Anomaly-based Bot Server (and more!) Detection

Jim Binkley
jrb@cs.pdx.edu
Portland State University
Computer Science

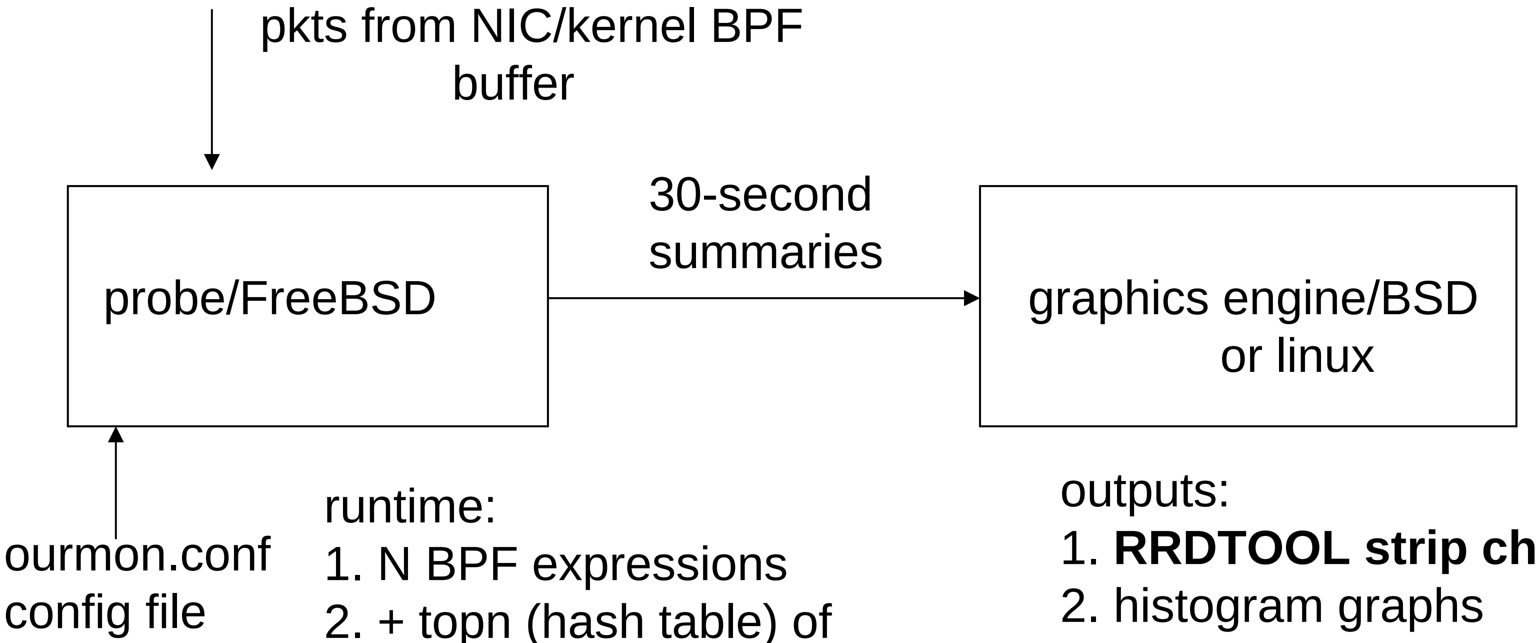
outline

- background
- experimental flow tuples
- botnet server mesh detection
- botnet client mesh detection
- conclusions

PSU's network

- 26k students/faculty/staff
- 350 Ethernet switches, 10k lit ethernet ports
- wide-spread wireless "pubnet", 802.11b/g
- typical daily traffic
  - 60k pps at peak periods
  - 200-300 mbits total, more to Internet, than from Inet
  - see next bullet item
- **we have dorms** (resnet) – resnet is typically infected
  - massive p2p bittorrent/gnutella traffic

