The Virtual Firewall

Vassilis Prevelakis
Computer Science Department
Drexel University

1. Introduction

The trend towards portable computing means that the traditional security perimeter architecture (where a firewall protects computers in the LAN by controlling access to the outside world) is rapidly becoming obsolete. This has resulted in a number of products described as "personal firewalls" that control that computer's access to the network and hence can protect it in the same way as a traditional firewall. Existing systems such as Windows and most Unix and Unix-like systems already provide security features that can be used to implement firewall functionality on every machine. However, the difficulty of securing general purpose operating systems has impeded the widespread use of this approach. Moreover, it is difficult to ensure that a secured system remains secure after the user has had the opportunity to install software and perform reconfigurations and upgrades.

Recognizing the futility of attempting to secure the user machines themselves, in [Prev03, Denk99] the authors proposed the use of a portable "shrink-wrapped" firewall. This was a separate machine running an embedded system that included firewall capabilities and was intended to be placed between the general purpose computer and the network. The problem of securing the firewall became much simpler as it utilized a special-purpose firewall platform with a highly controlled architecture. Sadly, the proposal saw limited adoption because carrying around yet another device is expensive and inconvenient. To make matters worse, if the external device is lost or damaged the user will be presented with a dilemma: remain disconnected from the network until the firewall box is replaced, or accept the risk and connect the laptop directly to the unprotected network.

In this paper we propose a compromise solution whereby the firewall is run under the host operating system within a virtual machine. The virtual machine environment we have used is VMware, which means that the technique described here can be used for both Windows and Linux platforms. The virtual firewall imitates the hardware firewall device with the exception that it is an entirely software-based system. We first describe the firewall itself and then the changes to the Windows host environment to ensure that the firewall controls access to all external networks, including wireless connections. Finally, we discuss some security considerations that affect the use of this platform.

2. Virtual Firewall (VF) Architecture

The Virtual Firewall platform supports tools for packet filtering, traffic monitoring and management. In addition we require IPsec support to allow a mobile station to be connected transparently with its home network. Secondary requirements include the ability to boot very quickly (increasing availability), minimal maintenance and a very small footprint (both in terms of RAM and virtual disk).

Figure 1 shows the integration of the VF within a Windows host environment. The host operating system has minimal access to the network (enough to support bridging between the guest VM running the Virtual Firewall and the network). As far as the host OS is concerned the VF is its default gateway (i.e. the only way for IP traffic to reach the outside world).

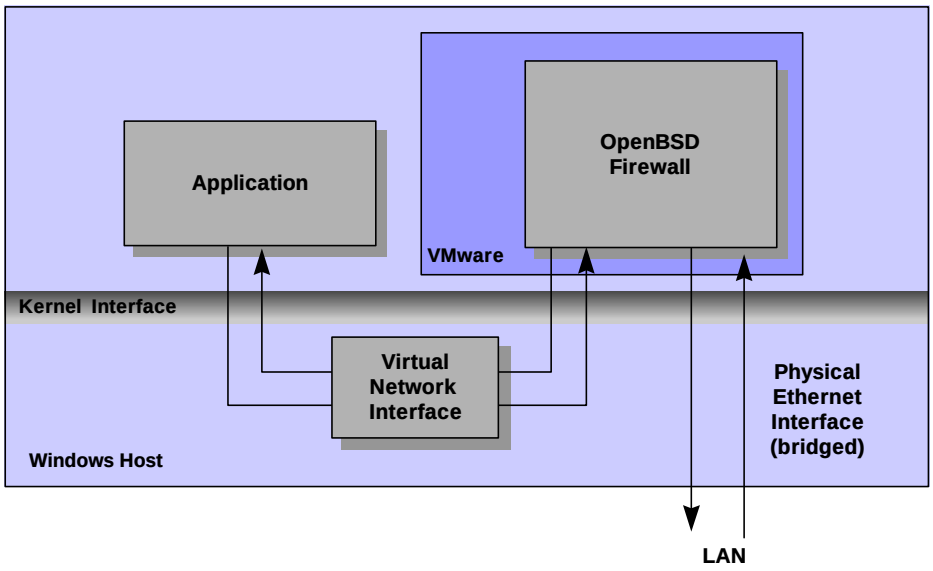

Figure 1: Communication links

The VF has at least two network interfaces, an internal (virtual) interface for communication with the host OS and the external, which is bridged to the outside network. The VF runs an embedded version of the OpenBSD 3.7 system which boots off a read-only medium and contains only firewall-related software (more on this later on).

