

Software Firewalls versus Wormhole Tunnels

Bob Rudis and Phil Kostenbader 2005-05-04

Hardware and software firewalls promise to protect your system and your network from the dangers of the Internet, but how well do they really fare against cutting-edge attacks? This article presents some of the major differences between hardware and software firewalls and illustrates the real challenges faced by software firewall vendors.

Hardware versus software

Firewalls come in many different flavors to meet varying needs of users. There are dedicated hardware firewall appliances, like the Cisco PIX, complete with cool flashing lights, that provide security for whole networks. There are also dedicated software firewalls, such as CheckPoint/ipchains/iptables/ipf/pf, running on robust operating systems used as gateways that also provide complete network security. Finally, there is a new generation of "personal" firewalls that are designed to run on and protect workstations and servers directly.

Dedicated hardware or software firewalls usually provide at least one-dimensional coverage -- port/IP blocking -- for your network. Some even provide two-dimensional coverage by adding active intrusion detection (IDS) or intrusion prevention (IPS) to their basic functions. Personal firewalls can even go one step further and provide three-dimensional coverage with the addition of active application blocking.

Most organizations will end up deploying a combination of dedicated and personal firewalls to provide comprehensive asset protection both on and off their internal networks.

Perils of personal firewalls (and other dedicated firewall software)

While dedicated and personal firewalls are usually managed in a similar fashion, personal firewalls are more challenging for IT departments to deploy and control. Local users with administrative privileges -- not an unlikely scenario in most shops running Microsoft Windows -- will be able to start and stop the firewall process at will. Similarly, users with sufficient privileges or malicious programs that exploit a vulnerability on a system will be able to install software that "climbs over the top" of the firewall. In situations where the user is notified of potentially dangerous activity (usually via a pop-up window), too often they will typically select "allow", fearing the loss of the ability to operate on the network. This is the most typical mode of operation for viruses or worms spread by e-mail or spyware installed through a web browser.

Since most large organizations worry about support costs, they are unlikely to even have a personal firewall configured in this highly restricted mode. Generally, the firewall will be deployed with the most basic policy possible to protect the system with no potential to interfere with core business functions.

Even if more restricted blocking is in place, web, e-mail and other types of standard Internet traffic will usually be allowed from "standard" components and thus provide an equally accessible conduit for malicious programs.

Once something comes in "over the top" it is, unfortunately, all too easy for it to dig its way out underneath or even go straight through the firewall.

Most personal firewalls operate like a bouncer at a nightclub: they are a part of the system and look closely at folks coming in and are aware of who's going out. But how closely can they really watch everything? Will a bouncer catch a patron stealing something they could be hidden in their pocket? Will a bouncer profile who they are going to watch and miss those that do not fit the definition?

In much the same way, personal firewalls operate as either an application or as a combined application/kernel component that sits and watches what it is fed by the network component of the kernel. Like a bartender, the operating system doesn't really even concern itself with what it's processing (remember, it's the bouncer's job to restrict access). If one could devise a way to consistently stay out sight of the bouncer and get direct access to the bartender, the drinks would never stop flowing. The practical question then remains: is it possible to have network traffic flow through a system and never be seen -- or at least cared about -- by any active personal or dedicated software firewall? The answer, unfortunately, is yes.

Wormhole-tunneling with PCAP

[PCAP](#) is a library designed to provide applications with the ability to access network interfaces using a standard API and capture packets directly. Perhaps the most common example of this is the [Ethereal](#) program. Ethereal is an open source, multi-platform packet sniffer and protocol analyzer. With it, one can get a raw view of what's coming in and going out of the PC. The following scenario uses just Ethereal to demonstrate how trivial it is to bypass these software firewalls.

Scenario 1

Find a Windows 2000/XP/2003 or BSD/Linux machine and either load a copy of your favorite firewall software or use the one that comes with it (most modern Windows and BSD/OS X/Linux systems come with a built-in software firewall). Configure the firewall to block all inbound TCP and UDP traffic and put the system on the network. Install Ethereal (which requires you to also install libpcap) and run it. If the system is on a switch, all traffic headed from or to it will be seen along with any broadcast traffic. If the system is on hub, all network traffic will be seen.

