IPSJ Journal Vol. 52 No. 7 2212–2222 (July 2011)

Regular Paper

Feasibility Study of Security Virtual Appliances for Personal Computing

Ahmad Bazzi[†1] and Yoshikuni Onozato[†1]

Computers connected to the Internet are a target for a myriad of complicated attacks. Companies can use sophisticated security systems to protect their computers; however, average users usually rely on built-in or personal firewalls to protect their computers while avoiding the more complicated and expensive alternatives. In this paper we study the feasibility —from the network performance point of view— of a VM configured as an integrated security appliance for personal computers. After discussing the main causes of network performance degradation, we use *netperf* on the host computer to find the network performance overhead when using a virtual appliance. We are mainly concerned with the bandwidth and the latency that would limit a network link. We compared the bandwidth and latency of this integrated security virtual appliance with current market products and in both cases, the performance of the virtual appliance was excellent compared with hardware counterparts. This security virtual appliance for example allows more than an 80 Mbps data transfer rate for individual users, while security appliances generally allow only 150 Mbps for small office users. In brief, our tests show that the network performance of a security virtual appliance is on par with the current security appliances available in the market; therefore, this solution is quite feasible.

1. Introduction

The Internet has become an essential medium for communication and collaboration. Yet, computers connected to the Internet continue to face major security threats that range from viruses, Trojans and worms to malicious hackers trying to break into them. Such attacks continue to increase in both scale and sophistication as well as the defense mechanisms for dealing with them.

In order to provide protection against these diverse threats, companies continue to invest huge amounts of money in information security systems and appliances. The average user, on the other hand, cannot afford such sophisticated expensive solutions, so many of them rely only on the default firewall shipped with their operating system (OS). The efficiency and customizability of these firewalls can vary greatly depending on the vendor and the OS version. Our objective is to provide the average user with an inexpensive, efficient and reliable security solution regardless of the security of his OS, while ensuring high network performance.

In past years, multi-core CPUs have become the standard not only for desktop processors but also for processors used in most laptop series. This situation has created abundant processing power that we can use for running a virtualization program for instance without slowing the host OS. Therefore we look once again to using computer virtualization technology to create a virtual machine (VM) with a preconfigured security function.

The use of a VM as a security virtual appliance (SVA) has become more common in the past years, in particular for companies with virtualized servers and for companies using Platform as a Service (PaaS) in cloud computing. For example, VMware Virtual Appliance Marketplace [1)] is dedicated to providing an extensive list of VA images with a diversity of applications such as Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM), IT administration, and security. In fact the market has become so specific that virtualizing the demilitarized zone (DMZ) that is logically and physically isolated from internal and external networks has lead Cisco to release a dedicated virtual switch [2)].

Considering the drawbacks of a software firewall program, there are several advantages that we can find in a SVA:

Isolation between host OS and guest OS The guest OS is logically isolated from the host OS. The guest OS is running a different operating system, consequently it is completely secure against the vulnerabilities of the host OS. Reference 3) tries to take advantage of the isolation between the guest OSs themselves. In our approach however we try to take advantage of the isolation between the host OS and the guest OS, similar to Ref. 4).

Fail-close Design In our implementation, the network interfaces are configured in a way to ensure that no packet can reach the host OS without passing through the guest OS.

Integrated Security Appliance In the guest VM, we have the chance to in-

†1 Graduate School of Engineering, Gunma University

stall an OS with a complete security solution, in contrast with being limited to installing one piece of software for protection. Moreover, because it is a complete OS, it can be easily equipped with new security functions by adding additional security software.

There is a small number of research publications about using a SVA on personal computers. Prevelakis's work [4] is an excellent example, where he discusses the technical challenges that he encountered and solved while running an OpenBSD firewall VM on a MS Windows 2000 laptop. Unfortunately the SVA was limited to the firewall function and provided no easy interface that the average user can access. The author did not study network performance of such a SVA but was mainly concerned with technical aspects of the proposed and implemented solution.

Since the publication of Ref. 4), many things have changed; in particular, the average speeds of the available Internet subscription plans nowadays are increasingly higher than the average speeds in 2005. Consequently, a personal firewall is now expected to process higher bandwidth. From the CPU manufacturer's side, computers with multi-core CPUs have become the standard not only for desktops but also for laptops. Moreover, AMD has added hardware-assisted virtualization to their newer processors using AMD Virtualization (AMD-V) [5] while Intel is equipping its new processors with Intel Virtualization Technology (Intel VT) [6].

In this paper, we are mainly concerned with investigating the network performance —bandwidth and latency— of SVAs on a PC. We are interested in virtualizing and testing a complete security appliance that incorporates an Intrusion Detection System (IDS), anti-virus, etc. We also focus on the firewall function where the entire appliance can be configured and managed through a graphical user interface (GUI). This is in comparison with Ref. 4) which only considers a basic command-line-based firewall with no study of the network performance aspects.

This paper is organized as follows: In Section 2 we briefly mention the motivation behind this paper and how virtualization can be used to add to the client's security. We discuss why the network performance might be affected and is worth benchmarking in more detail in Section 3. In Section 4 we present our test environment providing technical specifications of servers and clients along with the SVA that we setup. The network performance tests are presented next, where Section 5 deals with the bandwidth tests while Section 6 deals with the latency tests. We offer our conclusions in Section 7.

