Needle in a Haystack

Improving Intrusion Detection Performance in University settings by removing “good traffic” to better focus on “bad traffic”

The Person Talking

- Paul Tatarsky paul@tatarsky.com
- Network Intrusion Detection (NIDS) operator/UNIX sysadmin in some form since 1990
- Watched a lot of packets go by.
- Most good, some bad, more than a few ugly

The Analogy

- That one unpublished exploit stream that smashes your named daemon stack and gets a shell on your DNS server is the needle. (chrooted of course)
- Benign or understood traffic is the hay.
- It is easier to find needles with less hay.
- It is not always easy or <u>safe</u> to define hay

The Environment

- University of California Santa Cruz, School of Engineering (where I am a IDS operator)
- Growing from “one uplink, one core switch” to multiple buildings, numerous core switches, several Gbit uplinks.
- No true firewall in place
- Snort (www.snort.org) is the IDS engine of choice
- Using spans to capture traffic
- IDS is the main source of protection for the department

The Problem

- Need to better focus on unknown traffic while removing known sources of heavy “good” streams
- Several legit high flow sources in department
- Older Snort IDS platform running on Intel hardware showing signs of age.
- A need to consider some “internal” monitoring with all the new subnets and wireless
- Too many packets, not enough IDS operator time
- Budgetary issues

Why Does this Matter to Larger Networks? (Net/Com/Gov)

- Consider us a micro-version of what happens to “exposed” machines (no firewall, all sorts of crazy stuff, student run machines, botnets love us, etc)
- If we can better focus on our relatively small data flows using lower end hardware, could map up to major performance gains at larger networks.
- Less time spent watching known flows means more time (processing power to apply/IDS operator time) to spot exploits, botnets, spam proxies, and DDOS attacks.

Flow size/content R&D started

- How big are our flows normally?
- Traffic analysis needed to understand "normal" flows
- Are our IDS signatures even looking for things related to most of these flows?
- Can we compare what we are looking for better with what we are capturing for IDS alerting?

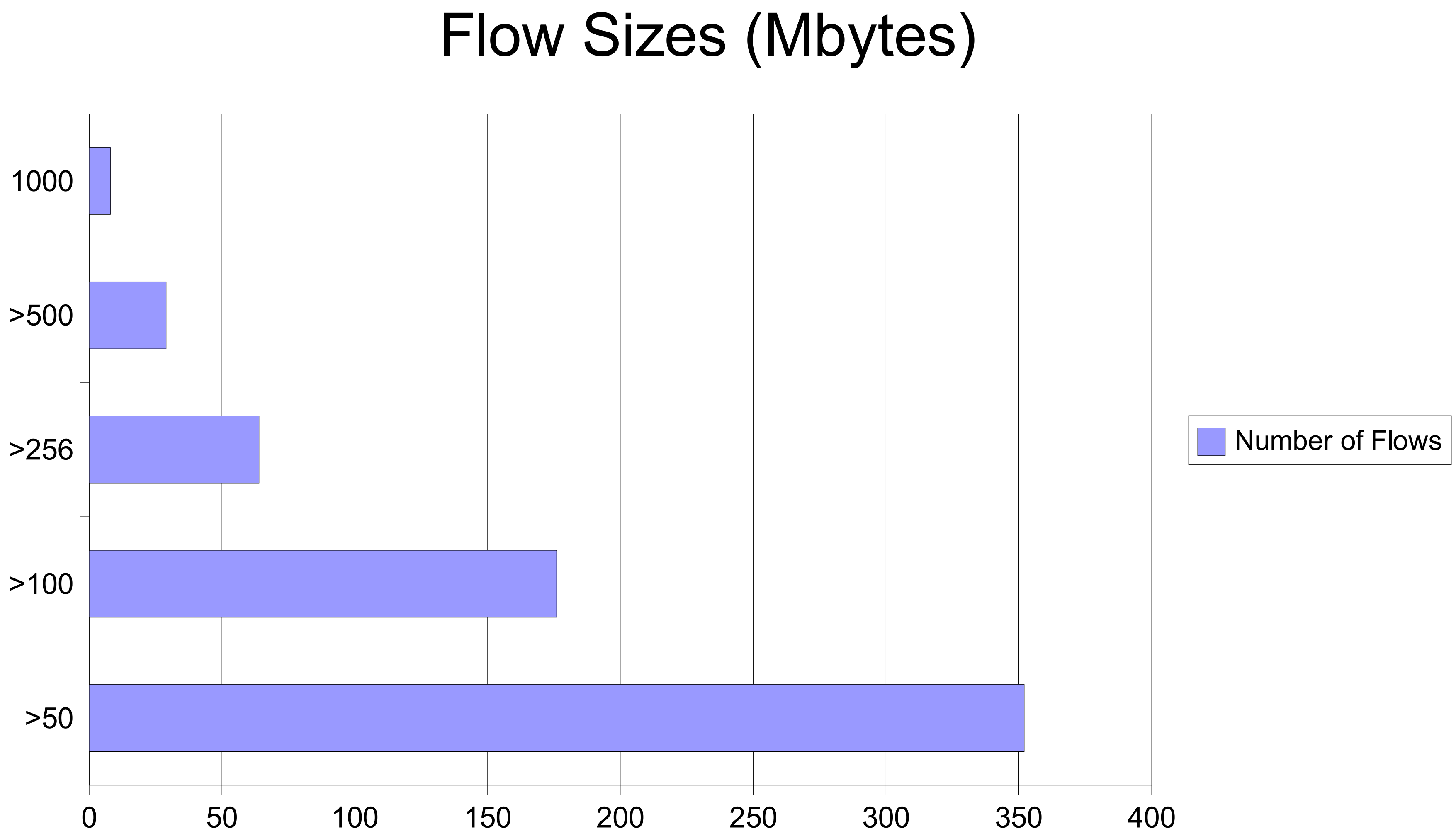
How to define “hay”

