Virtualization Efficacy for Network Intrusion Detection Systems in High Speed Environment

Monis Akhlaq, Faeiz Alserhani, Irfan U. Awan, John Mellor, Andrea J. Cullen, and *Pravin Mirchandani

Informatics Research Institute, University of Bradford, Bradford, BD7 1DP, United Kingdom and
*Syphan Technologies (www.syphan.com)
{m.akhlaq2,f.m.f.alserhani,i.u.awan,j.e.mellor}@bradford.ac.uk, a.j.cullen@braford.ac.uk, *pmirchandani@syphan.com

Abstract. The virtualization concept was developed a few decades back to facilitate the sharing of expensive and robust main-frame hardware among different applications. In the current scenario, virtualization has gone through a conceptual transformation from cost effectiveness to resource sharing. The research community has found virtualization to be reliable, multipurpose and adaptable. This has enabled a single system to dynamically map its resources among multiple instances of operating systems running numerous applications. The concept has been adopted on platforms dealing with network performance, application analysis, system design, network security and storage issues. This research work has focussed on analysing the efficacy of the virtualization concept for Network Intrusion Detection Systems (NIDS) in the high-speed environment. We have selected an open source NIDS, Snort for evaluation. Snort has been evaluated on virtual systems built on Windows XP SP2, Linux 2.6 and Free BSD 7.1 platforms. The test-bench is considered to be extremely sophisticated, ensuring current day network requirements. The evaluation has been targeted at the packet-handling capacity of operating systems/ applications (Snort) under different traffic conditions and on similar hardware platforms. Our results have identified a strong performance limitation of NIDS running on virtual platforms. It can be easily ascertained that virtual platforms are not ideal for NIDS in high-speed environments. Finally, the analysis has also identified the factors responsible for the unsatisfactory performance of IDS (Snort) on a virtual platform.

Keywords: Network intrusion detection systems, operating systems, performance evaluation, Snort, virtualization.

1 Introduction

Virtualization is a framework for abstracting the resources of a PC into multiple execution platforms by creating multiple machines on a single computer. Each machine operates on the allocated hardware and can afford multiple instances of applications [1]. This concept has been successfully incepted within the industry/ business community. The mechanics of system virtualization to implement network security tools has been considered as an appropriate choice for academia dealing with information security [2, 3].

Network Intrusion Detection Systems (NIDS) are considered to be an important security tool in any organization [4]. The effectiveness of any Intrusion Detection Systems (IDS) depends on its ability to handle various traffic-loads at maximum speed with minimal packet loss. The consequences of packet loss in any NIDS can be ascertained by considering the damages caused by single packet attacks-Witty worms [5]. Witty was the first widespread internet worm to attack a security product. It uses buffer overflow technique [6] to exploit a flaw in the firewall software [7]. From any perspective, packet capturing and handling are considered as major factors in deciding the efficiency of an IDS. Packet handling assumes an additional importance in relation to the use of virtual machines, where resources are shared between multiple platforms.

Snort [8], an open source NIDS has been selected because of its popularity and status as a de facto IDS standard. Snort relies on the packet capturing libraries (libpcap and winpcap) [9]. The libraries act as packet capturing interfaces to the underlying operating system [9]. Snort is a signature based IDS; it uses rules to detect hostile attempts to intrude onto a the network. The rules are a set of requirements created to generate an alert and have a particular syntax. On successful detection of a hostile intrusion, the detection engine sends an alert to a log file/storage resource [10].

Our research work focuses on evaluating the virtualization concept for NIDS in high-speed networks. Virtualization has found its acceptance in NIDS; however no comprehensive evaluation has done before. Mostly, the concept has been debated on perceived logics of resource conservation in virtualization without any experimental proof. We have analyzed the concept by utilizing open source NIDS- Snort under high-speed multi-Gbps environment. Our concept is unique in the sense that we have incorporated three different OS platforms and the evaluation criteria is based on packet handling capacity of Snort. The results are based on the response of Snort for different packet sizes and bandwidths representing actual network traffic.

The test-bench is built on VMware Server [11]; the VMware performs CPU virtualization by employing direct execution and binary translation. The software executes CPU instructions with minimum overheads. A guest operating system (OS) can run on the server without modification, thus ensuring a high degree of system compatibility.

Throughout the testing phase, VMware was installed on Windows Server 2008 [12]. Three virtual platforms were built on different OS platforms (Windows XP SP2 [13], Linux 2.6 [14] and Free BSD 7.1 [15]). These platforms were provided in all cases with equal hardware resources and subjected to similar traffic conditions. The responses of each platform were recorded and analyzed to reach the conclusions presented here.

This paper has been organized into sections; section 2 gives an overview of the virtualization concept employed, section 3 analyzes the factors affecting packet handling and section 4 describes the test-bench. Section 5 shows the results and finally in section 6 we present our analysis.

2 Virtualization

The concept has been developed to address the issues related to reliability, security, cost and complexity of the network/systems. It has successfully been used for the processing of legacy applications, ensuring load balancing requirements, resource sharing and tasking among virtual machines by using autonomic computing techniques. The technique has also shown merits in the situation where an application failure on one machine does not affect the other. In addition, ease of isolation allows multiple OS platforms to be built on one machine running variable instances of applications. This has made the concept quite fascinating for the research community [16].