2. Background and Previous Works

Most recent operating systems are shipped with a built-in firewall to help defend the systems from malicious packets. These firewalls are usually the main defense mechanism against network attacks that they would face in any non-secured network. In one example, a successful attack on a MS Windows XP can stop the firewall service and render the computer system completely exposed [7].

There are a number of reasons why such attacks are possible. In addition to the OS's vulnerabilities, these firewalls are installed and run as a service on the same OS they are protecting. A successful attack against the host system can therefore stop the firewall service. Moreover when the firewall service fails, the system goes into a fail-open state allowing all network traffic to pass.

One way to tackle this situation would be by isolating the firewall from the host OS and ensuring that the system will become inaccessible in case of a firewall failure. This can be achieved by using a hardware security appliance; however, this is relatively expensive and quite inconvenient for mobile users. We can achieve a certain level of isolation through virtualization technology. We refer the reader to Ref. 8) for an overview of virtualization, to Ref. 9) for current virtualization technologies, and to Ref. 10) for the network virtualization concepts. We recommend Ref. 11) for an in-depth technical overview.

Here, Ref. 3) covers virtual data center with multiple hypervisors forming a cluster together, and each hypervisor hosting several VMs. These VMs can be transferred from one hypervisor to another using "live migration". Moreover, these VMs should be protected by separate security appliances. The authors highlight the trend to virtualize the network security functions using SVAs. Their aim is to provide a distributed and scalable security function for the network flow and for the guests hosted in a virtual datacenter. Most importantly, the network performance tests they carried out show that these SVAs provide a performance level comparable to that of physical appliances while providing additional benefits.

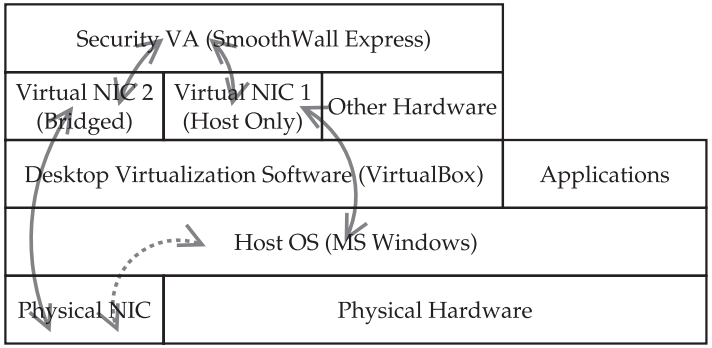

Fig. 1 Packet path: The solid line shows the network packet path when using passing through a SVA, while the dashed line shows the network packet path when no firewall VA is being used.

In contrast, we are interested in providing additional security to the average user. Instead of studying the network performance of SVAs for guests running on hypervisors, we are concerned with the performance of SVAs running on and protecting common operating systems such as MS Windows 7.

A SVA can be built from free open source software (FOSS) [12] to minimize the cost and can be saved on the computer like any other file. This makes it suitable for the average user as it adds to his system security without causing prohibitive inconvenience. This approach will create a certain level of isolation between the firewall and the host OS. Moreover, steps can be taken to ensure that the host OS will become unreachable in case of a firewall failure as discussed later in Section 4.1.

The SVA will run on the virtual hardware created by the virtualization software running on the host computer as described in Ref. 4). The network packets sent or received by the host have to pass through this VA. In **Fig. 1**, the dashed arrow indicates the usual path of the network packets. These packets first have to pass through the OS driver and then through the physical NIC to reach the network. However, when a SVA is used, the host OS will use a private IP on a virtual NIC that is only shared with the VM. Consequently, the packets leaving the host OS have to first go through the host-only virtual NIC to be processed by the SVA. Then the VA will deliver them again through another virtual NIC bridged to the physical network as indicated by the solid arrows. We will discuss this in more detail in Section 3. As one might expect, this will affect the network performance, so it is worth studying network performance degradation due to the SVA to check the feasibility of this approach.

3. Performance Overhead

There are many publications regarding the network performance of a VM. In Ref. 13), different network performance tests were run from a VM to show how close the network performance of VMs is to the physical ones.

The tests that we carry out in this paper are different. On one hand we are using common personal computers, while on the other we are testing the network performance of the physical host machine when all its network traffic passes through the VA running on this same host. In other words, the VA is not the final destination but just part of the path that the network packets have to travel through when leaving or entering the host.

In Ref. 14), the authors study the performance overhead in the Xen hypervisor. They state that the computational overhead of the hypervisor and the driver domain cause a reduced network throughput. Although in this paper we are concerned with network performance when using desktop virtualization software (instead of a hypervisor replacing the OS), a similar statement can be made as to the overhead caused by the virtualization software.

The discussion in Ref. 15) relates more to our experiments as they are studying the performance of a VM running on desktop virtualization software. Reference 15) explains the technical details that lead to the latency incurred when a packet is sent from or to the VM. We can see from their work that the performance overhead causes include the world switch (from virtual to physical and vice versa), the additional IRQs that need to be raised, and the CPU virtualization overhead.

The network packet would normally pass through the network card to be handled by the OS kernel which would deliver it to the related application. However observing Fig. 1, we can see that when using a virtual firewall, the network packet will be received by the network card and then delivered to the host kernel which in turn will deliver it to VirtualBox —as the target application— and then from

VirtualBox it will go to the firewall guest OS. Inside the guest OS, it will be processed according to the implemented firewall rules. If it is allowed to pass, it will go through the Virtual Box host-only interface to be delivered to the correct application that is waiting for this packet.

