Measuring the Performance of NetSniff: Testbed Design

Julie-Anne Bussiere,[*] Jason But
Centre for Advanced Internet Architectures. Technical Report 050623A
Swinburne University of Technology
Melbourne, Australia
julie-anne.bussiere@laposte.net, jbut@swin.edu.au

***Abstract*- NetSniff is an IP traffic analysis tool currently utilised in low traffic scenarios. We wish to explore the possibility of expanding its use to higher traffic situations and networks. This technical report explains our motivation for doing so, and the design of the tesbed we will construct to facilitate our determination of the processing performance of NetSniff.**

***Keywords*- NetSniff, performance, testbed**

I. Introduction

NetSniff is a multi-network-layered real-time traffic capture and analysis tool developed as part of the ICE^3 project being run out of the Center for Advanced Internet Architectures (CAIA). The NetSniff tool is currently deployed in low-bandwidth and low-traffic scenarios. To gather more useful information, we would like to deploy it within networks where the number of aggregate users is higher. This paper explains why we want to use NetSniff on a larger scale, and also discusses the design of the testbed we will use to measure the processing performance of NetSniff.

II. Purpose

The NetSniff tool facilitates capture and anlysis of IP network traffic. Its design and implementation concepts are described here [1]. The NetSniff tool is currently deployed in low-traffic scenarios, and is performing well. However, the levels of captured traffic is not sufficient to properly model the widespread use of the Internet and network applications. To do this, NetSniff must be deployed in larger networks with more users generating more traffic. An example would be deployment in large corporate LANs such as the Swinburne LAN or within ISP networks.

In this scenario, Netsniff – or indeed any network monitoring device – is required to correctly capture, parse and analyse both an increased amount of network traffic and an increased number of multiple, concurrent network flows. Unfortunately we are not able to simply deploy NetSniff in a real network scenario due to privacy concerns, as such we must build a testbed to generate traffic specifically for the purpose of testing the performance of NetSniff.

III. Methodology

Measuring NetSniff performance with high traffic implies generating IP traffic. The following subsections explain 2 different testbeds we considered.

A. Using virtual hosts on one real machine using RULE

The Remote Unix Lab Environment (RULE [2]) provides multiple virtual networked Unix hosts using one real workstation, see figure 1. By creating several virtual hosts on one machine, we can launch multiple different and concurrent IP/TCP applications. A single RULE enabled computer can easily manage up to 50 virtual hosts, each with a different IP address. This testbed allows us to create real traffic with multiple concurrent flows of different applications.

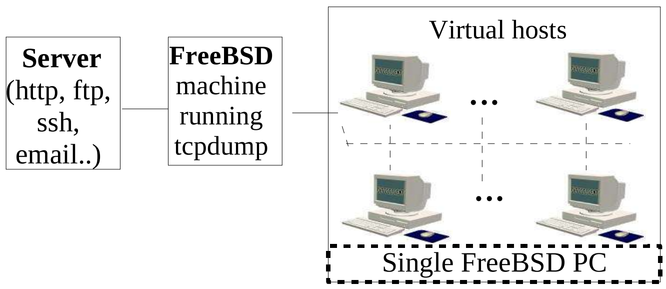

***Fig.1*Virtual hosts testbed**

The server will be a separate FreeBSD machine running HTTP, HTTPS, FTP, email (SMTP) and SSH services. Each virtual host will be configured to access these applications on the server to download / upload data using commands like wget. A third computer configured as an Ethernet bridge with FreeBSD will be located between the server and the RULE host. This machine will be running tcpdump [5], recording all traffic into one tcpdump file.

The recorded tcpdump file will either be directly processed by NetSniff, or be replayed across a network using tcpreplay. When replaying traffic, we will need two FreeBSD machines, one running tcpreplay to generate the traffic and one running NetSniff to capture and parse the generated traffic. The traffic replay speed can be modified, allowing analysis of NetSniff performance in higher bandwidth conditions. The configuration and hardware of the FreeBSD machine

[*] Julie-Anne Bussiere performed this work while a visiting research assistant at CAIA in 2005

running NetSniff will be changed to determine ideal traffic capture configuration.

B. Using Internet Traffic Generator Tool

Internet Traffic Generator [3] is a traffic generator tool constituting an alternative to the previous testbed to simplify setup and experiment duration. This tool is able to generate traffic at high data rate and in a concurrent fashion including TCP (Ipv4 and Ipv6) as well as UDP traffic, ICMP, DNS, Telnet, VoIP (G.711, G.723, G.729, Voice Activity Detection, Compressed RTP). However, this tool cannot generate traffic of upper layer applications (HTTP, E-mail). However, the critical processing point in NetSniff is TCP streams and, furthermore the applications generated by ITG belong to common used applications which are interesting too. The testbed is presented in figure 2.