Virtualization in information technology is the technique of creating a number of machines (guest machines) running on top of a physical machine (host machine). The guest machines use the hardware of the host machine and can manage multiple instances of applications running guest operating systems. Currently there are two kinds of server virtualization [17]:

2.1 Hosted Virtualization

This is also called as operating system (OS) based virtualization because the virtual layer on the host OS runs multiple instances of the guest OS. The guest OS operates as an independent platform and manages multiple applications as shown in Figure 1 [17].

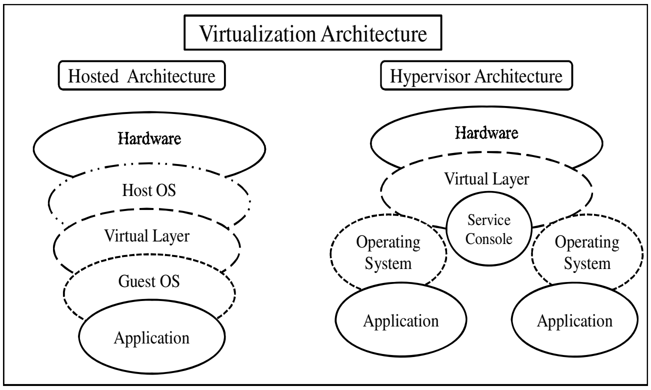


Fig. 1. Virtualization Architecture

2.2 Hypervisor Virtualization

Hypervisor; software manages the physical resources of the server by allocating the hardware resources of the machine to multiple virtual platforms. This concept uses two interfaces, a virtual layer and a service console. The virtual layer enables OS installation and the service console manages user applications as shown in Figure 1 [17].

3 Packet Handling

Packet capturing and handling is vital requirement for a NIDS. A good packet capturing response by the NIDS towards variant traffic reduces the probability of a system becoming compromised. Factors that affect the packet capturing performance of a NIDS in a Gigabit Ethernet environment include host configuration parameters (hardware and software) and application-specific parameters (NIDS).

3.1 System Hardware

3.1.1 Processor Speed and Architecture

The clock speed and the architecture of the processor directly impact the performance of a NIDS. Multi core and multithreaded architectures provide few performance gains in comparison with single processor as, many applications such as Snort Version 2.6 developed and optimized for legacy architectures fail to maximize these architectural advances [9].

3.1.2 PCI Bus and Disk Input/ Output (I/O) Operations

PCI bus architecture directly influences the operational efficiency of memory and storage devices. Current system architectures identify two bottlenecks in packet capturing accuracy – bus bandwidth and disk throughput. When writing to a disk, packet capture libraries pass data to the system bus twice, once from the network interface card to memory, and secondly from memory to disk. Thus the actual bandwidth available to the PCI bus is half that of the traffic bandwidth [18].

Table 1. PCI Buses and Disk System

Bandwidth – Current PCI Buses

PCI Bus Parameters	Bandwidth
Standard 32 bit, 66 MHz	264 Mbytes/s
Standard 64 bit, 133 MHz	1,066 Mbytes/s
PCIexpress x 1	250 Mbytes/s
PCIexpress x 16	4000 Mbytes/s
PCIexpress x 32	8000 Mbytes/s

Bandwidth – Disk System

Type	Transfer Rate
ATA/ATAPI Interface	133 Mbytes/s
SATA	**300 Mbytes/s**
SCSI (320)	320 Mbytes/s
Fiberchannel	512 Mbytes/s
SCSI (SAS)	348 Mbytes/s
SAS 2	768 Mbytes/s
SAS 3	1563 Mbytes/s

Data intensive applications and the huge amounts of data required to be stored in enterprise networks demand highly efficient I/O operations to avoid performance bottlenecks. The invention of multi-core processors has enhanced the capability of systems to support multiple virtual machines, yet system performance in relation to disk I/O operations remains limited [19]. Table 1 shows the types of I/O devices and their data transfer rates. The performance of these I/O devices in high-speed Gbps networks needs evaluation.

3.1.3 Memory

Alongside CPU capabilities, memory is another major factor affecting packet-handling ability. The memory controller is a part of the CPU sub-system and establishes information flow between memory and the CPU. The CPU sends a signal to the memory within a system clock cycle, which varies depending upon the speed of memory and bus speed. System performance is also affected by the speed at which the data can be transferred between system memory and the CPU. System bus and memory are the critical components when it comes to the efficiency of system in relation to packet handling [9].

When the system does not have enough main memory resources, virtual memory is used [9]. Virtual memory is allocated on the system hard disk. The performance of virtual memory in comparison to main memory is considered as a weak link in system performance [17].

3.1.4 Network Interface Card (NIC)

This component is directly connected to the physical media to interface to the data flow. The chosen NIC should correspond to the system capacity and network requirements. NICs are designed to offload the packet-capturing overhead by removing the system's CPU from this process. These devices are also capable of filtering, load balancing and regular expression functionality at hardware level [9].

