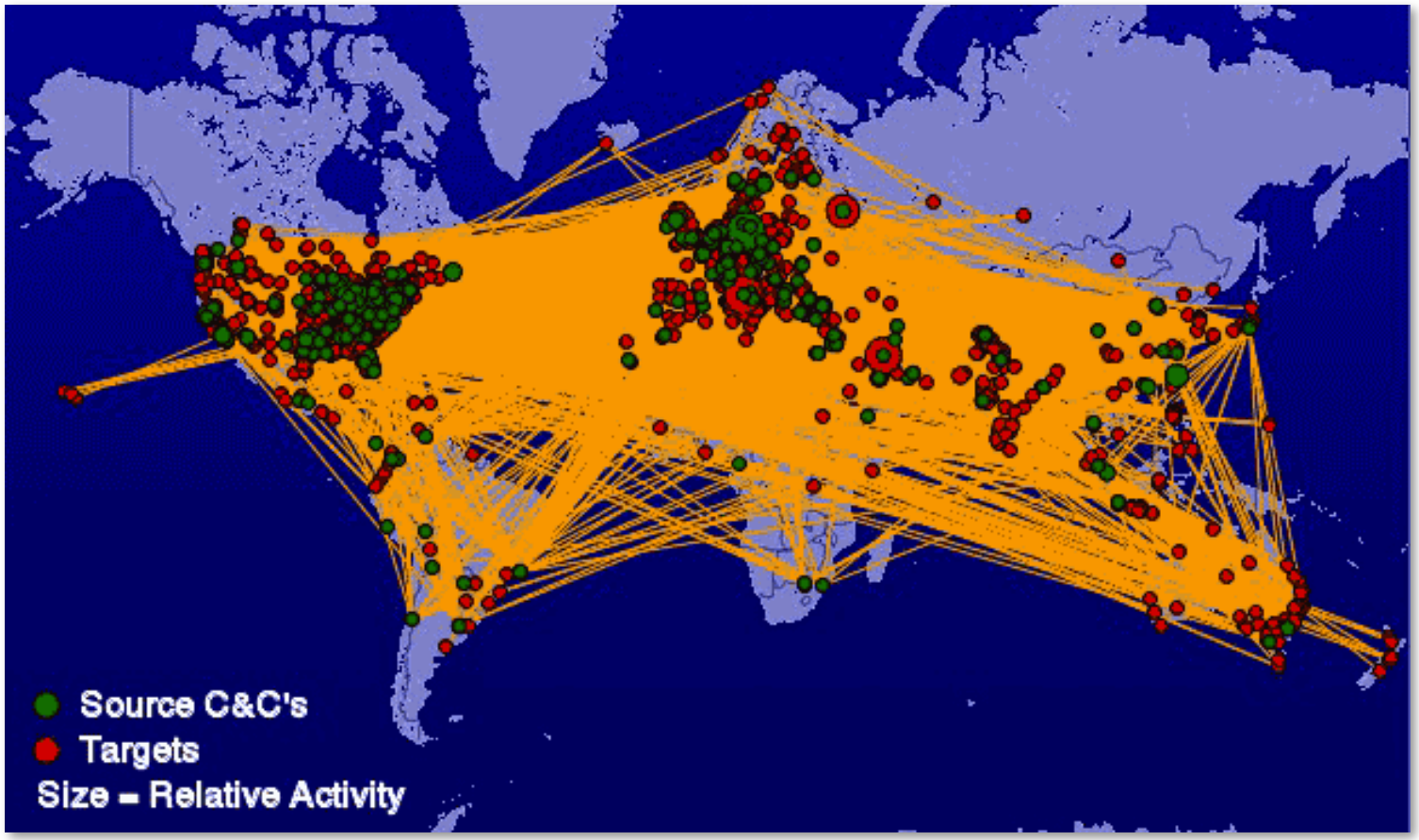
*http://en.wikipedia.org/wiki/DDoS#Distributed_attack

Automated DDoS Attack

Luckily this was not what happened to us…



World-Wide DDoS Attacks in 2007



Courtesy of shadowserver.ogr

The Attack: Symptoms

June 2008

- Web sites hosted by the server under attack become non-responsive.
- Load average on the server (Fedora Core Linux) was varying between 8 and 15.