The VF operating system is not aware that it is running under VMware, allowing us to migrate an existing version of our firewall that normally runs on a single-board computer. An earlier version of this system is discussed in [Prev02].

2.1 Firewall

The VF's default packet filtering policy allows traffic from the interior network to flow through it to the outside network and optionally via an IPsec VPN to some home network. At the same time it allows only a very restricted set of incoming connections. This implies three classes of restrictions:

- **Public Network:** This refers to packets coming in from the interface that is connected to the public network. Incoming connections are generally blocked except IPsec which has its own security mechanisms. Moreover, we allow ICMP echo and reply messages for network troubleshooting, but we block other ICMP messages.
- **IPsec VPN traffic:** Packets received from the interface connected to the local (protected) network and destined for the remote end of the VPN connection fall within this category of restrictions.
- **Local (internal) Network:** We generally do not allow connections to the VF itself. Exceptions to this rule include services such as DNS which is required for the operation of the node. We also allow certain types of ICMP packets for network troubleshooting.

When considering security mechanisms, there is always a need to strike a balance between security and convenience. Making life difficult for the Windows user is counter-productive as it will likely result in the VF being disabled. This consideration influenced the decision on outgoing connections. While there may be some justification in restricting such connections (e.g. to prevent spyware from leaking sensitive information) we decided to keep the default configuration of the packet filters relatively relaxed leaving the workstation user to decide on whether a more strict policy should be imposed.

Table 1 shows a typical configuration of the packet filter with the IPsec VPN rules removed to keep the table short. Note that we also perform Network Address Translation for the Windows host to allow it to have direct access to the outside network. Another approach would be to configure the VF as a layer-2 firewall but so far we have not encountered any problem with the NAT solution, which also ensures that outside parties cannot address the Windows host directly.

```
# interfaces
int_if = "le2"
ext_if = "le1"
# sshd (22)
tcp_services = "{ 22 }"
icmp_types = "echoreq"
priv_nets = "{ 127.0.0.0/8, 192.168.135.0/24, 192.168.136.0/24 }"
# options
set block-policy return
set loginterface $ext_if
# scrub
scrub in all
# nat/rdr
nat on $ext_if from $int_if:network to any -> ($ext_if)
# filter rules
block on  $ext_if all
pass quick on lo0 all
# no packets from/to private nets on the outside
block drop in  quick on $ext_if from $priv_nets to any
block drop out quick on $ext_if from any to $priv_nets
pass in on $ext_if inet proto tcp from any to ($ext_if) \
      port $tcp_services flags S/SA keep state
pass in inet proto icmp all icmp-type $icmp_types keep state
# packets for the Windows host
pass in  on $int_if from $int_if:network to any keep state
pass out on $int_if from any to $int_if:network keep state
#
pass out on $ext_if proto tcp all modulate state flags S/SA
pass out on $ext_if proto { udp, icmp } all keep state
```

Table 1: Sample packet filter configuration

2.2 Virtual Firewall Services

The Virtual Firewall platform runs two vital services: DHCP to acquire an address for the external network interface, and DNS. The latter may appear to be redundant, until we consider problem of ensuring correct name resolution for the Windows system. On the VF side, the DHCP client will ensure that the VF has the addresses for the local DNS proxies, but the Windows host will not normally have access to this information.

As a result, we run a small DNS server on the VF. This server is not consulted by the VF itself, because, as a firewall it only uses its own (static) host table. The DNS server is only for the benefit of the Windows environment which has the address of the VF statically assigned as a DNS server.

This configuration works satisfactorily, until we try to connect to some network with a split horizon DNS server. In this case our built-in DNS server will not have access to the internal DNS information. To deal with similar situations, we intend to install a DNS proxy on the VF and change the `dhclient-script` file to update the proxy's configuration with the IP address of the DNS server on the LAN.

3. Host Operating System Configuration

Although the discussion in this section assumes a Windows environment (Windows 2000 in particular), most of the comments and suggestions made below apply equally to newer versions of Windows as well as other platforms that support VMware (e.g. Linux). The techniques described have been tested with VMware 5.0, but they should work with earlier releases as well (with the exception of USB-attached devices, discussed below).

3.1 Installing the Virtual Firewall

Assuming that VMware is installed and running (see www.vmware.com for details) we need to configure a new virtual machine that will run the VF. Follow the wizard to create a new virtual machine with the minimum of allowed disk space and 64Mb RAM. The VM should have two Ethernet interfaces, one bridged to the external network, the other a host-only internal connection. With the VM running, the VF operating system is then installed on the virtual disk created by VMware.