The use of virtualization leads to performance overhead due to several reasons. These include:

(1) Privilege level switching [15]
(2) Trapping certain privileged instructions [9),16] (part of the CPU virtualization overhead)
(3) Mapping memory pages [9]

Other reasons that would affect performance for the case of a FW VA include:

(1) The FW uses up CPU time whenever it processes a packet according to firewall rules.
(2) The FW uses up CPU time when it performs the NAT function.
(3) There will be several additional necessary interrupts to send and receive any packet [15].

3.1 Protection Rings; Privilege Level Switching

Current CPUs have 4 protection rings, i.e., 4 privilege levels, as shown in Section 4.3.5 in Ref. 17). For normal OS functions, the OS usually switches between privilege level 0 and privilege level 3; the kernel would run in privilege level 0 and user applications in privilege level 3. Most virtualization software will use privilege level 1 to run the guest OS [16]. Eventually the host OS switches between the host kernel running in privilege level 0 and the guest OS running in privilege level 1 and the user programs running in privilege level 3 as shown in **Fig. 2**. This means additional privilege level switching needs to take place and this in turn consumes additional CPU cycles [15].

3.2 Trapping Certain Privileged Instructions

As already mentioned, the guest OS cannot run in level 0 but instead runs in level 1. However, the guest OS will run normally and even try to execute CPU instructions that require a level 0 privilege. The virtualization software has to implement some technique to replace the guest OS privileged instructions with suitable ones. For example, Oracle "VirtualBox contains a Code Scanning and Analysis Manager (CSAM), which disassembles guest code, and the Patch Manager (PATM), which can replace it at runtime code scanning and patching [16]."

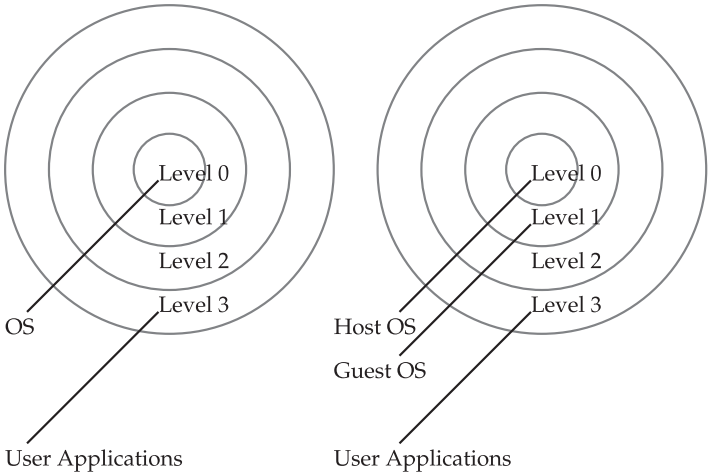

Fig. 2 CPU Protection Rings and Virtualization: The host OS runs in privilege level 0, while the applications run in privilege level 3. A virtualized guest OS usually runs in privilege level 1.

3.3 Virtual to Physical Memory Mapping

When the guest OS accesses its memory, it will actually be accessing virtual memory pages mapped to physical ones [9]. The real-time mapping between the physical and virtual memory space is another cause of virtualization overhead.

3.4 System Interrupts

A VM hosted using desktop virtualization software will require additional system interrupts to access the network [15]. Running the VM as a firewall with 2 NICs necessitates additional interrupts as the packet is first delivered to the VA, then it has to leave the VA to be delivered to the host OS.

We will use **Fig. 3** to shed more light on the additional necessary steps. Let's consider the action of receiving a network packet as an example. First the network packet reaches the physical NIC, which will lead to an interrupt. The CPU might be executing a certain program in privilege level 3 but it has to call the required function to handle this interrupt; this usually means switching to privilege level 0 to process the packet and deliver it to the OS.