The capacity of traditional NICs can also be enhanced by using specialized techniques such as [1]NAPI [20] and [2]PF_RING [21].

3.2 Operating System

The performance of an application also depends on the host operating system. In order to maximise performance gains, a suitable operating systems that supports the processor architectures is recommended. Ideally system performance can be increased by using an operating system with multithreaded kernel architecture on a multiprocessor machine.

3.3 NIDS and Packet Capturing Libraries

Snort has been selected as the target IDS platform for this research; this section describes the packet processing capabilities of Snort with an emphasis on its packet capture libraries [9].

- It calls libpcap by using the pcap_dispatch function to process waiting packets.
- For each available packet, libpcap calls the PcapProcessPacket function to perform packet processing. This function resets several per-packet counters, collects packet statistics and calls ProcessPacket.
- The ProcessPacket function handles all the actions involved in decoding the packet and printing in verbose mode. It can also call the logging functions if running in logger mode or call the pre-processors if running in IDS mode.

[1] NAPI ("New Application Programming Interface") is a modification to the device driver packet processing framework, which is designed to improve the performance of high-speed networking.

[2] Type of socket (PF_RING) optimization for packet capturing techniques based on a circular buffer.

A limitation of Snort is that it can only process a single packet at a time. Its pcap and inline API (Inline application programming interface) provide buffering resources; however any prolonged processing causes the buffers to fill and packets start to be dropped as a result. Additionally, the [3]Frag3 and [4]Stream4 pre-processors deal with multiple packets received by the system. Any missing packet forces Snort to wait till the session time out thus causing memory and CPU drain. This also creates a feedback loop where lost packet cause additional packet loss causing overall system degradation.

Another area that has an impact on packet drop rates is the technique of retrieving packets from the network interface using the pcap API (Packet Capturing application Programming interface). A few improvements in these techniques have been suggested in [21], [22] and [23], however increases in traffic-loads and speed considerably reduce the positive impact of these modifications.

4 Performance Test

4.1 Test-Bench

The test-bench is distributed into three parts and configured around a ProCurve series 2900 switch [24] as shown in Figure 2.

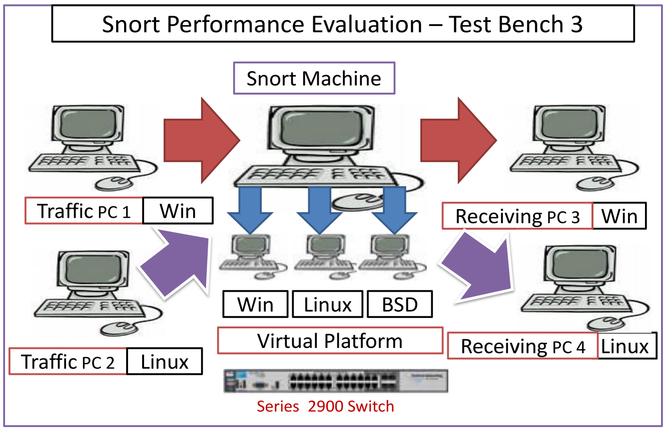


Fig. 2. Test-bench

The basic idea of the evaluation process revolves around packet capturing and evaluation by virtual platforms and Snort. We have selected two machines for traffic generation: Linux 2.6 and Windows XP SP2 platforms respectively. Similarly, the traffic reception machines were also deployed to fulfil network requirements. Details of the traffic generation tools are shown in Table 2.

[3] IP defragmentation preprocessor in Snort.
[4] Verify & reassembles TCP sessions in Snort.

The virtual platform running Snort has been configured on a dual quad-core processor. The machine hardware details are listed in Table 2. The system is built on the Windows 2008 Server platform and three separate virtual platforms have been created-Windows XP SP2, Linux 2.6 & Free BSD 7.1. Snort is running simultaneously on all the virtual machines and similar traffic-loads and type are injected onto all platforms.

4.2 Evaluation Methodology

In order to ascertain the capability of Snort to handle high-speed network traffic on virtual platforms we proceeded as follows:

- Parallel Snort sessions were run on all virtual machines.
- The machines were injected with similar traffic-load characteristics (UDP and TCP Traffic) for 10 minutes.
- Different packet sizes (128, 256, 512, 1024 and 1460 bytes) were generated and Snort's performance at the following traffic-load was evaluated: 100 Mbps, 250 Mbps, 500 Mbps, 750 Mbps, 1.0 Gbps and 2.0 Gbps respectively.
- Snort's performance characteristics were evaluated - packets received, packets analysed, packets dropped and CPU usage at various packet sizes and band widths levels.
- Packets received were compared at both the host OS and the virtual platforms running the Snort applications.
- During the course of the tests, no changes were made in OS implementation specifically Linux using NAPI [16] and [5]MMP and Free BSD using [6]BPF [27].