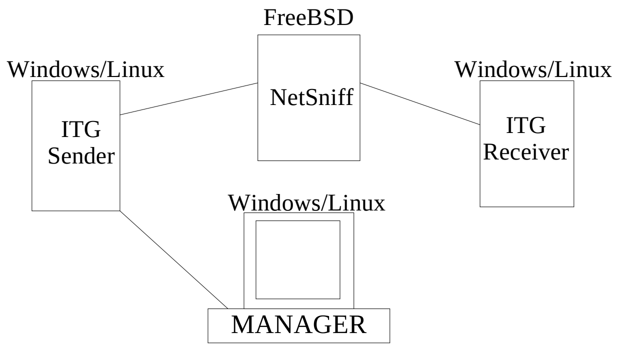

Fig.2 **Internet Traffic Generator testbed**

ITG Sender generates a set of flows; it operates as a multi-threaded application. One of the threads implements the TSP protocol and drives the generation process, while the others generate the traffic flows. The manager allows remote control of traffic generation. NetSniff will run and process directly the generated traffic, but we can also run tcpdump instead and record the traffic in a tcpdump file for later replaying.

C. Hardware and configuration of the capturing box

The "capturing box" represents the machine itself - including RAM, network card and the clock speed - in addition to NetSniff software. The performance testing will take in account NetSniff software's implementation as well as the machine's hardware the most convenient to run NetSniff in high traffic circumstances. In this way we will run NetSniff on different machines with the same traffic to compare performance depending on Hardware, and we will run NetSniff using the same machine with different traffic models to point out software implementation performance.

IV. Rule Based Testbed

In the first instance we will measure the performance of NetSniff using a RULE based testbed. The traffic captured in this scenario is generated by real networked applications running on a real network. As such, this traffic is more likely to conform to real-world traffic than that from the traffic generator. It may be possible to evaluate NetSniff performance using the traffic generator at a later stage.

Once the testbed is built, we need to configure the virtual jail hosts and the server to allow data exchange for each application. This is described in the following sections. Once configured, a shell script **main.sh** is written, this script will launch a series of commands in a loop. These commands will continuously use different applications to retrieve (or send) data from(to) the server. It is possible to execute the shell script at the same time from each jail host using the command "at" as jail root:

```
> at -f main.sh -t time
```

where time is in the format MMddhhmm.ss .

A. Enabling SSH

On the server, the SSH daemon is enabled by removing the comment character # on the corresponding line from the **/etc/inetd.conf** file. No specific configuration needs to be done on the jail hosts.

To allow the automatical usage of the ssh command in a shell script, we need to avoid the password prompt. To be connected directly from the jail host (as root or as user, depending on future needs) run the command:

```
> ssh-keygen -t rsa
```

This will first ask you for the directory to store the key file, type enter to keep the default path. For the passphrase, choose an empty one by typing enter twice. Now the key is generated, copy the public key in the host you want to ssh (the server), knowing the username and the password. From the jail host:

```
>cat ~/.ssh/id_rsa.pub | ssh
[server] "cat >>
~/.ssh/authorized_keys"
```

In the shell script, the command line to use ssh is:

```
>ssh -n [serverhost] [command]
```

The scp command can also be used directly :

```
>scp [filetocopy]
     [serverhost:directorytocopy]
```

B. Enabling FTP

On the server, the FTP deamon is enabled by removing the comment character # on the corresponding line from the **/etc/inetd.conf** file. No specific configuration needs to be done on the jail hosts.

To exchange files with ftp, we use a shell script defined as follows.

```
#File autoftp.sh
#!/bin/sh
HOST='serverhost'
USER='username'
PWD='serverpwd'
ftp -n $HOST<<END_SCRIPT
  quote USER $USER
  quote PASS $PWD
  get myfile1
  put myfile2
```

```
    quit
    END_SCRIPT
#end autoftp.sh
```

This shell script is then executed from the main shell script with the command line `sh autoftp.sh`.

C. Enabling HTTP and HTTPS

On the server, we install Apache [6] with mod_ssl [4] and OpenSSL [7]. On the jail hosts, we install the wget package which also requires the packages gettext and libiconv.