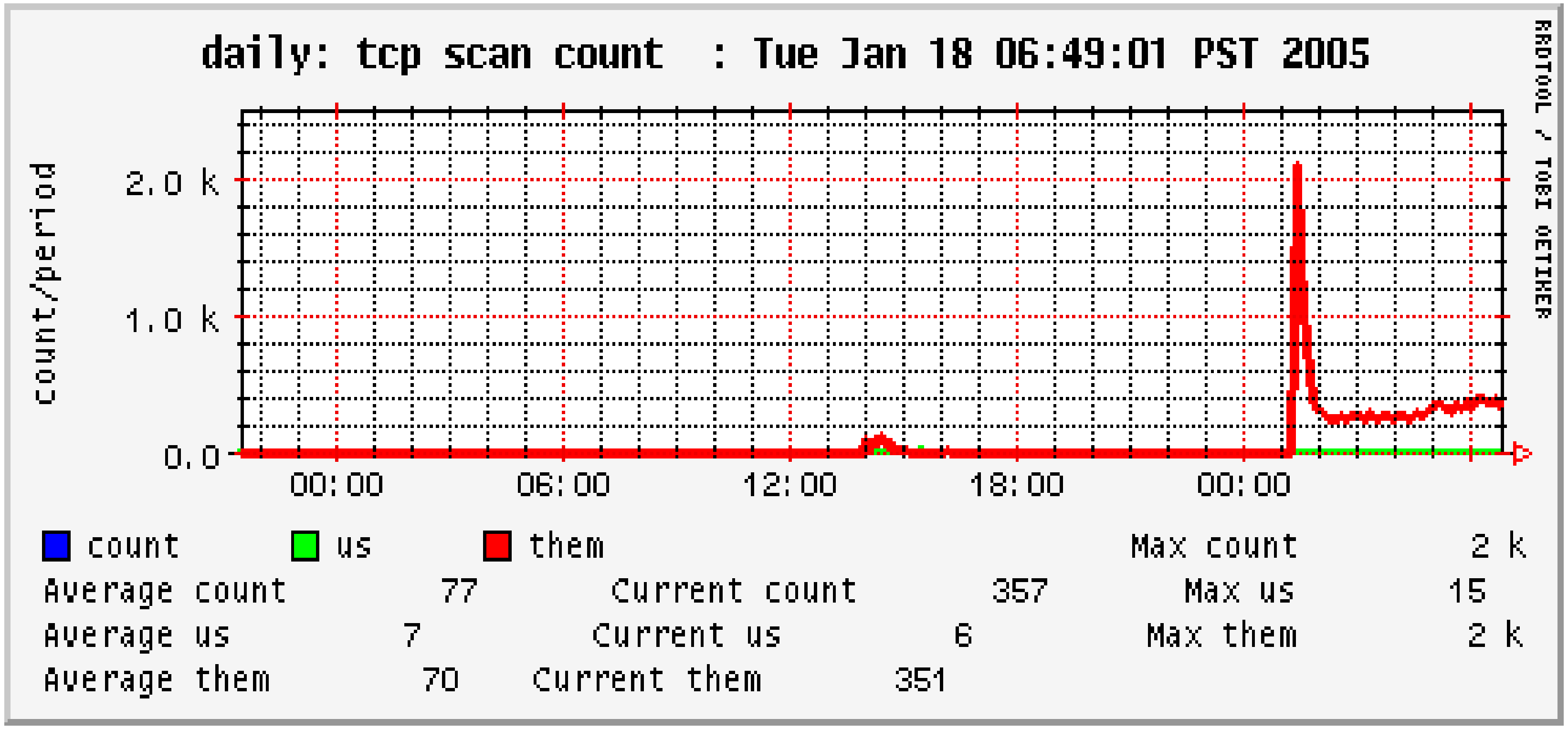
ourmon architectural breakdown



runtime:
1. N BPF expressions
2. + topn (hash table) of flows and other things (**tuples or lists**)
3. some hardwired C filters (scalars of interest)
4. PCRE tags for large-scale traffic analysis

outputs:
1. **RRDTOOL strip charts**
2. histogram graphs
3. **various ASCII reports, hourly summaries** or report period