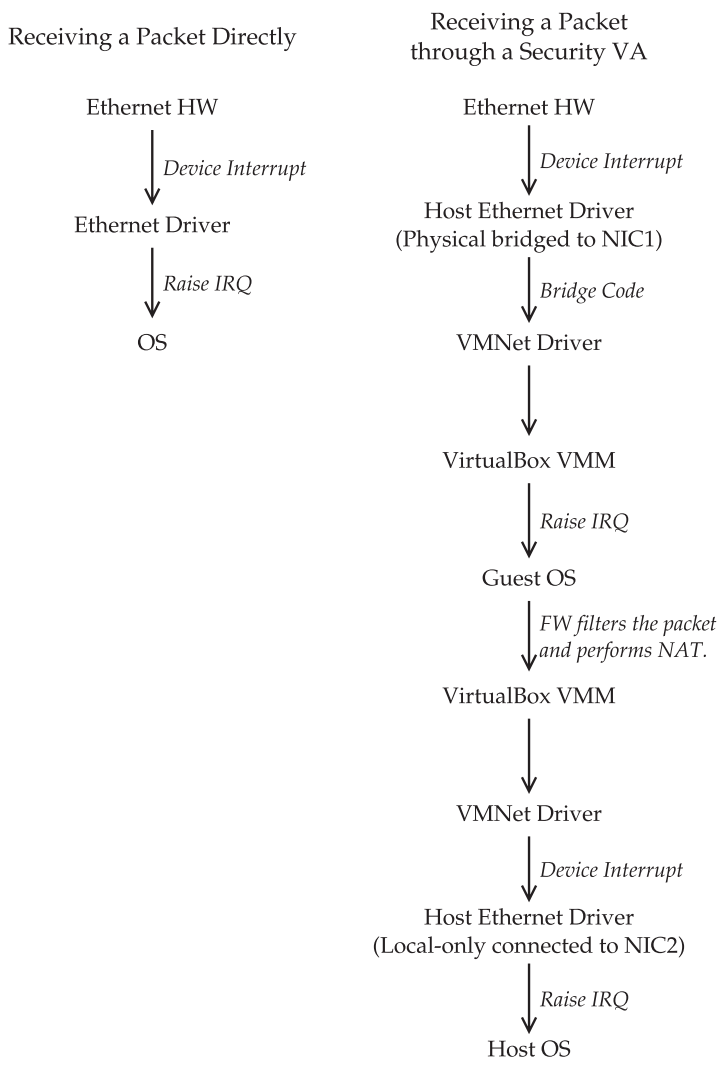

Fig. 3 Receiving a packet: The left side shows receiving a packet in usual cases and the right side shows receiving a packet when using a firewall VA. There are additional interrupts and processing overhead.

When using a firewall VA, instead of delivering it directly to the OS, "VirtualBox Bridged Networking Driver" will be invoked because the physical NIC is bridged to the virtual NIC2 as discussed in Fig. 1. Now the packet will trigger another interrupt as it needs to be processed by the guest OS. However the interrupt will this time be on the virtual CPU. In VirtualBox, the guest OS is run in privilege level 1. Inside the guest OS, the packet will be processed by the firewall. It might be filtered out depending on the imposed firewall rules. Moreover, we have enabled a NAT functionality, so the packet will be resent accordingly. There might be certain programs that will be triggered by the incoming packet and need to run in privilege level 3. After the guest OS finishes processing the packet, it needs to pass it to the host OS. This will issue an interrupt to use virtual NIC1. This will lead to another interrupt on the host OS as it has an incoming packet. Running in level 0, the host will handle the received packet as now it is addressed to its final destination.

Obviously this leads to an overhead on every single packet that is sent or received. We wanted to answer the question of, what level of network performance we can expect when using a virtual firewall. In order to answer this question, we created a set of network configurations similar to those that we would encounter in real network communications.

We wanted to know how such a system configuration would perform with real Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) traffic. Using netperf, we can use the bulk data transfer tests to study the bandwidth; netperf allows us to configure the socket and message sizes. Moreover, we used the Request/Response tests to study the latency; netperf allows us to set the send and receive packet sizes. Every test was run with two configurations: one with a direct connection to the server and another with a VM configured as a firewall.

4. The Network Test Environment

We begin by describing the implementation before introducing the network performance results. The network setup of the experiment is shown in **Fig. 4**, where a PC connects through a VA to a server that runs on physical hardware.

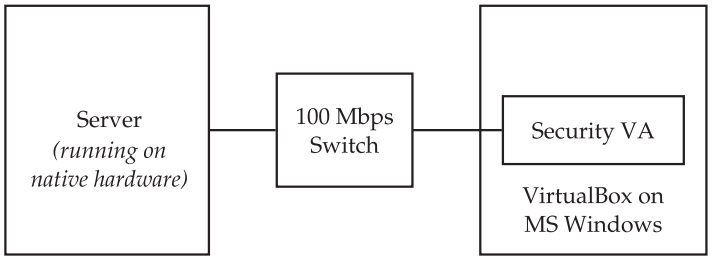

Fig. 4 The network setup for the experiment.

4.1 The Firewall Virtual Appliance

In Ref. 4), Prevelakis uses OpenBSD to create a minimal firewall to protect a MS Windows 2000 laptop. Conceptually, our setup is similar to the virtual firewall in Ref. 4) but with certain technical differences. First, we used VirtualBox [18] as our desktop virtualization software. VirtualBox is virtualization FOSS and it is quite rich in features. Moreover, being FOSS, there are no restrictions on publishing performance tests which makes it ideal for our purposes. Second, we used SmoothWall Express [19] to create the virtual firewall. SmoothWall Express is an open source firewall project based on GNU/Linux that started in 2000. It incorporates all the features expected in a modern firewall available in some of the expensive alternatives, such as a GUI interface with network utilization graphs, anti-virus and IDS and therefore is a suitable choice for creating the virtual firewall to use in our tests. In brief, instead of using a minimal firewall as Ref. 4), we used a complete firewall with a GUI, logging and additional security functions that range from IDS to anti-virus.

Using VirtualBox, we created a VM with SmoothWall Express firewall and configured it with 2 virtual network interface cards (NICs); the first virtual NIC is configured as "Host-only" and is completely isolated from the physical network, while the second NIC is configured as "Bridged" and therefore has to be attached to a physical NIC. On the MS Windows host OS, the different drivers/items used by the physical NIC can be turned off except for the "VirtualBox Bridged Networking Driver" which is the only item necessary for the VM to communicate with the physical network. The firewall is configured to use network address translation (NAT) in order to provide network access to the protected host.