Table 2. Network Description

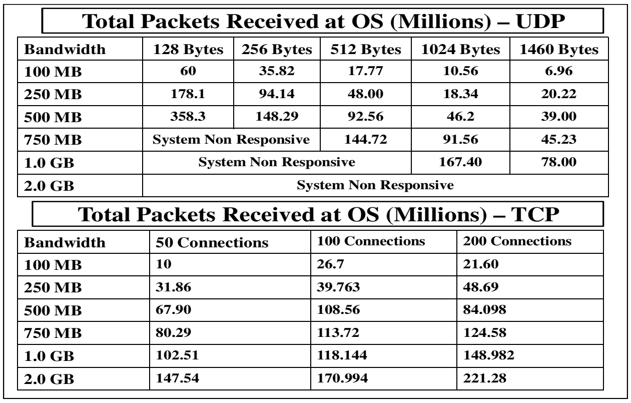
Machine Type	Description	Tools Used
Network traffic/ back ground traffic generator- PC 1 (Win)	Dell Precision T3400, Intel D Quad-Core, Q6600 2.40 GHz. 2 GB RAM, PCIe, 1Gb/s RJ45, Network Card (Broadcom NetXtremo gigabit Ethernet).	LAN Traffic Generator [25]
Network traffic/ back ground traffic generator - PC 2 (Linux)	Dell Precision T3400, Intel Quad-Core, Q6600 2.40 GHz. 2 GB RAM, PCIe, 1Gb/s RJ45, Network Card (Broadcom NetXtremo gigabit Ethernet).	D-ITG Traffic Generator [26]
IDS Machine (Snort)	Dell Precision T5400, Intel (Xenon) Dual Quad-Core, 2.0 GHz. 4.0 GB RAM, L2 Cache 12 MB, PCIe, Network Card , 10 GB Chelsio. Hard Disk: 1000 GB, Buffer 32 MB, SATA 300.	VM Ware Server, Virtual Machines (Win, Linux, Free BSD)
Network traffic/ back ground traffic Receiver- PC 3 (Win)	Dell Precision T3400, Intel Quad-Core, Q6600 2.40 GHz. 2 GB RAM, PCIe, 1Gb/s RJ45, Network Card , 10 GB Chelsio.	LAN Traffic Generator
Network traffic/ back ground traffic Receiver- PC 4 (Linux)	Dell Precision T3400, Intel Quad-Core, Q6600 2.40 GHz. 2 GB RAM, PCIe, 1Gb/s RJ45, Network Card (Broadcom NetXtremo gigabit Ethernet).	D-ITG Traffic Generator
Switch	ProCurve Series 2900, 10Gb/s switch with 24x1 Gb/s ports and 2x10 Gb/s 3CR17762-91-UK ports.	

5 Results

The results are distributed over UDP and TCP traffic types respectively. It was observed that the total packets transmitted from the traffic-generating PCs was

[5] Modified device drivers packet handling procedures.
[6] Berkley Packet filter.

Table 3. Packets Received at Host Operating Systems



Total Packets Received at OS (Millions) – UDP

Bandwidth	128 Bytes	256 Bytes	512 Bytes	1024 Bytes	1460 Bytes
100 MB	60	35.82	17.77	10.56	6.96
250 MB	178.1	94.14	48.00	18.34	20.22
500 MB	358.3	148.29	92.56	46.2	39.00
750 MB	System Non Responsive		144.72	91.56	45.23
1.0 GB	System Non Responsive			167.40	78.00
2.0 GB	System Non Responsive				

Total Packets Received at OS (Millions) – TCP

Bandwidth	50 Connections	100 Connections	200 Connections
100 MB	10	26.7	21.60
250 MB	31.86	39.763	48.69
500 MB	67.90	108.56	84.098
750 MB	80.29	113.72	124.58
1.0 GB	102.51	118.144	148.982
2.0 GB	147.54	170.994	221.28

equivalent to the number of packets received at the host machine/ OS running virtual platforms as shown in Table 3; however this is not the case once the system found [7]non responsive.

5.1 UDP Traffic

The results below are described in relation to packet size, bandwidth (i.e. traffic-load), and the virtual OS platform running the Snort application:

5.1.1 Snort Response for Packet Sizes – 128 & 256 Bytes

- Linux shows quite good performance for these packet-sizes upto 250 Mbps traffic-load; its performance declined at higher bandwidth levels as shown in Figure 2. The system found non responsive at the traffic-loads of 750 Mbps and above.

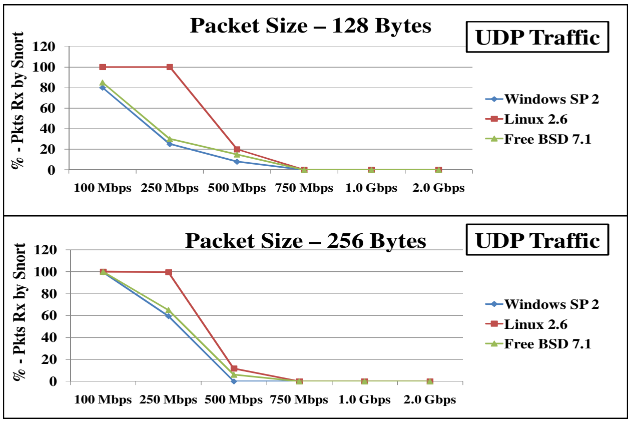


Fig. 3. Snort Packets Received (%) - UDP Traffic (128 Bytes & 256 Bytes)

[7] In non responsive situation we consider 100% packet loss.