3.2 Network Configuration

Assuming that VMware has been installed and the virtual firewall is running, we have to configure the network interfaces to ensure that Windows never tries to access the external network directly. We do this by defining an internal (virtual) network and instructing the Windows host to use a default gateway (the VF machine) on that network and turn off the IP services from the real Ethernet port. In this way communications from a Windows application (e.g. Firefox) will be directed to the OpenBSD VF which will perform NAT and send the packet on its way (see Figure 1). Similarly the reverse path is followed for incoming packets. Since we have turned off IP processing from the Windows Ethernet interface, Windows will not respond to IP packets arriving on that interface.

The Windows Ethernet interface should be operational (i.e. do not disable it) since the VF system will be using it as a bridge to the external network. Be sure to remove support for Internet Protocols from the interface (and *only* that interface, you still need IP to talk to the VF). After disabling IP from the external interface, it will stop showing up in `ipconfig` reports. Notice that in the Windows network configuration panel (Figure 2) all check boxes are clear except for the VMware bridge protocol.

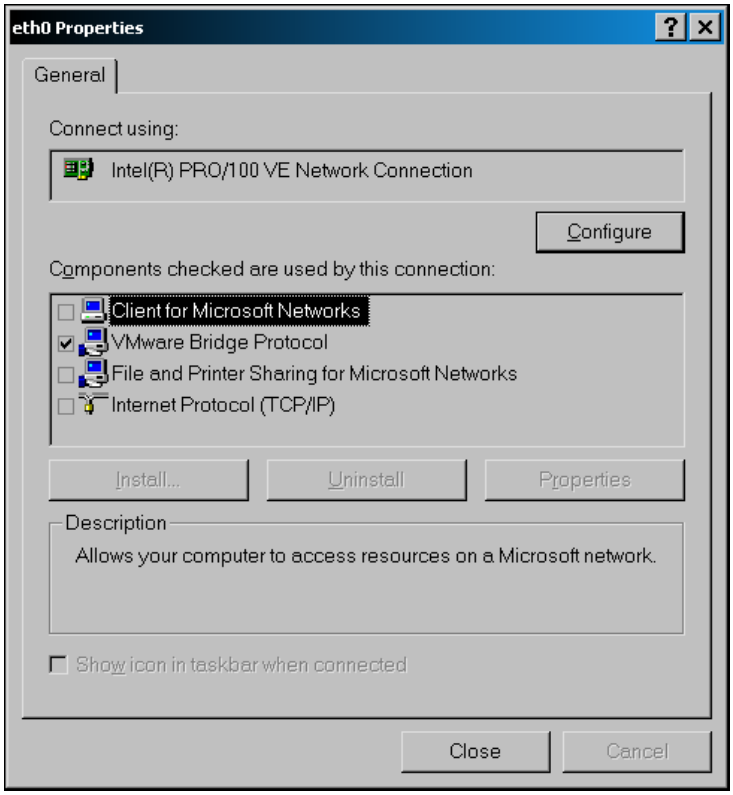

Figure 2: Configuration of Windows Ethernet Interface

The next step is to instruct Windows to use the virtual host-only interface (that links it with the VF) for its communication with the outside world. In other words the VF will be the default gateway for the Windows machine and its DNS server. Figure 3 shows the configuration in my system. Windows has the address `192.168.135.1/24` and the VF is `192.168.135.128/24`.

Note that both Windows and the VF have statically configured addresses (i.e. they do not use DHCP for their configuration) in their internal interface, meaning that VMware should not be running its DHCP service on the internal network.

```
Ethernet adapter VMware Network Adapter VMnet8:

 Connection-specific DNS Suffix  . :
 Description . . . . . . . . . . . : VMware Virtual Ethernet Adapter for VMnet8
 Physical Address. . . . . . . . . : 00-50-56-C0-00-08
 DHCP Enabled. . . . . . . . . . . : No
 IP Address. . . . . . . . . . . . : 192.168.135.1
 Subnet Mask . . . . . . . . . . . : 255.255.255.0
 Default Gateway . . . . . . . . . : 192.168.135.128
 DNS Servers . . . . . . . . . . . : 192.168.135.128
```

Figure 3: Configuration of Windows virtual Ethernet port

Wireless Networks

So far we have been looking at a wired Ethernet interface, but in many cases a laptop is likely to be using a wireless Ethernet interface. This configuration has caused us a number of headaches as in most cases Windows wants to configure the WiFi connection on its own and we have to prevent it from doing so, since we do not want it to acquire an IP address.