scan count graph (worm count) in Jan. 2005

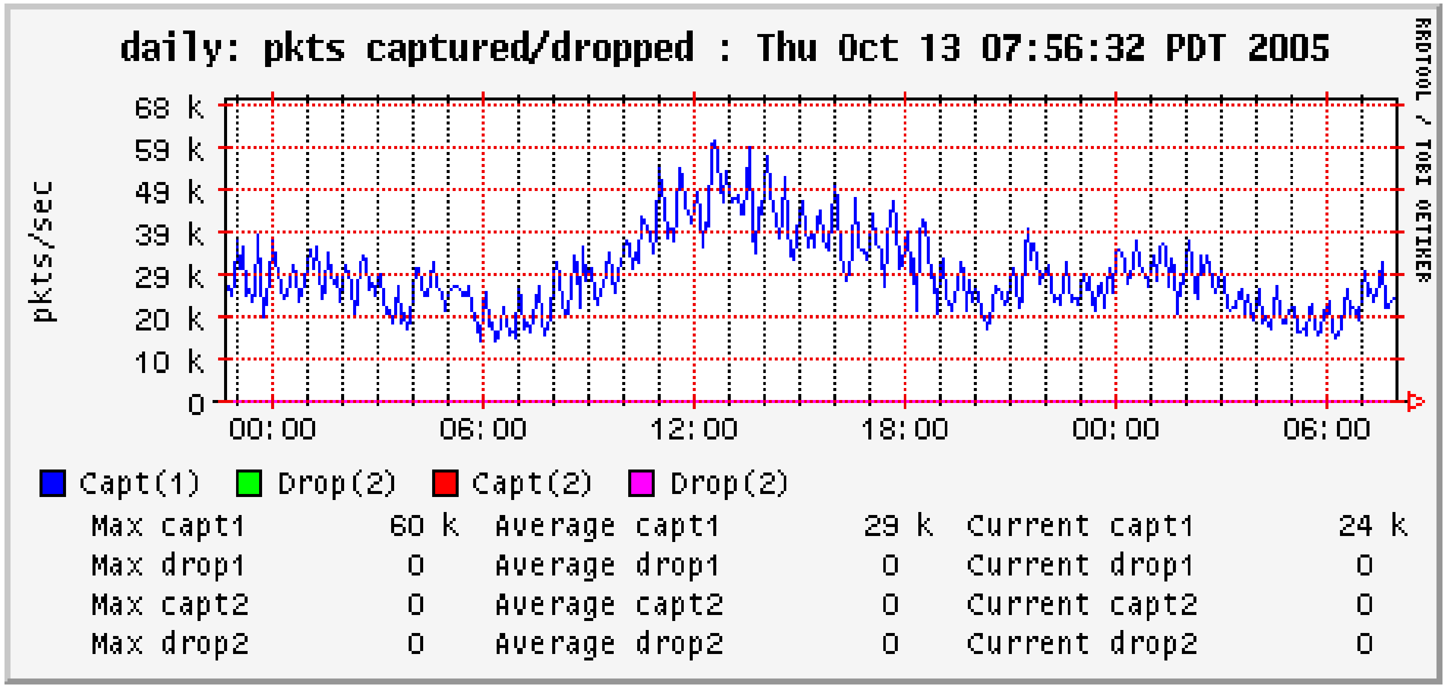


```
🟦 count        🟩 us        🟥 them                         Max count        2 k
Average count          77        Current count         357        Max us        15
Average us          7           Current us          6           Max them        2 k
Average them         70      Current them         351
```