At first it was not obvious why…

The Attack: Discovery

Not immediately obvious what was happening:

- Network traffic was not unusually high.
- Load average was cyclic.
- Server would become reasonably responsive.
- Upon restart of web services server would work fine for a while…

The Attack: Discovery

Forensic investigation began:

- Methodically verify all services running:

```
ps auxwww | less
```

  - No obvious problems

- Check all running network services:

```
netstat –atv, netstat –o –t, etc.
lsof –i
```

- *lsof* clearly showed a suspiciously large number of apache processes being spawned:

```
lsof –i | grep http | wc -l
```

The Attack: Discovery

Forensic investigation continued:

"`lsof –i | grep http | wc -l`" made it clear that all 256 available apache processes were in use. This was unusual for this box.

Next we dug a bit deeper:

- What were these processes attaching to?
  - First picked a few of the process IDs associated with http sessions in either `ESTABLISHED` or `CLOSE_WAIT` states:

lsof -i | grep httpd

```
httpd      8088    apache   32u  IPv6 3571963678       TCP limestone.uoregon.edu:http->180.98.61.58.broad.sz.gd.dynamic.163data.com.cn:wacp (ESTABLISHED)
httpd      8089    apache    4u  IPv6 2263806398       TCP *:http (LISTEN)
httpd      8089    apache    8u  IPv6 2263806403       TCP *:https (LISTEN)
httpd      8089    apache   32u  IPv6 3572005891       TCP limestone.uoregon.edu:http->123.65.209.149:15648 (CLOSE_WAIT)
httpd      8166    apache    4u  IPv6 2263806398       TCP *:http (LISTEN)
httpd      8166    apache    8u  IPv6 2263806403       TCP *:https (LISTEN)
httpd      8166    apache   32u  IPv6 3572011333       TCP limestone.uoregon.edu:http->112.81.71.171:siebel-ns (ESTABLISHED)
httpd      8167    apache    4u  IPv6 2263806398       TCP *:http (LISTEN)
httpd      8167    apache    8u  IPv6 2263806403       TCP *:https (LISTEN)
httpd      8167    apache   32u  IPv6 3572011561       TCP limestone.uoregon.edu:http->218.90.214.155:ms-rule-engine (ESTABLISHED)
httpd      8168    apache    4u  IPv6 2263806398       TCP *:http (LISTEN)
httpd      8168    apache    8u  IPv6 2263806403       TCP *:https (LISTEN)
httpd      8168    apache   32u  IPv6 3572013440       TCP limestone.uoregon.edu:http->112.81.71.171:netangel (ESTABLISHED)
httpd      8169    apache    4u  IPv6 2263806398       TCP *:http (LISTEN)
httpd      8169    apache    8u  IPv6 2263806403       TCP *:https (LISTEN)
httpd      8169    apache   32u  IPv6 3572014726       TCP limestone.uoregon.edu:http->218.90.214.155:cpqrpm-agent (ESTABLISHED)
httpd      8170    apache    4u  IPv6 2263806398       TCP *:http (LISTEN)
httpd      8170    apache    8u  IPv6 2263806403       TCP *:https (LISTEN)