- Windows shows good performance for 128 Bytes packet sizes at 100 Mbps loading only. Its performance is compromised at higher loading levels as shown in Figure 3. The system also found non responsive at traffic-loads of 750 Mbps and above.
- Free BSD performs slightly better than Windows as shown in Figure 3. The system also found non responsive at traffic-loads of 750 Mbps and above.

5.1.2 Snort Response for Packet Size – 512 & 1024 Bytes

- Linux shows quite good performance for traffic-load upto 500 Mbps for all packet sizes as shown in Figure 4. The Linux however system found non responsive at traffic-loads of 1.0 Gbps and above for 512 Bytes packet sizes and at 2.0 Gbps for packet sizes of 1024 Bytes.
- Windows also performed satisfactorily at traffic-loads of 250 Mbps and 500 Mbps for packet sizes of 512 Bytes and 1024 Bytes respectively as shown in Figure 4. The system found non responsive at traffic-loads of 1.0 Gbps and above for packet size of 512 Bytes and 2.0 Gbps for packet sizes of 1024 Bytes.

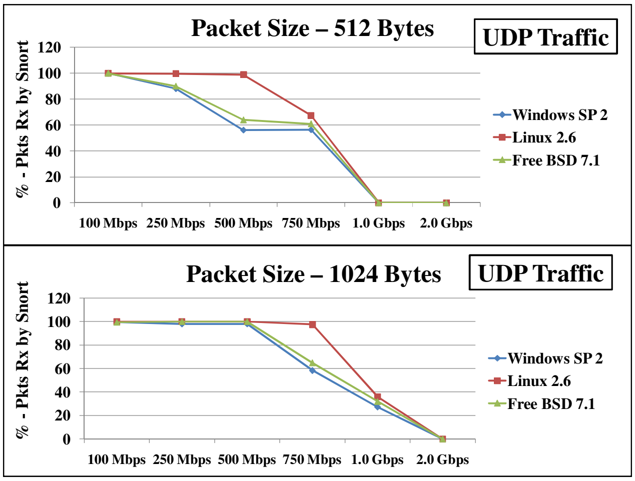


Fig. 4. Snort Packets Received (%) - UDP Traffic (512 Bytes & 1024 Bytes)

- Free BSD responds a bit better than Windows as shown in Figure 4. The system found non responsive at traffic-loads greater than 1.0 Gbps for packet sizes of 512 Bytes and 2.0 Gbps for packet sizes of 1024 Bytes.

5.1.3 Snort Response for Packet Size – 1460 Bytes

- Linux shows significantly better performance for packet sizes of 1460 Bytes of packet for traffic-loads upto 1.0 Gbps however, the system found non responsive at 2.0 Gbps loading as shown in Figure 5.
- Windows also shows good performance upto750 Mbps loading. The system found non responsive at 2.0 Gbps traffic-load as shown in Figure 5.

- Free BSD responds a bit better than Windows as shown in Figure 5. The system found non responsive at 2.0 GB traffic-load as shown in Figure 5.

5.2 TCP Traffic

We have included the results of 512 Bytes packet sizes in this section due to paucity of space. The results have been accumulated on the basis of successful connections (50, 100 and 200 respectively). Packets received at the host platform/ OS are shown in Table 3.

5.2.1 Snort Response for 50 Connections – 512 Byte

- Linux exhibits quite good performance upto 750 Mbps loading however, its performance declined at higher traffic-loads as shown in Figure 5.
- Windows was acceptable upto 250 Mbps loading and its performance reduced for higher traffic-loads as shown in Figure 5.
- Free BSD performed a bit better than Windows as shown in Figure 5.

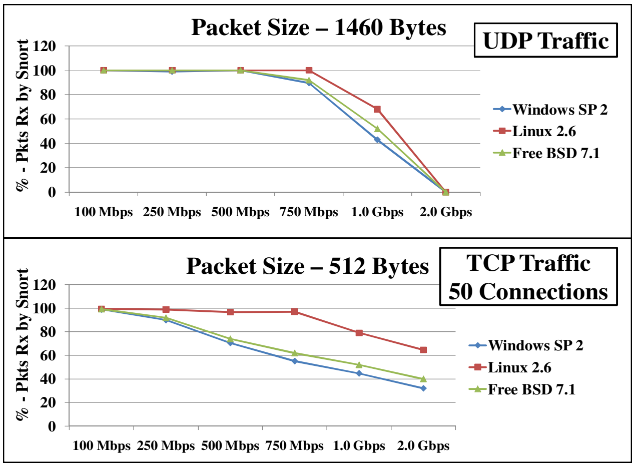


Fig. 5. Snort Packets Rx (%) - UDP (1460 Bytes) & TCP (50 Connections)

5.2.2 Snort Response for 100/ 200 Connections – 512 Bytes

- Linux exhibits quite good performance upto 250 Mbps loading with minimum packet loss, however, its response linearly declined for higher traffic-loads as shown in Figure 6.
- Windows also exhibits a similar performance level upto 250 Mbps loading levels and its performance declined for higher traffic-loads as shown in Figure 6.
- Free BSD performs a bit better than Windows as shown in Figure 6.