2k external host attack (DDOS) on infected host running IRC

recent large ddos attack

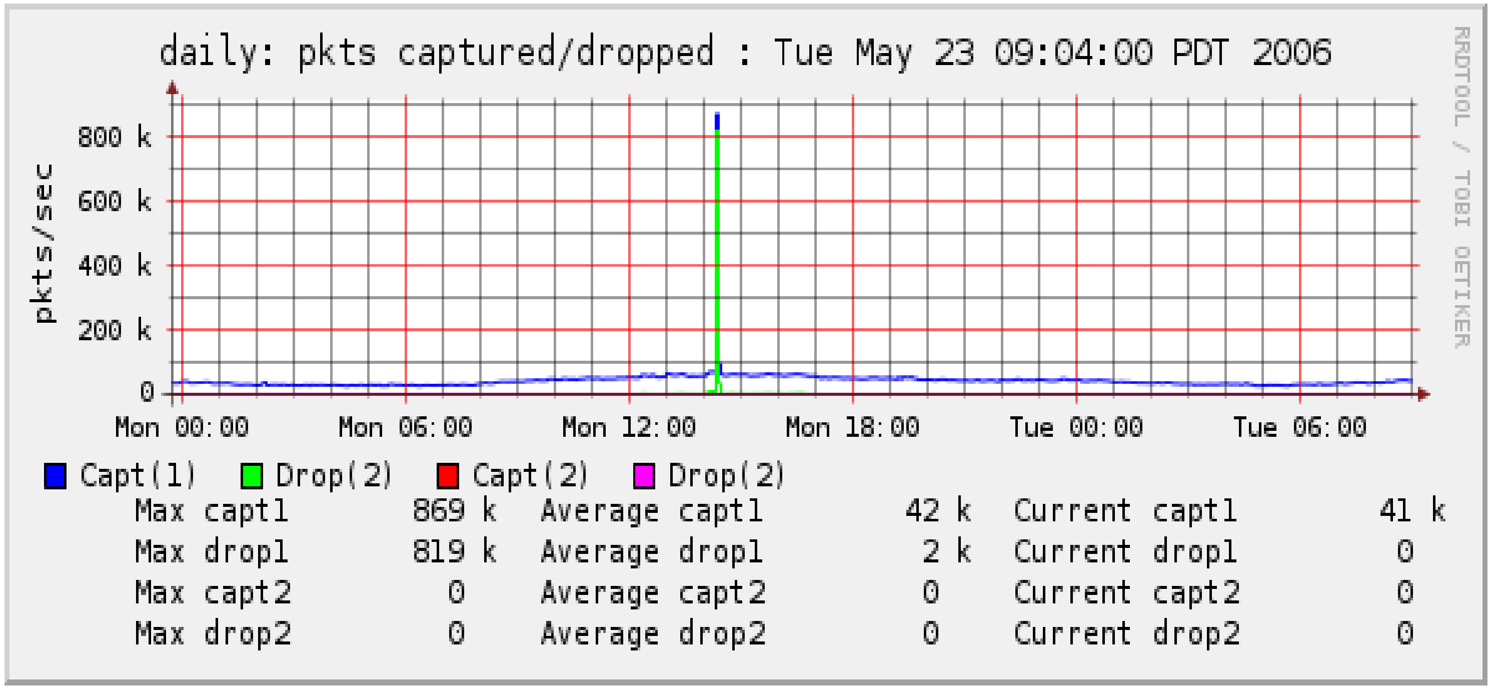
- fundamental pkts graph looks like this normally:



🟦 Capt(1) 🟩 Drop(2) 🟥 Capt(2) 🟪 Drop(2)

```
Max capt1      60 k   Average capt1    29 k   Current capt1    24 k
Max drop1       0     Average drop1     0     Current drop1     0
Max capt2       0     Average capt2     0     Current capt2     0
Max drop2       0     Average drop2     0     Current drop2     0
```

ouch ouch ouch



```
🟦 Capt(1)  🟩 Drop(2)  🟥 Capt(2)  🟪 Drop(2)
     Max capt1        869 k   Average capt1      42 k   Current capt1      41 k
     Max drop1        819 k   Average drop1       2 k   Current drop1       0
     Max capt2          0     Average capt2       0     Current capt2       0
     Max drop2          0     Average drop2       0     Current drop2       0
```

that's 869k pps – we have physical gE connection to Inet …

botnet situation

- over the last 2 years emerging picture
  - large percentage of our infections botnet related
- collateral damage common:
  - Jan 06/wireless subnet knocked off air due to DDOS attack
  - large and vicious DDOS attacks have occurred in OUS systems (previous pic)
- large amounts of TCP-based scanning aimed at ports 139/445
- decided to create IRC mesh detection module in ourmon to look for IRC-related malware
- goal: basic IRC statistics plus coupling of IRC to scanning module elsewhere in ourmon

infrastructure – 3 tuples in ourmon (irc new, tcp syn old)

- every thirty seconds extract 3 experimental flow tuples:
- **irc channel tuple:**
- **irc host tuple:**
- **tcp syn tuple**
  - coupled with scan detection attribute called
  - **tcp work weight**
- **IRC: we look at layer 7 IRC data, and use a snap size of 256 bytes.**

irc tuples and stats

- we extract these 4 IRC messages:
  - JOIN, PRIVMSG channel-name
  - PING, PONG for client/server connectivity
- we want: IP addresses in channel names
- also client/server information taken from directionality of IRC messages
- per host and channel stats counters
- also per network stats counters, total message kinds of all 4 kinds – graphed with RRDTOOL

irc measures

- irc channel tuples:
  channel name, message counts, list of IPs

- irc node tuples:

  ip address, message counts, weak tcp ww, client/server flag

- TCP work weight: (comes from syn tuple)