- Some of our known “good” streams:
  - Sql returned data queries to Genome site
  - HTTP/FTP/Rsync downloads known sources (ISO images)
  - P2P downloads?
  - File systems (NFS/SMB/GPFS/AFS)
  - Backups
  - Automatic updates for various platforms
  - VPN traffic (once established?)
  - Video conferencing/VOIP

Finding Flow Sizes Opensource

- Tcpflow (Jeremy Elson) to break them out to disk and take a look at them.
- Snort (Marty Roesch) in “session log” mode to do similar to above.
- Tcpdstat (Kenjiro Cho) to summarize flows by pcap capture. Modified slightly Dave Dittrich.
- Netflow statistics or other switch level stats
- Ntop gives a nice web based “traffic” summary
- Gives an idea of who is moving what around
- Are there repeated large items? Of course.

Sample Top Flow Sizes > 50MB



For a small sample period at UCSC SOE...

Top Flows as Percent of Data Capture

- Compared to these ~600 flows there were >300,000 smaller flows (which sort of skews the graph)
- However, the total data size of the 600 flows represented 33% of all traffic in bytes.
- The top twenty flows represented 7% of all traffic in sample.
- Sample period was small. Working on longer range statistics.

Hay definition with Libpcap "not" clauses

- Libpcap can define pretty elaborate "not" rules to exclude traffic by packet patterns
- When used with Snort can prevent engine from ever seeing the packets
- But assumes you know quite a bit about the protocol and "content" is hard to define
- No flow concept
- Get it wrong and away goes your IDS alerts

Snort Pass Rules and Flowbits

- More precise than libpcap filters since includes flow and packet content definitions.
- Flowbits construct can do some session level markers and is very powerful “hay” finder
- Still means Snort reads those packets into its engine just to discard them.
- Get pass rules wrong and there go your alerts.