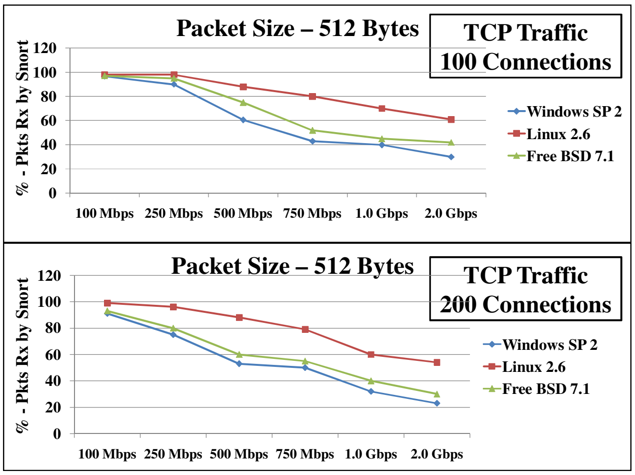


Fig. 6. Snort Packets Received (%) - TCP Traffic (100 & 200 Connections)

6 Analysis

We have identified two basic factors that contribute to the packet-drop limitation in virtual platforms running NIDS in high-speed environments.

6.1 OS and Application Incompatibility

The results have identified different packet capture performance levels by the respective OS platforms. The packets received by virtual platforms are actually the packets received by the Snort application. Overall Linux performed quite well in comparison to Windows and Free BSD for both UDP and TCP traffic. The results lead to the following conclusions:

6.1.1 UDP Traffic

- All platforms respond well for packet sizes greater than 512 Bytes.
- For packet sizes of 128 Bytes and 256 Bytes, Linux performs significantly better than others; however its performance declines above 250 Mbps loading. Windows and Free BSD performed well for 128 Bytes at 100 Mbps traffic-load only.
- All OS platforms non responsive at packet sizes of 128 Bytes and 256 Bytes above 500 Mbps of traffic-loads.
- There were practically no measurable results from all the platforms at 2.0 Gbps loading for all packet sizes.
- The overall performance standing measured was Linux, followed by Free BSD, with Windows in last position.

6.1.2 TCP Traffic

- The systems remain alive for all packet sizes and number of connections for traffic-loads upto 2.0 Gbps.
- The performance of the systems linearly declined in response to increases in the number of connections and traffic-loads.
- Linux outperforms Windows and Free BSD in all the tested scenarios.

6.1.3 Evaluating OS Packet Handling Competency

In order to reach a definite conclusion concerning OS incompatibility as regards the virtualization of NIDS in high-speed networks environments, we extended the research to conduct some additional tests. These tests comprised of three virtual machines built on the same OS platform (Free BSD). The Snort application was activated on all platforms and similar tests were conducted as described in section 4.2. In the first scenario, three machines were operational and in the second one, two were running. The performance of the system was analysed to reach a conclusion.

6.1.4 OS Performance Analysis

In the first scenario, with Free BSD configured on three parallel virtual platforms similar performance metrics were observed. As such the performance of Free BSD was found to be quite similar to the previously executed test-bench scenario and only a small amount of variation was observed. In the second two-machine scenario, an improvement in performance was observed; however performance levels declined at higher traffic-loads. Due to a paucity of space we have only included the results of 512 Bytes of packet size for UDP Traffic as shown in Figure 7. The graph shows the average performance of systems in each scenario.

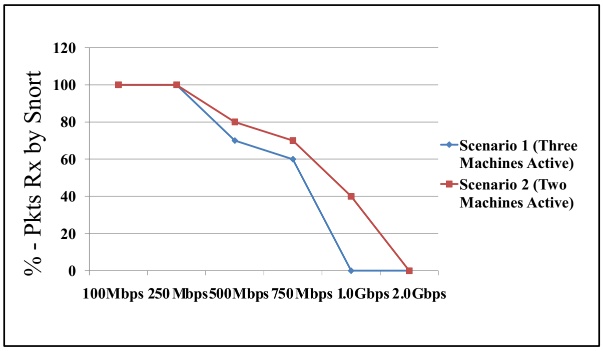


Fig. 7. Snort Packet Received (%) – Free BSD on Three/ Two virtual platforms

The performance of Free BSD in the two scenarios has identified a direct link between packet capturing ability of the system and the use of hardware resource sharing. The results shows that two platforms perform significantly well in comparison to the use of three virtual machines. Thus, it can be concluded that the packet capturing performance for NIDS when run as multiple virtual instances is limited due to the impact of hardware resource sharing and there is no direct relationship to OS itself. Similar tests were also conducted on Linux and Windows platforms; due to space restrictions we have not included the results here. Both platforms behaved in a similar patter as that of Free BSD thus confirming the hardware resource sharing limitation.