The VMware environment will also have to be configured to create a bridged connection between the WiFi interface and the VF. As in the case of the wired Ethernet interface, the WiFi card has to operate in bridging mode. Some WiFi cards cannot do this at all, others only allow this configuration to work if the card is configured without encryption, etc. If the card is compatible with the required configuration, then we can use it in the same way as the wired interface, otherwise we need to proceed to Plan B which involves the use of an external (USB-attached) Ethernet interface.

3.3 USB-attached Ethernet Interfaces

VMware allows USB devices to be connected to (and controlled by) a Virtual Machine. This feature allows us to connect a USB WiFi device to our computer in such a way that the Windows environment is oblivious to the existence of the device which is controlled entirely by the VF. Unlike the case of the wired Ethernet interface or the built-in WiFi interface, this method allows us to have an Ethernet interface that is invisible to the host environment.

The USB attached network device must be supported by the operating system of the VF (OpenBSD) since it is the VF that manages the device. We have had some trouble identifying a USB WiFi device that is available for purchase and is supported by OpenBSD, but fortunately the recent 3.7 release has greatly increased the number of supported devices.

The USB network device will have its own OpenBSD device identifier (e.g. `atu0`) which means that the network and `pf(4)` configuration in the default installation will need to be updated. A major benefit of the directly controlled device is that the VF can use it to examine the existing wireless networks for any antisocial activity before bringing up IP on that interface.

4. Security Considerations

When running a firewall as a service of a general purpose OS, there is always the risk that some software will interfere with the operation of the firewall. This is actually very common with “Personal Firewall” products that run under Windows [Ratt04]. In fact, recently released hostile software (malware) such as the Bagle-BK Worm [Esec05], have been known to turn off virus protection and firewall features as soon as they take over a machine.

Running the firewall in a separate VM should, therefore, be viewed as an improvement in the context of better management of the network connection (by channeling it though the firewall) rather than as bringing the security provided by an external firewall to your desktop.

Another concern is that a hostile application may not even need to deal with the VM. Since the OS has access to the network hardware (assuming the wired Ethernet case) a virus may contain its own IP stack and hence access the network directly (via the layer 2 interface). Moreover, since packets pass through the host OS to reach the firewall, it is possible that the host can still be attacked (via a layer-2 exploit). Normally I’d say that the chances of this happening are pretty remote, but Windows being Windows, we fear that some user-friendly feature of the OS will manage to get in the way. Alternatively, some combination of events may cause Windows to activate spontaneously IP services on the interface without asking the user. In the case of the wireless connection, the situation is even worse, with a lot of automated processing going on the host. Windows is so intrusive that in some cases it cannot even be convinced to keep the wireless hardware disabled. For this reason, the USB-based wireless solution (although more cumbersome) offers a direct path from the firewall to the hardware with the host seeing only the USB traffic. In general, the less the host OS knows about the network connection, the better.

A more comprehensive solution from the security standpoint is that proposed in [Meus00] where a stripped down host OS runs various VMs. One of the VMs may run the Windows user

interface and associated applications, while another may run the firewall. For performance reasons this approach is not yet feasible. For example, applications such as games or DVD players that create a high bandwidth connection between a mass storage device, the CPU and the video device will suffer unacceptable performance degradation when run in a VM. Internet audio, VoIP and chat applications, however, can be easily placed in such a sandbox. Such applications typically initiate a connection from the Windows host to some external server, which means that, the firewall can do little against an attack vectored through the outbound connection (assuming the user wants to run such an application, the firewall cannot prevent the application from connecting in the first place). Our solution is to run this software on a separate VM with non-persistent secondary storage. At least if/when they run amok they will not break anything.

Having discussed the case where the host attacks the VM, let us now consider the opposite where an intruder escapes from the VM and compromises the host. Although ultimately possible, the dual layer (host plus firewall) provides defense in depth, thus allowing time to detect the attack on the firewall and take appropriate action. Having said that, the existence of the firewall will likely exacerbate the already weak security posture of the Windows host by encouraging complacency (why bother to turn off service *foo* when the firewall will prevent anybody from connecting to it anyway). Still the firewall VM provides a good vantage point to monitor traffic and to launch port scanning checks on the Windows host to identify weaknesses.