httpd      8171    apache    4u  IPv6 2263806398       TCP *:http (LISTEN)
httpd      8171    apache    8u  IPv6 2263806403       TCP *:https (LISTEN)
httpd      8172    apache    4u  IPv6 2263806398       TCP *:http (LISTEN)
httpd      8172    apache    8u  IPv6 2263806403       TCP *:https (LISTEN)
httpd      8172    apache   32u  IPv6 3572014865       TCP limestone.uoregon.edu:http->112.81.71.171:globmsgsvc (ESTABLISHED)
httpd      8178    apache    4u  IPv6 2263806398       TCP *:http (LISTEN)
httpd      8178    apache    8u  IPv6 2263806403       TCP *:https (LISTEN)
httpd      8178    apache   32u  IPv6 3572014820       TCP limestone.uoregon.edu:http->218.15.22.132:csoft1 (ESTABLISHED)
httpd      8181    apache    4u  IPv6 2263806398       TCP *:http (LISTEN)
httpd      8181    apache    8u  IPv6 2263806403       TCP *:https (LISTEN)
httpd      8181    apache   32u  IPv6 3572011335       TCP limestone.uoregon.edu:http->112.81.71.171:2329 (CLOSE_WAIT)
httpd      8182    apache    4u  IPv6 2263806398       TCP *:http (LISTEN)
httpd      8182    apache    8u  IPv6 2263806403       TCP *:https (LISTEN)
httpd      8182    apache   32u  IPv6 3572014880       TCP limestone.uoregon.edu:http->112.81.71.171:ssm-cvs (ESTABLISHED)
httpd      8183    apache    4u  IPv6 2263806398       TCP *:http (LISTEN)
httpd      8183    apache    8u  IPv6 2263806403       TCP *:https (LISTEN)
httpd      8183    apache   32u  IPv6 3572013482       TCP limestone.uoregon.edu:http->112.81.71.171:netadmin (ESTABLISHED)
httpd      8184    apache    4u  IPv6 2263806398       TCP *:http (LISTEN)
httpd      8184    apache    8u  IPv6 2263806403       TCP *:https (LISTEN)
httpd      8184    apache   32u  IPv6 3572011872       TCP limestone.uoregon.edu:http->112.81.71.171:nexstorindltd (CLOSE_WAIT)
httpd      8185    apache    4u  IPv6 2263806398       TCP *:http (LISTEN)
httpd      8185    apache    8u  IPv6 2263806403       TCP *:https (LISTEN)
httpd      8185    apache   32u  IPv6 3572012783       TCP limestone.uoregon.edu:http->112.81.71.171:vrts-registry (ESTABLISHED)
httpd      8186    apache    4u  IPv6 2263806398       TCP *:http (LISTEN)
httpd      8186    apache    8u  IPv6 2263806403       TCP *:https (LISTEN)
httpd      8187    apache    4u  IPv6 2263806398       TCP *:http (LISTEN)
httpd      8187    apache    8u  IPv6 2263806403       TCP *:https (LISTEN)
httpd     30201    apache    4u  IPv6 2263806398       TCP *:http (LISTEN)
httpd     30201    apache    8u  IPv6 2263806403       TCP *:https (LISTEN)
httpd     30201    apache   32u  IPv6 3571983190       TCP limestone.uoregon.edu:http->180.98.61.58.broad.sz.gd.dynamic.163data.com.cn:abcsoftware (ESTABLISHED)
[root@limestone ~]# lsof -i | grep httpd
```