  per IP ww = (Syns sent + Fins sent + Resets returned)/total pkts

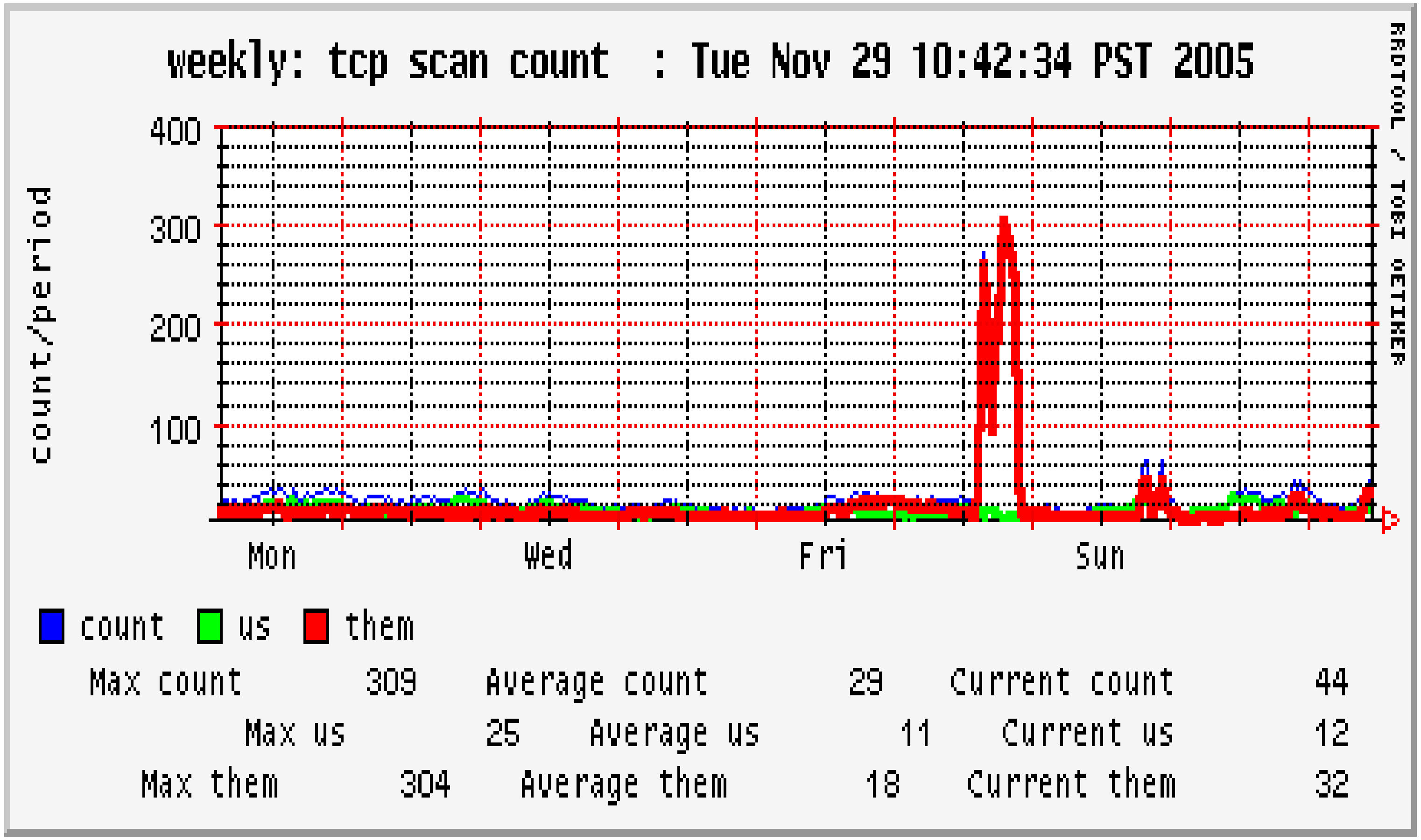
  view this as a **rude efficiency measurement**:

  100% means you are sending control packets.