Sample Snort Pass Rule

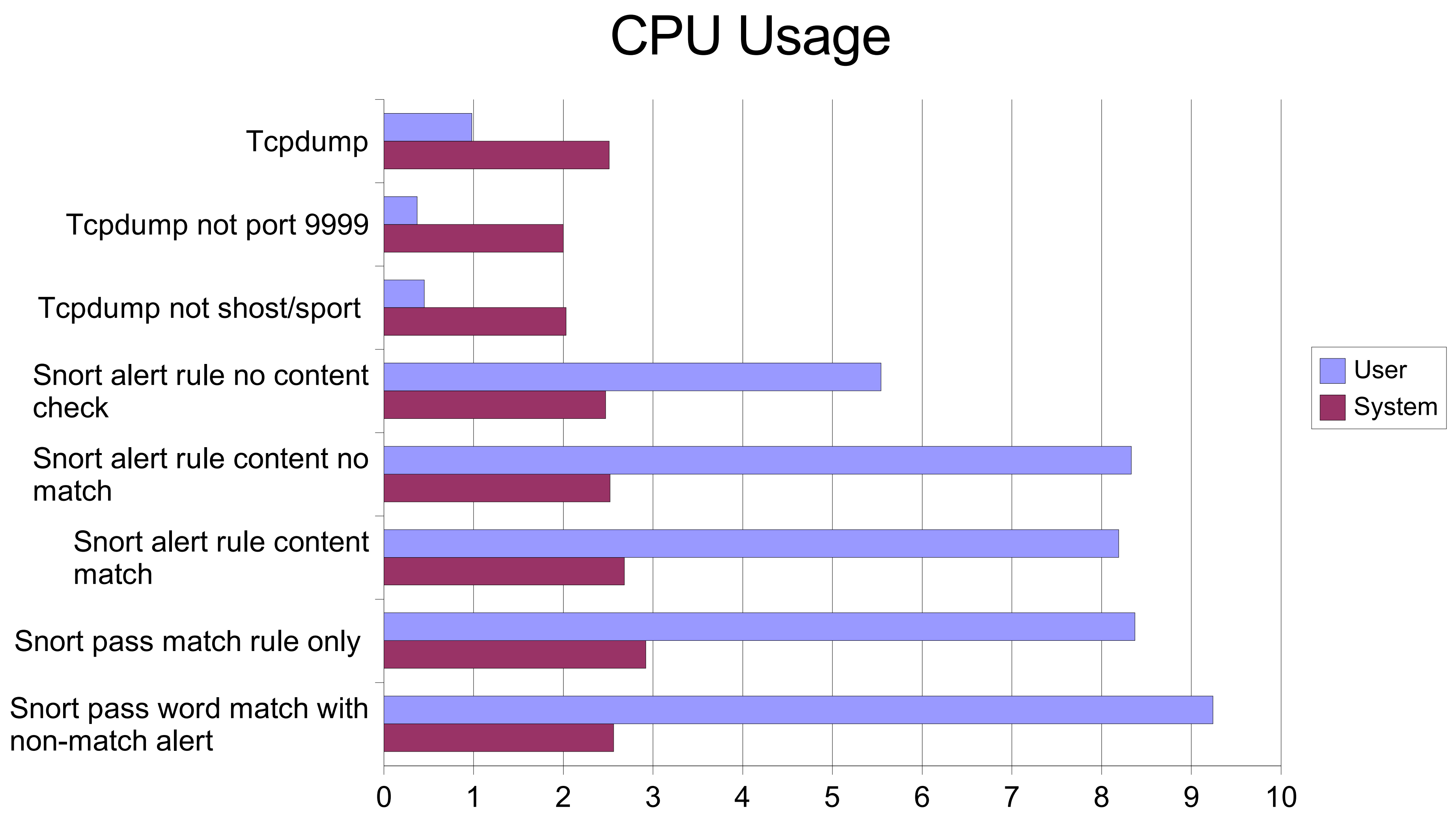
```
pass tcp any any -> any any (msg:"Likely ISO download";flow:from_server,established; content:"|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01|CD001|01 00|"; classtype:misc-activity; sid:1415086; rev:1; )
```

You can put a lot more “00 00” in there at the front.
Can I certify no attack will ever use this string? No.

Some simple tests show rough impact

- Send the Fedora 3 Disk 1 ISO image between two hosts using netcat on port 9999
- 638MB flow
- Capture packets in a few different ways
- Alter the method used to “ignore” parts of the flow