The specifications of the firewall VA are as follows:

OS SmoothWall Express 3.0 SP1
CPU 1 core of the host machine
Memory 256 MB RAM
Disk 2 GB
NIC1 PCnet-FAST III (Am79C973)
NIC2 PCnet-FAST III (Am79C973)

4.2 Client/Desktop Configuration

We wanted to create a setup that is similar to what we would find on current average computers. The two desktops have identical hardware:

OS MS Windows XP SP3 32-bit edition
and MS Windows 7 64-bit edition
CPU AMD Athlon64 X2 Dual Core 2.90 GHz (5600+)
Memory 2 GB
Disk Seagate Barracuda 7,200 rpm 500 GB
NIC Artheros L1 Gigabit Ethernet Driver

4.3 Server Configuration

The server was configured as follows:

OS (GNU/Linux) Ubuntu 9.10 Server 64-bit edition
CPU AMD Athlon64 X2 Dual Core 2.90 GHz (5600+)
Memory 4 GB RAM
Disk Seagate Barracuda 7,200 rpm 1 TB
NIC Intel Gigabit Ethernet Driver

4.4 Benchmark Tool

We used netperf [20] to carry out the bandwidth and latency tests. We ran each test for 10 minutes. Moreover each test was repeated at least 5 times in order to ensure consistent results with a $99\% \pm 0.5\%$ confidence level.

5. Bandwidth Tests

Using netperf, we carried out several bulk transfer tests on the MS Windows XP host and the MS Windows 7 host for both the TCP and UDP protocols. We set the socket buffer size to 56 KB on both the host and server and measured bulk transfer with a 4 KB message size. Then we set the socket buffer size to

Table 1 Results of the bulk transfer over TCP (with the TCP_STREAM) option on the MS Windows hosts.

Socket size (KB)	56	32	32
Message size (KB)	4	4	1
MS Windows XP direct connection (10^6 bits/sec)	94.38	94.38	94.38
MS Windows XP through VA (10^6 bits/sec)	84.79	83.76	83.72
Degradation percentage	10.16%	11.25%	11.29%
MS Windows 7 direct connection (10^6 bits/sec)	94.37	94.38	94.38
MS Windows 7 through VA (10^6 bits/sec)	91.20	91.03	91.01
Degradation percentage	3.36%	3.55%	3.57%

Table 2 Results of the bulk transfer over UDP (with the UDP_STREAM) option on the MS Windows hosts.

Socket size (KB)	56	32	32
Message size (KB)	4	4	1
MS Windows XP direct connection (10^6 bits/sec)	95.99	95.99	94.18
MS Windows XP through VA (10^6 bits/sec)	94.14	94.22	91.56
Degradation percentage	1.93%	1.84%	2.78%
MS Windows 7 direct connection (10^6 bits/sec)	95.97	95.95	94.17
MS Windows 7 through VA (10^6 bits/sec)	92.05	92.05	93.66
Degradation percentage	4.08%	4.06%	0.54%

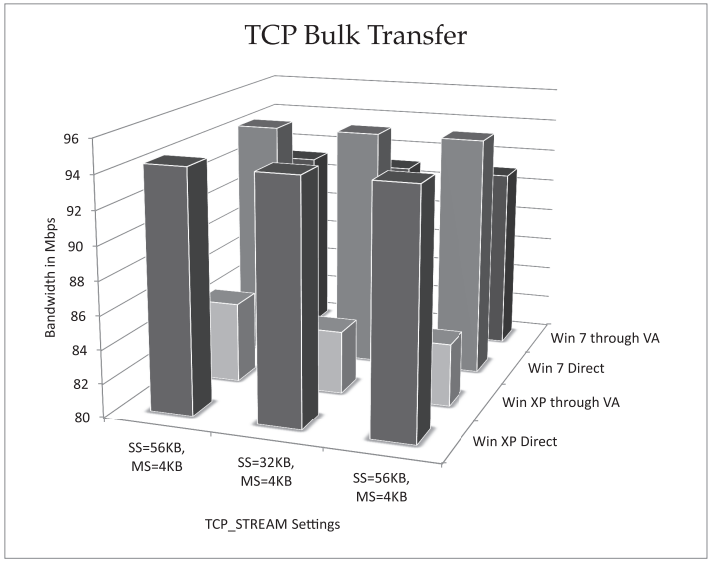

Fig. 5 Results of the bulk transfer over TCP (with the TCP_STREAM) option on the MS Windows hosts.

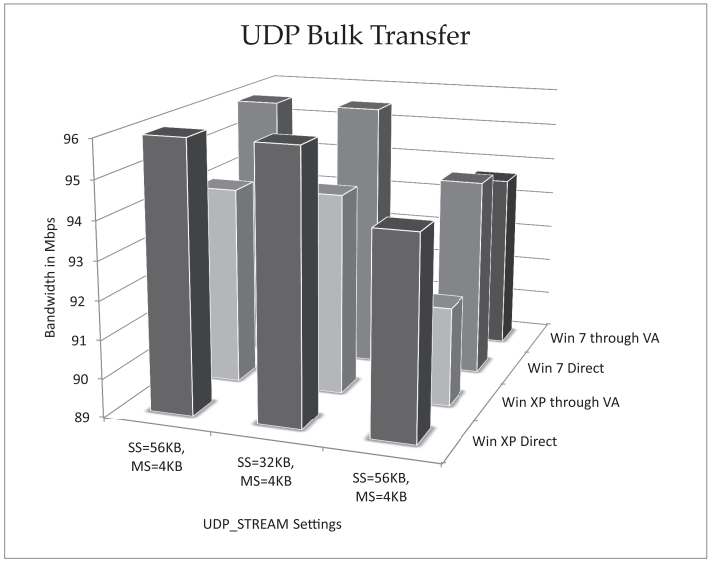

Fig. 6 Results of the bulk transfer over UDP (with the UDP_STREAM) option on the MS Windows hosts.