Follow the Process ID

We followed the trail of several of the web sessions that were in ESTABLISHED and CLOSE_WAIT states:

```
lsof –b –p PID | grep REG | grep –v mem
```

This gives lots of output, but only the last line is what we care about. This is the file the IP address of the process is, or was, accessing via http:

lsof –b –p PID | grep REG | grep –v mem

```
httpd   30201 apache  txt    REG        8,3     315280    1475010 /usr/sbin/httpd
httpd   30201 apache  DEL    REG        0,8            2147483647 /dev/zero
httpd   30201 apache   2w    REG        8,6  121347284    3859232 /var/log/httpd/error_log
httpd   30201 apache   6w    REG        8,6   68329371    3859368 /var/log/httpd/modsec_audit.log
httpd   30201 apache   7w    REG        8,6   21096567    3859383 /var/log/httpd/modsec_debug.log
httpd   30201 apache  12w    REG        8,6  121347637    3859232 /var/log/httpd/error_log
httpd   30201 apache  13w    REG        8,6        990    3859392 /var/log/httpd/ws.edu.isoc.org-error_log
httpd   30201 apache  14w    REG        8,6      10102    3859238 /var/log/httpd/gnuveau-error_log
httpd   30201 apache  15w    REG        8,6          0    3859386 /var/log/httpd/nsrc-error_log
httpd   30201 apache  16w    REG        8,6          0    3859254 /var/log/httpd/routeviews-error_log
httpd   30201 apache  17w    REG        8,6    1676923    3859387 /var/log/httpd/pythia-error_log
httpd   30201 apache  18w    REG        8,6     163355    3859199 /var/log/httpd/antc-error_log
httpd   30201 apache  19w    REG        8,6       5931    3859389 /var/log/httpd/ssl_error_log
httpd   30201 apache  20w    REG        8,6  194446083    3859143 /var/log/httpd/access_log
httpd   30201 apache  21w    REG        8,6       2406    3859390 /var/log/httpd/ssl_request_log
httpd   30201 apache  22w    REG        8,6          0    3859393 /var/log/httpd/ws.edu.isoc.orgc-access_log
httpd   30201 apache  23w    REG        8,6          0    3859391 /var/log/httpd/ws.edu.isoc.org-access_log
httpd   30201 apache  24w    REG        8,6       1862    3859234 /var/log/httpd/gnuveau-access_log
httpd   30201 apache  25w    REG        8,6          0    3859385 /var/log/httpd/nsrc-access_log
httpd   30201 apache  26w    REG        8,6          0    3859252 /var/log/httpd/routeviews-access_log
httpd   30201 apache  27w    REG        8,6    3828670    3859384 /var/log/httpd/pythia-access_log
httpd   30201 apache  28w    REG        8,6      98267    3859198 /var/log/httpd/antc-access_log
httpd   30201 apache  29w    REG        8,6       2082    3859388 /var/log/httpd/ssl_access_log
httpd   30201 apache  30w    REG        8,6       2406    3859390 /var/log/httpd/ssl_request_log
httpd   30201 apache  33r    REG       8,17 3580680192   42237953 /var/ftp/fedora/releases/9/Fedora/i386/iso/Fedora-9-i386-DVD.iso
[root@limestone ~]# lsof -b -p 30201 | grep REG | grep -v mem
```

The Attack: Discovery

Forensic investigation continued:

Running "`lsof`" from the previous slides lead us to the realization that a majority of the http processes were downloading the DVD ISO image:

/var/ftp/fedora/releases/9/Fedora/i386/iso/Fedora-9-i386-DVD.iso

The attack was to start the ISO download and then reset the connection.

This caused:

- Load average issues
- Blocked other client access to hosted web sites
- Generated minimal network traffic

The Attack: Mitigation

Next steps:

- Collect a list of IP addresses associated with DVD ISO download.
- Review the addresses to look for patterns.
- Determine (manually) what to block and then we used iptables firewall rules.

```
# lsof –i | grep httpd

httpd   22660 apache   25u  IPv6 9124821       TCP nsrc.org:www->59.252.183.12:23384
   (ESTABLISHED)
httpd   22660 apache   25u  IPv6 9124821       TCP nsrc.org:www->67.55.218.66:24566
   (CLOSE_WAIT)
```

The Attack: Mitigation

Involved IP Addresses:

- Upon inspection we were able to categorize a few address blocks.
- Blocks were all in .cn TLD.
- Not necessarily a directed attack from .cn as machines could be compromised from somewhere else.
- Once address blocks were determined iptables filter rules were applied.

The Attack: Firewall Rules

For involved IP Addresses:

```
iptables -I INPUT -s source_address -j DROP
```

Very simple. Worked as the attack was specific and not heavily distributed.

Specifically, from earlier slide this would be:

```
iptables -I INPUT –s 59.252.183.12* -j DROP
```

Or, to remove a range:

```
iptables -I INPUT –s 59.252.183.0/24 -j DROP
```

***ip address is example only**

firewall rules

Our attack, literally, looked like this…

More distributed or larger attacks require protection at your border routers.