CPU Usage Baselines



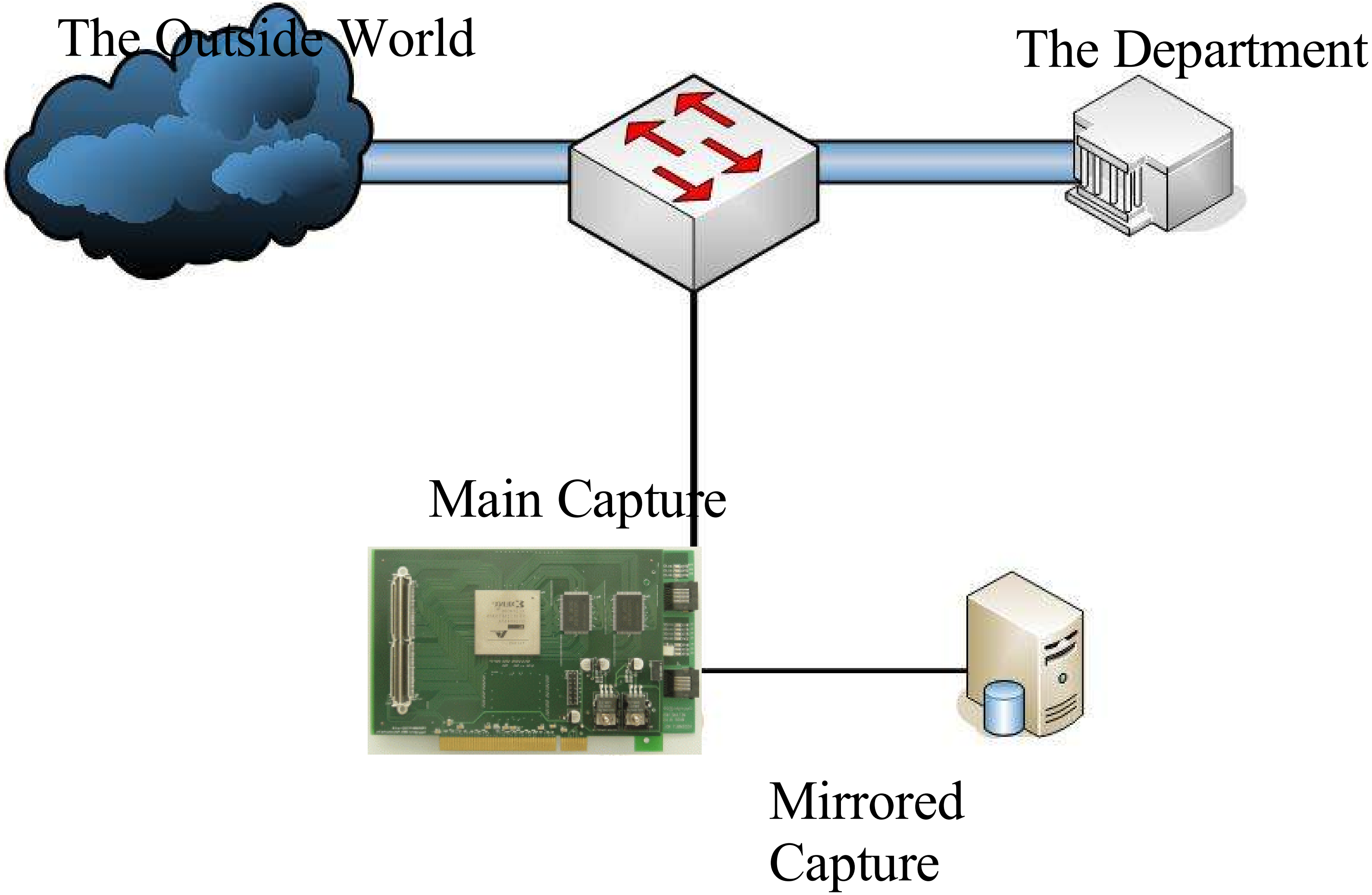
What if my capture card dropped the packets for me?

- Can it keep state on the streams?
- Can it detect some of the “good patterns” and then stop handing the stream to the interface?
- Can I still perhaps record the packets somewhere to make sure I didn't falsely drop a stream?
- Or at very least, can I get rid of some flows by using layer 3 defines?

Started looking around for such an animal

- Introduced to the Metanetworks MTP card
- Uses Snort format rules to define “capture” or “no capture” rules
- Keeps state and handles flows
- Has a “mirror” port that passes the capture without **the filtering** to another unit. Ala a mini-regeneration tap.
- Hands packets that are marked with capture rules “upstream” to a capture supporting UNIX interface