32 KB on both the host and server, and checked the bulk transfer rate with a message size of 4 KB then 1 KB. The bulk transfer results on the TCP protocol for Windows XP host and Windows 7 host are shown in **Table 1** and shown as a graph in **Fig. 5**. The results using the UDP protocol are shown in **Table 2** and shown as a graph in **Fig. 6**.

TCP is used quite often because of its reliable transmission mechanisms; most of the users' programs utilize the TCP protocol. If the computer is connected directly to the server using a 100 Mbps Ethernet connection, the maximum band-

width was measured as 94.38 Mbps (where 1 Mbps is equal to 1×10^6 bits/sec) as shown in Table 1. The performance dropped to approximately 91 Mbps in the case of MS Windows 7 and dropped to approximately 84 Mbps for XP due to the SVA configured as described earlier. Hence, the total performance degradation was less than 4% for MS Windows 7 and less than 12% for XP. Although 11.29% degradation might sound large, 84 Mbps is still way beyond the needs of the average user.

Actually a computer user rarely utilizes a 100 Mbps link to its full capacity; an average user's Internet utilization usually does not exceed 2 Mbps, and might reach 10 Mbps or 20 Mbps during heavy downloads for short periods of time. The reason for this is that bandwidth-demanding Internet applications such as online streaming radio stations usually transmit at a bitrate between 64 kbps and 256 kbps. Similarly for streaming video websites, the bitrate is up to 200 kbps for normal quality videos (screen resolution of 320×240 resolution), up to 900 kbps for high quality videos (screen resolution of 480×360 resolution), and finally around 2 Mbps for high definition (HD) quality videos.

On the other hand, even if the Internet Service Provider (ISP) subscription is 100 Mbps broadband speed or 1 Gbps for example, the download speed might reach 10 Mbps or 20 Mbps data transfer speed in best cases as it is usually capped by the remote download server. It can be noticed that the usual bottleneck on the Internet nowadays tends to be the remote servers and not the high-speed Internet subscription plan. Hence, even for the most bandwidth-demanding Internet applications, the necessary bandwidth is less than one fourth of what a SVA can process.

Finally, let's consider two products in the market from Juniper Networks and from Cisco. The Juniper IDP75 is an intrusion detection and prevention appliance for small and mid-size businesses and supports a maximum throughput of 150 Mbps data transfer speed [21]. The Cisco ASA 5505 is described as a "full-featured security appliance for small business, branch office,..." and it supports up to 150 Mbps data transfer speed [22]. Both of these security appliances are for small business, i.e., for several simultaneous users, and they support 150 Mbps as maximum. Hence, a SVA that supports above 80 Mbps data transfer speed for a single user is considered quite fast by current business standards.

Besides many of the DNS requests, UDP is used in many media streaming protocols that do not need to guarantee the delivery of all the packets. Because it is connectionless and requires less overhead than TCP, we can see that it generally allowed higher bulk transfer rates as shown in Table 2. Moreover, we can see that the degradation in performance didn't go beyond 4.08% in the case of MS Windows 7, while it reached 2.78% in XP's case.

Similarly, we can see that the most bandwidth-demanding applications nowadays don't reach even one fourth of the bandwidth that a complete SVA can process.

6. Latency Tests

We used netperf's "Request/Response" in order to study the effect of the firewall VA on the network latency. This test is "a synchronous, one transaction at a time, request/response test," where "a transaction is defined as the completed exchange of a request and a response [23]." The output of this test indicates the average number of transactions that took place during one second.

To study the latency when using TCP, we used TCP_RR, which stands for TCP Request/Response. TCP_RR works by establishing a TCP connection at the beginning then exchanging packets of preset sizes for the test duration and finally breaking the TCP connection. This is similar to the mechanism of downloading or uploading a single file over HTTP or FTP protocols for example. The TCP_RR test allows us to control the sizes of the request packet and the response packet. We set the request-response sizes to 1–1 (one byte request, one byte response), 32–256, 32–512, and 32–1024. The numerical results of our tests for Windows XP and Windows 7 hosts are shown in **Table 3** and shown as a graph in **Fig. 7**. The maximum degradation in performance was at the 1–1 case: in XP's case, it dropped from 4879 to 2248 which is around 54%.

The question that we pose now is whether a rate of 1590 transactions/second (the lowest we recorded) is acceptable. To answer this question, we setup a test machine to represent a heavy user. We ran multiple messaging and chatting programs, opened more than one webmail account simultaneously, connected to a streaming radio station, setup several downloads in the background, continuously browsed websites, etc. We noticed that none of these tasks reached even 20

Table 3 Results of the TCP Request/Response (TCP_RR) test with varying request and response packet sizes on the MS Windows hosts.