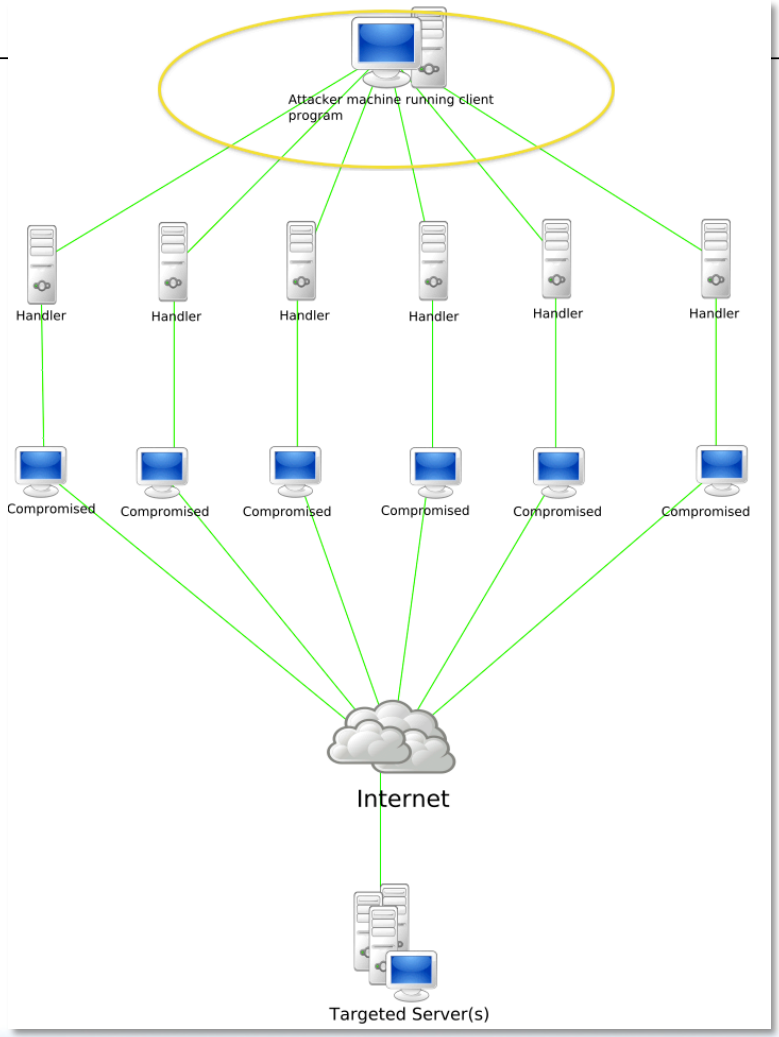


Image courtesy Wikipedia.

Blocking at the Router

- We can block incoming traffic from ranges of IPs to a local IP as Access Control List rules on your border router(s).
- What if your organization is *big*?
- UO is medium-sized… 30,000 devices, multi-homed.
- Many DDoS attacks don't go away in reasonable amounts of time…
- Origin address ranges are likely to change…

Blocking at the Router

Potentially a lot of work to update ACLs for each host under attack.

You can consider something like Botnet Traffic Filter systems.

Concept – auto=detect DDoS attack on a host or hosts. Block all traffic to these hosts…

There's a problem…

Botnet Traffic Filters

Issues blocking all traffic to the host:

- Some items to think about:
  - Short enough TTLs on DNS entries so that host IP addresses can be changed.
  - Coordination of this effort among multiple groups involved – particularly when you have multiple domains using a single IP (like we did).

Botnet Traffic Filters

Blocking all traffic to the host:

- TTL on A records for this hosts were long'ish…
- "BTFDs" imply you can change the IP of the host without too much difficulty.
- This host included multiple legacy sites, different domains, political bodies, etc…

Luckily for us simple host-based firewall rules solved the problem!

Issues

Dealing with DDoS attacks. A tricky issue:

- Lots of way to approach the problem
- Dependent on where you sit on the network
- Does your upstream provider have automated solutions?
- Can you talk to them if necessary? Often not.
- Automated solutions potentially expensive and can block legitimate traffic.

Summary

- Network and host monitoring to help detect and track-down DDoS attacks.
- Use of *NetFlow* and *top-talkers* on your routers to determine source(s) and types of DDoS attacks. Would not have worked in our case.
- Mitigation strategies dependent on:
  - Size of your organization
  - Available hardware
  - Your position as a customer
  - Severity of the attack
  - Ability to quickly change IPs