Picture of Setup



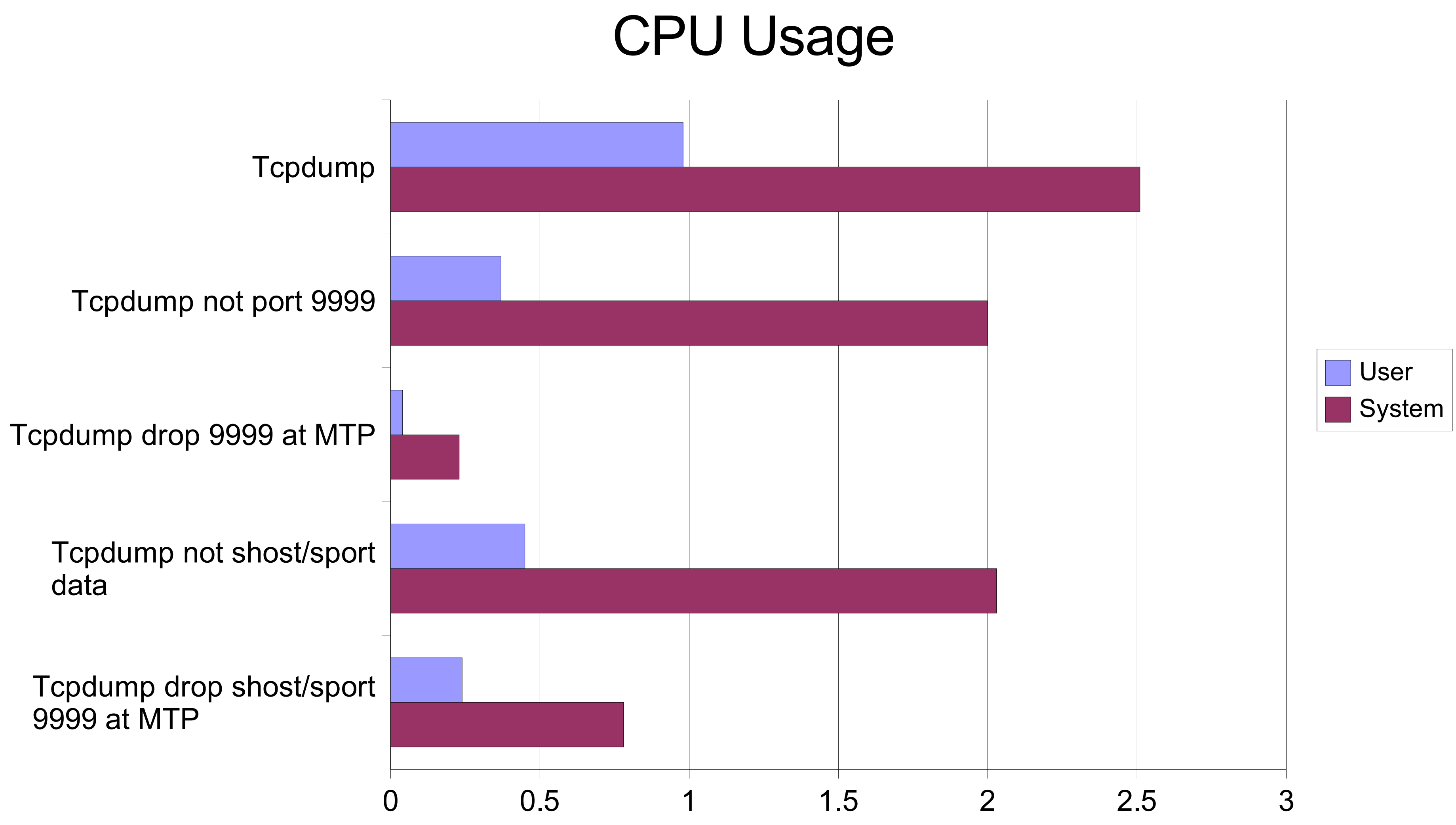
Started with translation of L3 “not” BPF rules into MTP card

not (host backup1 and port 13782)
and not host updates.redhat.com
...

Into snort format:

- pass tcp $UCNET any -> $BACKHOST 13782 (msg:"@backups";)
- pass tcp 66.187.224.40 any -> $UCNET any (msg:"@redhatupdates1";)
- pass tcp 209.132.176.40 any -> $UCNET any(msg:"@redhatupdates2";)

CPU/Flow size impact post MTP L3 filters



Continued Research into Content based rules

- Would like to move many content based “pass” rules down into hardware
- Would like some way to pass Snort flowbits state
- Generic ISO header pass rules into hardware
- Genome data pass rules with pcre content
- Passive FTP downloads. Trigger a pass based on “PASV/PORT” or “EPSV/EPRT” rule which reads ports to discard?
- Automatic update services (Windows Update, yum, cvsup, autoupdate, up2date)

What could it gain you?

- Ability to prune or focus on very specific targets in high volume traffic environments
- Perhaps focus on core gear for attacks against it. Perhaps remove known P2P flow types to focus more on attacks leaving your ranges.
- Less powerful PC platforms able to “keep up” due to offload of capture to card.

Slide O' URLs

- Libpcap
  - http://www.tcpdump.org/
- Ethereal
  - http://www.ethereal.com/
- Snort
  - http://www.snort.org/
- Tcpflow
  - ftp://ftp.circlemud.org/pub/jelson/tcpflow/
- Tcpdstat (modified)
  - http://staff.washington.edu/dittrich/talks/core02/tools/tools.html
- Ntop
  - http://www.ntop.org/
- MTP Card
  - http://www.metanetworks.net/