TINA DARMOHRAY

firewalls and fairy tales

Tina Darmohray, contributing editor of *;login:*, currently serves as Stanford's Information Security Officer. She is the editor of the Short Topics booklet *Job Descriptions for System Administrators* and was a founding member of SAGE.

tmd@iwi.com

THE RECENT ISSUE OF A POPULAR security rag had the following ad for a security appliance inside the front cover: "If You Believe Routers Can Secure Your Network, You'll Need Another Fairy Tale to Come True." This kind of sales and marketing FUD has been used to hype the differences in security products for years. With firewalls, in particular, there's some historical background to all the mud slinging which can give consumers some market insight and savvy when shopping for a solution.

Recall that the Internet was the outcome of a cooperative research project. In its early days, the challenge was to get and keep the computers talking to each other. Many professional friendships were made between humans at either end of the "next hop," as manual intervention and tweaking of routing tables kept the Net humming. A spirit of cooperation permeated the network, and guest accounts and remote access were de rigueur.

In 1988 the Morris worm hit the Internet hard. It marked the end of blind trust on the Net. Before the Morris worm, professional gatherings focused on simplifying administration and improving reliability of the machines and the network that connected them. After the worm, attention turned to securing the network as well. Network administrators started cobbling together network access controls. Increasingly, Net News discussions, BoFs, and conferences dealt with the new focus on computer and network security. It became a necessary evil.

Today's booming computer security market masks the early struggles of security-minded individuals. The early implementers of network access control often found themselves not just explaining the technology but defending it as well, as they were perceived as being counter to the spirit of cooperation that had been the norm. Giving an invited talk on network security was sometimes like volunteering to be the person in the dunking booth at the local carnival!

Despite their unpopularity in the user community, firewalls were deployed one network connection at a time. The earliest firewalls consisted of routers configured to filter out packets destined for particular internal network ports, thereby denying access to internal services. To those managing the network connections, this was an obvious countermeasure. The router hardware and software already existed and was in place. All that had to be done was to use an existing capability to implement a more secure stance.

Additionally, since the changes were made at the network layer (layer 3 of the textbook 7-layer network model), they were transparent to users' applications. A packet-filtering router yields a lot of bang for the buck. Even simple access control lists (ACLs) can dramatically enhance site security. And, as with all firewalls, the "conservation of energy" is huge as you protect many internal machines with a small set of perimeter devices.

A simple filtering router examines network packets for source and destination IP address, source and destination port numbers, and protocol type. Using this information, decisions are made to route or reject packets. Administrators choose the ports to be forwarded or dropped based on the way that ports are assigned to applications. A UNIX kernel reserves ports < 1024 for privileged processes; it randomly assigns ports 1024 and above, as needed, to processes that request them. The reserved ports are conventionally assigned as "well-known ports" with a particular service being associated with a particular port number. Thus, rejecting packets destined for a particular well-known port translates into filtering out the corresponding service. For example, rejecting all traffic to TCP port 23 is intended to disallow Telnet connections. Similarly, rejecting traffic for TCP port 80 would disallow HTTP connections, and so on. (For a list of well-known ports and their assignments, see http://www.iana.org/assignments/port-numbers.)

Simple packet filters provided a cheap and easy security solution, but the debate had just begun. Many argued that they didn't provide enough protection. Like most security solutions, firewalls are about narrowing the window of opportunity for intrusion. Critics pointed out that packet-filtering firewalls innately fell short of application-level firewalls. For the most part, the criticisms focused on the tenuous binding of port numbers to applications. Recall that this binding is based on the UNIX convention of assigning ports. This is not a standard that must be rigidly adhered to, but a convention, a traditional way of doing things. To that end, there's nothing stopping someone from running a service on a different-than-expected port and thereby slipping around the packet filter. The most common example of this is the abuse of the high-end ports, numbered 1024 and higher. These ports must be allowed through static filters in order to accept the return traffic of internally initiated connections.

My favorite story on this subject concerns a battered university administrator who implemented a few simple packet filters to disallow some of the most dangerous services, like inbound Telnet. Predictably, the students rebelled and publicized their own version of the well-known port assignment convention, with Telnet found on the port number matching the last four digits of someone's telephone extension. Most of the new port assignments were > 1024, and glided right through the packet filter, much to the dismay of the security-minded network administrator.

The critics of packet-filtering firewalls usually fell in the application proxy firewall camp. They argued that proxy firewalls are more secure because, since they work at the application layer, they could examine the payload of the packet, enabling them to be protocol-specific. They could also authenticate traffic at the user level. However, in those early firewall days, purists who felt that packet filters just weren't secure enough were forced to build their own proxy firewalls, since there were none commercially available at that time.

Writing proxy firewalls isn't for everyone. There's real coding involved and, since it's for security purposes, it better be well written and torture-tested. That's too tall an order for many system administrators, so packet filters remained a popular alternative. However, some able and willing administrators took the time to write proxies. A few of those early efforts were the DEC SEAL, SOCKS, and the TIS Firewall Toolkit. The latter two played a key role in what happened next in the evolution of firewalls.