Now, if the firewall is blocking all inbound TCP/UDP packets, how is it possible to see the traffic? Obviously, Ethereal is seeing the packets *before* they reach the firewall. Do a full nmap ([<http://www.securityfocus.com/print/infocus/1831> \(2 of 7\) \[7/4/2008 5:07:27 PM\]](http://</p></div><div data-bbox=)

www.insecure.org/nmap/) port scan with OS detection (which should fail) against the system. If the firewall is running any type of active protection mechanism, it would detect and block this scan even if the inbound policy was not set to block all traffic. However, the point to note is that *Ethereal still sees the traffic.*

Based solely on this simple scenario, it is clear that libpcap can be used to listen for inbound traffic that is supposed to be blocked. This provides an unfiltered incoming channel for malicious programs. If it were possible to send packets out via this same "wormhole-tunnel" then a malicious program would have a full, two-way communications channel to operate on.

For sending packets, once again we see that the libpcap library provides comprehensive support for this see [Sending Packets with libpcap](#). The outbound traffic goes completely underneath the firewall.

Unprotected on the road

While the scope of this paper is focused on software firewalls, the next scenario adds VPN into the mix since most "road warriors" run a combination of firewall and VPN in order to use resources on their home network while protecting the mobile asset. Let's look at this scenario.

Scenario 2

If you have a VPN client, add it to the configuration described in Scenario 1. Make sure split tunneling is disabled to ensure that all traffic is forced through the VPN connection. Start Ethereal and perform the same nmap scan against the local interface address (not the VPN address). The traffic that is supposed to be blocked by both the firewall and VPN software -- or at least be "invisible" to the VPN software -- is still available at the PCAP-level.

In a VPN configuration, most personal firewalls are configured to drop their shields (because all traffic is heading to and from a trusted source), so the VPN client is, in fact, a liability because there is no need to use a libpcap outbound wormhole-tunnel communications channel. The firewall will happily ignore whatever packets a malicious program might need and they go unfiltered through the "secure" VPN connection.

