What is IPSS and how can change my life?

1. What is IPSS?

IPSS (IP Surveillance Suite) is a set of tools that let you know exactly how your network is used at IP level. These tools were designed focusing on high scalability and best performance taking advantage for multiple CPUs or multicore processors and multi-platform environments.

IPSS let you keep track of IP network usage and could build useful reports to understand who and how your bandwidth is wasted (protocols, requested IP addresses...).

Official Homepage:

http://gabriel.verdejo.alvarez.googlepages.com/ipss

2. Source code license

Original source code is available at IPSS Homepage and was initially written using C/C++ language by Ivan Balañà (ivanb@confluencia.net) for his Computer Science degree under Gabriel (gabriel.verdejo.alvarez@gmail.com) guidance.

This suite was released under the Creative Commons license:

http://creativecommons.org/licenses/by-nc/2.5/deed.en

3. IPSS architecture (quick overview)

IPSS was designed to achieve a high performance system that could support high speed links (Gigabit networks and above) and take advantage of different architecture environments (heterogeneous systems).

We choose a three layer model because after studying several well known programs (ntop, nprobe, ipaudit...) with different approaches and lots of papers and bibliography, we identify three different main issues.

Data gathering

Before any process can be done, data should be collected. Over high speed networks this process needs all CPU attention but usually doesn't mean all CPU process power.

We must inspect every packet that runs our network so a lot of interaction is needed every time. Our process cannot wait for an idle CPU (or core) because data keep running over the network and don't wait.

Example. A Gigabit Network can move 2^30 bits per second.

2^30 = 1073741824 bits = 2^27 bytes = 134217728 bytes

Typical MTU over Ethernet is 1500 bytes so we have on a busy network at least
134217728 / 1500 = 89478 packets!!!!!!

With several interfaces on the same machine we got more fun...

Data processing

After data has been collected, we must normalize, compact and compress
it.

We must normalize data to get the same aspect for every capture data and keep track of IP flows. For example, packets from a same communication have source and destination IP address changed.

We must compact information on a second stage because the capture process must be quick and efficient (light CPU usage) so data is given “almost raw” from the collector module

We must compress information because we don't stop our capture process so this means that our data will grow ever and ever. Therefore try to get smaller data files seems a clever idea.

All this processes are very CPU intensive (greedy) and more interfaces will need more process time. Our approach let you run this batch processes on another machine.

Data visualization

This is a very useful part for almost everybody because we know the truth.....you’re not a BOFH.

“*everybody love graphs. Only real BOFH uses vi and text reports*”

Data visualization should build useful reports about network usage using the processed data. Must answer ad-hoc queries for specific reports and let you know the real state of your network.

At this time only simple graphs using gnuplot are displayed, but we hope soon a full WWW interface will be developed.

4. IPSS components: Listener

The Listener component entrust data collection process of every network interface. This tool has been designed to get a very high performance using thread paradigm and the standard capture library PCAP.

This tool listen every configured interface using a different thread and keeps track of every packet and every communication that happens in a user defined period of time. At the end of this defined period it forks another listener process and flushes capture data to disk.

Main features are:

- TCP, UDP, ICMP protocols supported
- Unix daemon style behavior (start/stop)
- Multi-cpu / multi-core support
- Multiple interfaces support,
- Custom sample time
- Custom memory usage

5. IPSS components: Flow Adder

The Flow Adder tools arranges captured data from every interface and perform a heavy CPU process that compacts, normalizes and compresses all data obtained from listener. It creates data summaries that viewer will display.

This program can be executed as normal user and doesn't require privileged access to any computer resource.

Main features are:

- Unix daemon style behavior (start/stop)
- No privileged user right needed
- Custom compression program and ratio
- Custom process priority (nice level)
- Custom sample time
- Custom top N selection
- Can work on a "non-listener" machine

6. IPSS components: Viewer

Viewer tool reads compressed data and summaries obtained from Flow Adder and calls standard GnuPlot program to create the graphic report. At this time reports are text file based but a great and wonderful web interface will be created soon.

Main features are:

- No privileged user right needed
- Custom date / time report selection
- Custom protocol report selection
- GnuPlot compliant
- No web interface (at this time)

7. Nice try...but how can IPSS change my life?

We provide a flexible and scalable three layer model that let you keep an eye on your network traffic. We also provide the full source code (for free!) of our implementation of this model to let you modify and adapt it this for your needs. Please check CC license.

This code has been developed using ANSI C/C++ and has been compiled and tested under x86 platforms (Linux, BSD) and Sparc (Solaris 7, 8).

So....to try it out and tell me about it!

Have fun.