When SGI acquired MIPS one afternoon in 1992, the MIPS proxy firewall named SOCKS[1] became publicly available. Finally, a proxy firewall was available to the masses. SOCKS was written by David Koblas, a system administrator for MIPS Computer Systems. It is an elegant solution, ultimately providing a mechanism to protect internal systems by means of transparent proxies that operate at the TCP protocol level.

However, there was a hitch. The transparency came at a price: Software needed to be deployed on each client computer. While this was reasonable for the small homogeneous MIPS Computer network, it can be logistically prohibitive on any large-scale network with multiple OS platforms.

In 1994, the White House was shopping for a vendor who could get it onto the Internet securely. Marcus Ranum, then of Trusted Information Systems (TIS), was up to the task. He brought up whitehouse.gov and in the process developed the TIS Firewall Toolkit (FWTK). The FWTK proxies actually implemented a secure subset of the most common network application protocols. Because his work was funded with taxpayer money, Marcus decided to release a version of the FWTK to the Internet community. The release wound up being an ingenious marketing move that gained a huge following for the TIS technology. It didn't require any software modifications on client systems and therefore ran quite easily in heterogeneous environments. But the lack of software mods came at the price of transparency to the users. Users were exposed to the "double hop" it took to get out of the

network via the proxy firewall. For example, to Telnet outbound from the protected network, a user would Telnet to the proxy server and then issue the command to connect to the final destination.

While functional, this wasn't nearly as elegant as the entirely behind-the-scenes connections that transparent proxies like SOCKS made on the users' behalf. A further drawback to the true application proxy approach was the necessity of providing a specially coded proxy program for each protocol that was to traverse the firewall. With the rapid proliferation of protocols in the latter half of the 1990s, proxy firewall vendors were forced to play a continuous game of catch-up. So proxy firewall fans were left to port code or provoke users, with no real alternatives in between.

Meanwhile, vendors began to warm up to the emerging firewall market. Livingston Enterprises was among the first to market routers as firewalls. They heeded Brent Chapman's call for more security functionality and implemented the features he outlined as being critical (and missing in most vendors' routers).[2]

At the time, and for a good while after Livingston's early entry into the firewall market, leading vendors such as Cisco didn't provide the increased functionality that industry spokesmen like Brent were calling for. Cisco eventually caught up, but their slow market entry was a black eye for them with the security enthusiasts for years, and a soft underbelly on which proxy firewall proponents hammered.

Probably the most innovative of the early vendor entries in the market was Checkpoint's stateful packet-filtering firewall. Checkpoint introduced a packet-filtering product that considered connection-state information when deciding to pass or drop traffic. Tracking connection state (including "virtual" connections for nominally connectionless protocols like UDP) enables you to permit traffic to pass through the packet-filtering firewall only if it is associated with a connection you've explicitly approved, such as the data connection of an FTP session, or the UDP response to a DNS query. Checkpoint also introduced dynamic packet filtering, which opens only the ports you need at the time you need them. So, for example, if you need to permit traffic on a high-level port in response to an outbound connection request, that port is allowed, but only for the duration of the connection. These two major improvements to vanilla packet filtering really raised the security bar for packet-filtering solutions.

Aside from these early breakthroughs, there hasn't been a huge change in the bottom line of firewalls in more than a decade: packet-filtering firewalls look at layer 3 information, while proxies are able to inspect the actual content of the packets. Once you open the payload portion of the packet, you can make decisions on that content as well, e.g., on user identity or whether the data matches the kind of bits that should be associated with a specific protocol. Beyond that, though, most of the "new" entries into this market space are just new marketing variations on the old theme. Eventually the commercial version of the Firewall Toolkit, Gauntlet, developed transparent proxies and incorporated packet-filtering features to respond to its perceived failings. Packet-filter vendors added the ability to filter on packet payload. And everyone moved toward "appliance" firewalls and Web GUIs for administration.

TIS's Gauntlet and the Livingston Firewall router are gone, and a horde of new names have taken their place. The mainstream commercial firewall products available today are actually hybrids of the packet filters and application proxies first developed in the early 1990s. "Deep Inspection" firewalls are just packet filters that are going a little further up the stack than traditional packet filters. "Intrusion Prevention" devices are usually combinations of signature-based intrusion detection systems and traditional firewall technology that shut down connections based on patterns in their payload, in addition to making decisions based upon ports or protocols.

What is true is that solutions continue to merge, as each of the two primary areas takes a little more from one or the other and incorporates it into its baseline. A close look at the new marketing rarely reveals actual new technology, though.

Next time you see a new firewall advertisement, take a step back and analyze the claims through the lens of history before you decide to fear or to fantasize that fairy tales have come true.

1. D. Koblas, "SOCKS," *Proceedings of the Third USENIX UNIX Security Symposium* (Baltimore, MD: USENIX Association, September 1992).

2. D. Brent Chapman, "Network (In)Security Through IP Packet Filtering," *Proceedings of the Third USENIX UNIX Security Symposium* (Baltimore, MD: USENIX Association, September 1992).