The server is launched as root using:

```
#/usr/loacal/apache/bin/apachectl startssl
```

To download an html page from the server we can issue the command:

```
wget http://serverhost/mypage.html
```

To use ssl encryption, we need to specify an option which indicates not to check the certificate (the certificate created is not signed and then appears insecure).

```
wget https://serverhost/mypage.html –no-check-certificate
```

D. Enabling SMTP

On the server, the sendmail daemon is enabled by adding the following line to **/etc/rc.conf**

```
sendmail_enable="YES'
```

Mail can be sent using the telnet application to connect to the SMTP server. An easy way to automatically use telnet in a script is with Perl. Install the Perl5 package on each jail host. The Net::telnet class script is required, then copy the file telnet.pm in .../perl5/site_perl/Net, creating the directory /Net.

The perl script to send mail is:

```
#File automail.pl
#!/usr/bin/perl -w
use Net::telnet;
$hostname= "serverhost";
$telnet = new Net::Telnet ( Timeout=>10);
$telnet->open( Host=> $hostname, Port=> 25);
$telnet->waitfor('/$/i');
$telnet->print('helo rulex.caia.swin.edu.au');
$telnet->waitfor('/$/i');
$telnet->print('mail from: user@rulex.caia.swin.edu.au');
$telnet->waitfor('/$/i');
$telnet->print('rcpt to: user@serverhost');
$telnet->waitfor('/$/i');
$telnet->print('data');
$telnet->waitfor('/$/i');
$telnet->print('subject: whatever);
$telnet->waitfor('/$/i');
$telnet->print('my message');
$telnet->waitfor('/$/i');
$telnet->print('.');
$telnet->waitfor('/EOF $/i');
$telnet->cmd('quit');
#end of automail.pl
```

This Perl script is then executed from the main shell script with the command line `perl automail.pl`.

E. Other requirements

The server and the virtual machines must belong to the same subnetwork (192.108.0.x). To change the network configuration on the server, modifications are needed in the /etc/resolv.conf and /etc/rc.conf files. In resolv.conf, comment the existing lines and add :

```
nameserver    192.168.0.y
```

y is a number between 1 and 255, different from all the numbers already used by the machines in our network (jails and server). In rc.conf, comment out the pre-existing defaultrouter and ifconfig lines and add:

```
defaultrouter 192.168.0.y
ifconfig_em0="inet 192.168.0.s netmask 255.255.255.0"
```

y must have the same value than previously, and s is the number associated with the indivudual machine being configured.

The files to be transferred during the experiment (ftp or ssh) are created with a C program. It creates a file of the specified size filled with random characters. To avoid the use of unnecessary space on the disk, the transferred files will be deleted immediately after the download. Some html pages of different sizes are be created.

In /etc/hosts file, the different jail host names have to be registered adding the following line for each:

```
192.168.0.x rulex.caia.swin.edu.au user
```

V. Main Sheel Script

The repetitive actions performed by the main shell script on each jail host are:

1. **http**: download a html page of 1.68 KB
2. **http:** download a html page of 3.2 KB
3. **ftp:** download 2 files with ftp, respectively of 10 KB and 200 KB, upload a 1MB file
4. **https:** download a html page of 1.68 KB with ssl
5. **smtp**: send a mail of 1KB
6. **ssh/scp:** ssh twice, executing a short command, upload a file of 2 MB with scp, and ssh to remove this file on the server.
7. **http:** download a html page of 3.22 KB

8. **http:** download a html page of 68.5 KB
9. **ftp:** download 2 files with ftp, respectively of 2 MB and 2 KB, upload a 500 KB file
10. **https:** download a html page of 3.22 KB with ssl
11. **smtp:** send a mail of 2.3 KB
12. **ssh/scp:** ssh twice, executing a short command and download a file of 1.4 MB with scp

VI. Conclusion and Future Work

For the needs of NetSniff performance measurements, a testbed to generate IP traffic has been designed. This paper explains the configuration and software installation requirements for automating in a shell script the generation of ssh, ftp, http, https and smtp traffic over a network of virtual jail hosts. Future research will involve building the aformentioned testbed and using it to generate conclusions on the processing performance of NetSniff under both raw processing and live capture situation.

References

[1] U. Keller, J. But, "Netsniff - Design and Implementation Concepts," (pdf) CAIA Technical Report 050204A, February 2005

[2] G. Armitage, "Maximising Student Exposure to Networking using FreeBSD virtual hosts", (pdf), July 2003, http://portal.acm.org/citation.cfm?doid=956993.957010

[3] Internet Traffic Generator, http://www.grid.unina.it/software/ITG/, , accessed June 2005

[4] Apache SSL Module, http://www.modssl.org, , accessed June 2005

[5] http://www.tcpdump.org

[6] The Apache Web Server, http://www.apache.org, accessed June 2005

[7] OpenSSL Project, http://www.openssl.org, accessed June 2005

[8] Perl Scripting Language, http://www.perl.com, accessed June 2005