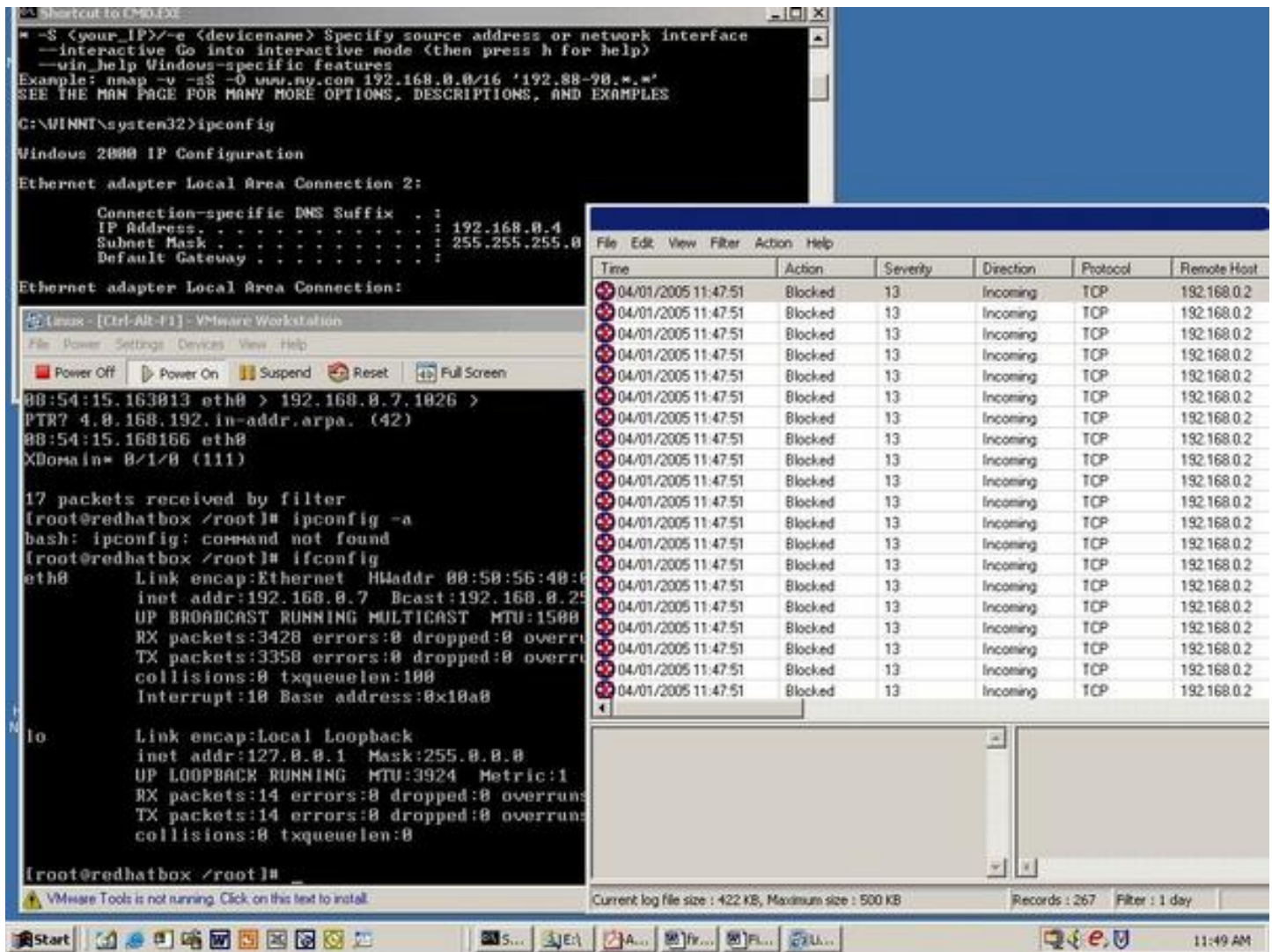
Emulating insecurity

Creating and using a wormhole-tunnel communications channel is not limited to malicious use by malware, spyware, viruses or worms. The following scenario illustrates how one can legitimately (and more robustly) bypass the firewall without the use of libpcap.

Scenario 3

Load a copy of [VMWare](#) to startup an emulated Linux environment on a Windows box. Configure it so that the emulated network is in bridged mode.

With the firewall blocking, run an nmap scan against both the PC's IP address and the emulated environment's IP address. The firewall will block the PC scan, but the emulated environment will gladly respond back to the probe attempt, as shown below in Figure 1.



The screenshot displays a Windows XP desktop with a VMware Workstation window and a Windows Firewall log window. The VMware window shows a Linux terminal with the following output:

```
VMware Tools is not running. Click on this text to install.
[root@redhatbox /root]# ipconfig -a
bash: ipconfig: command not found
[root@redhatbox /root]# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:5B:56:40:68
          inet addr:192.168.0.7  Bcast:192.168.0.255
          UP BROADCAST RUNNING MULTICAST  MTU:1500
          RX packets:3428 errors:0 dropped:0 overruns:0
          TX packets:3358 errors:0 dropped:0 overruns:0
          collisions:0 txqueuelen:100
          Interrupt:10 Base address:0x10a8

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924 Metric:1
          RX packets:14 errors:0 dropped:0 overruns:0
          TX packets:14 errors:0 dropped:0 overruns:0
          collisions:0 txqueuelen:0

[root@redhatbox /root]#
```

The Windows Firewall log window shows the following entries:

Time	Action	Severity	Direction	Protocol	Remote Host
04/01/2005 11:47:51	Blocked	13	Incoming	TCP	192.168.0.2
04/01/2005 11:47:51	Blocked	13	Incoming	TCP	192.168.0.2
04/01/2005 11:47:51	Blocked	13	Incoming	TCP	192.168.0.2
04/01/2005 11:47:51	Blocked	13	Incoming	TCP	192.168.0.2
04/01/2005 11:47:51	Blocked	13	Incoming	TCP	192.168.0.2
04/01/2005 11:47:51	Blocked	13	Incoming	TCP	192.168.0.2
04/01/2005 11:47:51	Blocked	13	Incoming	TCP	192.168.0.2
04/01/2005 11:47:51	Blocked	13	Incoming	TCP	192.168.0.2
04/01/2005 11:47:51	Blocked	13	Incoming	TCP	192.168.0.2
04/01/2005 11:47:51	Blocked	13	Incoming	TCP	192.168.0.2
04/01/2005 11:47:51	Blocked	13	Incoming	TCP	192.168.0.2
04/01/2005 11:47:51	Blocked	13	Incoming	TCP	192.168.0.2
04/01/2005 11:47:51	Blocked	13	Incoming	TCP	192.168.0.2
04/01/2005 11:47:51	Blocked	13	Incoming	TCP	192.168.0.2
04/01/2005 11:47:51	Blocked	13	Incoming	TCP	192.168.0.2
04/01/2005 11:47:51	Blocked	13	Incoming	TCP	192.168.0.2
04/01/2005 11:47:51	Blocked	13	Incoming	TCP	192.168.0.2
04/01/2005 11:47:51	Blocked	13	Incoming	TCP	192.168.0.2
04/01/2005 11:47:51	Blocked	13	Incoming	TCP	192.168.0.2

Figure 1. VMWare example.

The emulated environment is using similar techniques to "bypass" the firewall and use low-level network kernel access to perform its functions. In the example screen capture, the VMWare machine has an address of 192.168.0.7 and the host PC has an address of 192.168.0.4. The scan was performed from 192.168.0.2. The firewall -- running on 192.168.0.4 -- shows the blocked scan attempts.