TCP ww

- we have 2 years of experience with it
- < 50% is normal over some number of minutes
- not only attribute used for scan detection:
  - strength: typically use 1 syn/second at least
  - 2-wayness of data: typically look at this as additional attribute in 30-second scan determination
  - counts of L3 and L4 unique destinations
- strength and 2-wayness not used here:
  - IRC version of TCP work weight is weaker
- ww often affected by P2P lack of connectivity – especially with gnutella

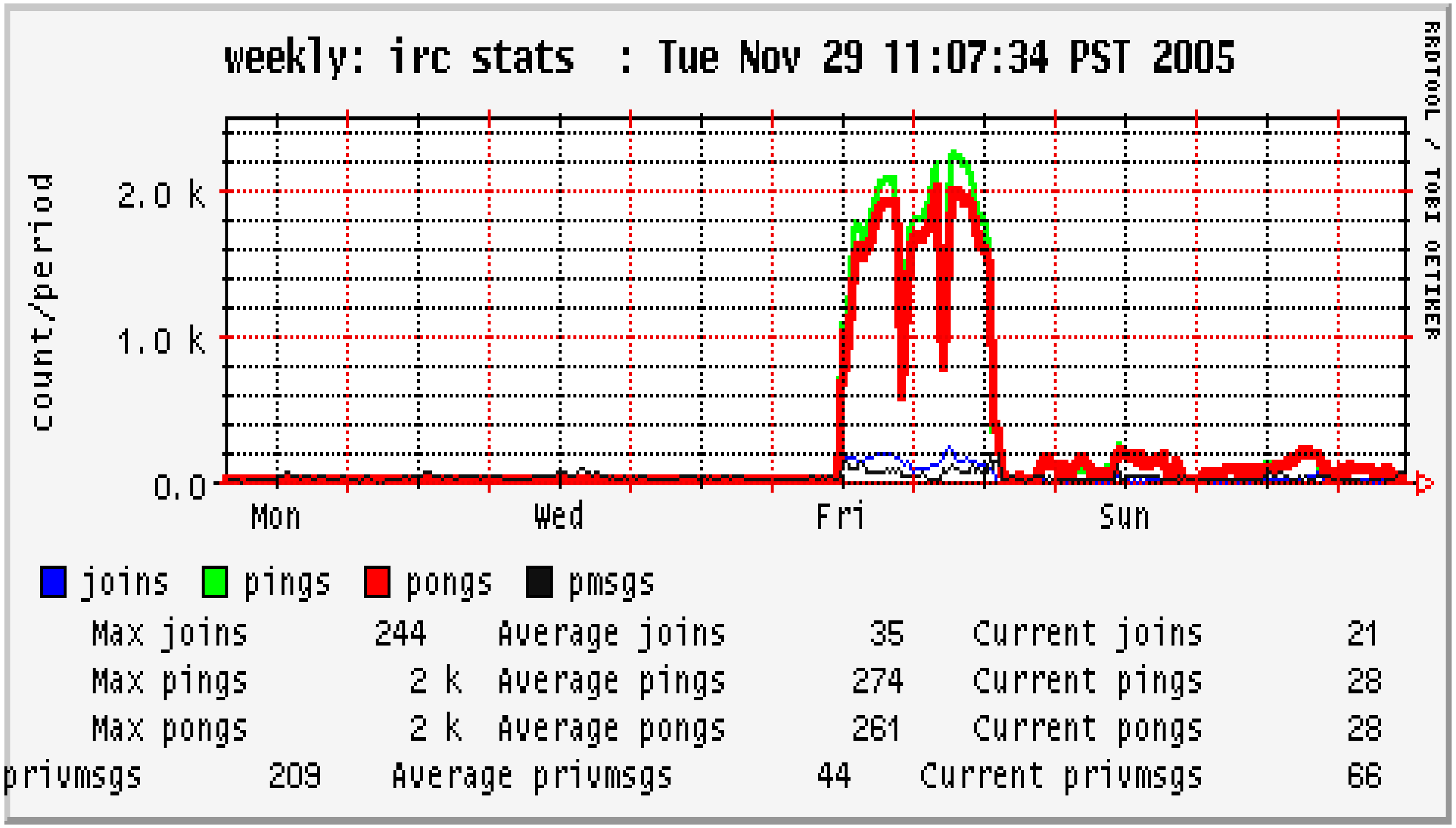
high abnormal scanner count – ironically was the real alert



```
Max count      309    Average count      29    Current count     44
       Max us         25    Average us        11    Current us     12
   Max them       304    Average them      18    Current them      32
```

some kinda distributed tcp syn scan right?, wait … let’s look at the IRC data

bot server detection: uh-oh, irc RRD has ping/pong way UP!