6.2 Hardware Incompatibility in Virtualization

The dynamics of virtualization requires the Host OS and the Virtual Machine software (VMware Server) to be stored in the physical memory (RAM) of the host machine. The virtual machines (Windows XP SP 2, Linux 2.6 and Free BSD 7.0) running on VMware Server have been respectively allocated virtual RAM and disk space on the physical hard drive of the host machine. The processes/ applications running on the virtual machines use these simulated virtual RAMs and hard disks for the various operations shown in Figure 8.

Our test-bench has multiple instances of Snort and packet-capture libraries running on different virtual platforms each with a different OS. The packets captured by each virtual machine are less than the packets received by the NIC, thus identifying packet loss somewhere in between. The basic cause of packet loss at each OS apart from the losses incurred by Snort during evaluation is the bottleneck caused by a low disk data transfer rate. The disk I/O statistics as shown in Figure 9 reflects the hardware limitations in handling multiple read/write operations. At 300 Mbps of traffic-load, the disk I/O capacity touches 100% thus its performance at higher loads can be easily ascertained.

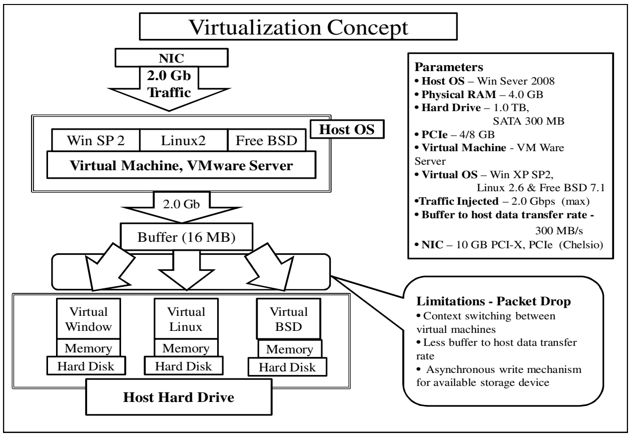

Virtualization Concept
NIC
2.0 Gb Traffic
Win SP 2
Linux2
Free BSD
Host OS
Virtual Machine, VMware Server
2.0 Gb
Buffer (16 MB)
Virtual Window
Memory
Hard Disk
Virtual Linux
Memory
Hard Disk
Virtual BSD
Memory
Hard Disk
Host Hard Drive
Parameters
• Host OS – Win Sever 2008
• Physical RAM – 4.0 GB
• Hard Drive – 1.0 TB, SATA 300 MB
• PCIe – 4/8 GB
• Virtual Machine - VM Ware Server
• Virtual OS – Win XP SP2, Linux 2.6 & Free BSD 7.1
•Traffic Injected – 2.0 Gbps (max)
• Buffer to host data transfer rate - 300 MB/s
• NIC – 10 GB PCI-X, PCIe (Chelsio)
Limitations - Packet Drop
• Context switching between virtual machines
• Less buffer to host data transfer rate
• Asynchronous write mechanism for available storage device

Fig. 8. Virtualization Concept

The memory and storage for each virtual machine has actually been allocated on the physical storage resources (i.e. hard disk) of the host machine. Packets received by the NIC without any loss are transferred to the hard disk buffer at the PCI rate (4/8 Gbps). From this buffer, these packets are required to be written to the disk at the buffer- to-host transfer rate of 300 MB/sec (SATA Hard Drive [28]); thus a huge gap between the disk-transfer rate and the incoming traffic load exists. In addition, traffic

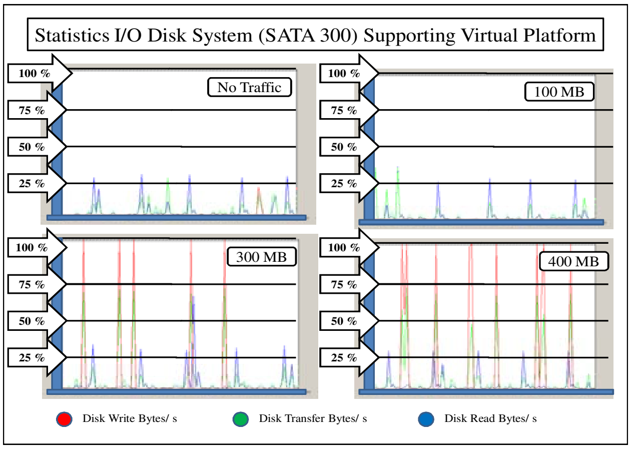


Fig. 9. Statistics I/O System (SATA 300) Hard Drive

is fed to all virtual machines simultaneously (in parallel mode), the disk is physically only able to write to one location at a time. Thus any disk-write instance to a virtual machine, will cause packet drops on another. There are also some additional packet losses due to context switching within the hard disk.

In order to augment our analytical stance of showing hardware as one of the major bottlenecks for the efficacy of the virtualization concept for NIDS in high-speed networks we have utilized the disk queue length counter as shown in Figure 10. In normal circumstances, the average disk queue length should be three or less (its ideal value) [29]. However; in our test network, it is observed to be always greater than the ideal value for the traffic ranges measured at 2.0 Gbps.