Request size (byte)	1	32	32	32
Response size (byte)	1	256	512	1024
MS Windows XP direct connection (trans./sec.)	4879	2449	2448	2447
MS Windows XP through VA (trans./sec.)	2248	2154	1605	1590
Degradation percentage	53.92%	12.05%	34.44%	35.02%
MS Windows 7 direct connection (trans./sec.)	4884	2457	2448	2447
MS Windows 7 through VA (trans./sec.)	2436	2156	1851	1713
Degradation percentage	50.12%	12.25%	24.39%	30.00%

Table 4 Results of the UDP Request/Response (UDP_RR) test with varying request and response packet sizes on the MS Windows hosts.

Request size (byte)	1	32	32	32	516
Response size (byte)	1	256	512	1024	4
Windows XP direct connection (trans./sec.)	4883	2452	2446	2448	2449
Windows XP through VA (trans./sec.)	2253	2244	2120	1774	2128
Degradation percentage	53.86%	8.48%	13.33%	27.53%	13.11%
Windows 7 direct connection (trans./sec.)	4884	4869	2449	2448	2450
Windows 7 through VA (trans./sec.)	2444	2408	2217	1822	2220
Degradation percentage	49.96%	50.54%	9.47%	25.57%	9.39%

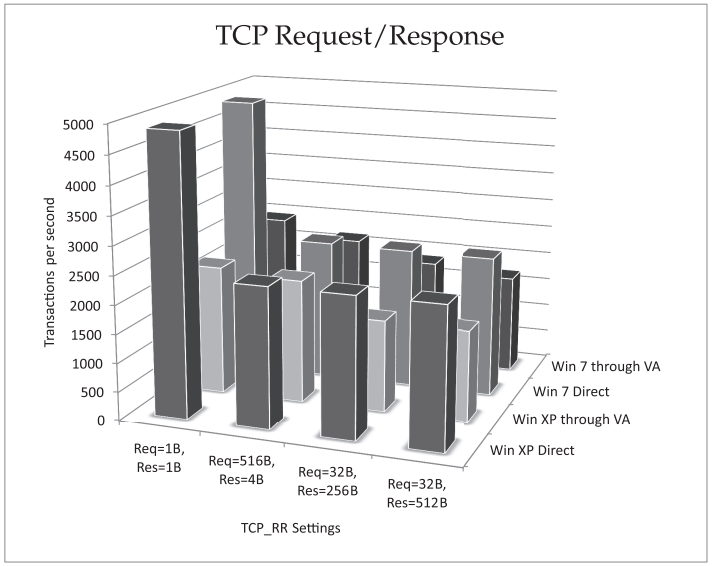

Fig. 7 Results of the TCP Request/Response (TCP_RR) test with varying request and response packet sizes on the MS Windows hosts.

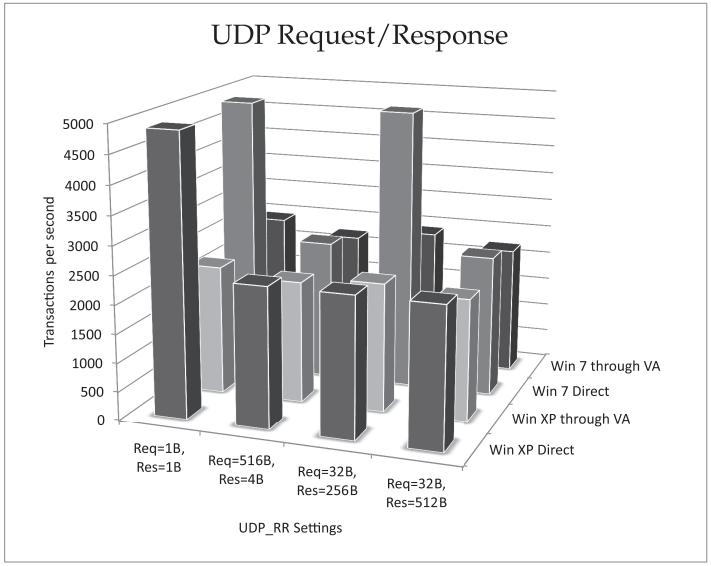

Fig. 8 Results of the UDP Request/Response (UDP_RR) test with varying request and response packet sizes on the MS Windows hosts.

transactions per second. Eventually, we can clearly see that this is negligible when compared to the possible number of transactions/second that the system can maintain.

Finally, we compare again the SVA performance with the Cisco ASA 5505. According to Ref. 24), the ASA 5505, which is intended for a small business, can support up to 4000 new connections/second. Consequently, the performance of the SVA can support more than 2000 connections/second for a single user and this is quite high according to the current industry standards.

In order to study the latency in UDP communications, we used UDP_RR short for UDP Request/Response. Again, this test exchanges UDP packets of preset sizes during the test duration. Since UDP is connectionless, there is no need to establish a connection and later break it as in the TCP case. One real scenario that uses UDP and resembles this test is a file transfer over TFTP. As with the TCP_RR test, we test the 1–1 (one byte request, one byte response) case, 32–256, 32–512, 32–1024, and we add one extra test with 516–4 to simulate a TFTP file upload. The numerical results are shown in **Table 4** and shown as a graph in **Fig. 8**. As with the bulk transfer tests, the request/response tests scored higher in the case of UDP since it is connection-free and simpler to implement.