🟦 joins 🟩 pings 🟥 pongs ⬛ pmsgs

```
    Max joins        244    Average joins        35    Current joins       21
    Max pings        2 k    Average pings       274    Current pings       28
    Max pongs        2 k    Average pongs       261    Current pongs       28
privmsgs       209    Average privmsgs       44    Current privmsgs       66
```

hourly irc summary stats like so:

channel	msgs	ips	scanners	evil
f	157k	36k	1700	you tell me
x	81k	13k	712	
normalirc	5k	20	0	

- about 50k remote hosts with one campus botserver in several IRC channels
- a botclient “just changed” into a botserver Friday about 10 am, and acquired many friends fast

botserver conclusions

- from pure IRC POV:
- 1. ping/pong counts
  - entire IRC nets at PSU 40/period, not 2k/period
- 2. number of IPs in channel
  - biggest IRC channel 20 per day, not 10-50k
- 3. total IRC server messages
  - pings/pongs/privmsgs elevate the server
- interesting: total number of high TCP wws
  - external hosts that cannot connect to on-campus bot server (running on windows system)

TCP syn point of view - stats

- 1. L3D/L4D: interesting but statistically weak result
- on the 2 days of the bot server
  - bot server IP had highest count of average L3 destinations per sample period for any campus host
  - 1100 versus next highest which was a web server
  - web server and/or p2p clients typically < 1000
  - all you really say: will score high for that attribute
- 2. Syn count per period
  - highest on day 1, less so (still bad) on day 2
  - but it was scanning on day 1 as a normal bot client
- 3. pkt count for sent/recv. pkts HIGHEST on day 2
  - RECV pkts/SENT pkts 10/1

botnet client detection

- typical IRC data gives us small meshes on campus of
  - max: 20, min: 2 IRC channels
  - ports used may be 6667, but may vary
  - some automated bots exist (devoted to traditional IRC phenomenon like audio/video dissemination)
  - we have dorms …
- what seems to happen though is that the botnet client meshes SCAN with greater than one host during the day
- we therefore need an hourly/daily summarization

ubuntu channel - benign

ip	tmsg	ping	pong	privmsg	ww	server
net1.1	11598	1912	1910	6494	43	H
net1.2	7265	619	622	5086	0	H
net1.3	17218	4123	4100	7069	37	H
net2.1	28152	3913	3904	17113	0	S

F7 - an evil client mesh

ip	tmsg	ping	pong	privmsg	ww	server
net1.1	1205	377	376	428	42	H
net1.2	113	39	43	25	**96**	H
net1.3	144	60	61	21	**94**	H
net1.4	46	12	14	17	**90**	H
net1.5	701	343	345	11	**90**	H
net2.1	1300	587	593	101	16	S

evil channel sort – rank channels based on simple metric

- f7 ahead of ubuntu –
  - given 4/6 scanners compared to none
- max work weight during day kept is important idea
  - out of set of N, how many were scanners at any time?
- key idea: > 1 scanner in channel
  - plus of course other attributes in logs help
  - including ports
  - length and intensity of scanning

conclusions/future work

- p2p vs malware scanners distinction is a problem
  - we have an algorithm for p2p id based on pure attributes
  - it’s not perfect but it’s not bad
  - we use signatures too (but they aren’t perfect)
- given a set of attackers N (scanbots/spambots)
  - **and not using IRC as a mesh organizing principle how can we determine the mesh?**
  - DNS?
  - p2p meshes are a problem here too
    - except when they are the target

more information

- see http://www.cs.pdx.edu/~jrb
- **"Locality, Network Control, and Anomaly Detection,"** James R. Binkley, Portland State University, John McHugh, Carnegie Mellon University, and Carrie Gates, Dalhousie University, PSU Technical Report 04-04. January 2005. ps
- **"Ourmon and Network Monitoring Performance,"** James R. Binkley and Bart Massey, Computer Science, PSU, Proceedings of USENIX '05: FREENIX Track, April 2005. ps
- **"An Algorithm for Anomaly-based Botnet Detection,"** James R. Binkley and Suresh Singh, Computer Science, PSU, USENIX SRUTI: '06 2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet", July 7 2006. pdf
- **"Anomaly-based Botnet Server Detection,"** James R. Binkley, Computer Science, PSU, FLOCON CERT/SEI, Vancouver WA, October 2006. pdf
- http://ourmon.sourceforge.net