7 Conclusion

This research has focused on ways of determining the efficacy of the virtualization concept for NIDS in high-speed network environments. The test scenarios employed involved the use of the widely deployed open-source NIDS, namely Snort. The results obtained have shown a number of significant limitations in the use of virtual NIDS, where both packet-handling and processing capabilities at different traffic loads were used as the primary criteria for defining system performance. We have confirmed that the underlying host hardware plays a prominent role in determining overall system performance. We have further shown that performance is further degraded as the number of virtual instances of NIDS is increased, irrespective of the virtual OS used. Furthermore, we have demonstrated a number of significant differences in the performance characteristics of the three different virtual OS environments in which Snort was run.

This work has identified specific and replicable bottlenecks in commonly used implementations of virtualization for a widely used NIDS in high-speed networks. The results obtained can be taken as a benchmark for improving the performance of these systems in future research work. These shall also provide an experimental data to the researchers which were felt missing in the previous efforts.

Acknowledgement. The authors would like to thank the anonymous reviewers for their insight comments. University of Bradford and Syphan technology also deserved deep appreciation for their assistance and support.

References

1. Singh, A.: An Introduction to Virtualization, `http://www.kernelthread.com/publications/virtualization`
2. Business value of virtualization: Realizing the benefits of integrated solutions, `http://h18000.www1.hp.com/products/servers/management/vse/Biz_Virtualization_WhitePaper.pdf`
3. Virtualization, `http://www.windowsecurity.com/whitepapers/Virtualization.html`
4. Inella, P.: An Introduction to IDS, `http://www.securityfocus.com/infocus/1520`
5. Shannon, C., Moore, D.: The spread of the Witty Worm. IEEE Security and Privacy 2(4), 46–50 (2004)
6. Buffer overflow, `http://www.mcafee.com/us/local_content/white_papers/wp_ricochetbriefbuffer.pdf`
7. The spread of Witty worms, `http://www.caida.org/research/security/witty`
8. Snort, `http://www.Snort.org/`
9. Baker, A.R., Esler, J.: Snort IDS and IPS Toolkit, Syngress, Canada (2007)
10. Alserhani, F., Akhlaq, M., Awan, I., Cullen, A., Mellor, J., Mirchandani, P.: Evaluating Intrusion Detection Systems in High Speed Networks. In: Fifth International Conference of Information Assurance and Security (IAS 2009). IEEE Computer Society, Los Alamitos (in press, 2009)
11. VMware Server, `http://www.vmware.com/products/server/`
12. Windows Server (2008), `http://www.microsoft.com/windowsserver2008/en/us/default.aspx`
13. Windows XP SP2, `http://www.softwarepatch.com/windows/xpsp2.html`
14. Linux 2.6, `http://www.kernel.org/`
15. Free BSD 7.1, `http://www.freebsd.org/where.html`
16. Xu, J., Zhao, M., Fortes, J.A.B., Carpenter, R., Yousif, M.: On the Use of Fuzzy Modelling in Virtualized Data Center Management. In: Proceedings of 4th International Conference on Autonomic Computing, ICAC 2007 (June 2007)
17. Virtualization and disk performance /pdf/Virtualization_, `http://files.diskeeper.comPerformance.pdf`
18. Schneider, F., Wallerich, J., Feldmann, A.: Packet capture in 10-gigabit ethernet environments using contemporary commodity hardware. In: Uhlig, S., Papagiannaki, K., Bonaventure, O. (eds.) PAM 2007. LNCS, vol. 4427, pp. 207–217. Springer, Heidelberg (2007)

19. Optimizing network infrastructure for virtualization,
`http://www.dell.com/downloads/global/power/ps3q08-20080375-Intel.pdf`
20. Salim, J.H., Olsson, R., Kuznetsov, A.: Beyond Softnet. In: Proceedings of USENIX 2001, November 2001, pp. 165–172 (2001)
21. Deri, L.: Improving Passive Packet Capture: Beyond Device Polling. In: Proceedings of the 4th International System Administration and Network Engineering Conference, Amsterdam (September 2004)
22. Biswas, A., Sinha, P.: A high performance packet capturing support for Alarm Management Systems. In: 17th International conference on Parallel and Distributed Computing and Systems (PDCS), Phoenix (2005)
23. Salah, K., El-Badawi, K.: Performance Evaluation of Interrupt-Driven Kernels in Gigabit Networks. In: The IEEE Global Telecommunications Conference, GLOECOM 2003, December 2003, pp. 3953–3957 (2003)
24. ProCurve Series 2900 switch,
`http://www.hp.com/rnd/products/switches/HP_ProCurve`
25. LAN Traffic V 2,
`http://www.topshareware.com/lan-traffic-v2/downloads/1.html`
26. D-ITG V 2.6, `http://www.grid.unina.it/Traffic/index.php`
27. Berkley Packet Filter,
`http://www.freebsd.org/releases/7.1R/relnotes.htm`
28. SATA Technology, `http://www.serialata.org/`
29. Disk Queue Length Counter,
`http://www.windowsnetworking.com/articles_tutorials/Windows-Server-2003-PerfTuning.html`