7. Conclusion

The increase in computer performance and development of multi-core processors for personal computers in recent years allow new applications including SVA that were not possible up until now. To study their impact on network performance, we configured an integrated SVA using SmoothWall. This SmoothWall is an all-inclusive firewall distribution that includes an IDS, anti-virus, and a Virtual Private Network (VPN) server among other security and administration functions. In other words, the SmoothWall VA contained all the expected functions from current commercial integrated security appliances including traffic monitoring and logging.

We tested the network performance of this SVA on MS Windows XP and 7 to study the virtualization overhead. The test results show that using a SVA, the network performance of the host machine is excellent compared to the current products in the market. For instance, this SVA allows 80 Mbps data transfer speed for a single user, while small hardware appliances usually allow up to 150 Mbps data transfer speed for a small office. One integrated security appliance can support up to 4000 new connections/second for the users of a small office combined. Our tests show that this SVA can support more than 2000 connections/second for a single user. Hence, we can clearly see that the performance level of this SVA was quite on par with physical hardware which makes this a solid solution.

Acknowledgments We would like to thank Professor Ushio Yamamoto for his insightful suggestions and for referring us to the work of Prevelakis [4].

This research was supported in part by the Grant-in-Aid for Scientific Research (C)22510139 from the Japan Society for the Promotion of Science (JSPS).

References

1) VMware Inc.: VMware Virtual Appliance Marketplace: Virtual Applications for the Cloud (Online), available: http://www.vmware.com/appliances/
2) Cisco and VMware Inc.: DMZ Virtualization Using VMware vSphere 4 and the Cisco Nexus 1000V Virtual Switch (2009).
3) Basak, D., Toshniwal, R., Maskalik, S. and Sequeira, A.: Virtualizing networking and security in the cloud, *ACM SIGOPS Operating Systems Review*, Vol.44, No.4, pp.86–94 (2010).
4) Prevelakis, V.: The Virtual Firewall, *;login: The USENIX Magazine*, Vol.30, No.6, pp.27–34 (2005).
5) Advanced Micro Devices Inc.: AMD-V Nested Paging (2008).
6) Neiger, G., Santoni, A., Leung, F., Rodgers, D. and Uhlig, R.: Intel virtualization technology: Hardware support for efficient processor virtualization, *Intel Technology Journal*, Vol.10, No.3, pp.167–177 (2006).
7) Reguly, T.: Microsoft ICS DoS FAQ, 2006 (Online), available: http://blog.ncircle.com/archives/2006/10/microsoft_ics_dos_faq.html
8) VMware Inc.: Virtualization Overview (2006).
9) VMware Inc.: Understanding Full Virtualization, Paravirtualization, and Hardware Assist (2007).
10) VMware Inc.: VMware Virtual Networking Concepts, pp.1–11 (2007).
11) Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. and Warfield, A.: Xen and the art of virtualization, *Proc. 19th ACM Symposium on Operating Systems Principles*, ACM, pp.164–177 (2003).
12) Miller, K.W., Voas, J. and Costello, T.: Free and Open Source Software, *IT Professional*, Vol.12, No.6, pp.14–16 (2010).
13) VMware Inc.: Networking Performance, pp.1–9 (2008).
14) Menon, A., Santos, J.R., Turner, Y., Janakiraman, G. and Zwaenepoel, W.: Diagnosing performance overheads in the Xen virtual machine environment, *VEE'05*, ACM (2005).
15) Sugerman, J., Venkitachalam, G. and Lim, B.: Virtualizing I/O devices on VMware workstation's hosted virtual machine monitor, *USENIX Annual Technical Conference*, pp.1–14 (2001).
16) Oracle Corporation: *Oracle VM VirtualBox User Manual*, ch.10, pp.152–161 (2011).
17) Intel: *Intel Architecture Software Developer's Manual Volume 1*, Vol.1, No.243190 (1999).

18) Oracle Corporation: Oracle VirtualBox (Online), available: http://www.virtualbox.org/
19) Manning, L.: SmoothWall Express: Express Open Source Firewall Project (Online), available: http://www.smoothwall.org/
20) Jones, R.: Netperf (Online), available: http://www.netperf.org/netperf/
21) Juniper Networks: IDP Series Intrusion Detection and Prevention Appliances (IDP75, IDP250, IDP800, IDP8200), pp.1–6 (2009).
22) Cisco: Cisco ASA 5500 Series Adaptive Security Appliances, pp.1–22 (2010).
23) Jones, R.: *Care and Feeding of Netperf*, Hewlett Packard Company (2007).
24) Cisco: Cisco ASA 5500 Series Adaptive Security Appliances Model Comparison, 2010 (Online), available: http://www.cisco.com/en/US/products/ps6120/prod_models_comparison.html#mid-range

(Received October 31, 2010)
(Accepted April 8, 2011)
(Original version of this article can be found in the Journal of Information Processing Vol.19, pp.378–388.)

Ahmad Bazzi received his Diploma in Computer and Telecommunications Engineering from the Lebanese University, Beirut in 2003. In 2011, he received his Master's degree in Computer Engineering from Gunma University, Japan, where he is currently pursuing his Ph.D.

Yoshikuni Onozato received his D.E. degree in Electrical and Communication Engineering from Tohoku University, Sendai in 1981. Since April 1992, he has been with Gunma University where he is currently a Professor. His research interests lie in the areas related to computer communications. He is a member of IPSJ.