5. Conclusions and Future Plans

Now that we have finished the description of the Virtual Firewall, we can return to the original claim in the introduction and discuss whether having one is actually justified. There is no doubt that, nowadays, a firewall on each computer is necessary; this is why Microsoft is bundling a firewall with its Windows XP platform. So the question is really why a separate firewall on a virtual machine, rather than a firewall as part of the base OS. It is fair to say that keeping the firewall separate simplifies its administration, as its configuration and maintenance is completely separate from that of the rest of the OS. This allows the management of the firewall to be carried out without requiring the cooperation of the workstation user which may be a considerable advantage in centrally managed environments. Within a large corporate environment the ability to have the firewall distinct from the rest of the machine may simplify the deployment of security policies and VPN operations [Arba98]. This may evolve in the Distributed Firewall concept [Ioan00] where global network policies are enforced by firewalls installed on each machine in the network.

Moreover, it also simplifies upgrades and security patches to the VF since these cannot affect the host OS. For example, we have a Windows 2000 machine that would only boot in safe mode after installing the latest OS service pack.. Such considerations may impede timely upgrades and hence open windows of vulnerability to the system. Finally, changes to the firewall configuration cannot be done via a user-friendly interface that may hide vital information from the administrator. Most importantly, configuring an application on the host will not result in an accidental change in the firewall policy.

The system described here has been in operation for about six months and has been “stress-tested” by linking the workstation to unprotected wireless networks, taking it to numerous conferences and trade shows, etc. We plan to use the platform to acquire long term attack data that will help us refine the security policy of the VF and create a wireless network forensic database. Another way that the VF choke point can be useful is in analyzing the DNS requests made by the host. We hope that by monitoring DNS lookups we can create a personality profile for the user and use that to detect the existence of spyware or other unauthorized programs on the host platform.

We also plan efficiency enhancements to ensure that the VF requires minimum resources from the hosting platform and becomes available with negligible delay during boot. VMware requires a minimum hard disk allocation of 100Mb. While this may not appear to be excessive, our firewall can boot from an 8Mb compact flash, so the other 92Mb are wasted and could be returned to the system. We are also in the process of evaluating exactly how much RAM is needed by our system in order to come up with a reasonable configuration for the virtual machine. VMware allows the user to suspend a virtual machine and then restart it with little delay. We are looking into creating a "frozen" configuration of the VF, one that is ready to be resumed, rather than one that boots when the virtual machine is initialized. This will allow almost instant availability for the firewall and very fast resets (since a reset will resume the frozen configuration).

Acknowledgments

I would like to thank Angelos Keromytis for suggesting that running a firewall under VMware might be possible. Microsoft was also instrumental in getting this work done, by releasing security patches, that cannot be installed on my machine (they crash it). Without these security patches the use of a separate firewall became imperative, so I had to develop one.

References

[Arba98] William A. Arbaugh, James R. Davin, David J. Farber, and Jonathan M. Smith, "Security for Virtual Private Intranets," IEEE Computer (Special Issue on Broadband Networking Security), Vol. 31(9), pp.48-55, September 1998.

[Denk99] Denker, John S., Steven M. Bellovin, Hugh Daniel, Nancy L. Mintz, Tom Killian and Mark A. Plotnick, "Moat: A Virtual Private Network Appliance and Services Platform," *LISA'99: 13th Systems Administration Conference*, Washington, November 1999.

[Esec05] http://www.esecurityplanet.com/alerts/article.php/3487701

[Ioan00] Ioannidis S., A.D. Keromytis, S.M. Bellovin and J.M. Smith, Implementing a Distributed Firewall, *Proceedings of Computer and Communications Security (CCS) 2000*, pp. 190-199.

[Meus00] Robert Meushaw and Donald Simard, "NetTop: Commercial Technology in High Assurance Applications," Tech Trend Notes, National Security Agency, Volume: 9 Edition: 4, Fall 2000.

[Prev02] Vassilis Prevelakis, Angelos Keromytis, "Designing an Embedded Firewall/VPN Gateway," Proceedings of the International Network Conference 2002, Plymouth, UK.

[Prev03] Vassilis Prevelakis, Angelos Keromytis, "Drop-in Security for Distributed and Portable Computing Elements," Journal of Internet Research, Volume 13 Issue 2, MCB Press, 2003.

[Ratt04] rattle, "Bypassing Windows Personal FW's," Phrack Magazine Vol 11 Number 62, Build 3, Jul 13, 2004.