Below, Figure 2 shows the results of the nmap scan of the VMWare client.

```

xterm

# /usr/sbin/ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
    inet 127.0.0.1 netmask ffffffff
hme1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
    inet 192.168.0.2 netmask fffffff0 broadcast 192.168.0.255
    ether 8:0:20:9a:af:af
# /usr/local/sbin/nmap -sS 192.168.0.7

Starting nmap V. 2.52 by fyodor@insecure.org ( www.insecure.org/nmap/ )
Interesting ports on redhat-box (192.168.0.7):
(The 1646 ports scanned but not shown below are in state: closed)
Port      State      Service
21/tcp    open       ftp
22/tcp    open       ssh
23/tcp    open       telnet
25/tcp    open       smtp
79/tcp    open       finger
111/tcp   open       rpcbind
113/tcp   open       auth
513/tcp   open       login
514/tcp   open       shell
587/tcp   open       submission
1024/tcp  open       kdm

Nmap run completed -- 1 IP address (1 host up) scanned in 4 seconds
# /usr/local/sbin/nmap -sS 192.168.0.4

Starting nmap V. 2.52 by fyodor@insecure.org ( www.insecure.org/nmap/ )
All 1657 scanned ports on the-primary-pc (192.168.0.4) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned in 1452 seconds
you have mail
#

```

Figure 2. Nmap scan of the VMWare environment.

If both scenarios two and three are combined, the setup will have a workstation with a VPN connection (with split-tunnelling disabled) and also a running VMWare client. While the VPN is engaged, the VMWare session will have complete access to the *local* network. With some configuration tweaking and some minor scripting, the VMWare client could act as a "bridge" between the local network and the VPN target network -- something that *should* be disabled when split-tunnelling is disabled.

Protecting systems/networks with (and from) PCAP

There are legitimate uses of the libpcap library, beyond traditional packet sniffers, that even go so far as to help protect networks from attack. Scenario 4 shows an example of this.

Scenario 4

Build a BSD/OS X/Linux system with a firewall set to block all except ssh traffic (used for management). Install a copy of the [SnortIDS](#), which is libpcap-based, and then run [IDSwakeup](#) against the system. It might be a good idea to update the IDSwakeup signatures with the latest snort signatures for a comprehensive scan. Snort will detect and report the generated intrusion attempts despite the fact that the firewall is blocking almost all the traffic. This provides

a mechanism to deploy a very secure snort IDS system without making it completely hidden from the monitored network.

Despite any potential good that libpcap-based programs might offer, there are some steps that can be taken to defend against malicious use of this wormhole-tunnel channel:

- *Disable admin access*

The libpcap library -- and others like it -- can only be installed with administrator-level privileges. If Windows users are not allowed/configured to run with those privileges, the likelihood of a libpcap exploit is significantly reduced. Un-patched systems with privilege escalation vulnerabilities would still be at risk, however.

- *Configure application restrictions in the personal firewall*

Most personal firewalls enable administrators to define a list of applications that are specifically allowed or denied access to run. Microsoft's Windows 2000 and newer operating systems have built-in mechanisms for software restriction policies as well. If possible, configure your workstations to only allow the specific programs that are necessary for users to do their daily work.

- *Monitor application usage*

Use tools such as [Altiris](#) or Microsoft's [Systems Management Server \(SMS\)](#) to regularly scan and report on all installed executables and instances of libraries such as libpcap. Review these reports regularly to ensure that only allowed applications are installed. You can even use tools such as Microsoft's [PromqryUI](#) to scan for systems that have their NICs in promiscuous mode, since some of the wormhole-tunnel hacks will end up placing an interface in this mode.

Microsoft's own [Malicious Software Removal tool](#) and their [AntiSpyware](#) software (currently in beta) can also be used to scan for these libraries and programs that use them and either block them outright or log their usage to a file for later processing.

- *Use network intrusion detection systems (NIDS) for a layered defense*

When compromised systems are on the internal network, well-maintained and monitored NIDS environments should detect anomalous use of network resources and will definitely detect known signatures of malicious network traffic.

Conclusion

As attacks become more sophisticated, the traditional multi-layered approach to security will become more and more crucial to ensure the integrity of systems and networks. These wormhole-tunnels have the potential to increase the number of "zombie" PCs on the Internet even further -- [estimated at one million](#) already -- since they can bypass unmonitored protection mechanisms. It is important that users be educated regularly on all of the threats they face when connected and that all systems are deployed with a defense in depth strategy. Using the approaches in this article will help you secure your network from these threats.

[Privacy Statement](#)

Copyright 2006, SecurityFocus